Szókincsbővítés?

A napokban ismét felmerült egy olyan kérdés, melyet már sokszor megrágtunk, ez pedig a szókincs alakulása. Gondolok itt arra, hogy miként változtatja meg egy szakma a nyelv szókincsét, illetve viszont. Az informatika világában ez különösen érezhető, hiszen egy rendkívül gyorsan változó területről van szó, gondoljunk csak bele, egy-két év alatt annyi új fogalom kerül be a szakmába, amennyi tíz év alatt nem született meg egy fél évszázaddal ezelőtt.

A kérdés tehát nem az, hogy szükség van-e új fogalmakra, sokkal inkább az, hogy hogyan „kereszteljük el" őket. Ha felmerül egy új (többnyire angol) fogalom, a magyarításának általában két útja van. Az első változat, amikor az angol szó „könnyen kimondható" (megjegyzem, más mondható ki könnyen egy angolul tudó informatikusnak, mint egy átlagembernek). Ekkor a szó magyarításán a tömeg nem is agyal – nagy örömmel és némi gőgös büszkeséggel használják a beszédben az angol szót. Később jönnek a gondok, amikor le kell írni, mert hogyan is írjuk le, hogy key-jel, vagy hogy array-ben? És ha eddig sem született „elég vagány" fordítás, akkor bizony a magyar nyelv szabályai alapján szembejönnek majd a domének, a rúterek, a flopik meg az ofiszok.

A második lehetőség, ha „nem mondható ki könnyen", egy kicsit keményebb dió, hiszen itt már az első pillanattól fogva még a hunglist beszélők is komoly igényt éreznek egy megfelelő szó használatára. Nagyon rövid ideig tartott a vita a billentyűzet körül (senki sem kívánja, hogy a kíbordon kelljen dolgoznia, de sokáig tartották magukat a klaviatúrapártiak), és alig gondolkodik el az ember, hogy mauz vagy egér.

Gyakran emlegetett érv az új jövevényszavak mellett, hogy „az új szavakkal bővítjük a szókincset". Ez az a pont, ahol komolyan elgondolkodom. Mert ha valaki szveppel a diszken, ha loggolja, amit csinál, és lagzik, ha nem éri utol magát, akkor biztos, hogy sok „új szót" használ, de vajon az illető szókincse hosszú távon egészséges és kerek marad-e. Továbbra is használni fogja-e a színes magyar szókincs ugyanazokra a fogalmakra illeszkedő tagjait? Kialakul-e egyáltalán egy széles szókincs ilyen tolvajnyelv mellett?

AP → elérési pont

acces point → elérési pont

arányos szélességű – proportional: olyan betűkészlet, melynél a betűk egyedi szélességgel rendelkeznek, tehát például négy M betű egymás mellett lényegesen több helyet foglal, mint négy I betű.

bérlés – leasing: olyan versenyhelyzetkezelési módszer, amikor nem történik tényleges zárolás, de versenyhelyzet alapján a rendszer a bérlőnek üzenetet küld, miszerint is a „bérlemény megszűnt", egy másik folyamat változtat az erőforráson.

ciklus – loop: programszervezési elem valamilyen feltétel alapján többször végrehajtandó programrészlethez.

címke – tag: több értelemben használjuk, az angol tag szó fordításaként a jelölőnyelvek (például html, xml stb.) egyes vezérlésre, tulajdonság-meghatározásra használt elemei. Használják rá a magyar tag szót is (taggal, tagok), értelmileg a címke pontosabb.

context menu → helyi menü

fibre channel → rostcsatorna

fogantyú → handle: grafikai elemeket kezelő programoknál egy grafikai objektum kitüntetett pontjai (általában kis négyzetekkel jelölve), melyek segítségével különböző módosításokat végezhetünk (nyújtás, forgatás, átméretezés stb.).

handle → (1.) kezelő, → (2.) fogantyú

helyi menü – context menu: programokban adott elemekhez tartozó menü, melyet például a jobb egérgombbal kattintva hívhatunk elő.

hivatkozás – link: (1.) egy hálózati erőforrásra mutató elem, melyhez akár tevékenységeket is lehet rendelni (például böngészőben a hivatkozáson való kattintáshoz), (2.) a fájlrendszeren belül egy `ln` paranccsal létrehozott fájl, mely lehet közvetett (direct, hard) és közvetlen (symbolic, soft). A közvetlen hivatkozás valójában az adott fájl tartalmához létrehozott újabb fájlrendszer-bejegyzés, külön fájlként viselkedik, míg a közvetett hivatkozás csupán a másik fájlbejegyzésre mutat.

elérési pont – access point, AP: vezeték nélküli hálózatoknál azok az eszközök, melyek hozzáférést biztosítanak az adott hálózathoz.

hurokeszköz – loop device: egy olyan hálózati csatoló, mely csak elméleti szinten létezik, közvetlen visszacsatolás („hurok") a gépre.

imagemap → képtérkép

képtérkép – imagemap: elsősorban weboldalakon használt ábrák, melyekhez egy „térkép" tartozik, mely megadja, hogy a kép különböző területein kattintva milyen tevékenységet kell a rendszernek végeznie.

kezelő – handle: programozásban a megnyitott fájlokat a kódon belül gyakran egy változóban tárolt értékkel tartjuk nyilván, ez a fájl kezelője vagy kezelőszáma.

lease, leasing → bérlés

link → (1.) összeszerkesztés, → (2.) hivatkozás, → (3.) összekötés

loop → (1.) ciklus, → (2.) hurokeszköz

monospaced → rögzített szélességű

nullmásolás – zero copy: a forrás és a cél közötti közvetlen másolás módszere. Például egy fájl tartalmának egy eszközre küldése több egymás utáni másolást igényel, ezek áthidalására használható egyetlen művelet. Lásd bővebben az 54. oldalon található cikket.

összeszerkesztés – link: a programfordítás utolsó lépése, amikor a lefordított programrészeket egy egységes állománnyá állítjuk össze.

összekötés – link: eszközök, elemek közötti kapcsolat kialakítása.

proportional → arányos szélességű

rostcsatorna – fibre channel: egy átviteli eszköztípus, pontosabban egy átfogó protokoll- és átviteli felületmeghatározás, a rostcsatorna célja, hogy egységes és gyors felületet biztosítson különböző elemek (akár a gép alkatrészei, akár önálló gépek) között. Több értelemben is használják.

A ➲ http://hsi.web.cern.ch/HSI/fcs/
➲ http://www.iol.unh.edu/training/fc/fc_tutorial.html
➲ http://www.fibrechannel.org/
címeken bővebb tájékoztatás olvasható.

rögzített szélességű – monospaced: olyan betűkészlet, melynek mindegyik karaktere azonos szélességű (a karakterhez tartozó térközzel együtt). Lásd még: arányos szélességű.

tag → címke

zero copy → nullmásolás.

Beköszöntő

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

Azt mondják, tavasszal a természet felébred, ilyenkor kap mindenki új erőre. A tavasz ereje a linuxos közösségben is érződik, számos olyan változás történt és történik a napokban, melyek hosszú távon éreztetik majd a hatásukat. Ott van például az új FSF alapítvány „létrejötte". Azért használtam idézőjeleket, mert a csapat valójában hosszabb ideje létezik, de csak a napokban történt meg az alapítvány bejegyzése. Ha valaki esetleg nem tudná hova tenni a csapat tagjait, nézzen körbe például az OpenOffice.org fordítói hétvégén résztvevők között.

Ezzel szinte egy időben lezajlott az LME rendszeres éves választmányi gyűlése is. Illetve nem is biztos, hogy helyesen fogalmazok, amikor rendszeres éves gyűlésről beszélek, hiszen az egyik legfontosabb meghozott döntés éppen az volt, hogy ne egy, hanem három évre válasszák meg az elnökséget, valamint az ellenőrző bizottságot.

Ezzel a két történéssel kapcsolatban csak nem tudom megállni, hogy hangosan el ne gondolkozzam. Az elmúlt években láthattuk, hogy bizony nagyon komoly igénye van a linuxos közösségnek a képviseletre, az érdekvédelemre és a reklámra. Bár vannak kisebb csoportok, akik a saját „vonzáskörzetükben" tevékenykednek, de eleddig hiányoltam azt a szervezetet, amelyik elég erős és – mondjuk így – elég rámenős, hogy fel tudja venni az üzleti alapokon működő ellenfelekkel a harcot.

Ahogy azt már régebben is papírra vetettem, az LME-től vártam volna el, hogy az országos összefogás létrejöjjön, és az ország különböző pontjain lévő csoportok között erősítse kapcsolatot. Azt gondolom, hogy sajnos még mindig a turáni átok ül a linuxosokon is, és – bár ne legyen igazam – könnyen lehet, hogy a több szakmai szervezet nem segíteni fogja az előbb említett célt, hanem éppen ellenkezőleg. Csak abban bízom, hogy ez a „megerősödési időszak" minél gyorsabban, minél fájdalommentesebben zajlik le, és hamarosan tényleg erős képviselet lesz – számomra mindegy, hogy egy szervezet vagy tíz, a lényeg, hogy hatékonyan működjön.

Szóval tavasz és kikelet. Nálunk is több huncutság gyűlt össze erre a lapszámra. Nem csak arra gondolok, hogy két gép találta ki lapzárta közben, hogy ők nem hajlandóak tovább folytatni a robotmunkát, meg nem is arra, hogy a nyomdába adás napján derült ki több oldalról, hogy nincs meg, sokkal inkább például nyugtalan grafikusunkra, aki – meglátva a gimpes cikket, megviccelte egy kicsit olvasóinkat. Ha alaposabban megnézitek a cikkben található babát, hogy mit is olvas, látjátok majd, mire gondolok.

Új tervem is szép lassan körvonalazódik a nyílt forrású „vállalatirányítási rendszerről", a múlt havi számban megjelent cikkem kapcsán többen jelentkeztek, és remélem, hogy hamarosan egy komolyabb találkozót hozhatunk össze.

Az már sejthető, hogy egy olyan modulos felépítésű rendszert kell kialakítsunk, ami a felhasználó igényeitől függően a különböző rétegeket (például raktárkezelés, partnernyilvántartás, rendelések stb.) tetszőleges mélységben kezeli, de a szolgáltatásokat mindenhol megtartja. Komoly kihívás lehet például, hogy ahol nem foglalkoznak engedményekkel, ott ne kelljen a felhasználónak bajlódnia a felesleges adatok kitöltésével, mindezek ellenére a rendszer egységes maradjon. No, de erről majd a következő számban írok részletesebben. Addig is várom mindenki levelét, aki – akár vállalat nevében, akár szakemberként – szívesen részt vesz a fejlesztésben.

A márciusi számhoz is kellemes olvasgatást kívánok, remélem, mindenki talál benne kedvére valót!

Programvadászat

Lassan itt a tavasz – télen, a számítógép előtt ülve elgémberedett tagjainkat ideje kinyújtóztatnunk, a korongról frissíteni a Linuxunkat, szétnézni az Interneten frissítésekért és biztonsági javításokért. Javítsuk ki a hibákat, és menjünk ki a levegőre – magára hagyva az oly sokat kényeztetett vasszörnyeteget. Mint mindennek, a számítógépnek is van azonban lelke, még ha ezt néha tagadjuk is, úgyhogy egyelőre ne hagyjuk túl sokáig magára, nehogy megsértődjön. A további időtöltéshez pedig kiváló segítséget fog nyújtani az MPlayer és a Gimp.

MPlayer

Ez méltán népszerű, és nekünk, magyaroknak különösen szívmelengető, hatalmas tudású médialejátszó programnak a legfrissebb változata (0.90rc4) került fel korongunkra. Ez az utolsó RC kiadás, a következő csomag már 0.90 névre fog hallgatni.

Nézzük meg – a teljesség igénye nélkül –, hogy mi változott:

- -ac hwac3 kijavítva (az rc3-ban rossz volt)
- vo_svga: 4bpp- és 8bpp-javítás
- javítva a ./configure --cc="ccache gcc" hiba
- -loop-javítások, így a -loop 2 tényleg kétszer játszik
- mp3lib: layer-2-dekódolásjavítások
- libavcodec: DivX 5.03-dekódolás
- ao_oss: korlátozott csatornakezelés javítva
- vorbis dekódolás kijavítva
- -ao mpegpes + -ac hwac3 kijavítva
- -ao pcm bogus wav fejléc javítva
- -vo x11 + -wid javítva
- sig11 a második hangfájl lejátszásánál – javítva
- configure: cdda, nas, i18n, svgalib, faad2, lame felismerés javítva
- -af/-af-adv támogatás a mencoderben
- libmpdvdkit2: frissítve libdvdcss 1.2.5-re

Újdonságok:

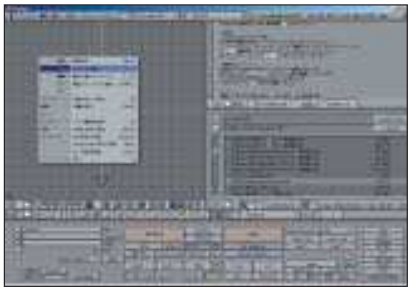
- nyersvideo-támogatás (-rawvideo, hasonlít a -rawaudio-ra)
- kísérleti mpeg4-es támogatás (bekapcsolása a -demuxer 27 -fps xxx kapcsolóval)
- dvd/vobsub-fejlesztések: pozícionálás, választható Gauss életlenítés (gaussian blur scaler)
- vf_bmovl: 400%-os sebességnövekedés
- libavcodec: natív DV audiodekóder
- GIF demuxer (animált GIF-ekhez)
- új zajeltávolító szr: -vop denoise3d
- gamma és MMX-hez javított fényerősség stb.
- támogatás a -vop eq2-ben
- a xvid és divx4/5linux könyvtárak egy idejű támogatása
- libavcodec: néhány B-frame-mel kapcsolatos enkódolási hiba javítva

Blender 3D

Miután a Blender 3D szerkesztőprogram jövője megnyugtatóan eldőlt, forrását a fejlesztői nyílttá tették. Ezentúl a népszerű 3D-modellező, animációs és leképezőprogramot készítő cég nonprofit alapítványaként (Blender Foundation) folytatja működését. Az alapítvány emellett egy e-boltot is üzemeltet, amelyben könyvek és egyéb kiegészítők kaphatók a Blenderhez.
A tulajdonosok a Blender teljes forráskódját a GNU GPL felhasználási szerződés alatt hozták a nyilvánosságra, és a weboldalon lehetőséget teremtenek a Blender köré egy, az egész világra kiterjedő fejlesztői közösség kialakulására. Ez természetesen nagymértékben elősegítheti a Blender további fejlődését. Már most is egy igen nagy tudású és megbízható modellező-, animációs és leképezőrendszerhez van szerencsénk. A forráskód nyílttá tételével rengeteg fejlesztőhöz eljutottak, így a Blender rajongótáborából kikerülő programozók belevethetik magukat kedvenc rendszerük fejlesztésébe. A japán változat – mint a képen láthatjuk – már el is készült.
A program a 45. CD Blender könyvtárában található, több operációs rendszerre előre fordított csomagban.

Gimp

A legújabb kiadása, vagyis az 1.2.3-es megbízható és a 1.3.11-es fejlesztői változat forráskódja és egyéb kiegészítők kaptak helyet a korongon a Gimp könyvtárban.



OpenOffice.org 1.0.2

Az egyik legjobban használható linuxos irodai csomag új változata is helyet kapott a CD-n – mind előre lefordított, azonnal telepíthető, mind forráskód formájában megtalálható korongunkon. A forráskódot azoknak ajánljuk, akik egy kicsit belülről is szeretnék megismerni ezt a nagyszerű programot.
A sok bosszantó hibát a készítők már kijavították.
A rendszermagfa változásait is közzé tesszük a fejlesztői rendszermag és a különféle foltok formájában. Természetesen a Magazin könyvtárban a kapcsolódó anyagok ugyancsak megtalálhatók.

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

NITROX, az új biztonsági lapka

Az ABIT legújabb kiszolgálókba szánt alaplapjára különleges biztonsági lapka került. A Cavium Networks NITROX Security Macro Processor nevű egysége 400 Mb/s sebességű IPSec-forgalmat képes támogatni, vagy 3500 RSA művelet elvégzésére képes másodpercenként. Az alaplapot elsősorban egy egység

magas, állványra szerelhető számítógépekbe szánják, amelyek például VPN-átjáróként vagy SSL-kapcsolatokat nagy számban fogadó webkiszolgálóként alkalmazhatók. Az alaplapon a különleges processzor mellett nem sok dolgot találunk: a memória és a Pentium 4 processzorok fogadására képes foglalatok mellett egy szál PCI-foglalat árválkodik, mellette két ethernetcsatoló, egy PS/2 aljzat, egy soros és egy párhuzamos kapu, illetve némi USB-szórvány teremt lehetőséget a külvilággal való kapcsolattartásra; képet a szintén az alaplapra épített VGA-vezérlőn keresztül kapunk. Az Abit a közeljövőben hasonló, de több processzor fogadására képes alaplapot is tervez megjelentetni. A nagyobb teljesítményű változatra Gigabit ethernetvezérlő kerül majd, illetve a nagyobb sebességű hálózati forgalom támogatásához is elegendő teljesítményt nyújtó NITROX lapka.

➲ http://www.abit.com.tw

Portocom-nehézségek

Egymilliárd forintos csempészés vádjával előzetes letartóztatásba helyezték a Portocom Rt. négy vezetőjét. A feljelentés szerint a cég egy olyan vállalkozástól vásárolt számítógép-alkatrészeket, ami értéken alul vámkezeltette azokat. A Portocom közleménye szerint a Vám- és Pénzügyőrség Országos Parancsnoksága már három éve vizsgálja a vállalkozás működését, de szabálytalanságot nem talált, ellenben jogosulatlanul igénybe vett vámkedvezmények gyanúja miatt az ügyészség elrendelte a négy érintett személy őrizetbe vételét. A Portocom szerint a feljelentés hátterében egy a cég jelentős piaci részesedésére áhítozó versenytársuk áll, amely így szeretné kiütni a nyeregből a hordozható gépek magyarországi piacának egyik meghatározó szereplőjét. A Portocom működését az ügy nem befolyásolja.

Bio-üzemanyagcellák

A Sharp és a RITE kutatóintézet közös bejelentése szerint a két szervezet kifejlesztette azokat az alapmegoldásokat, amelyek révén a jövőben bio-üzemanyagcellákat építhetnek. A cellák egy „hagyományos” üzemanyagcellából, illetve egy mikrobákat tartalmazó egységből fognak állni. Az utóbbi feladata az, hogy cseppfolyósított és finomított konyhai hulladékból származó glükózt feldolgozva hidrogént termeljen, amelyet az üzemanyagcellához vezetnek, így nem lesz szükség külön hidrogén-, metanol vagy egyéb gázforrásra, mint a már meglévő megoldásoknál. A biocellák kisméretűek lesznek, egy matchbox nagyságú készülék egy LCD TV-t, egy kétliteres egység pedig egy egész háztartást képes lesz táplálni.

A kutatás során a legnagyobb gondot az jelentette, hogy a korábbi hasonló megoldásoknál a mikrobák túlságosan kevés hidrogént termeltek. Módosításukkal sikerült elérni, hogy hosszabb ideig éljenek, és több százszor hatékonyabb hidrogéntermelést folytassanak.

➲ http://www.rite.or.jp

Első magyar PHP Konferencia

2003. március 29-én a PHP-levelezőlista közössége és az ELTE Radnóti Miklós Gyakorlóiskola megrendezi az Első Magyar PHP Konferenciát. A rendezvényen a részvétel ingyenes lesz, a szervezők egyaránt várják a számítástechnikai cégek döntéshozó vezetőit, a programozókat, az érdeklődő tanárokat és diákokat; illetve általában mindenkit, akit érdekel a webes fejlesztések témaköre. A számítógépes partik hangulatát idézi az 5k compo megrendezése, amelyre legfeljebb 5120 bájt méretű, tetszőleges célt szolgáló programmal lehet nevezni – a verseny győztesei webtárhely-szolgáltatást, folyóirat-előfizetéseket és könyvet nyernek. (Erről bővebben lásd a 14. oldalon beszélgetésünket a szervezőkkel.)

➲ http://www.phpconf.hu

Csődbe került a telnet?

Lapzárta előtt nem sokkal szinte robbant a hír: csődvédelembe menekült a telnet Magyarország Rt. A cég honlapján több mint féléves a legújabb sajtóközlemény, nem mondhatjuk el tehát, hogy buzgón cáfolnák a saját összeomlásukkal kapcsolatos híreket. A telnet által üzemeltetett weblapok a hírek szerint egy időre elérhetetlenné váltak, külföldi vonalait lekapcsolták, szolgáltatásai leálltak, ügyfélszolgálatán a telefont nem veszik fel – ha tehát esetleg kiderülne, hogy az internet magyarországi történetének egyik szinte kultikussá vált szereplője háza táján minden rendben van, hírnevének és ügyfélkörének semmiképpen nem tettek jót a történtek.

Új Lindows-termékek

A Lindows.com legújabb terméke újabb köntösbe bújtatva próbálja meg a felhasználók asztalára csempészni a Lindows operációs rendszert. A cég sajtóközleménye szerint a Lindows Media

Lindows Media Computer

Computer különféle szórakoztató elektronikai készülékek egész halmazát tüntetheti el tulajdonosa lakásából, aki egyetlen olcsó számítógép megvásárlásával kiválthatja az otthoni CD-, MP3- és DVD-lejátszót is. A mindössze 10 másodperc alatt elinduló gép képernyőjén a felhasználónak elegendő azt kiválasztania, hogy éppen mit szeretne nézni-hallgatni, majd a multimédiás billentyűzettel és az egérrel könnyedén eligazodhat a képernyőn megjelenő kezelőszervek között. Ezeket a lehetőségeket az Elegent nevű cég etDVD megoldása, a gép BIOS-ába ágyazott DVD-lejátszó program teszi elérhetővé. Ha a felhasználónak mégis egy normál számítógépre volna szüksége, egyetlen kattintással elindíthatja és megjelenítheti a LindowsOS szokványos felületét, és játszhat, internetezhet, szöveget szerkeszthet.

A viszonylag kis méretű, VIA C3 processzorral, DVD-meghajtóval és ethernetcsatolóval felszerelt gépet a tengerentúli vásárolók 350 dolláros áron vásárolhatják meg, amivel a – tulajdonképpen semmi különlegességre nem képes, mégis figyelemre méltó – Lindows Media Computer szinte az eldobható termékek árszintjére lépett le.

➲ http://www.elegent.com
info.lindows.com/lmc

Új Sony fájlkiszolgáló

A Sony új fájlkiszolgálója nagyjából akkora, mint egy nagyobb zsebrádió, mégis rögvest két feladatra is alkalmas: egyrészt egy 2,5"-os merevlemez révén fájlkiszolgálóként használható, másrészt

vezeték nélküli hálózati hozzáférési pontként szolgál. A fájlkiszolgáló fogalma hagyományosan valamilyen nagyméretű, szigorúan helyhez kötött gépet takar – itt azonban messze nem erről van szó. A készülék beépített akkumulátorával egy órán át tud üzemelni, ha pedig elektromos hálózat van a közelben, akkor a bölcsőjébe helyezve tetszőleges ideig rendelkezésre áll, illetve ilyen módon vezetékes hálózathoz is csatlakoztatható. Természetesen igazi hordozhatóságról a csekély akkumulátoros üzemidő miatt még nem beszélhetünk, de olyan helyzetekben, amikor rövidebb időre – például egy megbeszélés idejére – vezeték nélküli hálózatot kell létrehozni és fájlokat kell megosztani, a készülék hasznos segítőtárs lehet. Operációs rendszere Linux, egyszerre akár 250 ügyfelet is ki tud szolgálni, a fájlok továbbítását CIFS/SMB és NFSv3 protokoll felett végzi.

BEA Weblogic Itanium 2 alapon

A BEA WebLogic kiszolgáló 2002. február 12-től a Hewlett-Packard HP-UX 11i v 1.6 operációs rendszerrel dolgozó Itanium 2 alapú kiszolgálógépein is futtatható. Az Itanium szerkezetnek köszönhetően az ügyfelek és a rendszerintegrátorok a korábbinál gyorsabban végezhetik el a régebbi és a webalapú alkalmazások összevonását, és ily módon nagyobb hatékonyságot és jobb válaszsebességet érhetnek el.
A HP Itanium alapú kiszolgálóinak széles kínálatából választó nagyvállalatok, független programszállítók (ISV-k) és rendszerintegrátorok – jelenleg HP-UX, illetve néhány hónapon belül Linux és Windows operációs környezetben is – hatékonyan hozhatják létre és vonhatják össze üzleti alkalmazásaikat és webszolgáltatásaikat. A BEA és a HP közös munkája nyomán várhatóan tovább gyorsul az Itanium alapú megoldások elterjedésének üteme, és a két cég ügyfeleit még inkább hozzá tudja segíteni az Itanium szerkezetben rejlő előnyök: a kedvező ár-teljesítmény-arány, a nagy teljesítmény és kapacitás kiaknázásához.
➲ http://www.hp.hu

Fujitsu-Intel-együttműködés

A Fujitsu és az Intel bejelentették, hogy a jövőben közösen fejlesztenek csúcskategóriájú vállalati kiszolgálókat. Az új Linux vagy Windows operációs rendszert futtató gépek a Fujitsu nagygépes és Unix alapú kínálatát fogják kiegészíteni, így a cég minden területre megfelelő megoldást tud majd kínálni ügyfeleinek.
Az együttműködés eredményeként 2004 végére Intel Xeon alapú kiszolgálók jelennek meg a Fujitsu kínálatában, 2005 végére pedig akár 128 Intel Itanium processzort tartalmazó monstrumot is vásárolhatunk tőle.
Az együttműködés segítése, illetve októberben bejelentett linuxos tervei támogatására a Fujitsu egy új, kifejezetten linuxos rendszerekkel foglalkozó üzleti részleget is létrehozott. A több mint háromszáz mérnököt foglalkoztató részleg a Fujitsu nagy megbízhatóságú rendszerek építésében szerzett tudására építve biztonságos, nagy rendelkezésre állást és méretezhetőséget biztosító számítógépeket, valamint alapfeladatok elvégzésére alkalmas programokat fog fejleszteni.
➲ http://www.fujitsu.com
➲ http://www.intel.com

Xandros Desktop 1.0

A Xandros kiadta az ex-Corel Linux hamvain szárba szökkent Xandros Desktop 1.0 rendszerének Standard Edition névre keresztelt változatát, amely lényegében a mostani bejelentéssel Deluxe változattá előléptetett Xandros Desktop 1.0 kissé lebutított változata. A Standard Edition is tartalmazza a Xandros előnévvel ellátott összetevőket, mint az Installer, az Enhanced Desktop és a File Manager, irodai feladatokra az OpenOffice.org, böngészésre pedig a Mozilla áll használói rendelkezésére. A Deluxe változattal ellentétben viszont telepítője nem tudja átméretezni a már meglévő NTFS-lemezrészeket, hogy helyet szorítson az újabb operációs rendszernek, nem tartalmazza a CodeWeavers CrossOver Office és Plugin termékeket, nem jár hozzá nyomtatott kézikönyv, és a vásárlónak mindössze egyetlen alkalommal van joga – elektronikus levélben – támogatást kérni. A sok ***nem*** mellett némi vigaszt jelenthet, hogy így 5 centtel 40 dollár alá süllyedt a rendszer ára, ami tulajdonképpen barátinak is mondható.
➲ http://www.xandros.com/

Bajban a Mandrake

Január közepén röppent fel a hír, hogy a francia MandrakeSoft, a Mandrake Linux-terjesztések készítője csődvédelmet kért a helyi bíróságtól. Szerencsére szó sincs arról, hogy el kellene siratni a céget, egyszerűen „csak” annyi történt, hogy a MandrakeSoft nem tudja teljesíteni kötelezettségeit – vagyis nem tud fizetni a hitelezőinek. A cég működése nem állt le, tartozásait pedig megpróbálja átütemezni, újratárgyalni a hitelezőkkel. A megfelelő átszervezések és a pénzügyi egyeztetések után a MandrakeSoft jó eséllyel talpra állhat. Derűlátásra ad okot például, hogy a cég az utóbbi időben eredményesen csökkentette működési költségeit és növelte bevételeit; igaz, nyereséget ennek ellenére sem termelt.
A ➲ http://www.mandrake.com oldalra látogatva nyomát sem látni annak, hogy történt valami: február elején elérhetővé vált az áprilisra várható 9.1-es terjesztés harmadik próbaváltozata és a Mandrake Corporate Server 2.1 összeállítás is.

Működés közbeni rendszerfrissítések

Az alkatrész- és programfrissítések, illetve a szükséges felügyeleti feladatok elvégzése érdekében sok cégnél elkerülhetetlenek a tervezett rendszerleállások. A legutóbbi elemzések tanúsága szerint a tervezett állásidő a teljes rendszerleállásoknak mintegy 80 százalékát teszi ki. A Sun Microsystems bebizonyította, hogy mára a tervezett állásidő jelentős része kiküszöbölhetővé vált a Sun egyedülálló, működés közbeni rendszerfrissítési (Live System Upgrade) szolgáltatásainak köszönhetően; amelyeknek része a teljes eszközredundancia, a hatékony Solaris operációs rendszer, a kifinomult erőforrás-használati eljárások. A Sun Fire kiszolgálók – olyan Sun-szolgáltatásokkal ötvözve, mint például a Sun Remote Services (SRS) Net Connect szolgáltatás – a működés közbeni rendszerfrissítés alatt is páratlan rendelkezésre állást biztosítanak a Unix-rendszerekben, legkisebbre csökkentve a tervezett és az előre nem számolt állásidőt egyaránt. Mindez azt jelenti, hogy a tulajdonosa magasabb színvonalon szolgálhatja ki vásárlóit és partnereit, valamint magasabb fokú rendelkezésre állást biztosíthat felhasználóinak.
➲ http://www.sun.hu
➲ http://www.sun.com

Minősítés és biztonság

Az Oracle és a Red Hat közösen szeretnék megszerezni a Red Hat Linux Advanced Server számára a Common Criteria EAL2-es szintjének megfelelő minősítést. A bejelentés szorosan kapcsolódik az Oracle azon szándékához, hogy Linux operációs rendszeren futó Oracle9i Database Release 2 terméke számára formális biztonsági értékelést nyújtó EAL4-es minősítést szerezzen.
A két cég a minősítések megszerzésével a biztonságos, pontosabban valamilyen formális módszerrel, a biztonságot és minőséget szem előtt tartó folyamat során fejlesztett terméket kereső vállalkozásoknál és szervezeteknél juthat előnyhöz. A Red Hat és az Oracle a későbbiekben magasabb szintű minősítéseket is meg szeretne szerezni, ezzel közös megoldásuk igencsak zártnak mondható körbe lép majd be, hiszen fontosabb minősítéseket elég kevés operációs rendszernek sikerült szereznie. Törekvésük várhatóan az egész Linux-közösség számára fontos előnyökkel jár majd, hiszen egyrészt ráirányítja a figyelmet a biztonság kérdésére, másrészt az értékeléssel kapcsolatos anyagokat nyilvánosan is elérhetővé fogják tenni.

True24 Initiative

A VIA Technologies True24 Initiative néven jelentette be új kezdeményezését, amely a számítógépekbe épített hangeszközök – egyébként is lendületes – fejlődését hivatott segíteni. Jelenleg az asztali számítógépekben jellemzően 16-bites, 44 kHz-es mintavételezésre képes, kétcsatornás hangkártyák vannak, és túlnyomórészt az alaplapokra épített hangeszközök is ilyen jellemzőkkel bírnak. A DVD-filmek és zenei lemezek megjelenésével egyre nagyobb lett az igény a 24-bites, 96 kHz-es és hatcsatornás hangra, ezt tükrözi az egy ideje már megindult áttérés. A VIA jó érzékkel ismerte fel, hogy nem elég a számítógépek szokványos építőelemévé tenni a csúcsminőségű hangeszközöket, hanem szabványos megoldást kell létrehozni erre a célra, illetve a felhasználókkal is meg kell ismertetni a rendelkezésükre bocsátott eszközök tudását – a True24 Initiative ezt a három területet fogja össze egyetlen tervezetté.
➲ http://www.via.com.tw/en/multimedia/true24_a.jsp

Harc a himlő ellen

Az IBM, a United Devices és az Accelrys bejelentett egy új gyógyszerek kifejlesztésére irányuló, világméretű kutatástámogatási tervezetet, amely jó eséllyel veheti fel a harcot a himlővírus ellen.
A tervezet alapja egy hatalmas számítógépes „grid” lesz, amely világszerte több millió számítógép-tulajdonos számára teszi lehetővé, hogy gépének kihasználatlan erőforrásaival hozzájáruljon a himlő ellen esetleg bevethető gyógyszerek kifejlesztéséhez és vizsgálatához. A himlő ellen jelenleg nincs orvosság, a fertőzés egyedül védőoltással előzhető meg.
A United Devices hálózata korábban rákkutatási célokra jött létre, ám amikor az internetes közösség az eredetileg tervezettnél is jóval több adattal látta el a kutatókat, a hatalmas osztott hálózat kissé céltalanná vált, illetve irányítói az anthrax elleni oltóanyag keresésére kezdték használni.
A megújuló világméretű hálózatba a várakozások szerint több mint kétmillió számítógép csatlakozik majd, ezzel a világ legnagyobb számítógépének teljesítményénél harmincszor nagyobb, 1,1 teraflop számítási teljesítményt nyújtó osztott rendszer jön létre. Az érdeklődők egy képernyővédő letöltésével csatlakozhatnak a hálózathoz – furcsa és sajnálatos módon az ügyfélprogram csak Windows operációs rendszerek alá érhető el.
➲ http://www.grid.org

Ismét elítélték a Microsoftot

A bíróság arra kötelezte a Microsoftot, hogy legújabb termékeibe a továbbiakban ne a saját Java virtuális gépét (JVM) építse be, hanem a Sun hasonló célú megoldását. A Sun egymilliárd dollárra perli a Microsoftot, mivel az az eredeti, a Sun által fejlesztett Java-géppel nem teljesen egyező működésű Java-megvalósításával erősíteni kívánta a személyi számítógépek piacán megszerzett monopóliumát, illetve az együttműködési gondokkal a felhasználókat a saját .NET keretrendszere felé terelte. Az ítélet nyomán a Microsoft összeállította Windows XP operációs rendszereihez kiadott SP1 javítócsomagjának SP1a jelzésű változatát, amelyből nemes egyszerűséggel elmaradt a Microsoft JVM, illetve hamarosan elkészül az SP1b jelzésű változat is, amelybe már a Sun legújabb – külön, a Sun oldaláról egyébként is beszerezhető – Java virtuális gépe kerül.

Cégvilág

Az IBM felhagy az Itaniumot támogató linuxos fejlesztésekkel

Forrás: hwsw.hu

Az IBM felhagy az Intel Itaniumot támogató linuxos fejlesztéseivel, és a jövőben elsősorban arra összpontosít, hogy a nyílt forrású operációs rendszert saját Power processzoraira igazítsa. Elemzők szerint a lépés az Intel és az IBM közötti kiéleződő verseny előjele.

Ron Favali, az IBM szóvivője az IDG-nek elmondta, hogy a vállalat azért döntött a Linux itaniumos fejlesztésének leállítása mellett, mert az Intel 64-bites processzora iránt igen visszafogott a kereslet. „Jelenleg úgy gondoljuk, hogy az Itanium egy tudományos projekthez hasonlít: nincs piaca" – tette hozzá. A cégnek azok a fejlesztői, akik korábban az Itanium processzorokon futó Linux finomhangolásán dolgoztak, immár az IBM Powert támogató linuxos projekteket segítik.

Az Intel természetesen nem ért egyet az IBM-mel. Mint a processzorgyártó szóvivője elmondta, az Itanium iránti kereslet néhány vertikális piacon meghaladta a várakozásokat, noha számadatokat nem említett. Közlése szerint az Intel és a HP továbbra is teljes erőbedobással dolgozik a Linux Itanium-támogatásának továbbfejlesztésén.

Linuxra épülő mobiltelefonok a Motorolától?

Forrás: hwsw.hu

A Motorola még az idén piacra dobja első linuxos mobiltelefonját, amelyet a jövőben több hasonló készülék követ majd. Az amerikai mobiltelefon-gyártó a harmadik negyedévben mutatja be A760 készülékét, ami színes kijelzőt, digitális kamerát, videolejátszót, Java alkalmazásfuttatási környezetet és Linux operációs rendszert tartalmaz. Az A760 először Ázsiában lesz kapható, az Egyesült Államokba és Európába csak később érkezik meg.

„A Motorola A760 csúcskategóriás készülék, de a vállalat feltett célja, hogy alsóbb árkategóriás telefonjait is Linux operációs rendszerre építse" – mondta *Scott Durschlag*, a Motorola mobiltelefonokért felelős részlegének üzletfejlesztési igazgatója. Durschlag szerint a Linux azért jó választás, mert csökkenti a fejlesztési költségeket és a fejlesztésre fordítandó időt, ugyanis az operációs rendszer már számos olyan elemet tartalmaz, amelyet egyébként a telefon gyártójának kellene megírnia. A Motorola nem fejleszt saját Linux-változatot, hanem a kifejezetten beágyazott eszközökbe szánt programokkal foglalkozó MontaVista Software termékét használja majd.

Megkétszereződött a linuxos kiszolgálók forgalma az USA-ban

Forrás: hwsw.hu

A Gartner Dataquest piackutató vállalat jelentése szerint az Egyesült Államokban tavaly a negyedik negyedévben közel megkétszereződött a linuxos kiszolgálók forgalma, miközben a Unix-kiszolgálók piaca három százalékkal visszaesett.

A Gartner adatai szerint a negyedik negyedévben az USA-ban 384,6 millió dollár értékben értékesítettek linuxos kiszolgálókat, ami 90 százalékos növekedés a megelőző év hasonló időszakának 202,2 millió dolláros forgalmához képest. Ezzel szemben a régió összes kiszolgálóértékesítései mindössze öt százalékkal növekedtek. A legnagyobb növekedést a piacon az IBM érte el: a linuxos kiszolgálók eladásából származó forgalma 159,9 millió dollárra nőtt a megelőző év megfelelő időszakának 75,6 millió dollárjáról. A második helyen a HP áll 80,2 millió dolláros eladással (ami éves szinten 81 százalékos növekedésnek felel meg), míg a harmadik helyet a Dell szerezte meg 77,1 millió dolláros forgalommal (ami 66 százalékos éves növekedést jelent). A piacra csupán tavaly belépő Sun 1,3 millió dollár értékben értékesített linuxos kiszolgálókat a vizsgált időszakban.

Kiszorít az XP – újabb eljárás a Microsoft ellen

Forrás: terminal.hu

Az Európai Bizottsághoz újabb eljárási kezdeményezés érkezett be a Microsoft ellen. A Computer and Communications Industry (CCIA) szövetségébe tömörülő Sun Microsystems, AOL Time Warner, Oracle és Nokia keresetet nyújtott be a brüsszeli Európai Bizottsághoz a Microsoft ellen annak piaci fölényével való visszaélése miatt. A vizsgálat megindítására vonatkozó kérelem január 31-én érkezett be, melyben a fent említett cégek azzal vádolják a redmondi székhelyű óriást, hogy a Windows XP operációs rendszerébe beépített szolgáltatások miatt a versenytársak kiszorulnak az adott területekről. Az EU versenypolitikai ügyekért felelős szerve alapos vizsgálat alá kívánja vetni a panasztételt, ugyanis a brüsszeli Bizottság már azt is szabályellenesnek tartotta, hogy a Microsoft Windows 2000 operációs rendszerébe beépítette a Media Playert.

A közbeszerzési pályázat nyertesei

Az Axelero Internet és a GTS Datanet győzelmével zárult le az MKGI által kapcsolt távbeszélő-hálózati internetszolgáltatásra kiírt közbeszerzési eljárás.

Az Axelero Internet a Miniszterelnökség Közbeszerzési és Gazdasági Igazgatósága által kiírt tárgyalásos közbeszerzési eljárás nyomán keretszerződést kötött 1010 költségvetési intézmény számára nyújtható korlátlan, alapszintű kapcsolt vonali internetszolgáltatás értékesítésére.

A keretszerződés megkötését követően az Axelerótól a költségvetési intézmények közvetlenül rendelhetnek meg, összesen 16 ezer telefonos internethozzáférést, amelyek havidíja egyenként bruttó 1499 forint. Az érintett költségvetési intézmények, így többek között az IHM (Informatikai és Hírközlési Minisztérium) is jogosultságot szerzett arra, hogy akár belső használatra, akár más programjaihoz e keret terhére vásároljon internetszolgáltatást.

Kelényi Attila (attila@kiskapu.hu)

LME-események

A Linux-felhasználók Magyarországi Egyesületében februárban egymást érték az események.

Az LME évzáró vacsoráját február 14-én este Budán, a Mongolian Barbecue étteremben tartotta. Az egyesület rendezvényére 61, a Linux és a szabad programok terjesztése érdekében buzgón tevékenykedő Linux-hívőt hívtak meg. A rendkívül jó hangulatú vacsorán sokan most találkoztak először egymással, hiszen a számítástechnika berkeiben néha évek telnek el, mire személyesen is megismerkedhetünk egy-egy jól ismert becenév tulajdonosával. Az est fénypontjaként értékes szakmai könyveket vehettek át az elmúlt évben a nyílt forráskódú programok népszerűsítésében kiemelkedően segédkezők: *Agrai Ferenc, Balázs Tibor, Halász Gábor, Kósa Attila, Kósa Lajos, Lajber Zoltán, Nagy Attila, Pásztor György, Pozsár Balázs, Sári Gábor, Takács István, Vasas Csaba, Zelena Endre* és jómagam.

Küldöttközgyűlés

Az LME február 15-én délelőtt tartotta éves rendes tisztújító közgyűlését a budapesti MÁV Bevétel Ellenőrzési Igazgatóság (BEIG) épületében. A pontosan 10 órakor megkezdődött küldöttválasztó gyűlésen a megjelentek kis létszáma miatt 10:15-kor megismételték a szavazást, majd körülbelül 10:30-kor megkezdődött küldöttközgyűlés, az elnök (*Sári Gábor*) beszámolójával. Ezt követte az Ellenőrző Bizottság elnökének (*Laky Norbert*) beszámolója, majd az egyesület titkára, (*Balázs Tibor*) megtartotta pénzügyi beszámolóját. Rövid szünet után a közhasznúsági beszámoló következett, ebből idéznénk néhány érdekesebb részt.

Konferencia

A IV. GNU/Linux Szakmai Konferencia az Infoszféra Kft.-vel közösen került megrendezésre a Budapesti Best Western Grand Hotel Hungaria szállóban. A színvonalas rendezésnek köszönhetően a konferencia nagyon sikeres volt. A résztvevők rendkívül hasznosnak tartották az előadásokat, ahol új gondolatokat, megoldásokat és szakembereket ismerhettek meg. A konferencia fő támogatója az UHU-Linux Kft. volt. Idén az előadókkal karöltve *Zelena Endre* az LME történetének legszínvonalasabb konferenciakiadványát készítette el.

Positive E-Privacy díj

Az Egyesület eddigi tevékenysége elismeréseként tavaly megkapta a nemzetközileg tekintélyesnek számító „Positive E-Privacy" díjat.

Linuxos konferenciasorozat a MeH támogatásával

A Miniszterelnöki Hivatal támogatásával létrejött sikeres Linux Konferenciasorozat tavaly áprilisban ért véget. Az egyesület szakemberei hat egésznapos rendezvényen próbálták megismertetni az résztvevőket a Linux és a Szabad Szoftverek világával.

CD-írás

Ez volt tavaly a legsikeresebben működő projekt, ugyanis *Balázs Tibor* közel hatszáz korongot írt meg. *Tóth Csaba* és *Szakszon Mihály* közreműködésével szinte minden szakmai rendezvényen megjelentek, legutóbb a „Linux a közoktatásban" címűn. Ők ketten egy alkalommal a West End City Center Média Markt üzletében is tartottak Linux-bemutatót.

Sajtó

Varga Csaba Sándor és jómagam gondoskodtunk a Linux népszerűsítéséről. Szervezésünkben és közreműködésünkkel a Fix.tv-ben beindult egy heti rendszerességgel sugárzott Linux témájú műsor. A Számítástechnika című folyóiratban *Magosányi Árpád* cikkét olvashatták, a Budapesti Nap című napilapban Varga Csaba Sándor indított cikksorozatot. Több rádióműsorban is felbukkantak az LME képviselői. Legutóbb a „Pénz Piac Profit" című műsorban, a Kossuth rádióban, de szerepeltünk a Szabad Világ Számítástechnikai Magazinban, a Civil Rádióban, ahol Varga Csaba Sándor, *Czakó Krisztián* és jómagam a szabad programokról beszéltünk.

Szabadon projekt

A Szabadon kampány első része 2002. november 1-jétől 2003. január 31-ig tartott. Kitűzött céljának megfelelően felhívta a figyelmet a szabad programok létére és jelentőségére. A kampány folytatásában együttműködünk a hasonló célú civil szervezettekkel (Magyar BSD Egyesület, FSF.hu Alapítvány, FSN Alapítvány).

Az ftp.fsn.hu háttértár támogatása

Az egyesület a szabad programok legnagyobb európai tárházaként ismert ftp://ftp.fsn.hu/ szolgáltatást kilenc 120 GB-os merevlemez beszerzésével támogatta.

Szabad programok a kormányzati szférában

Az Egyesület nyílt levelet intézett *Kovács Kálmán* informatikai és hírközlési miniszterhez a szabad szoftverek kormányzati szinten történő felhasználása érdekében.

Szerencsi Linux-tábor

Czakó Krisztián szervezésében tavaly ismét nagy sikerű Linux-oktató tábor került megrendezésre. Az LME több tevékeny tagja és külső szakértők oktatóként vettek részt a táborban.

LME-private levelezőlista

A levelezőlista létrehozása a most leköszönő elnökség első döntéseinek egyike volt. Az info@lme.linux.hu és az elnokseg@lme.linux.hu címek az lme-private@lme.linux.hu listára lettek átirányítva. A lista sikeresnek bizonyult, mert meggyorsította és hatékonyabbá tette az elnökség és a tagok kapcsolatát, továbbá az egyesület tevékenysége is átláthatóbb lett.

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

Linux az oktatásban

Január végén a Bercsényi Miklós Élelmiszeripari SZKI adott otthont a „Linux az oktatásban" című rendezvénysorozat első előadássorozatának.

A történet meseszerű. Egy budapesti iskola lelkes számítástechnika tanárai megelégelték az egyik alapítvány csúf, bilincses kampányának komor fenyegetését, az örökös felhasználásiszerződés-díjak körüli bonyodalmakat, a vírusok elleni végeláthatatlan harcot, a megbízhatatlan programműködést. Bátran szembenéztek az iskola vezetésével, és kimondták a bűvös szót: „Linux!" Az első döbbenetet azonnali józan és meggyőző érvek fegyverével győzték le, mire az igazgatóság az egyetlen keresztülvihető választ adta: „Legyen!" Kezdő lépésként a szakma hozzáértő mesterei új életet leheltek kiszolgálójukba, amelyen a számukra leginkább kézenfekvő Mandrake Linux tagadhatatlan előnyeit aknázták ki. Szívük ifjonti heve mint elsöprő áradat terelte őket a munkaállomások felé is, hiszen számukra nem a programok betanítása az igazi kihívás, sokkal inkább a számítástechnika oktatása. Ennek az évnek az elejére sikeresen megtisztították a gépparkjukat, és kidobtak minden ablakos programot az ablakon. Szinte adta magát a kérdés: ha ez ilyen egyszerű, ennyire jó, megbízható és az oktatás kitűzött céljainak tökéletesen megfelel, vajon miért nem ezt használja mindenki? A válaszon sem kellett sokáig töprengeni: „Nem ismerik!" *Rózsár Gábor*, *Sallai András* (Zalaegerszeg), *Wolczán György*, azaz a három tanár új cél tűzött ki maga elé. Belefogtak a „Linux az oktatásban" szakmai konferencia szervezésébe.

Gibizer Tibor (Gibzo): *Egy élelmiszeripari iskolában mennyire fontos a számítástechnika oktatása?*

Rózsár Gábor: Minden középiskolásnak meg kell tanulnia a számítástechnikát, a gazdasági informatikusoknak pedig különösen.

Gibzo: *Miért Mandrake?*

R. G.: Számos Linux-változatot megnéztünk, tényszerűen elég nehéz lenne megmondani, hogy miért pont a Mandrake nyerte meg a tetszésünket. Egyszerűen rokonszenves volt. Most már elég sok rejtett apróságát kiismertük, így nem áll szándékunkban váltani. E terjesztésben rögtön tudjuk, hogy mikor hova kell nyúlni, ha valami galiba adódna.

Gibzo: *Honnan jött az ötlet, hogy egy ilyen konferenciát szervezzetek?*

R. G.: Már a rendszer építésénél sokat beszélgettünk arról, hogy milyen jó lenne, ha a többi iskola is megismerné a Linuxban rejlő lehetőségeket. December közepén fogtunk hozzá a szervezéshez, tehát csupán másfél hónap kellett a megvalósításához.

Gibzo: *Akkor ez egy teljesen önállóan szerveződött rendezvény, nem áll mögötte semmilyen cég?*

R. G.: Nem, az iskola teljesen önállóan vállalta fel ezt a szerepet. Közben természetesen cégek is bekapcsolódtak támogatóként. Ezúton is köszönjük a Kiskapu Kft.-nek, a Linuxvilágnak, a Linux-felhasználók Magyarországi Egyesületének, a SuSE hazai képviseletének és természetesen az UHU-Linux Kft.-nek, hogy megjelenésükkel emelték a rendezvény fényét. Több linuxos fórumon meghirdettük az eseményt, illetve azt, hogy előadókat keresünk. Legnagyobb meglepetésünkre többen jelentkeztek, mint ahány előadást meg tudtunk tartani.

Gibzo: *Mekkora volt az érdeklődés? Tudjátok, hogy hány iskola, mennyi tanár vett részt a konferencián?*

R. G.: A rendelkezésünkre álló termek mérete szabta meg a részvevők számát, így hatvan érdeklődőt tudtunk csak fogadni, ami sajnos azt jelentette, hogy a későn érkező jelentkezéseket már nem tudtuk elfogadni. Egy iskolából átlagosan két kolléga jött el, ez nagyjából azt jelenti, hogy több mint harminc oktatási intézményből érkeztek érdeklődők. Többségük középiskolákból, általános iskolákból jött, de a nyíregyházi tanárképző főiskolából is érkeztek hozzánk vendégek. Érdekességként megemlíteném, hogy a jelentkezők nagyobb része hölgy volt, és talán ennek is köszönhető, hogy rendkívül jó hangulatúra sikeredett ez a két nap.

Gibzo: *Terveztek-e folytatást?*

R. G.: Természetesen. Szeretnénk évente 2–3 ilyen rendezvénnyel megörvendeztetni a kollégákat. Terveinkben az is szerepel, hogy a jövőben nemcsak a tájékoztatás, eddigi tapasztalataink átadása szerepeljen, de a hétköznapi életben is jól használható ismeretanyagokkal is segíteni szeretnénk a munkájukat. Tehát mindenképpen szeretnénk oktatni is, ezért próbálunk fokozatokat meghatározni, hogy szintekre tudjuk bontani az előadásokat, illetve a gyakorlatokat. Most egy teljesen kezdőknek szóló anyaggal jelentkeztünk.

Gibzo: *Talán még korai megkérdezni, de véleményed szerint mennyire volt eredményes az elmúlt két nap?*

R. G.: Kicsit visszamennék az időben: amikor a diákokat odaültettük a Linux elé, meglehetősen idegenkedtek tőle, de az első óra végére megérezték, hogy a számítástechnika nem a Windowsnál kezdődik, és főleg nem ott ér véget. Hihetetlenül gyorsan eljutottunk odáig, hogy zökkenőmentesen dolgoztak az új környezetben.

A kollégáknál ugyanezt tapasztaltuk. Az első bizonytalan egérkattintások után csillogó szemekkel fedezték fel a lehetőségek szinte végtelen tárházát. Ehhez talán az is kellett, hogy azok, akik ide eljöttek, már nyitottak voltak az új felé. Ők már kezdik érezni, hogy a Linuxé a jövő.

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

OpenOffice.org maratoni súgófordító hétvége

A mai kiélezett helyzetben a GNU/Linux legkomolyabb segítsége abban, hogy az irodákba is betörjön, az OpenOffice.org irodai programcsomag.

Bár vannak más szövegszerkesztő, táblázatkezelő és egyéb feladatok ellátására alkalmas „irodai" programok, de vagy fizetős termékekről van szó, vagy messze nincsenek olyan szinten, hogy felvegyék a versenyt például a Microsoft Office csomaggal. Az OpenOffice.org egy lelkes csapatnak köszönhetően már magyarul szól a felhasználókhoz, sőt helyesírás-ellenőrző résszel is büszkélkedhet. Ami eddig hiányzott, az a magyar nyelvű súgó. Már közel egy hosszú éve várat magára az angol súgó lefordításának nehéz kérdése. Tovább nehezítette a helyzetet, hogy több érdekcsoport különböző célokat szem előtt tartva nem egy irányban „húzta a szekeret".
A civódást most is, mint ahogy tavaly, *Somogyi Péter* oldotta meg, aki magára vállalta egy következő fordítóhétvége megszervezését. Szerencsére a mostani hétvége szervezésében ismét sokan segítettek. De miért is külön kihívás a súgó lefordítása, és meddig jutott a csapat? Hogy ezt jobban megértsük, tudnunk kell, milyen állapotban volt a fordítás. A programot már lefordították. Elméletileg. Azért írom, hogy elméletileg, mert nem történt meg a program fordításának komoly és egységes lektorálása, az eredmény pedig – valljuk be – sokszor kusza, következetlen kezelőfelület. Ráadásul egy olyan programról van szó, amelynek a tudása rendkívül szerteágazó. (Már-már szállóigévé vált a fordítók között az alábbi mondat: „Nahát, fordítás közben megtanulom használni a programot!".)
De nem itt volt az egyetlen zavar. A súgó, ami alapján dolgoztunk (amit a program mellett megtalálhatunk), valójában egy korábbi termékhez készített német nyelvű súgó fordítása volt, ebből kifolyólag egyrészt gyakran értelmetlen mondatokkal találtuk szembe magunkat, másrészt olyan leírásokkal, amik már rég nem voltak pontosak. Szerencsére az egyik komoly diót, a súgó xml formátumának kezelését *Noll Jánosék* programja levette a fordítók válláról, cserébe viszont néha komoly fejtörést okozott, ha valamelyik mondat szerkezetét kívántuk átalakítani.
No, ezeken az akadályokon mind átverekedtük magunkat, majd elkezdtük a fordítást, az összegyűltek a Rózsa Művelődési házban (a nagybetűs Helyszínen), a többiek pedig az ország számos pontjáról, weben keresztül. (Pontosabban kezdték, mert én, több sorstársammal egyetemben, órákon át a gépem felélesztésével bajlódtam, de ez egy másik történet…)
Szerencsére a rendkívül nagy falatnak tűnő munkát a program kis „rekordokra" osztotta, így a fordítás menetét folyamatosan követhettük. Sőt volt, aki kifejezetten a minél több rekord lefordításának délibábját követve döbbenetes sebességgel gyártotta a magyar mondatrészeket. Igaz, akadt, aki csak a harmadik napon vallotta be, hogy valójában nincs is angol nyelvű súgója, mi több: az OpenOffice.org sincs nála telepítve. Ráadásul volt, aki nem is Linux vagy Windows alatt dolgozott (ugyanis az OpenOffice.org Windows alatt is fut), hanem nagy büszkén egy Macintosh világító almácskája mögé búvva kalapálta a billentyűket.
A csapat hangulata jelentősen javult, amikor a pénteken hozott ellátmány részét képező mélyfagyasztott diókrémes palacsinták keménysége szombat délelőttre emberi fogyasztáshoz megfelelő szintűre csökkent.

És végül elkészült a magyar súgó is

Egy terület viszont komoly vitákat váltott ki, ez pedig a szóhasználat. Aki rendszeres olvasója magazinunknak, pontosan tudja, hogy én elsősorban a „magyar magyarításokat" részesítem előnyben, így gyakran előfordult, hogy egy-egy szónál igyekeztem a nem létező fordítói szótárat módosítani (mint például a szerintem szörnyű kombipanel vagy a tezaurusz esetén). Sőt ami a szabad fejlesztésekre jellemző, itt is kialakult: ahány ember, annyi vélemény. A jobb gombra előugró helyi menüt például több csodanévre is elkeresztelték. Ha valakit érdekelnek ezek a szavak (a teljesség igénye nélkül), látogasson el a szotar.kiskapu.hu címre, és keressen rá az „(oo)" szövegrészre.
Nagy előny volt viszont, hogy Péter a program teljes forrásával jelen volt, így a fordítás közben a programban talált következetlenségeket azonnal ki tudtuk javítani. Amíg az ember nem kerül közvetlenül szembe a program felhasználói felületével, eszébe sem jut, hogy milyen rendkívül nehéz egy párbeszédablakot úgy kitalálni, hogy a gyorsbillentyűk a helyükön legyenek, a rövid szövegek találóak legyenek, és a felhasználó gyorsan kiismerje magát.
Végül is a munka döntő többsége elkészült, és azzal együtt, hogy a fordítás nem teljes, hiszen csupán a közös alapmodult és a szövegszerkesztőhöz tartozó részt fordítottuk le, a súgó így is eleget nyújt ahhoz, hogy egy lelkes titkárnő megismerkedhessen az OpenOffice.org szövegszerkesztőjének alapjaival.

Szy György (Szy.Gyorgy@linuxvilag.hu)
A Linuxvilág főszerkesztője,
a Kiskapu Kiadó vezetője.
Mindenki levelét örömmel várja.

Első Magyar PHP Konferencia – beszélgetés a szervezőkkel

A PHP-nak az utóbbi években bemutatott diadalmenetét sokféleképpen lehet mérni a kiszolgálókimutatásokon túl. Elég, ha megnézzük, hogy világszerte hány témába vágó, vagy egyenesen PHP-ra szakosodott helyi vagy nemzetközi konferenciát rendeznek havonta. Elnézve a PHP hivatalos honlapjának rendezvénynaptárát és híreit, 2003 tavasza a PHP-konferenciák évszaka lesz. Idén először hazánkban is lehetőség nyílik a szakma eszmecseréjére a magyar PHP-levelezőlista közössége által szervezett Első Magyar PHP Konferencián.

Heilig Szabolcs (Cece): *Gábor, ugye nem ez az első PHP Konferenciád?*

Hojtsy Gábor (Goba): Valóban, már két nemzetközi konferencián is részt vettem Frankfurtban.

H. Sz.: *A legutóbbin nemcsak látogatóként szerepeltél, de előadást is tartottál. Mégis, hogy lesz valakiből egy nemzetközi konferencián előadó?*

H. G.: 2001-ben sokat forgolódtam az előadók körül. Az első estén még a nekik szánt különvacsorára is benéztem, ezért mindenki azt gondolta, hogy én is előadással készültem. Rendszeresen meglepődtek, amikor kiderült, hogy hallgatóként jöttem el. A PHP fejlesztői közül jó párat ismerek levelezés révén, különösen a leíráson dolgozókat. Ezért annak ellenére, hogy magyar résztvevő rajtam kívül csak egy akadt (*Rábai Gyula* előadó), sikerült néhány ismerőssel találkozni. Ők is azt javasolták, hogy a következő évben én is készüljek egy előadással.

H. Sz.: *Megfogadtad a tanácsukat?*

H. G.: 2002-ben a konferenciát egy nappal meghosszabbították, így több idő jutott a szakmai programokra. Ezzel együtt az előadáshelyekre nagy volt a túljelentkezés. A terveket már nyár elején le kellett adni, így bőven jutott idő a témák kiválasztására és az átgondolásukra. Mivel én a dokumentációs csoportban is dolgozom, kézenfekvő volt, hogy a PHP-parancsfájlok leírásáról beszéljek. Két előadással kellett jelentkezni, így másodikként a számomra akkor még izgalmas újdonságként jelentkező „rich client" témát jelöltem meg. Szerencsére mindkét előadásomat elfogadták. Ezután egy komoly felkészülési időszak következett. A forrásanyagok, és háttéranyagok felkutatása mellett angol nyelvű tévéadásokat néztem, és angol nyelvű újságokat olvastam még a tömegközlekedési eszközökön is. Sőt egy főpróbát is rendeztem még itthon, egy lelkes kis csoporton próbáltam ki idegen nyelvű előadói készségemet.

H. Sz.: *Korábban is volt szó hazai PHP-konferenciáról a magyar PHP-levelezőlistán. Az akkori véleménynyilvánítások szerint még nem tűnt életképes vállalkozásnak egy ilyen hazai rendezvény megtartása…*

H. G.: Számomra is túl nagy falatnak tűnt ennek a megszervezése, és nem is voltam biztos benne, hogy egy olyan jó kis csapat fog kialakulni, mint amilyen végül is a magyar konferencia szervezésére összejött. *Pálócz István* már korábban is ajánlott helyszínt a PHP-levelezőlistán, de akkor én ezt az elképzelést nem vettem komolyan.

H. Sz.: *István, te itt lépsz be a képbe?*

Pálócz István: (más néven ***PI***): Majdnem. Idén januárban gondoltam egy nagyot, és újra elővettem az ötletet. Kijelöltem egy időpontot, amikor márpedig PHP-konferencia lesz Magyarországon. Összeütöttem egy oldalt, ahol jelentkezni is lehetett. A „program" ekkor még csak előadás-időpontokat tartalmazott. Megírtam az oldal címét a PHP-levelezőlistára, és kíváncsian vártam a fogadtatást. Igencsak meglepődtem, mert még aznap jelentkezett tíz ember, köztük volt az első előadó is, *Bártházi András*. Nem sokra rá Hojtsy Gábortól is kaptam egy levelet, amiben előadónak ajánlkozott. Öt előadást terveztem, és mivel egyet magam is beválaltam, már csak két előadó hiányzott.

H. G.: Közben néhányan elkezdtünk tippeket adni, mit lehetne javítani a programkiíráson. Majd a honlappal kapcsolatos kifogások megvitatására került sor.

P. I.: Amin közben folyamatosan jelentkeztek a résztvevők. Egyre többen rágták a fülem, hogy illene valami szebb honlapot készíteni.

H. Sz.: *A honlapnak új arca lett. Ezt követte rövidesen a harmadik változat is?*

P. I.: Ez már profi grafikus keze munkáját dicséri, és már végleges. Bejegyzésre került a phpconf.hu tartomány, itt lelhető fel a hivatalos honlap. Összeszerkesztésére egy hihetetlen teherbírású honlaphegesztőt is találtunk *Simon Benjámin* személyében.

H. G.: Millió más dolog is történt, nem csak a honlap külalakján és a logón vitáztunk. Első támogatónk, a helyszínt biztosító ELTE Radnóti Miklós Gyakorlóiskola mellé más támogatók megnyerésén törtük a fejünket. Időközben a teljes előadói kör is kialakult.

P. I.: Hála *Papp Győző*-nek és *Szabó Dénes*-nek.

H. Sz.: *A helyszín kiválasztása nehéz volt?*

P. I.: Egyáltalán nem. Több minden is az iskola mellett szólt. Egyrészt kitűnően felszerelt konferenciateremmel rendelkezik, ahol mind a hangosítás, mind a számítógépek monitorának kivetítése ingyen megoldható. A legfontosabb érv talán az, hogy az iskola alapítványának segítségével képesek vagyunk anyagi támogatást is fogadni.

H. Sz.: *István, továbbra is te maradtál a főszervező?*

P. I.: Amolyan koordinátorféleség. Amikor elkezdtem, álmomban sem gondoltam volna, hogy ilyen hatalmas segítséget kapok. Több listás összejövetel után az a kép alakult ki a többiekről, hogy nagyon sok mindent csinál-

nának szívesen, csak nem mernek belekezdeni.

H. Sz.: *De eleinte mégiscsak teljesen egyedül voltál a terveiddel?*

P. I.: A tervekkel igen, de a döntésemet nagymértékben segítette az is, hogy a hátam mögött tudhatok olyan barátokat (volt tanítványokat), akikkel egy ilyen rendezvény könnyedén lebonyolítható. Szerencsére iskolánk igazgatója is nagy örömmel fogadta kezdeményezésemet. Arra számítottam, hogy a rendezvény teljes megszervezése az én feladatom lesz. Azonban még egy hét sem telt el, és azon kaptam magam, hogy egy ütőképes csapat tagja vagyok.

H. Sz.: *Mekkora munka ez? Mennyi időtöket viszi el?*

P. I.: A koordinációs munka javarésze abból áll, hogy a különböző felvetett ötleteket ne hagyjam elsikkadni, és találjak valakit, aki felvállalja és véghez viszi azt. Mostanság az időm nagy részét valóban a rendezvényszervezés teszi ki. Általában éjfél körül olvasom az utolsó leveleimet, ennek ellenére hajnalban hat körül már a szervezői levelezőlistát bújom. A többiek reggel nyolc táján kezdenek bekapcsolódni, és hajnali három–négy óráig vérre menő vitákat folytatnak. Mikor és mennyit alszanak, arról fogalmam sincs, de azt hiszem ezt jobb, ha nem is tudom.

H. Sz.: *Most hol tartotok?*

P. I.: Eleinte abban reménykedtem, hogy sikerül egy ötvenfős rendezvényt összehoznom. Ehhez képest ma már kétszáz feletti a jelentkezők száma.

H. G.: Külföldi támogatónk is akadt már, két angol nyelvű PHP-magazin is védőszárnyai alá vonta kezdeményezésünket. Bővül egyéni támogatóink köre is. Úgy tűnik, többen gondolják azt, hogy érdemes ezt a rendezvényt megszervezni, mint amennyire kezdetben számítottunk.

H. Sz.: *Gonosz leszek. Rengeteg olyan kezdeményezésről tudok, amelyek nagy lendülettel indultak, majd a lelkesedés múltán elvesztek a süllyesztőben. Hasonló helyzet tapasztalható példaképp a PHP-kézikönyv fordítócsapatának munkásságával kapcsolatban is.*

H. G.: A konferenciával együtt céljaink közé tartozik az is, hogy a hazai PHP-közösség életét felpezsdítsük, és a jó kezdeményezéseknek segítséget nyújtsunk. Ráadásul most már a felelősségtudat is hajt minket a fent említett nagy számú jelentkező és a támogatók következtében.

P. I.: Felmerült az is, hogy szervezettebb formában lehetne támogatni akár a leírás fordítását vagy egy tankönyv elkészítését is.

H. Sz.: *Láttam, hogy a konferencia programjában a „PHP Szakosztály" alapító ülése is szerepel. Mit takar ez pontosan?*

P. I.: A szervezet létrehozásának célja elsősorban az, hogy összefogja azokat a magyarországi kezdeményezéseket, amelyek a PHP nyelvvel kapcsolatosak. Legyen az egy leírás fordítása (például: Smarty, PHPDoc, PHPGtk stb.), vagy akár egy szabadon hozzáférhető tartalomkezelő, esetleg egy webáruház elkészítése. Rengeteg szabad kapacitás van ezen a téren, de mivel nincs megfelelő adatcsere, ezek az erők szétforgácsolódnak. Egy ilyen szervezet alakításához elsősorban sok ember kell, ezért van a konferencia napján az alakuló ülés. A Szakosztályról egyelőre annyit, hogy egy már működő, nagy múltú szervezethez próbálunk csatlakozni. Jelenleg is folynak a tárgyalások.

H. Sz.: *Az összeállt programról alig beszéltünk még. Mire számíthat, aki ellátogat a rendezvényre?*

P. I.: Az elsődleges célunk az volt, hogy felmérjük az igényt egy ilyen rendezvényre, valamit hogy beindítsuk a párbeszédet az egyes elszigetelt csoportok között. Igyekeztünk minél szélesebb réteget megszólítani az előadásainkkal. Lesz bevezető előadás a PHP nyelvről, egy portál felépítésének tervezéséről, elkészítésének eszközeiről, valamint a PHP oktatásának nehézségeiről. Előzetes felméréseink szerint a legnagyobb érdeklődés Szabó Dénes Smarty sablonrendszerről szóló előadását övezi.

H. G.: Szeretnénk megtalálni az ifjú tehetségeket, ezért a konferenciát egy szabad stílusú, úgynevezett freestyle programozási versennyel kötöttük egybe.

H. Sz.: *A kiírásban ez szerepel 5k Compo néven?*

P. I.: Igen. A demoscene hagyományain alapuló versenyformát választottuk, azaz minél összetettebb, többet tudó alkotásokat várunk, amelyek csupán 5120 bájtot igényelnek. Az ilyen méretkorlátos pályaművek vélhetőleg külsőre semmi különöset nem fognak nyújtani. Egyedül a szakma képviselőinek eshet le az álla, ha abba is belegondolnak, mekkora méretbe van mindez belegyömöszölve.

H. G.: A verseny győztesei között értékes ajándékok kerülnek kisorsolásra. Többek között PHP-vel és minden extrával ellátott tárhely, külföldi PHP-s folyóirat, szakkönyvek.

P. I.: Az ingyenesség ellenére a rendezvény végén tombola is lesz, ahol hasonló földi jókra számíthat minden résztvevő.

H. Sz.: *Mi az az eredmény, amivel már elégedettek lennétek?*

P. I.: Én akkor lennék elégedett, ha egy igazán jó hangulatú találkozót sikerülne összehozni, ahonnan a legtöbben elégedetten távoznának.

H. G: Örülnék, ha a rendezvényünket zökkenőmentesen sikerülne lebonyolítani, amire minden esélyünk megvan. Ha látogatóink azt érzik, hogy egy profi rendezvényen voltak, és hasznosan töltötték az idejüket, akkor jól végeztük a feladatunkat.

H. Sz.: *Köszönöm a beszélgetést!*

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

KAPCSOLÓDÓ CÍMEK

A PHP-konferencia honlapja
➲ http://phpconf.hu
Az ELTE Radnóti Miklós Gyakorlóiskola honlapja ➲ http://www.radnoti.hu
A magyar PHP-levelezőlista
➲ http://weblabor.hu/vl-phplista

Minden egyben

Az Egyesült Államok Kereskedelmi Minisztériuma egy olyan új rendszer kidolgozását tervezi, amely nagymértékben egyszerűsíthetné a polgárok és a szervezetek közötti kapcsolattartást – a használt átviteli csatornától függetlenül. Az ENUM rendszerben az előfizetők egyetlen számon keresztül lennének elérhetők, függetlenül attól, hogy telefonon, levélben, üzenőrendszerben vagy mobilon keresik-e őket. „Itt az idő, hogy az Egyesült Államok is megmozduljon" – írta egy levelében a Kereskedelmi Minisztérium. „Biztosítanunk kell, hogy az ENUM vásárlóközpontú, biztonságos és versenyhelyzetet teremtő módon kerüljön megvalósításra." A minisztérium nagy hangsúlyt fektet arra, hogy olyan megoldás kerüljön kidolgozásra, amely nem csorbítja a polgárok személyes szféráját, biztosítja a piaci versenyt és a folyamatos fejlesztést is. Az ENUM-ot – amely máris 13 tagállam támogatását élvezi – a későbbiekben nemzetközi szabvánnyá is kívánják tenni. A rendszer lényege egy a telefonszámok és internetcímek közötti megfeleltetést biztosító rendszer, melyben az azt támogató eszközök a címfordítást önműködően végzik el. Ilyen módon az internetcím ismeretében az előfizető akár fel is hívható, a telefonszámot a webböngészőbe beírva pedig önműködően a weblapjára navigálhat a vásárló.

forrás: LinuxFórum

PHPRecipeBook

Ha receptkönyvre van szükséged, akkor ez a te programod. Beviheted a hozzávalókat, valamint az elkészítési eljárást és mentheted őket. Amikor kotyvasztani akarsz valamit, a programmal összeállíttathatod a hozzávalókat tartalmazó bevásárlási listát, amit azután kinyomtathatsz. Ha még az is lehetséges volna, hogy a listát a

weben keresztül elküldd a helybéli élelmiszerüzletnek, ahonnan házhoz szállítanák a szükséges összetevőket, ki sem kellene mozdulnod otthonról. Sajnálatos módon a program nem tartalmaz beépített recepteket. Előfeltételek: PHP- és SQL-támogatással (PostgreSQL vagy MySQL) rendelkező webkiszolgáló, SQL-kiszolgáló és webböngésző.

➲ http://phprecipebook.sourceforge.net

David A. Bandel

Linux Journal 2003. február, 103. szám

Techtables

Ez a hibajelentési és eszközfigyelő rendszer egy kicsit eltér a szokásostól. Jóformán kizárólag hibabejelentésekkel és eszközökkel foglalkozik, nem ügyfelekkel és más dolgokkal. Attól függően, hogy mire van szükséged, ez a program teljesen jó megoldás lehet. A telepítése és a használata egyszerű. Ha bármit védeni akarsz, htpass-

word-öt, illetve biztonságos webes támogatást kell alkalmaznod (ami könnyűszerrel megoldható). Futtatásához PHP- és SQL-támogatással (PostgreSQL vagy MySQL) rendelkező webkiszolgáló, SQL-kiszolgáló és webböngésző szükséges.

➲ http://techtables.sourceforge.net

David A. Bandel

Linux Journal 2003. február, 103. szám

diff -u: Rendszermag-fejlesztési hírek

Jó hír azoknak, akik arra vártak, hogy megjavítsák a rendszermagban az LVM1-et. Fél évig tartó működésképtelen állapot és karbantartáshiány után végre eltávolításra került. *Joe Thornber* adta közzé azt a javítófájlt, amely ténylegesen kiveszi a rendszermagból. Úgy tűnik, mintha ez is a „szedjük rendbe magunkat a 2.6-os változat előtt” mozgalom jegyében történt volna. A levelezőlistán szóba került, hogy az LVM1-et az LVM2, a Device Mapper (DM), esetleg az EVMS váltaná fel, de egyik sem aratott osztatlan sikert. A DM-ből sok szolgáltatás hiányzik, míg az EVMS túl sokkal rendelkezik. Még az is felvetődött, hogy – ahelyett, hogy kivennék – meg kellene próbálkozni az LVM1 javításával, de legalábbis keresni kellene egy a helyettesítésére alkalmas programot. A jelenlegi állás szerint az EVMS az elsődleges jelölt a 2.6-os változat számára, de ezt még túl korai lenne kijelenteni.

A rendszermagfejlesztők ottawai találkozóján (Ottawa Kernel Summit) egyetértettek abban, hogy a driverFS nevét megváltoztatják. Mindössze az a gond, hogy senki nem tudja, mi lesz az új név. *Patrick Mochel* a „kfs” mellett kardoskodik, mások viszont azt mondják, hogy túl sok egybetűs és fs típusú név létezik már most is. *H. Peter Anvin* a „kernelfs” vagy egyszerűen „kernfs” nevet javasolta. Pillanatnyilag annyit lehet biztosan tudni, hogy a név változni fog.

Bevették a Japánban igen népszerű NEC PC-9800-as kiépítés támogatását, amely nagyjából egyenértékű a nyugati világban elterjedt Intel alapú PC-vel. Habár lehetett rajta futtatni az MS-DOS és a Microsoft Windows megfelelő változatát, sohasem volt teljesen IBM-megfelelő. Miután ez a processzor uralja a japán PC-piac 40–50 százalékát, e javítófájlok segítségével a Linux rengeteg olyan ember számára elérhetővé válik, aki korábban nem juthatott hozzá.

A *Jeff Dike* által írt User-Mode Linux most már rendelkezik SMP-támogatással. Eddig akárhány processzor volt a rendszerben, az UML-folyamatok teljes mértékben egyprocesszorosak voltak. Ez egyebek között azt jelentette, hogy az SMP-programok és különösen magának a rendszermagnak a próbafuttatása nem vezetett messzire. Ezzel az új programmal azonban lehetővé válik az olyan alkalmazások gyorsabb kipróbálása, amelyek egyébként sok hosszadalmas újraindítást tennének szükségessé, esetleg a fájlrendszer épségét is veszélyeztetve.

Az `ioctl` felület elavulttá vált. Az új illesztőprogramoknak fájlrendszer alapú csatlakozást kell létrehozniuk a `libfs`-sel, ami újonnan került be a 2.5-ös fába. Az I/O-vezérlő függvényeket éveken át ostorozták, amiért nem tarthatók karban, leírás nélküliek és egyre kuszább csoportot alkotnak, ám hiába – nem lehetett őket megkerülni. Ettől fogva a Linux-fejlesztők egy értelmes felületet használhatnak, amely nem okoz több kárt annál, mint amennyi hasznot hajt.

2002. október 31-én befagyasztották a rendszermag bővítését. Még túl korai lenne arról nyilatkozni, hogy ennek eredményeképpen viszonylag rövid idő alatt eljutunk-e a 2.6-os változathoz, vagy ismét gyors fejlesztések beláthatatlan sorozata kezdődik. *Linus Torvalds* és sokan mások már jó ideje azért küzdenek, hogy a rendszermagújítást viszonylag rövid időkeretek közé szorítsák, de a megbízható változatok között eltelt idő még mindig években mérhető. Ha a 2.6-os változat 2003 áprilisa előtt megjelenik, az nagy előrelépés lesz a fejlesztési folyamat fejlődése terén.

Zack Brown

Linux Journal 2003. február, 103. szám

Új hozzáférési pont

A WiFi (802.11 vezeték nélküli ethernet) hozzáférési pontok (access point, AP, a hálózatra kapcsolódni képes eszköz) már jelen vannak egy ideje, a többségük azonban igen korlátozott programozási lehetőséget biztosít, különösen ahhoz, hogy az egyediesítésükre üzletet lehessen alapozni. A lapkakészletgyártással foglalkozó Intersil (➲ http://www.intersil.com) gondjaiba vette ezt a hiányosságot, és PRISM AP néven bemutatta új, saját tervezésű, fejlesztési mintarendszerét. Értékesíthetőségének a titka az operációs rendszere, ami beágyazott Linux. A PRISM AP-hoz teljes linuxos fejlesztőkörnyezet tartozik, amelyen többek között webkiszolgáló, DHCP-kiszolgáló, DHCP-ügyfél és SNMP-kiszolgáló futtatható. Miután az operációs rendszer Linux, az egyes elemeket a saját igényeidnek megfelelően testreszabhatod, illetve tetszőleges terméket fejleszthetsz. A kódot flashmemóriába égetheted, és azután mindenféle engedélyeztetési költség nélkül eladhatod.

Más szavakkal: nehéz elképzelni bármi mást, ami ugyanilyen mértékben programozható és értékesíthető volna, vagy lényegesen eltérő módon nyújtaná ugyanezt.

Doc Searls

Linux Journal 2003. február, 103. szám

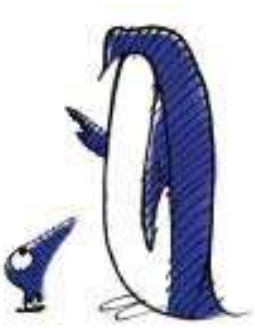
Ők mondták

Szükséges a 64-bites támogatás, szükség van terabájtos fájlrendszerekre. Az IBM azt állítja, hogy a Nyílt Forrás Közösség ezt meg fogja oldani. Én másképp gondolom. Azt valakinek meg is kell építenie. *(Jonathan Schwartz, Sun Microsystems)*

A HP azt hirdeti, hogy a HP-UX képes lesz Linux-alkalmazások futtatására, a Sun a Solarisról állítja ugyanezt. A független programgyártók pedig ezt fogják kérdezni maguktól: „Miért kellene azzal bajlódnom, hogy különböző Unix-változatokra fejlesszek programokat, amikor elég, ha Linuxra fejlesztek, és az szinte minden Unix-környezetben futni fog?” *(Dan Kusnetsky)*

Középkategória? Az valami olyan lehet, ami a hasznos és az érthető között helyezkedik el. *(David Sifry)*

Linux Journal 2003. február, 103. szám



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

A hónap szakmai tanácsai

Újraindíthatom a gépet?

Éppen telepítem a Red Hat Linux 7-et (azt a terjesztést használom, amelyet a Red Hat Linux 7 for Dummies könyv mellékleteként kaptam). A telepítés látszólag elakadt, miután 153 MB-ot telepített a 753 MB-ból. Biztonságos-e újraindítani a számítógépemet?
Marcus, marcus@lesniak.co.uk

Csak remélni tudom, hogy a számítógéped nem vár azóta is a válaszomra. Igen, biztonságos az újraindítás, de mivel nem sikerült a telepítés, nem fogod tudni elindítani a Linuxot. Lehet, hogy a telepítő CD hibás volt.
Christopher Wingert, cwingert@cwingert-mail.qualcomm.com

Ha minden lefagyott, akkor nincs más választás, csak az újraindítás. Karcmentesek a telepítő CD-k? Biztos vagy benne, hogy a merevlemezek jó állapotban vannak? Kezdd elölről a telepítést, de ezúttal a lemezrészek létrehozásakor válaszd a hibás blokkok ellenőrzését. A telepítés emiatt tovább fog tartani, de fény derülhet a lemezek bizonyos hibáira.
Felipe E. Barousse Boué, fbarousse@piensa.com

A rendszer sem hajlékonylemezről, sem CD-ről nem indul

Van egy Sony Vaio PCG F340-esem, és Slackware-t szeretnék telepíteni rá. Egy rossz tanácsra hallgatva formáztam a `c:` meghajtót. Ezután akármit tettem a CD-meghajtóba vagy a lemezmeghajtóba, a rendszerindításnál érvénytelen rendszerlemez-hibaüzenetet kapok.
John Krissinger, kriskrosx@aol.com

Lépj be a BIOS-ba, és változtasd meg a rendszerindító eszközök sorrendjét. A telepítésre használt meghajtó – akár a hajlékonylemez, akár a CD-ROM – a merevlemez előtt legyen.
Christopher Wingert, cwingert@cwingert-mail.qualcomm.com

Nem vagyok biztos benne, hogy a Slackware telepíthető a Vaióra; a hordozható gépek támogatása nem egyszerű feladat. Ha nem sikerül, próbálkozz inkább a Red Hat, a SuSE vagy a Linux-Mandrake terjesztésekkel. Ezeknek a telepítője jobb a hordozható gépek alkatrészeinek a felismerésében.
Marc Merlin, marc_bts@google.com

Hordozható gép nVidia videokártyájának támogatása

Kérlek, javasoljatok egy olyan Linux-változatot, amelyik képes működni a Dell Inspiron NoteBook számítógéppel. A gép UXGA megjelenítővel (1600×1200 képpont) és nVidia GeForce 2 Go grafikus rendszerrel rendelkezik. Sem a Dell, sem a Red Hat nem tud segíteni. Próbáltam telepíteni a Caldera régebbi változatait és a Mandrake 9.0-t, de ezek sem működtek.
Kamalakar Rao, kmlkr@juno.com

A gondod arra vezethető vissza, hogy a GeForce-hoz való jó illesztőprogramokat az nVidiától kell letölteni, mert ezek nem nyílt forrásúak. Telepítsd a neked tetsző terjesztést az Xfree 4.1-es vagy 4.2-es változatával. Ezután látogasd meg az nVidia honlapját, töltsd le a zárt forrású illesztőprogramokat, és kövesd a telepítési utasításokat (➲ http://www.nvidia.com/view.asp?IO=linux_display_1.0-3123).
Marc Merlin, marc_bts@google.com

A DNS és a DHCP beállítása

Egy Red Hat 7.3-at futtató számítógépet szeretnék iskolám kiszolgálójaként használni. Szükséges-e DNS- és DHCP-kiszolgálót beállítani?
Richard Whiteside, Hendel4ever@hotmail.com

A Red Hat remek eszközökkel rendelkezik ezeknek a szolgáltatásoknak a beállításához. Olvasd el az Installation Guide-ot, ez elég jól elmagyarázza a teendőket (➲ http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/install-guide).
Christopher Wingert, cwingert@cwingert-mail.qualcomm.com

Kiváló ötleteket kaphatsz a DNS-HOWTO-ban és a DHCP-HOWTO-ban. Ezek a leírások sok másik mellett megtalálhatók a terjesztés CD-jén, vagy az Interneten a ➲ http://tldp.org webhelyen és tükörkiszolgálóin.
Mario Bittencourt, mneto@argo.com.br

Hol a nyomtató?

Szeretném tudni, hogyan kell használni a rendszernyomtatókat az OpenOffice.org programból. A SuSE 8.0 Pro csücsül a gépemen, és egy Epson Stylus Photo 1280 van hozzá beállítva. Az OpenOffice.org telepítése után csak a „generic” nyomtatót tudtam elérni.
Nathan M. Fowler Jr., nfowler1@bellsouth.net

Az `spadmin` programot kellene futtatnod, amit az ~/OpenOffice.org1.0.1/program könyvtárban találsz meg. Ennek segítségével lehet beállítani a nyomtatókat az OpenOffice.org alá.
Felipe E. Barousse Boué, fbarousse@piensa.com

Hol az IP-cím?

Van egy Linksys útválasztóm. A Red Hat Linux felismeri az alaplapra épített ethernetkaput, de sehogy sem tudom rávenni a kapcsolódásra.
Tim Kuder, tim@kuderized.com

Próbáld meg a Red Hat `netconfig` nevű beállítóprogramját használni. Lehet, hogy be kell állítanod a DHCP-t, hogy az IP-címet a DHCP-kiszolgáló adja, ahelyett, hogy magadnak állítanál be egyet. Ezután pingeld meg az útválasztó címét. Ha ez sikerül, akkor a beállítás jó.
Felipe E. Barousse Boué, fbarousse@piensa.com

Linux Journal 2003. február, 103. szám

Új termékek

ProStore Backup Appliance

A ProMicro és az Avail Solutions összefogásából született meg a ProStore Backup Appliance, amely adattárolásra, biztonsági mentések készítésére és az adatok helyreállítására alkalmas. A ProStore készülék a ProMicro ProStore NAS-kiszolgálójából és az Avail Integrity nevű adatmentő programjából és önmű-

ködő szalagkönyvtárából lett összerakva. A ProStore jellemzői: 360 GB tárterület, Intel processzorok, 10/100-as ethernetkapuk, három PCI bővítőhely és legfeljebb hat IDE-meghajtó. Mindez egy 2U formájú házban kapott helyet. A beépített nyolckazettás szalagostár 640 GB tömörített adat átvitelére képes 6 MB/s sebességgel. Minden jellemző beállítható a felhasználó által.
Adatok: ProMicro Solutions, 12635 Danielson Court #203, Poway, California 92064, telefon: 866-776-6427,
➲ http://www.promicro.com;
Avail Solutions, 2430 Vineyard Avenue, Suite 205, Escondido, California 92029, telefon:760-743-7200,
➲ http://www.availsolutions.com

ACCPAC Advantage Series 5.0

Az ACCPAC Advantage Series Enterprise Edition 5.0 olyan többrétegű, webalapú vállalatirányítási rendszer, amely a teljes könyvelési rendszerhez hozzáférést ad a böngészőn vagy az ACCPAC munkaasztalon keresztül. Az Enterprise Edition a következő feladatköröket láthatja el: rendszerkezelő, főkönyvi könyvelés, tartozások, követelések, állóeszköz-nyilvántartás, rendelések kezelése, beszerzés, valamint amerikai és kanadai rendszerű bérszámfejtés. A főkönyv konszolidálása és a vállalatközi tranzakciók modul szintén rendelkezésre áll. Az Enterprise Edition képes több nyelv és több pénznem kezelésére, korlátlan számú felhasználót támogat, és együttműködik az Oracle, az IBM DB2 és a Pervasive adatbázisokkal.
Adatok: ACCPAC, 6700 Koll Center Parkway, Third Floor, Pleasanton, California 94566,
telefon: 925-461-2625,
➲ http://www.accpac.com

SCO Linux 4.0

A SCO Group a Caldera márkanév használatának abbahagyása óta első alkalommal jelent meg új termékkel a piacon; bejelentette a SCO Linux 4.0-t. A SCO Linux 4.0 a UnitedLinux 1.0-n alapul, és kitüntetett fontosságú üzleti alkalmazásokhoz tervezték. Az SCO Linuxhoz programok, támogatás és szolgáltatások is járnak a kis- és középvállalatok számára a több mint 16 ezer viszonteladón keresztül. A SCO Linux 4.0 négy változatban érhető el: Base, Classic, Business és Enterprise. Mindegyikhez egyéves támogatás jár.
Adatok: The SCO Group, 355 South 520 West, Suite 100, Lindon, Utah 84042, telefon: 801-765-4999,
➲ http://www.sco.com

Frequency Clock: Free Media System

A Frequency Clock: Free Media System nyílt forrású programrendszer a hang- és képfolyamok csatornáinak kezelésére. A felhasználók médiafolyamaikat dinamikus csatornákba rendszerezhetik, amelyeket webalapú médiafolyam-lejátszóval lehet megtekinteni. A Radioqualia által létrehozott Free Media System legfontosabb szolgáltatásai közé tartozik a testreszabott médiafolyam-lejátszó, amely többféle fájltípust ismer, valamint a kereshető archívumok és a valós idejű statisztikai elemzés. Minden médiafolyam, többek között a Windows Media, a Real és a QuickTime, ugyanazzal a lejátszóval – a Frequency Clock Playerrel – játszható le. Ráadásul a felhasználók a Playert saját tervezési szempontjaik alapján testreszabhatják, a tervezést nem szükséges a lejátszóhoz igazítani.
Adatok: The Frequency Clock, e-mail: radioqualia@va.com.au,
➲ http://radioqualia.va.com.au/freqclock/central.html

Zaurus SL-5600

A Zaurus PDA-családjának legújabb tagja az SL-5600, ami egy, az üzleti felhasználók számára készített vezeték nélküli kapcsolattartásra képes tenyérgép. Az 5600-as nagyfelbontású, színes QVGA LCD-vel, QWERTY billentyűzettel, 64 MB védett flashmemóriával, 32 MB SDRAM memóriával, kettős bővítőhellyel (Compact-Flash és SD/MMC kártyahelyek), beépített hangszóróval és mikrofonnal rendelkezik. A Qtopia alapú PDA-ba Intel XScale 400 MHz-es processzort szereltek, amely a memóriával 100 MHz-en tart kapcsolatot. Az SL-5600 virtuális mobil merevlemezzel is rendelkezik, ami a flashmemóriában megvédi az adatokat, az alkalmazásokat és az állományokat.
Az új Zaurus illesztőprogramokat tartalmaz a 802.11b vezeték nélküli LAN-csatolókhoz, CDPD vezeték nélküli modemekhez és 10/100-as ethernetkártyákhoz.
Adatok: Sharp Zaurus,
telefon: 201-529-9459,
➲ http://www.myzaurus.com
➲ http://www.sharpusa.com

Cubix BladeStation

A Cubix Corporation BladeStation nevű terméke egy kétprocesszoros Pentium 4 Xeon alapú pengekiszolgáló legfeljebb négy PCI-X/PCI kártyahellyel pengénként. A BladeStation FSB sebessége 533 MHz, és minden egyes penge összesen négy SCSI-meghajtót támogathat. Legfeljebb hét kétprocesszoros Xeon penge építhető be egy 6U magas keretbe, mindössze 21 hüvelyk mélységet felhasználva a keretben. Minden penge egy teljes hosszúságú 64 bites 133/100 MHz-es PCI-X bővítőhellyel, legfeljebb 8 GB DDR RAM-mal, egy 1 Gb-es ethernetkapuval és két 10/100-as ethernetkapuval rendelkezik. A PowerStation tápegységek rendszere hibatűrő n+1-es tápellátást biztosít.
Adatok: Cubix Corporation, 2800 Lockheed Way, Carson City, Nevada 89706, telefon: 800-829-0550,
➲ http://www.cubix.com

Linux Journal 2003, 106. szám

Kevesebb törődéssel

Ha Linus többet törődött volna azzal, mi történik a rendszermagon kívül, a Linux talán sokkal kevésbé hasznos operációs rendszer lenne.

Tavaly októberben több mint száz Linux-hívő egy boldog hetet töltött a Geek Cruise fedélzetén ***Linus Torvalds*** társaságában. Noha úgy tűnt, hogy több szó esett a családról, mint a Linuxról (ha már itt tartunk, nekünk négy hatévesnél fiatalabb gyermekünk van), elégszer hallottam Linust technológiai témáról beszélni ahhoz, hogy megállapítsam: ennek az embernek a mantrája három szóból áll: nem törődöm vele.

Linus ezzel a nyilatkozattal kezdte beszédét a hajón: „Én csak a rendszermaggal foglalkozom. A felhasználói szintű dolgokkal is törődtem tíz évvel ezelőtt, de csak azért, mert nélkülük a rendszermag nem használható. Nem tudom, mi történik a rendszermagon kívül, és nem is nagyon érdekel. Csak azzal törődöm, ami a rendszermagon belül történik."

Az, hogy nem törődik mással, nem jelenti azt, hogy Linusnak nincs is véleménye. Mint másoknak, neki is igen szerteágazó véleménye van. Velünk ellentétben azonban ő főszereplő, akinek a véleménye nagy súllyal esik latba. Linus kínosan ügyel arra, hogy kifejtse, milyen csekély mértékben törődik bizonyos dolgokkal, amelyek talán érdekesek, ám elvonják az ember figyelmét. Tökéletes példa erre a politika. Fent említett beszédében Linus ezt mondta: „Számomra mindenféle politikának csak szórakoztatási értéke van. Én nem törődöm vele." S mivel nem törődik vele, a Linux hatalmas siker lett. Sőt a sikere napról napra növekszik.

A Linux kezdett meghatározó rendszerré válni – mi több, talán „a" meghatározó operációs rendszerré –, mégpedig pusztán gyakorlati okokból. A Linux olcsó és könnyen telepíthető. Olyan egyszerű és hasznos, amilyen egy operációs rendszer csak lehet. Ez ugyan a Linux-közösség számára nem újdonság, de annál inkább új azoknak a vezetőknek, akik nem szoktak hozzá, hogy az operációs rendszer nem válik elavulttá holmi politikai irányváltás következtében.

Az egyik fórumon felmerült az elavulás kérdése. Linus rámutatott, hogy a kereskedelmi programok bizonyos fokig olyan alapmodellre épülnek, ahol az elavulás értéknek számít. Például a Windows 98 megjelenésekor megkérdezték ***Bill Gates***-et, hogy mennyire tartja veszélyesnek a Mac OS-t. Gates azzal hárította el a kérdést, hogy a Windows 98 valódi ellenfele a Windows 95. Nyíltan az volt a cél, hogy újabb bevételt nyerjenek ki a teljes ügyfélkörükből. Az Apple tervei is nyilvánvalóan hasonlóak, amikor megjelenteti az újabb Mac OS X-es változatokat. A vásárlók immár két évtizede elkerülhetetlennek tartják ezeket a váltásokat – eddig ugyanis nem volt más választásuk.

A Linuxszal azonban olyan operációs rendszert választhatnak az ügyfelek, ami nem törekszik arra, hogy elavulttá tegye saját magát. Ez a választási lehetőség sokak érdeklődését felkeltette múlt nyáron, amikor a Microsoft megemelte a Windows felhasználási díját. A kedvezőtlen gazdasági körülmények között a drágulás hatására a vezető beosztásúak sokkal nagyobb kedvet éreztek a Linux kipróbálására.

A rendszermag szintjén a Linuxnak nincsenek kereskedelmi tervei. Pusztán az a célja, hogy hasznos legyen. Ha emellett még pénzt is lehet vele keresni, az jó. Linus és az ő rendszermagja egyébként nem törődik vele. Bizonyos dolgok menet közben természetesen elavulttá válnak. Linus elmondta a beszédében, hogy a 2.6-os rendszermag eszközrétege teljesen megújul majd. Ezek a változások azonban nem azért születnek, hogy elavulttá tegyenek bármit is. Azért történnek, hogy a rendszermag több szempontból is jobb legyen, a lehető leghosszabb ideig.

A Linux gyakorlatiassága másra is kihat a rendszermag által támogatott számtalan lehetőség révén. Ellenpéldaként igen nehéz elképzelni, hogy a Microsoft vagy az Apple célja az lenne, hogy minél több olyan asztali géppel vagy felhasználói felülettel együttműködjön, amelyet nem saját maga fejlesztett ki. A Linux pedig pontosan ezt teszi. Úgy biztosítja az együttműködést az ilyen asztali gépekkel és kezelőfelületekkel, hogy nem törődik velük.

Dave Sifry így magyarázza ezt a folyamatot: „Azáltal, hogy határozottan elkülönül a rendszermag és a felhasználói terület kódja, a rendszermag szilárdabbá válik, és erős fejlesztések indulhatnak a felhasználói területen. Például a rendszermagon belüli kód visszaszorításával a Sambához hasonló kezdeményezések nem központosított módon tudtak újat alkotni, illetve megbízható, sokoldalú programokat készíteni. Nincs szükség a Linux-rendszermagot érintő javítófájlra, ha meg akarják változtatni a Sambát. Ugyanez elmondható a Linux többi lényeges alrendszeréről is; ilyen például az XFree86, a Gnome, a KDE, a böngészők, sőt még a glibc is (habár a glibc nem olyan jó példa, mint az előzők). Amiatt sem kell aggódni, hogy Linus esetleg valami rejtett API-t ír, hogy az OpenOffice lekörözze az AbiWordöt, vagy a Mozilla az Operát, netán a KDE a Gnome-ot – bármelyik program legyen is Linus kedvence.

A törődés hiánya a végső pálya, és úgy tűnik, ez adja a legjobb alapot más rá építő pályáknak. Miközben a Debian talán a legkevésbé kereskedelmi érdekeltségű Linux-változat, kimagaslóan hasznos építőanyagnak bizonyult a Lindows, a Xandros és a hasonló rendszerek számára. Ha a Debian ezekkel a megvalósításokkal foglalkozott volna, valószínűleg sokkal kevésbé lenne gyakorlatias.

Az „átláthatóság" is olyan régi jó Linux-tulajdonság, amelyre a vezetők mostanában egyre inkább figyelnek. Vajon mikor jön el az az idő, amikor az emberek ugyanolyan áttekinthetőséget követelnek meg az általuk használt operációs rendszertől, mint a könyvelési rendszerüktől? Ugyan, ne törődjünk vele. Úgyis bekövetkezik majd.

Linux Journal 2003. február, 106. szám

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

Ne fejlesszünk Linuxra!

Jobb, ha akkor választjuk ki a felületet, amikor a program már elkészült.

Rég elmúlt az az idő, amikor megengedhettük magunknak, hogy egyetlen felületre fejlesszünk programot. Miért? Mert minden felület legalább egy olyan előnyt nyújt, amelyet egyetlen másik sem. A Windows, a Linux/Unix, a Mac OS X, a beágyazott Linux és az összes többi felület egyedi előnyöket kínál. Ugyanakkor a változó piaci körülmények között lehetetlen megjósolni, melyik felület kínálja majd a remélt versenyelőnyt.
A megoldás: nem kell választani. Nézőpontunk szerint a programfejlesztők kihasználhatják minden egyes felület legjobb tulajdonságait, amennyiben a többfelületes fejlesztés mellett döntenek.
Ez nemcsak a személyi számítógépekre igaz, hanem kiszolgálók, hálózatok, hordozható eszközök és minden más, kapcsolatot biztosító eszköz esetében is.
Az egyre kevésbé helyhez kötött munkavégzés a mai osztott hálózatokhoz és világméretű szervezetekhez illeszkedő hordozható adatokat és hordozható alkalmazásokat követel.
Ha egy cég talpon akar maradni a piaci versenyben, föl kell ismernie, hogy az operációs rendszerek sokfélesége megkerülhetetlen adottság, és válaszul olyan alkalmazásokat kell fejlesztenie, amelyek a lehető legtöbb felületen gyorsan, tisztán és közvetlenül futtathatók. Az így megírt alkalmazás kihasználja minden egyes felület legjobb tulajdonságait, anélkül, hogy minden esetben újra kellene írni – ez az eljárás ugyanis korlátozza a fejlesztő céget és óriási időveszteséget jelent. A hosszabb távban gondolkodó cégek már fölismerték, hogy az egyetlen felületre történő fejlesztés kudarcra van ítélve, és jobb módszer mellett döntöttek.

Az egyfelületes fejlesztés drága

Ha egynél több felületre szeretnénk fejleszteni – ilyen módon bővítve piacunkat –, költségeink nyomasztóan megnőnek. Minden egyes felülethez teljes fejlesztőgárdára lesz szükségünk. Talán még ennél is fontosabb, hogy minden egyes felülethez teljes karbantartó és terméktámogatási csapatot kell alkalmaznunk. Ez minden egyes felülethez a költségek egyenes arányú növekedését jelenti, ami hihetetlenül alacsony hatékonyságú gazdálkodás.

Az egyfelületes fejlesztés egyirányú utca

Egy alkalmazást egyetlen felületre fejleszteni a kockázat növelésével egyenlő, mivel a felületek között választanunk kell még , mielőtt világosan felmérhetnénk a bennük rejlő lehetőségeket.
Ki tudja, igazunk lesz-e? Ez a döntés felemelt és tönkre is tett már programfejlesztő cégeket. A közelmúltban szinte mindenki azt mondta, hogy a Windows (lendületes terjeszkedése és uralkodó piaci helyzete okán) a kézenfekvő választás – de várjunk csak! A Linux komoly versenyben áll a kiszolgálók világában és egyre erősebben terjed a személyi számítógépek és a beágyazott rendszerek világában. Világméretű vevő- és profitközpontú cégek döntenek a Linux által nyújtott teljesítmény, rugalmasság, biztonság és alacsony költségek mellett. A korábban kézenfekvőnek tartott felületválasztás többé már nem annyira egyértelmű. Meg tudja valaki mondani, mikor (vagy hol) következik be ehhez hasonló villámgyors átalakulás? Én nem tudom.

> Ha egy cég talpon akar maradni a piaci versenyben, föl kell ismernie, hogy az operációs rendszerek sokfélesége megkerülhetetlen adottság…

Az egyfelületes fejlesztés nem nyitott a hordozható eszközök felé

Talán a legfontosabb szempont az, hogy egyetlen személyi számítógépes vagy kiszolgáló felületre fejlesztve azonnal megnehezítjük magunknak a hozzáférést a világ leggyorsabban bővülő programpiacához, ami nem más, mint a hordozható eszközök piaca. Ha például Microsoft Windows NT/2000 rendszerre írunk egy alkalmazást, eleve kizárunk minden költséghatékony módszert, amellyel az alkalmazást hordozható eszközökre vihetnénk át, mivel a kódot újra kell írnunk. Figyelembe véve, hogy az alkalmazásokat szinte kötelező hordozhatóvá tenni, egy alkalmazásnak még az elkészülte előtt a halálos ítéletét jelentheti, ha egyetlen személyi számítógépes vagy kiszolgáló felületre fejlesztjük.
A programfejlesztő ipar régóta küzd azzal, hogy gazdaságosan alkalmazható módszert dolgozzon ki a többfelületes fejlesztések számára. Az iparág története tele van olyan cégekkel, amelyek megpróbálták, és elbuktak. Miért?
Az egyik nehézség a teljes körű megvalósítás hiánya volt. Sok eszközkészlet bizonyos felületeken a szolgáltatásoknak csupán egy részhalmazát nyújtja.
További gondot okozott az emulációra vagy virtuális gépekre való hagyatkozás. Mindkettő jelentős és többnyire elfogadhatatlan teljesítménycsökkenéssel jár, különösen a hordozható eszközök számára, amelyeknek mindennél inkább szükségük van a jó futási sebességre.
Közismert tény, hogy a virtuális gépek közti különbségek a megvalósítás során különutas megoldásokhoz és teljesítménynövelő trükkök alkalmazásához vezetnek, és ezzel együtt növekedő karbantartási igényekhez is. Ez újabb költség, azonkívül megkeseríti az ilyen munkával megbízott fejlesztők életét.
Ma azonban már kipróbált módszerek léteznek arra, hogy egy alkalmazást egyszer írjunk meg, és az azután bárhol lefordítható és futtatható legyen. A többfelületes fejlesztést végző cégek olyan munkakörnyezetet hoznak létre, amelyben a fejlesztést előrevivő újítások végre megint a mindennapos munkát jelentik – nem pedig a ritka kivételt.

Linux Journal 2003. február, 103. szám

Haavard Nord
A Trolltech ügyvezetője és egyik alapítója. Programozói pályafutása kezdetén elfogadható többfelületes fejlesztőkészletet keresett adatbázis-fejlesztéshez. A cég termékei megkönnyítik az alkotó munkát, mert segítségükkel a fejlesztők az alkalmazás kódját egyszer megírva közvetlenül futtathatják Windows, Linux, Unix, Mac OS X és beágyazott Linux felületeken.

Az IBM naplózó fájlrendszere

Naplózó fájlrendszerrel létfontosságú kiszolgálód mindig újra tud indulni.

A Linux rendszermagja folyamatosan új lehetőségekkel bővül, ezek egyike a naplózó fájlrendszerek támogatása. Az IBM JFS-e a Linuxhoz létező naplózó fájlrendszerek egyike. Ebben a cikkben a JFS működését és jellemzőit tárgyaljuk. Szót ejtünk arról, hogyan kell telepíteni és beállítani egy Linux-kiszolgálón, valamint milyen tapasztalatokra tettünk szert a használata során az Ericsson Research Labnél (Montrealban).

Egy fájlrendszer arra szolgál, hogy a merevlemezeken lévő felhasználói adatokat tárolja és kezelje. Biztosítja, hogy az adat, amit a lemezre írunk, teljesen megegyezik azzal, amit a lemezről visszaolvasunk. A fájlok tárolása mellett a fájlrendszer számos egyéb adatot is kezel, úgymint a rendelkezésre álló szabad terület mértékét, vagy az `i` csomópontok számát, amelyeket a saját céljaira használ fel. A fájlrendszer efféle szerkezeteit kiegészítő adatoknak (metadata) szokás hívni, ezekbe a fájl tényleges tartalmán kívül minden egyéb beletartozik. A fájllal kapcsolatos adatokat, mint a fizikai elhelyezkedését vagy a méretét is a kiegészítő adatok között tárolják.

Naplózó fájlrendszerek a nem naplózó fájlrendszerek ellenében?

Egy naplózó fájlrendszer tartalmilag sokkal összefüggőbb, ugyanakkor a helyreállítása is jóval egyszerűbb, mint nem naplózó társainak. Ennek következtében naplózó fájlrendszerek esetén az újraindításhoz szükséges idő is rövidebb.

A nem naplózó fájlrendszerek ki vannak téve annak a veszélynek, hogy esetleg megsérülnek, mivel egy-egy logikai fájlművelet gyakorta több I/O műveleten keresztül kerül elvégzésre, így egy váratlan leállás esetleg a folyamat kellős közepén szakítja meg a műveletsort. Például egy egyszerű állományba történő íráshoz a következő műveletek szükségesek:

- Terület lefoglalása a fájlrendszeren.
- A területek mutatóinak a frissítése.
- A fájlméret frissítése.
- Az adat tényleges kiírása.

Ha a rendszert ezeknek a műveleteknek az elvégzése közben szakítjuk meg, akkor a nem naplózó fájlrendszer egy köztes, nem összefüggő állapotban marad. Ilyen esetekben ezek a fájlrendszerek az `fsck` nevű eszközre bízzák, hogy feltárja a kiegészítő adatokban keletkezett hibákat (például a könyvtár- és lemezcímző szerkezeteket), és lehetőség szerint javítsa is őket. Azonban az `fsck` elég időigényes lehet, függően a lemez méretétől, a könyvtárak és a bennük található fájlok számától. Tehát egy nagyobb fájlrendszer esetén a naplózó tulajdonság elengedhetetlen, hiszen egy naplózó fájlrendszer kevesebb mint egy másodperc alatt képes újraindulni.

Bemutatkozik a JFS

A JFS-t úgy tervezték, hogy bármilyen üzemzavart követően gyorsan képes legyen helyreállni, ezenkívül felkészítették nagy lemezrészek és nagyméretű állományok kezelésére, azonban ugyanígy megbirkózik nagyszámú könyvtárakkal és fájlokkal is. Ahhoz, hogy megfeleljen ezeknek a követelményeknek, a JFS a másodperc törtrésze alatt képes helyreállni, úgy, hogy csak a kiegészítő adatokban bekövetkező változásokat naplózza. A JFS támogatja a 64-bites rendszereket is, akár petabájt (2^{50} bájt = 1024 terabájt) méretű fájlokkal és lemezrészekkel. Ráadásul a fájlrendszerben található összes szerkezet B+ fákra épül. A nagyobb teljesítmény elérése érdekében a fájlrendszerben az összes lehetséges helyen B+ fákat használnak a hagyományos lineáris fájlrendszerszerkezetek helyett.

Fájlrendszerek

Az állományok úgynevezett fokok sorozatában tárolódnak. Foknak nevezzük az egymást követő szomszédos blokkok sorozatát, ami a JFS-ben egy egységet jelent. A fokot teljes egészében egy csoport belsejében (és ezáltal egyetlen lemezrészen) találjuk meg. Nagyméretű fokok azonban több tárterületi egységre is kiterjedhetnek. Egy fok mérete bármekkora lehet 1 és 2^{24} -1 blokk között. Példának okáért a JFS 24-bites értéket használ a fok hosszának meghatározására. 4 k-s blokkméret esetén a legnagyobb fokméret 4 k 2^{24} -1 lehet, ami megközelítőleg 64 GB-tal egyenlő. Fontos, hogy ez a határ csak egyetlen fokra vonatkozik – a fájl teljes méretét ez egyáltalán nem befolyásolja. A fokokat egy B+ fa alapján tartják nyilván a nagyobb teljesítmény érdekében, legyen szó akár új fokok hozzáadásáról, akár fokok kereséséről stb. Általánosságban a JFS elhelyezési politikája (allocation policy) a lehető legfolyamatosabb tárterület-elrendezést próbálja meg kialakítani úgy, hogy egy állományhoz minél kevesebb fokot használjon fel, és ezek a fokok lehetőség szerint minél nagyobbak legyenek. Ennek következtében az I/O műveletek hatékonysága növekszik, ami a teljesítmény javulását is magával vonja.

A JFS története

Az IBM a Unix fájlrendszerét Journaled Filesystem, vagyis Naplózó fájlrendszer néven mutatta be, amelynek első változata az AIX 3.1-gyel jelent meg. Ezt a fájlrendszert az AIX-en most JFS1-nek hívják. Az elmúlt tíz év során ez volt az AIX elsődleges fájlrendszere, és

vásárlók millióinak a gépein található meg. 1995-ben kezdtek el dolgozni azon, hogy a fájlrendszer méretezhető legyen, és támogassa a többprocesszoros rendszereket. A másik célkitűzés az volt, hogy a fájlrendszer más operációs rendszerekre egyszerűbben átültethető legyen.
Történelmileg úgy alakult, hogy a JFS1 fájlrendszer erősen kötődött az AIX memóriakezelőjéhez. Az effajta kialakítás jellemző a zárt forrású operációs rendszerekre, illetve azokra a fájlrendszerekre, amelyek csak egyféle operációs rendszert kívánnak támogatni.
Az új Journaled Filesystemet, amelyen a Linux-változat is alapul, először 1999 áprilisában az OS/2 Warp Server for eBusiness nevű termékükhöz adták, amelyet sokévnyi tervezés, kódolás és kipróbálás előzött meg. Az OS/2 Warp Client nevű termék is tartalmazta a fájlrendszert, amit 2000 októberében adtak ki. Ezzel párhuzamosan még 1997-ben a JFS fejlesztői csapatának néhány tagja visszatért az AIX operációs rendszer fejlesztői csoportjába, és elkezdtek dolgozni azon, hogy az új JFS az AIX-on is működjön. 2001 májusában Enhanced Journaled Filesystem (JFS2) néven kiadták a naplózó fájlrendszer második változatát az AIX 5L-hez. Mindeközben 1999 decemberében felfedték az OS/2-höz adott JFS forráskódját, és ezzel egy időben elkezdődött a JFS átültetése Linuxra.

Tapasztalatok a nyílt forrású projekttel

1999 decemberében három naplózó fájlrendszer is fejlesztés vagy átültetés alatt volt Linuxra. Az `ext2`-höz naplózási képességeket adtak, és a létrejövő új fájlrendszert `ext3`-nak nevezték. XFS fájlrendszerüket az SGI az elkezdte átültetni az Irixről. A harmadik fájlrendszer fejlesztését ***Hans Reiser*** kezdte el, így ennek a neve ReiserFS lett. De Linuxon e fájlrendszerek egyike sem volt teljes egészében működőképes 1999-ben. Az IBM hitt abban, hogy a JFS egy nagyon fejlett és markáns fájlrendszer, és átültetésével értékesebbé tehetnék a Linuxot.

Felvették tehát a kapcsolatot a vezető linuxos fájlrendszer-fejlesztőkkel, és tisztázták hogy lehetséges egy újabb naplózó fájlrendszer hozzáadása.
A Linux alapvető szellemisége, hogy meg kell adni a választás lehetőségét, így ennek fényében elfogadták az újabb naplózó fájlrendszer ötletét.
Az IBM 1999 decemberében kezdte el átültetni a fájlrendszert Linuxra, és 2000 februárjára már előálltak az első forráskóddal. Ez az első változat tartalmazta a bemutató forráskódot, támogatta a kötetek befűzését és leválasztását, és lehetővé tette az `ls` parancs használatát a JFS-lemezrészen.

A JFS telepítése különálló lemezrészre

A JFS a 2.5.6-os fejlesztői változat óta része a rendszermagnak, de az ***Alan Cox***-féle 2.4.x-ac rendszermagok is tartalmazzák a 2002 februárjában kiadott 2.4.18-pre9-ac4 változattól kezdődően. Alan rendszermagfoltjai a 2.4.x-es sorozathoz a ➲ http://www.kernel.org címen érhetők el. A 2.4-es rendszermagot külön is letöltheted, és saját magad hozzáadhatod a JFS-foltokat. E cikk írásának időpontjában a legújabb rendszermag a 2.4.18-as, a legújabb JFS-változat pedig az 1.0.20-as. A cikk további részében ezekkel fogunk dolgozni. A JFS-folt a JFS honlapjáról tölthető le. Szükséged lesz a JFS segédeszközeire (`jfsutils-1.0.20.tar.gz`) és a fájlrendszerfoltokra (`jfs-2.4.18-patch` és `jfs-2.4-1.0.20.tar.gz`) is.
Több Linux-terjesztés már eleve tartalmazza a JFS-t: legújabb változataiban a Turbolinux, a Mandrake, a SuSE, a Red Hat és a Slackware mind mellékeli a JFS-t.

A rendszermag felkészítése a JFS-re

Ha a fentebb felsorolt terjesztések bármelyikét használod, nem lesz szükséged a rendszermag foltozására. Csupán annyit kell tenned, hogy a rendszermagot le kell fordítanod JFS-támogatással. Elsőként töltsd le a hivatalos Linux-rendszermagot. Ha a rendszereden már létezik a */usr/src/linux* könyvtár, akkor nevezd át, így nem fogja felülírni a linux-2.4.18-as forrásfa. Miután letöltötted a rendszermagot tartalmazó `linux-2.4.18.tar.gz` fájlt, mentsd a */usr/src* könyvtárba, és csomagold ki. Így létre fog jönni egy új */usr/src/linux* könyvtár.
Következő lépésként töltsd le a JFS segédeszközeit és a 2.4.18-as rendszermaghoz tartozó foltot. Hozz létre egy könyvtárat a JFS forráskódjának */usr/src/jfs1020* néven, és ebbe a könyvtárba töltsd le a foltot. Ezt követően lépj be a 2.4.18-as rendszermag könyvtárába, és helyezd fel a foltot:

```
% cd /usr/src/linux
% patch -p1 < /usr/src/jfs1020/jfs-2.4-18-
  ➥patch
% cp /usr/src/jfs1020/jfs-2.4-1.0.20.tar.gz.
% tar zxvf jfs-2.4-1.0.20.tar.gz
```

A rendszermag beállításakor a `make menuconfig` vagy `make config` parancsot követően a ***Filesystems*** almenüben (`CONFIG_JFS_FS=y`) engedélyezd a JFS-t. Arra is lehetőséged nyílik, hogy a JFS-t különálló bővítményként állítsd be. Ebben az esetben elég csak a bővítményeket lefordítanod és feltelepítened:

```
% make modules && make modules_install
```

Máskülönben, ha a JFS-t közvetlenül a rendszermagba akarod befordítani, a teljes rendszermagot újra kell fordítanod:

```
% make dep && make clean && make bzImage
```

Majd fordítsd le és telepítsd a bővítményeket is, ha vannak fordítandó bővítményeid:

```
% make modules && make modules_install
```

Végül telepítsd fel az új rendszermagot:

```
% cp arch/i386/boot/bzImage /boot/bzImage-jfs
% cp System.map /boot/System.map-jfs
% ln -s /boot/System.map-jfs /boot/System.map
```

Ne feledd lefuttatni a LILO-t sem! (Ha még sosem fordítottál rendszermagot, olvasd el a Kernel-HOWTO-t, hogy megtudd, hogyan kell.)
Ha a rendszermag fordításával és telepítésével elkészültél, a JFS kiegészítéseit is le kell fordítanod és telepítened kell. Másold a `jfsutils-1.0.20.tar.gz` fájlt a */usr/src/jfs1020* könyvtárba, majd:

```
% tar zxvf  jfsutils-1.0.20.tar.gz
% cd jfsutils-1.0.20
% ./configure
% make && make install
```

Ha ezzel is megvagy, a következő lépés, hogy egy JFS-lemezrészt hozz létre. A következő példában mi egy tartalék lemezrészt használunk erre a célra; a következőkben megtudod, hogyan kell átköltöztetni egy meglévő lemezrészt JFS-re.
Ha még van a lemezeden felosztatlan hely, hozz létre egy új lemezrészt az `fdisk` paranccsal. A mi rendszerünk tartalmaz egy */dev/hdb3* tartalék lemezrészt, így mi ezt alakítottuk JFS fájlrendszerűvé. Miután létrejött az új lemezrész, indítsd újra a rendszeredet, hogy megbizonyosodj arról, hogy az új lemezrész képes a JFS fogadására.
A JFS létrehozásához add ki a következő parancsot:

```
% mkfs.jfs /dev/hdb3
```

Miután a fájlrendszer létrejött, be kell fűzni a rendszerbe. Ehhez először létre kell hozni egy könyvtárat, ahova a fájlrendszert be lehet fűzni. Legyen ez a könyvtár a */mnt/jfs*, majd adjuk ki a `mount` parancsot:

```
% mount -t jfs /dev/hdb3 /mnt/jfs
```

Ha a fájlrendszert sikeresen befűzted, készen állsz rá, hogy kipróbáld a JFS-t.
A fájlrendszer lecsatolásához az `umount` parancsot kell kiadni az előzőleg létrehozott csatlakozási pontra:

```
% umount /mnt/jfs
```

Átállás ext2-ről JFS-re

Az előző szakaszban bemutattuk, hogyan hozzunk létre JFS fájlrendszert egy meglévő tartalék lemezrészen. Most azt szemléltetjük, hogyan lehet átalakítani egy tetszőleges fájlrendszert, például az `ext2`-t JFS-re. Megnézzük, miként lehet összeismertetni a Linuxot egy JFS-lemezrésszel, majd hogyan tehetjük ezt a lemezrészt a gyökérfájlrendszerré.
Milyen elrendezés szükséges egy JFS-gyökérfájlrendszerhez? Az átállási folyamathoz egy üres lemezrészre lesz szükségünk. Tételezzük fel, hogy a */dev/hda5* tartalmazza a jelenlegi `ext2` gyökérfájlrendszert. Mi a */dev/hda6*-ot használjuk JFS-gyökérfájlrendszerként. Először az `ext2` lemezrész tartalmát átmásoljuk a JFS-lemezrészre. Ezt követően, ha nem akarod megtartani az `ext2`-es fájlrendszert, anélkül leformázhatod, hogy Linux-rendszeredet elveszítenéd.
Hogy a JFS-re épülő gyökérfájlrendszert használhasd létre a */dev/hda6*-on, kövesd az előzőekben leírt utasításokat, hogy rendszermagod tartalmazza a JFS-bővítményt. Ahhoz, hogy erről az új lemezrészről el lehessen indítani a rendszeredet, Linux-telepítésedet egy az egyben át kell másolnod az új fájlrendszerre. Ennek legegyszerűbb módja: másolj át minden állományt a JFS-lemezrészre. Elsőként fűzd be a fájlrendszert:

```
mount -t jfs /dev/hda6 /jfs
```

Majd másolj át minden fájlt az `ext2` fájlrendszerről a JFS-re:

```
% cd /
% cp -a bin etc lib boot dev home usr var
  ➥[ ] /jfs
```

A */proc* és a */tmp* különleges bánásmódot igényel:

```
% mkdir /jfs/proc
% chmod 555 /jfs/proc
% mkdir /jfs/tmp
% chmod 1777 /jfs/tmp
```

Nagyon fontos, hogy a */proc*-ot és a */tmp*-t a megfelelő jogosultságokkal hozd létre. Az 1777-es jogosultság azt jelenti, hogy az ilyen könyvtárban csak a fájlok és a könyvtárak egyéni tulajdonosai nevezhetik át és törölhetik a fájlokat vagy a könyvtárak tartalmát, vagy a rendszergazda. Az utolsó lépés magában foglalja a */etc/lilo.conf* és */etc/fstab* fájlok megváltoztatását. Elsőként a *lilo.conf*-ot változtatjuk meg, hogy rendszerünket a JFS fájlrendszeren lévő rendszermaggal lehessen indítani. Figyeld meg, hogy a gyökérfájlrendszer az első bejegyzésben különbözik. Mindez azért van így, mert a kötet, amiről a rendszerünket el szeretnénk indítani, nem a */dev/hda5/boot* könyvtárban van, hanem a */dev/hda6/boot* könyvtárban:

```
image=/boot/vmlinuz-jfs
       label=jfs-kernel
       read-only
       root=/dev/hda6
```

Végül a */etc/fstab* fájlon keresztül Linux-rendszerünknek meg kell mondanunk, hogy milyen fájlrendszereket fogunk használni. Változtassuk meg a következő sort:

```
/dev/hda5     /      ext2   defaults   1      1
```

hogy így nézzen ki:

```
/dev/hda6     /      jfs    defaults   1      1
```

Most újraindíthatod a rendszeredet, és betöltheted a `jfs-kernel` rendszermagot – így a Linux a JFS gyökérfájlrendszerrel fog felállni.
Meghibásodást követően a napló önműködően visszajátssza a műveleteket. A megszokott `fsck` üzenetek helyett a JFS naplójának üzeneteit fogod látni. A napló visszajátszása elengedhetetlen, ha a fájlrendszer sérült.

Az Ericsson Research Lab tapasztalatai a JFS-sel

Az Ericsson Research Open Systems Labjének egyik feladata az, hogy olyan rendszereket tervezzen, hozzon létre és mérjen fel, amelyek távközlési alkalmazások alapjául szolgálhatnak. Az ilyen távközlési rendszereknek nagyon magas és szigorú követelményeknek kell megfelelniük a méretezhetőség, a megbízhatóság és a magas rendelkezésre állás terén. Folyamatosan működniük kell, függetlenül bármilyen eszköz- vagy programhibától, ugyanakkor lehetővé kell tenniük, hogy hiba esetén a rendszerfelügyelők a szolgáltatás megszakítása nélkül cserélhessék ki a meghibásodott eszközöket vagy programelemeket. Ezért biztosítani kell a lehető legnagyobb megbízhatóságot és a lehető legmagasabb rendelkezésre állást, amire gyakran csak mint öt-kilences megbízhatósági szintre hivatkoznak (azaz: 99,999%-ban üzemképes).
Az ehhez hasonló távközlési programokat úgy fejlesztették ki, hogy a frissítést a szolgáltatás szüneteltetése nélkül lehetővé tegyék, emellett hibatűrők és tartalék hálózati kapcsolatokkal rendelkeznek, hogy szerencsétlenségek – akár földrengés – esetén is megfelelően működjenek.
Bár a rendszer tervezésekor mindent elkövetnek, hogy mindenféle esetlegesen felmerülő hibát megakadályozzanak,

Követelmények a naplózó fájlrendszerekkel szemben

- Gyors és megbízható hiba-helyreállítási képesség: a kiegészítő adatok helyreállítása (visszaforgatásukkal vagy újbóli alkalmazásukkal), miután a fájlrendszert egy hibás leállást követően befűzzük. A helyreállításhoz szükséges idő nem növekedhet a lemez felosztásának méretével arányosan.
- Legyen méretezhető: a fájlrendszernek méretezhetőnek kell lennie, vagyis támogatnia kell a nagy lemezrész- és fájlméreteket egyaránt (beleértve a véletlenszerű hozzáférést), valamint a nagyszámú fájlokat.
- Nagy teljesítmény biztosítása: a naplózási képesség hozzáadása nem bírhat jelentős hatással az alapvető műveletekre, se az olvasásra (beleértve a véletlenszerű hozzáférést), se az írásra.
- Befejezés: miután egy művelet sikeresen befejeződött, a műveletet nem lehet visszaforgatni (vagyis a tranzakció befejezettnek tekintendő, miután a vezérlés visszakerült a felhasználóhoz).
- Megfelelő eszközök: fájlrendszer létrehozása, a fájlrendszer helyreállítása rendszerleállás után, mentés (dump) és a fájlrendszer visszaállítása mentésből, töredezettségmentesítés (néhány fájlrendszer esetén ez az eszköz szükségtelen, mivel a fájlrendszer belső folyamatai végzik a töredezettségmentesítést).
- Rugalmasság: lehetővé kell tenni a fájlrendszer átméretezhetőségét akár különböző fizikai lemezrészeken keresztül is, miközben a fájlrendszer használatban van (az LVM felületének felhasználásával).
- Megbízható műveletvégzés: a fájlrendszer működésének minden körülmények között megbízhatónak kell lennie.

mindig van egy picinyke esély, hogy egy processzor (vagy egy kiszolgáló csomópont) téveszteni fog. Ebben a nagyon kivételes esetben képesnek kell lennünk újraindítani és újra üzemképes állapotba hozni ezt a részt, hogy amilyen gyorsan csak lehet, a lehető legkisebb kieséssel újra ki tudja szolgálni a kéréseket.
A naplózó fájlrendszerek iránti érdeklődésünk a távközlési Linux-rendszereken annak köszönhető, hogy az ilyen fájlrendszerek képesek a gyors újraindulásra. Egy esetleges meghibásodás esetén a naplózó fájlrendszer a fájlrendszert a lehető leggyorsabban képes újra összefüggővé tenni, ezáltal jóval megbízhatóbb, mint az egyéb fájlrendszertípusok.
Az IBM JFS-sel való kísérletezést valamikor 2000 elején kezdtük el. A JFS-csapat nagyon segítőkész volt, és képviselőjük, ***Steve Labs*** 2001 januárjában meglátogatta a laboratóriumunkat. Azóta szorosan követjük a JFS fejlesztését és kiszolgálóink mindig a legfrissebb változatot futtatják.
Első JFS-telepítéseinknek egy 1U lemeztömb adott otthont, melyben egy 500 MHz-es Celeron processzor dolgozott, 256 MB RAM-mal és 20 GB-os IDE-lemezekkel. Ezek az eszközök biztosították számunkra a munkakörnyezetet, amelyben a JFS-t kipróbálhattuk, és alkalmazásaink használata során tapasztalatokra tehettünk szert. Amióta a JFS 1.0.0-s változata 2001 júniusában megjelent, úgy döntöttünk, hogy kísérleti Linux-rendszerünkre is JFS-t telepítünk (*2. kép*).
Linux-rendszereink úgy lettek megtervezve, hogy rövidtranzakció-alapú kéréseket szolgáljanak ki. A JFS biztosít egy napló-alapú, bájtszintű fájlrendszert, amellyel elsősorban a tranzakció-középpontú rendszereket célozzák meg, ez illik a mi rendszerünkre is a legjobban.
Távközlési szempontból a JFS előnye, hogy a fájlrendszert nagyon összefüggő szerkezeti egységekben tárolja, könnyen helyreállítható, és sokkal gyorsabban képes újraindulni, mint nem naplózó társai, a hagyományos Unix-fájlrendszerek.
A rendszer meghibásodása esetén a többi fájlrendszer rendszerint könnyen megsérül. Az ilyen fájlrendszerek egy helyreállító eszközt tesznek szükségessé, mint amilyen az `fsck` is. Az ilyen helyreállító eszközök végigfésülik a fájlrendszer kiegészítő adatait, és kijavítanak minden hibát, amire a fájlrendszer szerkezetében rábukkannak. Ez a művelet azonban rendkívül időpazarló tevékenység, amelynek során könnyedén hiba léphet fel, és a legszerencsétlenebb esetben rossz helyre vagy rosszul állít vissza adatokat. A távközlési rendszerek nem engedhetnek meg maguknak olyan műveleteket, amelyek tovább növelik a kiesést.
A JFS-sel meghibásodás esetén a fájlrendszer egy összefüggő állapotba kerül vissza, úgy, hogy a fájlrendszer a megfelelő tranzakciókhoz megismétli a naplóban rögzült műveleteket. Az ilyen naplóalapú rendszereknél a helyreállításhoz szükséges idő lényegesen kevesebb, mivel a helyreállítás során nem kell az összes bejegyzést átfésülni, hanem csakis azokat, amelyek a leállás során működőek voltak.

Összegzés

A JFS-nek kulcsszerepe van, mivel egy esetleges rendszerleállást követően nagyon gyors fájlrendszer-újraindítást eredményez. A JFS-csapat elsődleges és legfontosabb célja, hogy megbízható, magas rendelkezésre állású fájlrendszert hozzon létre. A JFS-csapatnak nagy szerepe van a JFS Linuxra való átültetésében. A teljesítmény szempontjából sok különféle mérést alapul véve a JFS megint csak győztesen kerül ki. A JFS Project honlapjáról a developerWorksön további érdekességeket tudhatsz meg.

Köszönetnyilvánítás

Köszönet az Ericsson Research Open Systems Lab csapatának, hogy támogatták a Linuxszal és nyílt forrású rendszerekkel folytatott munkánkat.

A cikkhez kapcsolódó anyagok a 45. CD Magazin/JFS könyvtárában találhatóak.

Linux Journal 2003. január, 103. szám

Steve Best (sbest@us.imb.com)
Austinban (Texas), az IBM Linux Technology Centerben dolgozik. Jelenleg a Linux naplózó fájlrendszer projekten munkálkodik.

David Gordon (gordd00@dmi.usherb.ca)
A Québecben lévő Sherbrook Egyetem Számítástechnikai tanszékének hallgatója, épp most készül megszerezni a diplomáját. Az Ericsson Research Labjében segédfelügyelőként dolgozik.

Ibrahim Haddad (Ibrahim.Haddad@Ericsson)
Az Ericsson Corporate Unit of Research kutatója Montrealban. Harmadik nemezedékbeli vezeték nélküli IP-hálózatok felépítésével foglalkozik. Ibrahim képviseli az Ericssont az OSDL (Open Source Development Lab) műszaki alcsoportjában. Jelen pillanatban a Concordia Egyetem DrSc jelöltje.

Linux új méretarányban: az SGI Altix 3000 rendszer

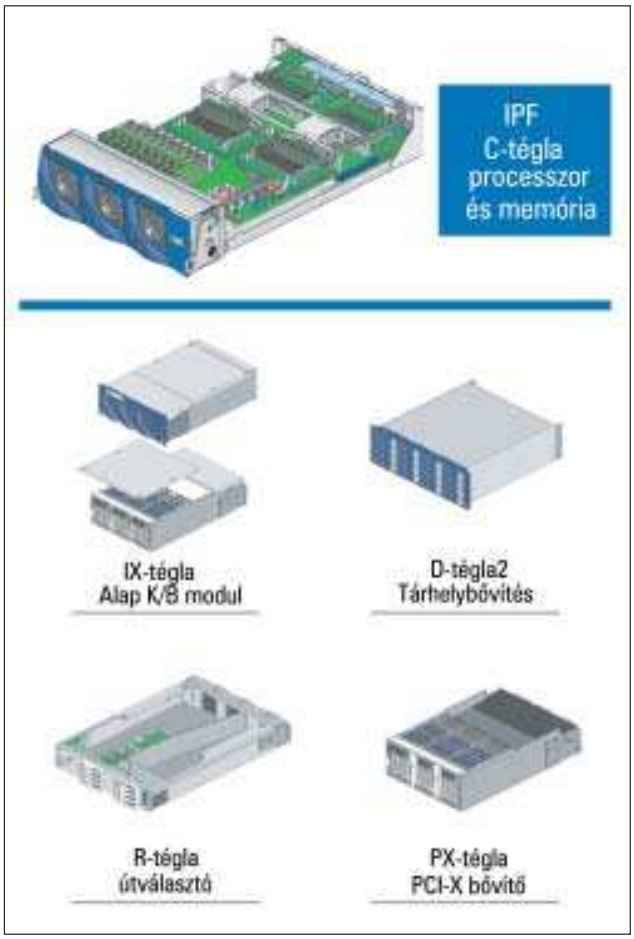
64 processzorával és az 512 GB memóriájával az SGI igényt tart a világ legerősebb Linux-rendszerének a címére.

Az SGI nemrégiben lépett színre új 64-bites, 64 processzoros, Intel Itanium 2 lapkákon alapuló Linux rendszerével – ami jelentős lépés mind a cég, mind a Linux számára. Ez a rendszer új távlatokat nyit, hiszen az összetett és igényes magas teljesítményigényű számításokon (High-Performance Computing, azaz HPC) dolgozó tudósok olyan körülmények között használhatnak és telepíthetnek Linuxot, amelyre ez idáig még nem volt példa. A HPC környezetek mindig az operációs rendszerek végső határait feszegetik, folyton több központi egységet, magasabb K/B sávszélességet és gyorsabb, hatékonyabb párhuzamos programozási támogatást követelve. A rendszer fejlesztésének korai szakaszában az SGI úgy döntött, hogy a Linuxot választja új felületének kizárólagos operációs rendszeréül, mivel bizonyítottan erős és megfelelő operációs rendszer az SGI által megcélzott számítási környezetekhez. A SGI NUMAflex globálisosztottmemória-rendszerével, az Intel Itanium 2 központi egységekkel és a Linux használatával már jóval a rendszer tényleges bemutatása előtt megdöntött minden rekordot.

Az új rendszer, amely az SGI Altix 3000 nevet kapta, legfeljebb 64 processzorral és 512 GB memóriával rendelkezik. A következő változatok azonban már 512 processzort és 4 TB-ot kínálnak majd. Ebben a cikkben az új SGI-rendszer mögött rejtőző alkatrésztervezést fedezzük fel, leírjuk, hogy milyen programfejlesztéseket kellett végrehajtani, hogy az új rendszert kivihessük a piacra, illetve megmutatjuk, milyen készségesen méretezhető és alkalmazható a Linux a legigényesebb HPC-környezetekben is.

Alkatrészháttér és rendszerfelépítés

Az SGI Altix 3000 rendszer Intel Itanium 2 processzorokat használ, és az SGI NUMAflex memóriakezelési szerkezetén alapul, ami a nem egységes memóriaelérés- (Non-Uniform Memory Access – NUMA) szerkezet SGI-féle megvalósítása. A NUMAflex 1996-ban mutatkozott be, és azóta használják a cég megújított SGI Origin kiszolgálócsaládjában, illetve a MIPS központi egységen alapuló szuperszámítógépeiben, valamint az Irix 64-bites operációs rendszerben. A NUMAflex-tervezés lehetővé teszi, hogy a processzort, memóriát, K/B rendszert, a kapcsolatokat, a grafikát és a tárakat moduláris alkotórészekbe, úgynevezett téglákba csomagoljuk. Ezek a téglák hihetetlen rugalmassággal kombinálhatók és állíthatók be, hogy az eredmény a vásárló erőforrás- és munkaterhelési igényeinek mind jobban megfelelhessen. Ezt a harmadik nemzedékbeli tervezést módosítva az SGI ilyen téglák haszlatával képes volt felépíteni az SGI Altix 3000 rendszert a Ki/Bemenet (IX- és PX-téglák), a tárhely (D-téglák) és a kapcsolatok (útválasztó téglák/R-téglák) részekhez. Az új rendszer fő eltérése a processzortégla (C-tégla), amely az Itanium 2 processzorokat tartalmazza. Az SGI Altrix 3000 rendszerében alkalmazott téglatípusokat az *1. ábra* mutatja be. A 2. *ábra* azt írja le, miként lehet ezekből a téglákból két keretet összeállítani, létrehozva egy egységes rendszerlenyomatú 64 processzoros rendszert (single-system-image 64-processor system).

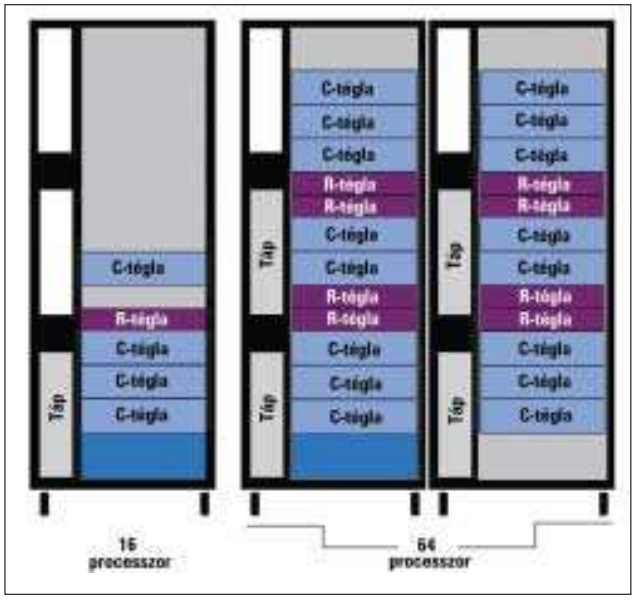


1. ábra NUMAflex-téglatípusok

A Linux felkészítése az új alkatrészkészletre

Egy olyan jól megtervezett és kiegyensúlyozott alkatrészrendszeren, mint a NUMAflex, az operációs rendszernek kell gondoskodnia arról, hogy a felhasználók és az alkalmazások az alkatrészeket teljes mértékben kihasználhassák anélkül, hogy közben a pocsékoló erőforrás-kezelés vagy a valahol egy szűk keresztmetszet hátráltatná őket. Hogy a nagy NUMA-rendszeren kiegyensúlyozott alkatrészerőforrás-kezelő rendszert tudjunk létrehozni, a rendszermagfejlesztést jóval azelőtt meg kellett kezdenünk, hogy az első Itanium 2 lapkák és alkatrészprototípus-rendszerek megérkeztek volna. Jelen esetben felhasználtuk az Itanium lapkák első nemzedékét, hogy a keresett HPC-környezethez szükséges processzorméretezést, a K/B teljesítménynövelést és az egyéb változtatásokat a Linux rendszeren elvégezhessük.

A program előkészítésének első lépése, még mielőtt az alkatrész-prototípusok megérkeztek volna, annak a lehető legpon-



2. ábra Két lehetséges NUMAflex-összeállítás

tosabb megállapítása volt, hogy milyen alacsony változásokat kell a rendszermag alacsony szintű (regiszterek és alkatrészek szintjén) kódjában eszközölni, hogy elinduljon és megbízhatóan fusson. és futtatásához. Azok a rendszerkészítők, akik saját, különösen fejlett rendszerekhez szánt ASIC tervezésébe fognak, általában szimulációs programokat és eszközöket használnak alkatrészterveik kipróbálásához. Mielőtt még a vasat megkaptuk volna, kifejlesztettünk és széles körben használtunk szimulátorokat a rendszerprogramokhoz (firmware) és a rendszermag fejlesztéséhez egyaránt, hogy segítségükkel elkészíthessük a rendszerszintű programokat.

Amikor az első nemzedékbeli Itanium processzorokkal ellátott eredeti alkatrész-prototípus megérkezett, eljött az üzembe helyezés ideje. Az egyik legfontosabb mérföldkő a rendszer első bekapcsolása, a processzor újraindítása (reset), majd az első utasítások PROM-ból történő kiemelése és végrehajtása volt. Az indítás után az alkatrészfejlesztési laborban hosszú órákon és hétvégeken keresztül tartott az igazi móka. Ez volt az a labor, ahol az alkatrészeket, a kipróbálást végző és a felületet tervező mérnökök szorosan együttműködtek egymással a rendszer hibaellenőrzésében, keresztülsegítve a processzort számos lényeges állomáson: eljutottak az PROM-tól az indítási promptig, a Linux-rendszermag futtatásától a felállásig, továbbá túljutottak a gyökérfájlrendszer olvasásán és befűzésén, az egyfelhasználós mód elérésén, majd a többfelhasználós módba lépésen, végül a hálózati csatlakozáson. Ezt követően ugyanezt több processzorral és több csomóponttal – többnyire párhuzamosan haladva – néhány, más állomásokon dolgozó felhozó csapattal is végigcsináltuk, akik szorosan követték a vezető-csapat fejlődését.

Miután az első nemzedékbeli Itanium processzoros prototípus-rendszereken sikerült a Linuxot futásra bírni, a programmérnökök nekiláthattak a munkának, immár tudva, hogy a Linux fut, és ami talán még fontosabb: jól méretezhető a NUMA-rendszereken. Számos házon belüli, első nemzedékbeli Itanium alapú rendszert építettünk fel és használtunk, így meggyőződhettünk róla, hogy a Linux a nagyrendszereken tényleg jól teljesít. 2001 elején kategóriájában elsőként sikeresen futtattunk egy 32-processzoros Itanium alapú rendszert.

1. kép Az alkatrészeket tervező mérnök, a PROM-ot megalkotó mérnök és a rendszermérnök megvitatnak egy hibát

2. kép A szerző fia egy korai 32 processzoros Itanium alapú rendszer előtt 2001 nyarán

3. kép Az Itanium 2 alapú C-tégla első indulása

Ezek az első nemzedékbeli Itanium alapú rendszerek kulcsfontosságúak voltak, mert a segítségükkel tudtuk a Linuxot az igényes HPC-követelményeknek megfelelőre formálni. Így már jóval azelőtt, hogy az Intelnél az első Itanium 2 processzorok

Feladatmegoldások valós környezetben

Következő példáinkban három tudományos HPC-alkalmazás teljesítményét mutatjuk be Linuxot futtató SGI Altix 3000 rendszeren. Három példarendszerünk a bioinformatikához szánt FASTA, a számítógépes kémiai feladatokra tervezett Gaussian és a folyadékdinamikai modellezésre használt STAR-CD lesz. Az összes próbát az SGI vezette.

FASTA bioinformatikai példa

Bár a biokémia és a számítógépes biológia már a 1980-as évek közepétől kezdve létezik, a bioinformatika viszonylag fiatal tudományága, amelyet az adatigényes élettudományok és a laboratóriumautomatizálási technológiák közeledése, illetve a hatalmas adatmennyiséget gyorsan szervező, feldolgozó és szétosztó számítógépes adatbázisok és algoritmusok megjelenése tett megvalósíthatóvá. A bioinformatika segítségével az új gyógyszerek hamarabb kerülhetnek piacra, megakadályozhatjuk a genetikai fertőzéseket, fertőzés- vagy szárazságtűrő kukoricát hozhatunk létre, meghosszabbíthatjuk az ételek szavatosságát, választási lehetőségünk lesz az olajkérdésre vonatkozóan, és létrehozhatjuk a jövő ételeit, amelyek segítenek majd a koleszterinszint szabályozásában és megelőzik a rákot. A gyorsan növekvő nyilvánosan elérhető biológiai adatok mennyisége miatt a szekvencia-adatbáziskeresés a bioinformatika egyik legkényesebb területe. A Smith–Waterman-féle (*T. F. Smith* és *M. S. Waterman*, 1981, Journal of Molecular Biology 147: 195–197) klasszikus keresési módszerek nyújtják a biológiai adatbázisok leghatékonyabb módszerét a szekvenciahasonlóságok kereséséhez. Csakhogy a Smith–Waterman–módszer elég számításigényes, így az eljárás hatékony felhasználásához különösen fontos a párhuzamosítás. Az egyik ilyen párhuzamosított Smith–Waterman-megoldás a FASTA bioinformatikai csomag (W. R. Pearson, 1991, Genomics 11: 635–650). A Smith–Waterman-algoritmus FASTA 3.4 változatát (ssearch34_t) – ami egyébként a Virginiai Egyetem lapján (alpha10.bioch.virginia.edu/fasta) fellelhető – használtuk 64 processzoros SGI Altix 3000 rendszerünk párhuzamos teljesítményének a mérésére.
A Smith–Watermant P-szálakkal (Pthread) párhuzamosítottuk, az SGI ChemBio Applications csapat pedig a magalgoritmust tovább finomhangolta, hogy az még jobban kihasználhassa az SGI Altix 3000 rendszer képességeit. Végeredményül a Smith–Waterman-algoritmus közel eszményi méretezhetőséget mutatott (a tökéletes méretezhetőséget a pontozott vonal jelzi), 64 processzoron futtatva közelítőleg 59×-es sebességnövekedést értünk el.

I. ábra
A FASTA teljesítménye csaknem lineáris

Gaussian számítógépes kémiai példa

Nemzeti kutatólaboratóriumok, egyetemek, gyógyszeripari és biotechnológiai cégek és vegyészeti vállalatok is használnak a Gaussian Inc. (➲ http://www.gaussian.com) Gaussian 98 rendszeréhez hasonló számítógépes kémiai alkalmazásokat a molekuláris energiák, tulajdonságok és reakciók kutatásában, elektronikus szerkezetekkel igen nagy molekularendszereket modellezve.

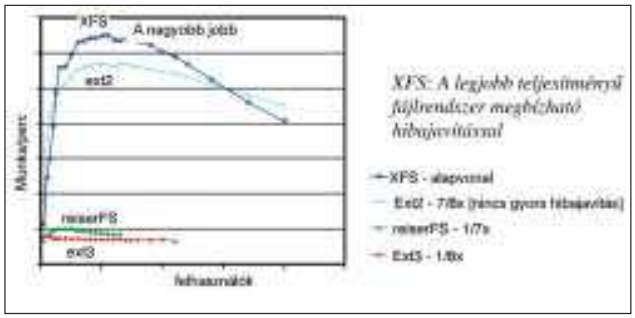
II. ábra
Gaussian 98 eredmények

A *II. ábrán* a Gaussian 98 méretezhetőségére vonatkozó eredményeket figyelhetjük meg, amelyeket a Gaussian QA-suite nevű, széles körben alkalmazott teszttel készítettünk, egy korai SGI Altix 3000 prototípusrendszeren az inkább Itanium, semmint Itanium 2 processzorokhoz szánt Intel-fordító korai változatát használva. Az itt látható eset a Valynomycin molekulának (C54H90N6O18) a sűrűségalapú elmélet (Density Functional Theory) szerinti erőszámításait mutatja be. A grafikonon megfigyelhetjük a tesztben kialakult párhuzamos sebességnövekedést. Az eltelt időt másodpercben adtuk meg. Az eszközök és a rendszer korai változatai ellenére a számításhoz szükséges idő az SGI Altix 3000 rendszer húsz processzorának felhasználásával körülbelül 4,5 óráról mindössze 25 percre csökkent.

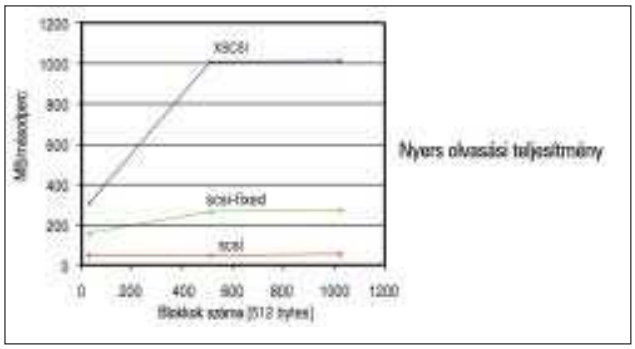
STAR-CD számítógépes folyadékdinamikai példa

A számítógépes folyadékdinamikát (Computational Fluid Dynamics, azaz CFD) számos hagyományos ipari ágazat is használja, többek közt az autóipar, az űrkutatás és az energiatermelési ágazat. A Computational Dynamics Limited (➲ http://www.cd-adapco.com) STAR-CD programja a folyadékáramlás témakörében a CFD-módszer egyik vezéralakja. A CFD-felhasználót végigsegíti a kezdeti alapvető tervezéstől kezdve a valós értékekkel bíró tanulmányokon át a hatékonnyá tételig, felajánlva fejlett fizikai modelljét, illetve azt a képességét, hogy strukturálatlan hálókkal összetett geometriai formákat is kezelni képes. A STAR-CD minden vezető Unix- és NT-felületen üzemel. Párhuzamosított változata, a STAR-HPC osztott memóriájú kiszolgálókon, erősen párhuzamosított rendszereken és munkaállomások telepein fut. A STAR-HPC az MPI könyvtárat használja a magas szintű méretezhetőség eléréséhez. SGI Altix 3000 rendszeren a STAR-CD előzetes kiadását és az autó áramlástani modelljéhez szánt „Model A Class Dataset" felhasználását futtatva az előzetes teljesítménypróbák alapján a Linux ismét bizonyította kitűnő processzorméretezhetőségét – egészen 64 processzorig. A 2002 novemberében a ➲ http://www.cd-adapco.com/support/bench/aclass.htm lapon közölt adatok szerint a Linux jobban méretezhető, mint két másik kereskedelmi rendszer: a Hewlett Packard HP-UX és az IBM AIX rendszere.
A *III. ábrán* a STAR-CD összehasonlító eredményeit láthatjuk a Linux, a HP-UX és az AIX rendszerek között.

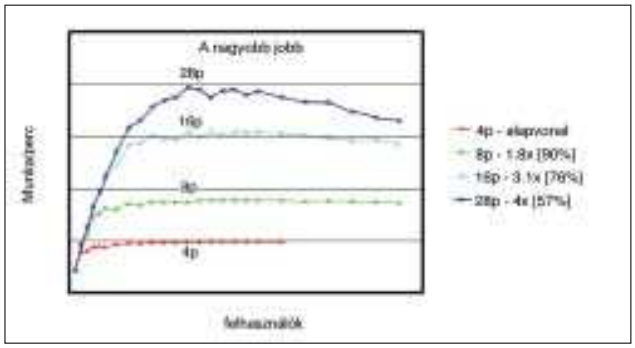
III. ábra
STAR-CD eredmények Linux,
HP-UX és AIX összehasonlításban



3. ábra Fájlrendszerteljesítmény-összehasonlítás: AIM7 többfelhasználós rendszermag terhelése, 2.4.18-as rendszermag 28 P Itanium mintapéldánnyal, 14GB, 120 lemez; különböző fájlrendszerek, az SGI álltal kiegészített és finomhangolt rendszermaggal



4. ábra Linux XSCSI teljesítménypélda: a 2.4.16-os rendszermaggal; 120 folyamat 120 merevlemezről olvas



5. ábra Processzorskálázási példa az AIM7-el: AIM7 többfelhasználós rendszermag terhelése, 2.4.16-os rendszermag; SGI-kiegészítések és finomhangolás a rendszermagban

elérhetővé váltak volna, már fejleszteni lehetett és ki lehetett próbálni a méretezést, a K/B teljesítményt és az egyéb változtatásokat.
Miközben az SGI-programmérnökök a teljesítményen, a méretezésen és más feladatokon dolgoztak az első nemzedékbeli Itanium processzorokkal felszerelt prototípusokon, az alkatrészeket tervező mérnökökből és felületet késztő mérnökökből álló másik csapat felkészítette az indításra a következő nemzedékbeli Itanium 2 processzoros SGI C-téglát, elölről megismételve a teljes fejlesztési folyamatot.
2002 közepére a fejlesztő csapat nagyszerű fejlődést mutatott: egyetlen processzor indításától eljutottak a 64 processzoros rendszer működéséig. Az Itanium 2 lapkákkal felszerelt 64 processzoros rendszer ismét úttörőnek számított a maga nemében. Mindezek természetesen egyetlen rendszerlenyomaton lévő (single system image) Linuxon futottak.
Az egész folyamat alatt minden változtatást és hibát, amit csak a Linuxban találtunk, visszaküldtünk a rendszermagfejlesztőknek, hogy a ja vításokat későbbi Linux-terjesztésekbe már beletehessék.

Közelkép a Nagy Vasról

Más Linux-fejlesztők gyakran kérdezik: „Miféle változtatásokat kellett alkalmaznotok a Linuxon, hogy egy ilyen méretű rendszeren fusson?” vagy „A Linux-processzor méretezhetősége nincs nyolc vagy hasonló számú processzorra korlátozva?”. Ahhoz, hogy válaszolhassunk ezekre a kérdésekre, előbb meg kell vizsgálnunk, mit használ az SGI programalapnak, milyen kitűnő változtatásokat végzett el a közösség, és hogy milyen más HPC-vel kapcsolatos fejlesztéseket és eszközöket adott az SGI, hogy a Linux messze túlszárnyalja az ismert nyolcprocesszoros határt.
Az SGI Altix 3000 rendszerek rendszerprogramja az Itanium processzorokhoz szánt szabvány Linux-terjesztést és a Linuxot további képességekkel felruházó SGI ProPack bővítményt tartalmazza. Az SGI ProPack termék egy újabb 2.4 alapú Linux-rendszermagot, HPC könyvtárakat, amelyek az SGI alkatrészeinek legjobb kihasználására vannak kiélezve, valamint NUMA-eszközöket és meghajtókat tartalmaz.
Az SGI Altix 3000 rendszeren használt 2.4 alapú Linux-rendszermag az Itanium processzorokhoz szánt szabványos 2.4.19-es rendszermagot (kernel.org), illetve néhány továbbfejlesztést tartalmaz. Ezek a továbbfejlesztések három osztályba sorolhatók: általános hibajavítások és felülettámogatás, fejlesztések a Linux-közösség más munkáiból, végül SGI-változtatások.
A rendszermag-változtatások első csoportjába a kipróbálás alatt talált hibák javításai tartoznak, illetve az alapot képező felület továbbfejlesztései, valamint a NUMA-támogatás. Ezeket a változtatásokat az SGI a rendszermagot fejlesztő csapat megfelelő karbantartójával együttműködve végezte, hogy e módosítások visszakerülhessenek a rendszermag főáramába.
A rendszermagfejlesztések második csoportjába azok a kitűnő munkák és teljesítményfoltok kerültek, amelyeket a közösségből mások fejlesztettek ki, de hivatalosan még nem fogadtak el, vagy átütemeztek a 2.5 fejlesztői vonalra. Ezek a fejlesztések a következő VA Software SourceForge lapokon találhatók meg: „Linux on Large Systems Foundry” (large.foundries.sourceforge.net) és a „Linux Scalability Effort Project” (sourceforge.net/projects/lse). Mi a projektekből a következő foltokat használtuk: a processzorütemezőt, a nagy rendszermagzár-felhasználást (Big Kernel Lock) csökkentő javítást, a Read–Copy–Update (olvass-másolj-frissíts) spinlock elven alapuló `dcache_lock-usage` csökkentésjavítást, illetve az FRlock zárolási elven működő `xtime_lock` (gettimeofday) felhasználáscsökkentő fejlesztést.
A Linux eszközkezelő fájlrendszerét (`devfs` ➲ http://www.atnf.csiro.au/people/rgooch/linux/docs/devfs.html) is beállítottuk és használjuk, hogy a rendszerünkön elérhető rengeteg lemezt és K/B sínt kezelhessünk. A `devfs` biztosítja számunkra, hogy az eszközelérési utak újraindítás után is megmaradjanak, mégha időközben lemezeket vagy vezérlőket helyeztünk be vagy távolítottunk is el. Az egyik legkellemetlenebb dolog, amivel a nagy rendszerek gazdái találkozhatnak, ha az egyik rendszer tönkremegy, és hirtelen ötven, esetleg még több lemez átszámozódik és átneveződik. A `devfs` megbízhatónak bizonyult olyan különlegesen igénybe vett rend-

szerkörnyezetekben is, amelyekben akár 64 processzorral és rostcsatornák (Fibre Channel) tucatjaival rendelkező összeállítások működnek együtt százszámra befűzött lemezekkel.
A `devfs` kiegészítő része a 2.4-es Linux-rendszermagnak, így további foltra nem volt szükség.
A rendszermag-változtatások harmadik csoportjába kerültek az SGI által végzett módosítások, amelyeknek a Linux fővonalába történő illesztése jelenleg is folyik, s a 2.4-es változatkövetőkben kapnak helyet, vagy a folt különleges felhasználási területe vagy természete miatt elkülönítve maradnak. Ezeket a nyílt forrású fejlesztéseket az „Open Source at SGI" honlapon találjuk (➲ http://oss.sgi.com). Az általunk elvégzett módosítások a következők voltak: az XFS fájlrendszer-program, folyamat-aggregátumok (Process AGGregates, PAGG), CpuMemSets (CMS), rendszermag-nyomkövető (kdb) és a Linux rendszermag összeomláslistázó (Linux kernel crash dump, azaz lkcd). Ezen felül az SGI beépítette az Irix alól áthozott SCSI alrendszerét és vezérlőit. A Linux 2.4 SCSI K/B alrendszerrel végzett első kipróbálások megmutatták, hogy a terület komolyabb fejlesztése nélkül nem tudjuk vásárlóink jelentős tárigényeit kielégíteni. Bár a fővonalbeli rendszermagfejlesztők a későbbi kiadásokhoz már dolgoznak ezen a feladaton, az SGI-nak azonnali megoldásra volt szüksége a 2.4 alapú rendszermagokhoz, így az Irixről áthozott SGI XSCSI hátteret és meghajtókat használtuk ideiglenes megoldásként.
A *3–5. ábra* azt mutatja be, milyen kezdeti javulást értünk el az SGI Altix 3000 rendszeren futó Linux alatt a fent említett változtatásokat követően. A *3. ábra* az XFS fájlrendszert hasonlítja a többi Linux-fájlrendszerhez. (Megjegyzés: ha a Linux-fájlrendszerek teljesítményét összehasonlító részletesebb írást keresünk, nézzük meg a 2002-es Usenix Annual Technical Conference „Filesystem Performance and Scalability in Linux 2.4.17" című cikket, amely az ➲ http://oss.sgi.com lapról szintén elérhető). A *4. ábra* az XSCSI-t hasonlítja 2.4-es Linux SCSI rendszeréhez, végül a *5. ábra* az AIM7-el elérhető processzor méretezhetőségét szemlélteti.
Bár az SGI inkább a nagyteljesítményű, illetve műszaki számítási környezetekre összpontosít – ahol a processzorciklusok többsége általában a felhasználói szintű kódra és alkalmazásokra fordítódik, nem a rendszermagra – az AIM7 teljesítménypróba megmutatta, hogy a Linux a főként az üzleti környezetekre jellemző, más típusú terhelés alatt is jól méretezhető. A HPC-alkalmazások linuxos teljesítmény- és méretezhetőségi példáit a széljegyzetben „Feladatmeghatározások valós környezetben" címmel olvashatjuk.
A Stream Triad teljesítménypróba segítségével az SGI megmutatta, hogy kettőtől 64 processzorig csaknem lineáris méretezhetőség érhető el, és eljutott a másodpercenkénti 120 GB-os sebességig. Ez az eredmény jelentős mérföldkőnek számít az iparban, hiszen új világrekordot állít fel a mikroprocesszor vezérelte rendszerek világában, és ezt az eredményt egy egyetlen rendszerlenyomatot futtató Linuxon értük el! Ez a lenyűgöző eredmény egyben azt is megmutatja, hogy a Linux a megszokott nyolcprocesszoros korlátozáson felül is ténylegesen jól használható. A Stream Triadról további tájékoztatást a ➲ http://www.cs.virginia.edu/stream oldalon olvashatunk.
Ha megnézzük az SGI ProPackben felsorolt rendszermagfrissítés-listáját, a felsorolást meglepően rövidnek találjuk majd, ami ékesen bizonyítja a Linux eredeti kitűnő tervezését. Ami talán még lenyűgözőbb, hogy e javítások nagy része már benne van a 2.5-ös fejlesztői rendszermagban. Azt mondhatjuk, hogy a Linux gyorsan HPC operációs rendszerré növi ki magát.

Egyéb változtatások a HPC Linuxon

Az SGI ProPack tartalmaz pár olyan eszközt és könyvtárat, amelyekkel a nagy NUMA-rendszerek teljesítményét fokozhatjuk, ha olyan összetett feladatokat akarunk megoldani, amelyek sok processzort és memóriát használnak, vagy amikor több alkalmazás fut egy időben ugyanezen a nagyrendszeren. Linux alatt az SGI a `cpuset` és a `dplace` parancsokat használja, amelyek jól becsülhető és fejlett CPU és memóriafelhasználás-vezérlést biztosítanak a HPC-alkalmazások felett. Ezek az eszközök segítenek nekünk kimetszeni a szükségtelen folyamatokat, segítenek úgy használni a minden feladathoz a szükséges erőforrásokat, hogy ne keresztezzék egymás útjait, illetve megakadályozzák, hogy a kisebb feladatok véletlenül nagyobb mennyiségű erőforrást pazaroljanak el, mint amennyit hatékonyan használni tudnak. Ilyen módon a rendszer erőforrásait hatékonyan használjuk fel, és az eredményeket rendszeres időközönként kapjuk meg – ami két jellemzően kényes pontja a HPC-környezeteknek.
Az SGI ProPackben található SGI Message Passing Toolkit (MPT) az SGI számítógépekre optimalizált, ipari szabványú üzenetküldő könyvárat teszi elérhetővé. Az MPT-ben megtaláljuk az MPI és a SHMEM API-kat, amelyek átlátszóan helyezik üzembe és használják az SGI-alkatrészek alacsony szintű képességeit, például a gyors memóriá belüli másolásokhoz használt blokkátviteli motort (BTE), és a memóriaeszközvezérlő-kiragadó művelet (fetchop) támogatását. A fetchop-támogatás közvetlen kapcsolattartást és összehangolást tesz lehetővé több MPI folyamat között, miközben semlegesíti az operációs rendszer rendszerhívásaival kapcsolatos többletmunkát.
Az SGI ProPack NUMA-eszközök, a HPC könyvtárak és a szabványos Linux-terjesztésre épített egyéb programtámogatás együtt hatékony HPC programozási környezetet jelent a nagy számítás- és adatigényű munkaterhelésekhez. Ahhoz hasonlóan, ahogy az egyedi ASIC „ragasztólogikaként" felhasználhatóvá teszi a processzorokat, a memóriát és a K/B részeket az alkatrészeken, az SGI ProPack program ahhoz biztosítja a „ragasztó logikát", hogy a Linux operációs rendszert a nagy HPC-környezetek általános építőkövévé tegye.

Összefoglalás

Senki sem hitte volna, hogy a Linux ilyen jól és ilyen hamar méretezhetővé válik. A Linux és az SGI NUMAflex rendszerkiépítés ötvözésével, valamint az Itanium 2 processzorokkal az SGI megépítette a világ legerősebb Linux-rendszerét. Az SGI Altix 3000 rendszer piacra viteléhez rengeteg erőfeszítés kellett, és úgy véljük, ez még csak a kezdet. Az SGI által folytatott kemény szabványalapú stratégia, amit az Itanium 2 alapú rendszereken futó Linuxoknál használt, feljebb helyezi a Linux képességeit jelző lécet, miközben vásárlóinak érdekes, megalkuvások nélküli választási lehetőséget kínál a HPC-kiszolgálók és szuperszámítógépek terén. Az SGI mérnökei – így tulajdonképpen a teljes cég – tökéletesen megbízik a Linux képességeiben és folytatni szeretné a megkezdett utat, még több érdekes áttörést hozva a Linux- és a HPC-közösségnek.

Linux Journal 2003. február, 106. szám

Steve Neuner
Az utóbbi 19 évben Unix-rendszermagfejlesztéseken dolgozik. Jelenleg az SGI-nál Linux-mérnök-igazgató, és négy éve, amióta csatlakozott az SGI-hoz, Linuxon és Itanium alapú rendszereken dolgozik.

Fák a Reiser4 fájlrendszerben (1. rész)

Az új ReiserFS felépítése – fák, csomópontok, kulcsok és cikkelyek.

Az adatszervezés egyik módja az, ha fákban helyezzük el őket. Ha egy számítógépen adatokat rendezünk, rendszerint úgynevezett halmokban, vagyis csomópontokban helyezzük el őket, amelyek mindegyike tartalmazza a többi halom nevét (vagyis mutatóikat). A csomópontok némelyike mutatókkal rendelkezik, ezeket végigböngészve általában azt találjuk, hogy más hasonló csomópontokra mutatnak.
Minket elsősorban a szervezési rész érdekel, így a dolgokra akkor bukkanhatunk rá, ha keressük őket. A fa egy szervezési módszer, amely ennek megfelelő tulajdonságokkal rendelkezik. Ennek következtében a fát a következők alapján határozzuk meg:

1. A fa csomópontok sokaságából áll, melyeket a gyökércsomópont köt össze; és nulla vagy több csomóponttal rendelkezik, amelyeket részfáknak hívunk.
2. A részfák mindegyike önállóan is fát alkot.
3. A fa egyetlen pontja sem mutat a gyökércsomópontra, és a csomópontból minden egyes nem gyökércsomópontra egyetlen mutató irányul.
4. A gyökércsomópont az összes részfájára, vagyis a részfákhoz tartozó gyökércsomópontokra rendelkezik mutatóval.

Az egyetlen pontból álló, mutatók nélküli csomópontot is fának nevezzük, mivel az egy gyökércsomópont. Hasonlóképpen fának nevezzük az egymásból egyenesen kiinduló, elágazások nélküli csomópontokat is.

A meghatározás sarokpontjai

Érdekes dolog azon vitatkozni, hogy vajon a *véges* is a fák meghatározásai közé tartozik-e. Fákat többféle módon határozhatunk meg, és a legjobb meghatározás mindig a célfeladattól függ. ***Donald Knuth*** professzor is többféle meghatározást használ a fákra. Elsődleges meghatározásaként egy olyan fát mutat be, amelynek egyáltalán nincsenek mutatói, se élei, se összekötővonalai, se egyebei – egyszerűen csak csomópontokból áll. Knuth a fát véges számú csomópontok halmazaként határozza meg. Az Interneten a végtelen fákról is lehet találni írásokat. Úgy gondolom, sokkal megfelelőbb a végest mint tulajdonságot használni, semmint beleerőszakolni a meghatározásba – bár kutatásaimban csak véges fákkal foglalkozom.
Ha fákkal foglalkozunk, gyakran találkozhatunk az él fogalmával. A mutató egyirányú, ami azt jelenti, hogy a mutató követhető abból a csomópontból, ahonnan kiindul, de a folyamat visszafelé nem lehetséges. Nem lehetséges egy csomópontból visszakövetni, hogy melyik másik csomópontból hivatkoznak rá. Ennek megfelelően az él kétirányú, vagyis mindkét irányból nyomon követhető.
Az alábbiakban három különböző fameghatározást sorolunk fel, melyekben az a különös, hogy matematikai szempontból mégis egyenlők. Az általam felvázolt meghatározásnak azonban nem felelnek meg, mivel az él nem egyezik meg a mutatóval. Mindhárom meghatározás esetén legyen egyetlen él, amely ugyanazt a két csomópontot köti össze:

- Élekkel összekötött csúcsok (vagyis pontok) esetében az élek száma pontosan eggyel kevesebb, mint a csúcsok száma.
- Csúcsok élekkel összekötött halmaza, mely nem tartalmaz kört (körnek nevezzük a csúcsból önmagába visszahajló összekötővonalat).
- Csúcspontok csoportját, amelyeket élek kötnek össze, és két csúcspont csak egyetlen útvonalon érheti el egymást.

Ezekben a meghatározásokban a fáknak nincsen egységes gyökerük, és az ilyen fákat szabad fáknak hívjuk.
Az általam használt meghatározás egy gyökérrel rendelkező fát takar, nem egy szabad fát, amelyben nincsen kör sem, és pont eggyel kevesebb mutatóval rendelkezik, mint ahány csomópontja van, emellett minden csomópont bármely másikat csak egyetlen útvonalon érhet el.

Gráf vagy fa?

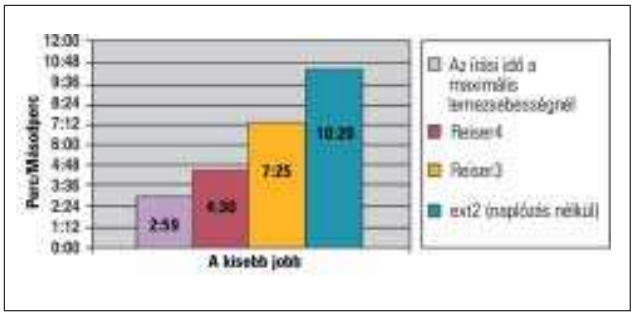
Vedd sorba azokat a feladatokat, amelyeket inkább gráf segítségével oldanál meg, és azokat, amiket fa segítségével valósítanál meg. Egy fában a fa minden egyes csomópontjához a gyökérből kiindulva csak egyetlen útvonal létezik, ezenkívül a fa minimális követelménnyel rendelkezik a mutatók számát illetően, hogy minden csomópontot össze lehessen kapcsolni. Ez teszi a fát egyszerű és hatékony szerkezetté. A fák akkor használatosak, amikor kevésbé bonyolult és hatékony szerkezetre van szükségünk, ahol nem jelent gondot, hogy minden csomópontot csak egyetlen útvonalon érhetünk el. A Reiser4 fákat és gráfokat egyaránt használ. Fákat olyankor, ha a fájlrendszer választja ki a szükséges szerkezetet (ezt hívjuk tárolási rétegnek, amelynek egyszerűnek és hatékonynak kell lennie), és gráfokat, amikor a felhasználó választja ki a szükséges szerkezetet (a tartalmi szinten, melynek sokoldalúnak kell lennie, hogy a felhasználó azt tehesse, amit csak akar).

Kulcsok

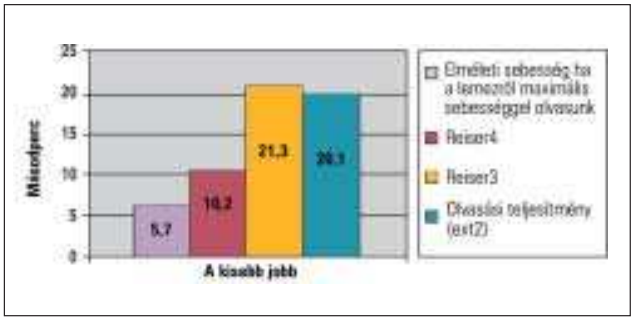
A fában minden elemhez kulcsot rendelünk, ennek alapján találjuk meg a keresett elemeket, használatuk pedig óriási rugalmasságot biztosít a dolgok rendezésekor. Ha a kulcsok rövidek, akkor kevés adat birtokában is megtalálhatjuk, amit keresünk. Azt is behatárolja egyúttal, hogy milyen adatok alapján kereshetjük meg a dolgokat.
Ez a határ korlátozza a kulcs használhatóságát, emiatt rendelkezünk egy tárolási réteggel, ami a dolgokat a kulcsok alapján keresi meg, és egy tartalmi réteggel, ami nagyon gazdag elnevezési rendszerrel rendelkezik (erre cikkünk 2. részében térünk ki). A tárolási réteg csak a tároló megszervezésére használ kulcsokat, így növelve a teljesítményt, ugyanakkor a tartalmi réteg olyan nevekkel dolgozik, amelyeknek a felhasználó számára jelentése van. Talán felteszed a kérdést, hogy vajon ez a megfelelő elrendezés-e, olyan, ami biztosítja a szabadságot, hogy újabb dolgokat adjunk a tárolási réteghez, így növelve teljesítményét – ugyanakkor a tartalmi réteg elnevezéseinek kezelésekor el kell viselnünk a mellékhatásait.

A részfa kiválasztása

A gyökérnél kezdjük a keresést, mivel innen az összes csomópont elérhető. De hogyan dönthetjük el, hogy a gyökérből elindulva melyik részfán folytassuk az utunkat? A részfákra a gyökérben található mutatók segítségével juthatunk el. Minden egyes részfára irányuló mutatóhoz tartozik egy bal oldali körülhatároló kulcs. A részfákra irányuló mutatók és maguk a részfák is baloldali körülhatároló kulcsuk szerint vannak rendezve. Egy részfa mutatójának a bal oldali körülhatároló kulcsa megegyezik a részfában található legkisebb kulccsal. Ennek a jobb oldali körülhatároló kulcsa nagyobb, mint a részfa legnagyobb kulcsa, és ez a csomópont következő alfájának a bal oldali kulcsa is egyben.



1. ábra Írási teljesítmény (30 másolata 2.4.18-as rendszermagnak)



2. ábra Olvasási teljesítmény
(a 2.4.18-as rendszermag forráskódját beolvasva)

Az egyes részfák csak olyan dolgokat tartalmaznak, amelyeknek vagy a mutatójukhoz tartozó bal oldali körülhatároló kulcsa legalább egyenlő, vagy a jobb oldali körülhatároló kulcsa legfeljebb ilyen nagyságú. Ha a részfában nincsenek többszörösen előforduló elemek, akkor minden részfa csak olyan dolgokat tartalmaz, amelynek a kulcsai kisebbek a jobb oldali körülhatároló kulcsánál. Ha nincsenek kettőződések, akkor egy csomópont részfákra irányuló mutatóinak és azok körülhatároló kulcsainak alapján egyértelműen eldönthető, hogy melyik ág tartalmazza a keresett dolgot.
A többszörösen előforduló kulcsokról egy másik alkalommal lesz szó. Most csak annyit jegyeznék meg, hogy először a részfák között a kettőzött kulcsú elem egyik kulcsát kell megtalálni, majd az ehhez tartozó párokat kell egyesével végigvizsgálni, mígnem megtaláljuk a megfelelő elemet. A kettőzött kulcsokkal kisebb méretű kulcsokhoz jutunk, így alkalomadtán egyetértésre juthatunk a kulcsok méretét és az ilyen nem hatékony keresések mennyiségét illetően. A fa minden csomópontjában a rendezés a csomópontokon belül zajlik. Ennek alapján az egész fát kulcsok szerint rendezzük, így bármely kulcs birtokában tudjuk, hol keressük a kulcshoz tartozó elemek egyikét.

Csomópontméret

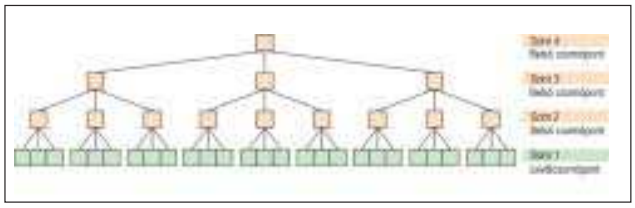
A csomópontok méretét egyformának választjuk, így könnyebb meghatározni a csomópontok közti szabad területet, mivel az a csomópont méretének valahányszorosával lesz egyenlő. Ilyen módon nem okoz gondot, ha rendelkezünk ugyan hellyel, de a rendelkezésre álló hely túlságosan kicsi egy csomópont tárolásához. Emellett a merevlemezek olyan csatolófelülettel rendelkeznek, ami feltételezi az egyenlő méretű szakaszokat, ami kényelmesen illeszkedik a hibajavító műveletekhez.
Ha nem fontos, hogy a csomópontok egyforma méretűek legyenek, mert mondjuk elférnek a RAM-ban, érdemes lehet egy pillantást vetnünk az átugró listákra.
A Reiser4 csomópontjai alapesetben a lapmérettel egyeznek meg, amely – ha Linuxot használsz Intel processzorral – 4 k (4096 bájt). Semmilyen tapasztalati bizonyíték nem támasztja alá, hogy ez a méret jobban megfelelne, mint bármely másik; az egyetlen indok, amiért mégis ezt használjuk, az az, hogy a Linux ezt teszi a legegyszerűbben programozhatóvá. Az igazat megvallva: inkább csak nem volt még időnk egyéb méretekkel kísérletezgetni.

Takarékoskodjunk a hellyel

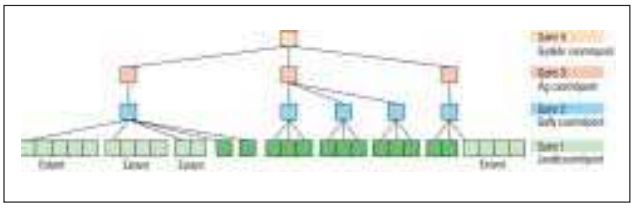
Ha a csomópontok mérete megegyezik, hogyan tároljunk nagyobb dolgokat? Darabokra vagdossuk őket: az így létrejövő darabokat cikkelyeknek (item) hívjuk. A cikkely mérete pont akkora, hogy elférjen egyetlen csomópontban.
A hétköznapi fájlrendszerek az állományokat teljes szakaszokban tárolják. Ez körülbelül annyit jelent, hogy fájlonként egy fél szakaszt elveszítünk. Ha a fájl jóval kisebb, mint a szakasz mérete, akkor a veszteség a fájl méretének sokszorosa is lehet. Ennek következtében nem ajánlatos szokványos adatbáziselemeket (címek, telefonszámok) tárolni ilyen hétköznapi fájlrendszeren, mivel az elfoglalt terület legalább 90 százaléka kárba vész. Azáltal, hogy a Reiser4-ben egyetlen csomópontban több dolgot is képesek vagyunk tárolni, az is lehetséges, hogy egy-egy szakaszt több kisméretű fájl között osszunk meg. Hatékonyságunk a kisméretű fájlok helykihasználását illetően megközelítőleg 94 százalék.
Ez a szám természetesen nem tartalmazza a fájlonkénti többlettárhelyet, ami függ az egyes fájlok méretétől, emiatt nehezen megbecsülhető.
A fájlok 4 k-s szakaszméretre igazítása azonban nagy fájlok esetén kifejezetten előnyös. Ha egy program közvetlenül szeretne egy fájllal dolgozni, tehát anélkül, hogy különféle rendszerhívásokhoz kellene folyamodnia, használhatja az `mmap()` függvényt, amellyel a fájl tartalma közvetlenül az alkalmazás címteréből válik elérhetővé. Bizonyos megvalósítási tényezőkből következően az `mmap()` megköveteli, hogy a fájl tartalma 4 k-s határra legyen igazítva. Ha az adatok már eleve 4 k-ra vannak igazítva, az nagymértékben felgyorsítja az `mmap()`-ot. A jelenlegi alapértelmezés szerint a Reiser4-ben a 16 k-nál nagyobb fájlok 4 k-s határra vannak igazítva. Jelenleg azonban még nem rendelkezünk elég tapasztalattal és adatokkal ahhoz, hogy meg tudjuk ítélni, a 16 k-s fájlméret valóban megfelelő-e.

Levelek, ágak, gallyak

Fás hasonlatunknál maradva, a levelek gyermekkel nem rendelkező csomópontok. A belső csomópontok alatt a gyermekkel rendelkező csomópontokat értjük.
A keresés a gyökérnél kezdődik, ezt követően bejár két belső csomópontot, majd egy levélen fejezi be az útját, mely egy adatokat tartalmazó csomópont, gyermekek nélkül. A cikkely



3. ábra Egy Magasság = 4; 4-szintű; Elkülönítés = 3; Kiegyensúlyozott fa



4. ábra Egy Reiser4 fa belső, vagyis ágcsomópontokkal

egy adattároló szerkezet, ami teljes egészében egy szakaszon belül helyezkedik el. Azt a csomópontot, ami cikkelyeket tartalmaz, formázott csomópontnak nevezzük. Ha dolgokat tárolunk egy fában, az egyes darabokat cikkelyekben és formázatlan levelekben tároljuk el. Formázatlan leveleknek hívjuk az olyan leveleket, amelyek semmilyen formázási adatot nem tartalmaznak, csakis adatot. Egyedül a levelek tartalmazhatnak formázatlan adatokat. A mutatók cikkelyekben tárolódnak, ennek következtében minden belső csomópont egyúttal szükségképpen formázott csomópont is.

A formázott csomópontokra irányuló mutatók különböznek a formázatlan levelekre irányulókról. A jobb megértés kedvéért vegyük a következő példát: a fokmutatók formázatlan levelekre hivatkoznak. Foknak nevezzük az egymást követő, szomszédos, szakaszon belüli vagy szám szerint egymást követő, egyazon dologhoz tartozó formázatlan leveleket. A fokmutató tartalmazza a kezdő szakasz számát és a hosszt. Mivel a fok csak egyetlen dologhoz tartozik, a fokhoz csak egyetlen kulcsot tárolhatunk, és még ezután is kiszámolhatjuk minden egyes bájtnak a kulcsát a fokon belül. Ha a fok legalább két szakasz hosszúságú, a fokmutatók sokkal tömörebbek, mint az általános csomópontmutatók.

A csomópontmutatók formázott csomópontokra irányuló mutatók. Jelen pillanatban még nem rendelkezünk a csomópontmutatók tömörített változatával, de hamarosan elkészülnek. Fontos, hogy fokmutatók esetén nem szükséges tárolni a hivatkozott csomópontok körülhatároló kulcsait, míg a csomópontmutatók használatakor ez muszáj. Valószínűleg a tömörített kulcsok a tömörített csomópontmutatókkal egyidőben kerülnek bemutatásra. Valaki talán azt várná, hogy a kulcsok tömörítve legyenek, csak azért, mert növekvő sorrendbe vannak rendezve. Szeretnénk, ha csomópont- és cikkelybővítményeink ilyen tulajdonságokkal valamikor a későbbiekben egyszerűen kiegészíthetővé válnának.

A gallyak a levelek szülői, és a fokmutatók csakis gallyakban léteznek. (Ez egy elég vitatott tervezési elmélet, amelyet a cikk következő részében még tárgyalunk.) Az ágak belső csomópontok, és nem egyeznek meg a gallyakkal.

Azt hihetnéd, hogy a gyökeret egyszerűen első szintnek is nevezhetnénk, de mivel a fa a felszínen nő, sokkal ésszerűbb első szintnek nevezni azt a részt, ahol a levelek is találhatók és az adatok tárolódnak. A fa magassága attól függ, hogy hány dolgot kell tárolnunk benne, és hogy a belső és gallycsomópontok hány gyermekkel rendelkeznek.

A cikkelyek típusai

A Reiser4 többféle típusú cikkelyt tartalmaz a különféle típusú adatok tárolásához:

`static_stat_data`: a tulajdonost, a jogosultságokat, az utolsó hozzáférés idejét, a létrehozás idejét, az utolsó módosítás idejét, a méretet és a fájlra mutató hivatkozások számát tartalmazza.

`cmpnd_dir_item`: a könyvtárbejegyzéseket és a fájlokra mutató kulcsokat tartalmazza.

Fokmutatók: lásd feljebb.

Csomópontmutatók: lásd feljebb.

Törzs: a fájlnak azon részét tartalmazza, ami nem elég nagy ahhoz, hogy egy különálló formázatlan levélben tárolódjon.

Egységek

Egységnek nevezzük azt az elemet, amelyet teljes egészében egy cikkelyben kell elhelyeznünk, anélkül, hogy több cikkely között szétdarabolnánk. Egy cikkely tartalmát gyakran egységekben egyszerűbb végigolvasni:

- A törzsben található cikkelyek esetén az egység a bájt.
- Könyvtárcikkelyek esetén az egység az egyes bejegyzéseket jelenti. A könyvtárbejegyzések tartalmazzák a fájl nevét és kulcsát (amelyek a gyakorlatban akár tömörítettek is lehetnek).
- Fokcikkelyek esetén az egység a fok. Fokcikkelyek csak az adott fájlhoz tartozó fokokat tartalmaznak.
- A `static_stat_data` számára az egész statisztikai adat egyetlen oszthatatlan, rögzített méretű mezőt képez.

Összegzés

Elmagyaráztam a Reiser4 fa alapvető szerkezeteit, de az igazán szórakoztató rész még csak most következik. Még nem magyaráztam el, hogy más kutatók hogyan alakítják a fáikat, és még azt sem tudod, hogy az elemek tartalma miért a fa alján helyezkedik el, miért szükséges a pontos elkülönítés, és miért van szükség különféle kiegyenlítőkre. Még csak nem is utaltam eddig arra, hogy miért jobbak a kiegyensúlyozott fák, és hogy miért a táncoló fa a legjobb. Amiről pedig aztán végképp nem beszéltem, az az, hogyan változik meg egy bonyolult és vitatott faszerkezet, amelyet ebben a cikkben is bemutattunk, úgy, hogy a Reiser4 kétszer gyorsabban olvasson, mint a Reiser3. Az erről szóló írás – helytől függően – a Linux Journal következő számában fog megjelenni.

Forrás

Donald E. Knuth A számítógép-programozás művészete

Linux Journal 2002. december, 104. szám

Hans Reiser (reiser@namesys.com)
Záró dolgozatát a nehéz tudományok és a számítástudomány filozófiájának különbözőségeiről írta, amelyre egy eredeti elnevezési rendszert is kifejlesztett. Ezt a rendszert még most is fejleszti, ahol is a Reiser4 képezi a tárolási réteget. 1993-ban Oroszországba utazott, hogy összeszedjen egy programozókból álló csapatot, és belefogjanak a ReiserFS fejlesztésébe. 1999 körül a rendszer olyannyira jól kezdett működni, hogy édesanyja felhagyott az unszolással, hogy a fia valamilyen jól fizető állást keressen egy nagy cégnél.

Dióhéjban az ext3 fájlrendszerről

Ha biztonságban szeretnénk tudni féltett adatainkat, használjunk valamilyen naplózó fájlrendszert, például az ext3-at.

A Linux „hagyományos” fájlrendszere az `ext2`. Ez egy jól kipróbált, megbízható rendszer, azonban akad egy gyenge pontja: érzékeny a hirtelen leállásokra. Az `ext2` a hatékonyság növelése érdekében a merevlemezt úgynevezett aszinkron módban használja. Ennek az a hátránya, hogy ha az adatátvitel akár csak egyetlen pillanatra is megszakad, fennáll az adatvesztés kockázata.

A metaadatok szerepe

Az, hogy történik-e adatvesztés, csak azon múlik, hogy mikor következett be a hirtelen leállás. Ha az adatok lemezre mentése után történik, akkor nincs is semmi baj, megúsztuk. Ha a kiírás előtt, akkor sajnos az új (még ki nem írt) adatok örökre elvesznek, de legalább a régiek megmaradnak. A legrosszabb eset az, amikor pont az írás folyamata közben következik be a sajnálatos esemény. Ekkor ugyanis könnyen előfordulhat az, hogy az állomány első fele az új, a másik fele még a régi adatokat tartalmazza. Ez rendkívül kellemetlen, de korántsem annyira, mint amikor az úgynevezett metaadatok írásakor áll le a rendszer. A metaadatok olyan adatok, amelyeknek a nyilvántartását a lemezen lévő állományokról maga a fájlrendszer végzi. Ilyen például a fájl neve, a mérete, létrehozásának a dátuma, a jogosultságok és a legfontosabb: hogy hol helyezkedik el a lemezen. Ha a metaadatok nem tudnak rendesen kiíródni, az is előfordulhat, hogy az egész fájl örökre eltűnik.

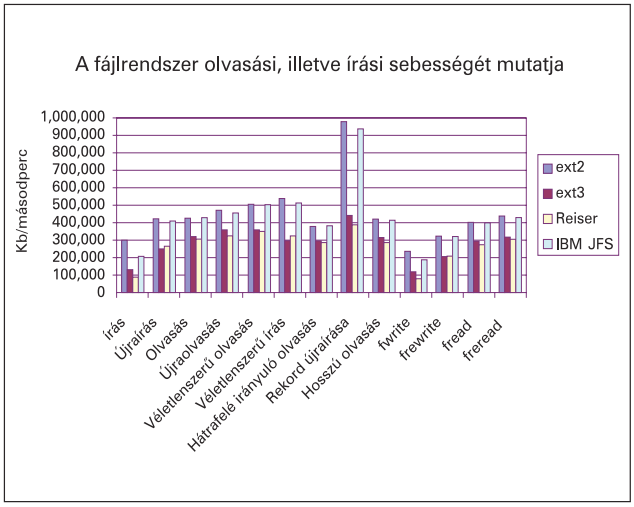
Azért nincs minden veszve, mert a legtöbb esetben (ha nem is teljes mértékben) helyreállítható a károsodás. Erre szolgál az `fsck` nevű program is, ami minden szabálytalan leállítás után a rendszer indulásakor önműködően elindul. Ez egy fájlrendszerelemző program, ami az esetleges hibákat próbálja meg felderíteni és helyreállítani a fájlrendszerben. Amennyiben a metaadatok megsérültek, általában vissza lehet állítani őket, ha volt róluk másolat. Ha nem volt, akkor a sérült állomány tartalma véglegesen elveszett.

Ezekután kétségtelen, hogy aki létfontosságú adatokat `ext2`-n tárol, az a tűzzel játszik (hacsak nem tart otthon szünetmentes tápot – bár hirtelen leállást más is okozhat, nem csak áramszünet). A másik nagy gond az `ext2`-vel az, hogy az `fsck`-val történő ellenőrzés időigényes, bizonyos esetekben több óráig is eltarthat. Ez különösen kellemetlen lehet egy hálózati kiszolgáló esetében, ugyanis amíg az ellenőrzés tart, addig nem tudja ellátni a feladatait.

A naplózó fájlrendszerek áttekintése

Mi lehet a megoldás? Használjunk úgynevezett naplózó (journaling) fájlrendszereket.

Az ezek mögött meghúzódó alapgondolat az, hogy mielőtt az írási műveletet végrehajtanánk a lemezen, egy csakis erre a célra fenntartott területre (a naplóba) beírjuk, hogy mit kellene elvégeznünk. A napló tulajdonképpen egy adatbázis, amelybe szekvenciálisan írjuk be a lemezen elvégzendő változtatásokat.



Mit nyerünk ezzel? Tegyük fel, egy hirtelen leállás történik. A rendszer újraindítása után a fájlrendszer ellenőrzi a naplót. Ha nem talál benne új bejegyzést, akkor két dolog lehetséges: vagy minden írási művelet befejeződött, vagy a leállás még az új művelet naplóba történő felvétele előtt bekövetkezett. Az utóbbi esetben az új adatok sajnos véglegesen elvesztek, de a régiek sértetlenül megmaradnak.

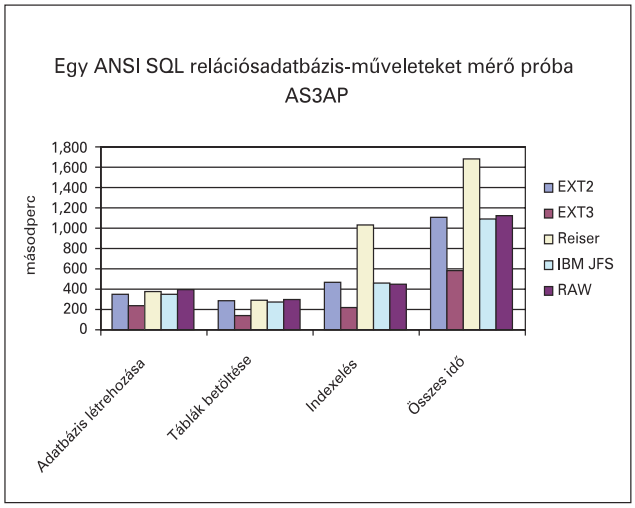
Amennyiben új bejegyzést talál a naplóban, akkor a leállás minden bizonnyal a lemezre való írás pillanatában következett be. Ilyenkor azonban nem kell számolnunk az `ext2` esetében előforduló adatvesztés lehetőségével, mivel a napló segítségével a fájlrendszer a félbe maradt műveletet be tudja fejezni. Ez az egész folyamat nem tarthat tovább pár másodpercnél, tehát az órákig tartó „fsckázásnak” is búcsút inthetünk.

A naplózó fájlrendszerek másik előnye, hogy sok esetben gyorsabbak, mint némely hagyományos fájlrendszer. Az írási kérések ugyanis rendszerint a lemez különböző helyein lévő blokkokra érkeznek, így a fejet állandóan pozícionálni kell. Ez komoly visszaesést jelent a teljesítményben. A naplózó fájlrendszerek esetében először mindig szekvenciálisan a naplóba írunk, így a fejet több írási művelet esetén is csak egyszer kell pozícionálni. A naplóban feltüntetett tranzakciókat pedig ráérünk elvégezni akkor, amikor a lemeznek nincs más feladata.

A naplózó fájlrendszerekre jellemző még, hogy nem annyira szétszórtan tárolják az állományokat, mint más fájlrendszerek. Ezenkívül különböző egyszerűsítéseket és javításokat is végeznek. Példaként említhetjük, hogy az AIX fájlrendszere például a kis könyvtárak tartalmát a fájlleíróban (i-node) tárolja.

A naplózó fájlrendszer tehát jó dolog. Nézzük, milyen kínálat áll belőlük rendelkezésre Linux alá. Használhatjuk az IBM által kifejlesztett, az OS/2 megvalósításra épülő JFS-t (lásd még a 22–25. oldalon). Ezzel a legnagyobb gond az, hogy a többihez képest viszonylag kevés segédprogram lelhető fel hozzá.

Az XFS szintén egy másik rendszerből átültetett naplózó fájlrendszer. Ez az SGI Irix nevű operációs rendszeréből származik, és számos alkalmazás támogatja. A ReiserFS egy szintén nagyon megbízható fájlrendszer, fejlesztése folyamatosan zajlik (bővebben a 34–36. oldalon olvashatunk róla).
A fenti fájlrendszerek fejlesztésével (illetve azoknak Linuxra történő átültetésével) párhuzamosan egy másik naplózó fájlrendszer is megjelent, az `ext2`-vel teljesen együttműködő `ext3` fájlrendszer.



Az `ext3` kifejlesztésére több okból is szükség volt. Először is a Linux-felhasználók többsége `ext2`-t használt, ezen tárolták több gigabájtnyira duzzadt adataikat. Ahhoz, hogy egy másik, korszerűbb fájlrendszerre térjenek át, meg kellett oldaniuk adataiknak egy másik lemezrészre történő mentését, majd az átformázás után az adatok visszamásolásának is zökkenőmentesen kellett mennie. Aki már csinált ilyet, tudja, hogy elég körülményes művelet. (Nem is beszélve arról, ha az adatok mellett a kérdéses lemezrész magának a rendszernek is az otthona.) Másrészt az `ext2` akkorra már kiforrott fájlrendszer volt, a felhasználók többsége bízott benne.
Így nem csoda, ha örömmel fogadtak egy olyan naplózó fájlrendszert, amelyik megbízhatóan együttműködik az `ext2`-vel. Ez azt jelenti, hogy a felhasználó egy egyszerű művelet segítségével – a rendszer működése közben – régi fájlrendszerét `ext3`-ra alakíthatja át, anélkül, hogy adatai ideiglenes tárolásáról gondoskodnia kellene. Sőt az `ext3` bármikor visszaalakítható `ext2`-vé.
A másik nem elhanyagolható pozitív tulajdonsága az `ext3`-nak az, hogy megszabadulhatunk az `fsck`-val történő állandó lemezellenőrizgetésektől. Egy hirtelen leállás utáni ellenőrzés, illetve az esetleges hibák kijavítása egy több gigás lemezrész esetében akár órákat is igénybe vehet. Az `ext3` használatakor ez a művelet azonban nem tarthat tovább, mint pár másodperc. Az igazsághoz hozzátartozik az is, hogy az `ext3` igazából nem egy teljesen új fájlrendszer. Valójában nem más, mint az eredeti `ext2` kibővítése a naplózó szolgáltatással. A napló itt nem más, mint egy különleges rejtett állomány a gyökérben.
Ennek a megoldásnak az az ára, hogy az `ext3` nem tartalmaz olyan lemezgyorsító szolgáltatásokat, mint a többi korszerű fájlrendszer. Elméletileg azonban így is gyorsabb az `ext2`-nél, mert hatékonyá teszi a fej mozgását. (A különböző sebességmérő próbák ez ügyben azonban nem mindig szolgálnak meggyőző bizonyítékkal.)

Átállás ext2-ről ext3-ra

A továbbiakban az `ext3`-as fájlrendszerrel fogunk foglalkozni, mivel erre pár másodperc alatt bárki „áttérhet”, és a Red Hat is ezt a fájlrendszert támogatja. Megnézzük, miképp formázhatunk egy üres lemezrészt, illetve hogyan alakíthatjuk át meglévő `ext2`-es fájlrendszerünket `ext3`-sá.
Mindenekelőtt a rendszermagnak támogatnia kell az `ext3` fájlrendszert. A 2.4.15-ös rendszermagtól kezdve minden változatban megtalálhatjuk, ha ennél régebbi rendszermagunk van, akkor külön foltot kell letöltenünk. Ezt a
➲ http://www.zip.com.au/~akpm/linux/ext3/ címről tehetjük meg. A foltot másoljuk be a rendszermag forrását tartalmazó könyvtárba, majd ugyaninnen adjuk ki a következő parancsot:

```
gunzip < ext3-2.4-x-y.gz | patch -p1
```

Az x az `ext3` folt változatszámát, az y pedig a rendszermag változatát jelöli.
Vigyázzunk arra, hogyha a gyökérlemez is `ext3`-as, akkor semmiképp se fordítsuk a támogatást modulba. A másik dolog, amire fel szeretnénk hívni a figyelmet: lehetőleg ne használjuk a rendszermagban található általános tárkorlát-támogatást (quota) az `ext3`-as fájlrendszerrel, mert rendszeromlást okozhat. Ha tárkorlátot szeretnénk használni, telepítsük az ***Alan Cox***-féle `-ac` foltokat, amivel efféle baleset elvileg nem fordulhat elő.
A rendszermagoldali támogatáson kívül szükségünk lesz még két csomagra is. Az első a `util-linux`, ami minden bizonnyal már telepítve is van. Arra figyeljünk, hogy ne legyen 2.11-nél idősebb változatú. A másik csomag az `e2fsprogs` névre hallgat, ez is megtalálható az összes terjesztésben.
Az `ext3` lemezrészt az `mke2fs` program segítségével hozhatunk létre a következő módon:

```
mke2fs -j /dev/eszk z
```

Egy meglévő `ext2`-es lemezrész átalakításához a `tune2fs`-t használhatjuk:

```
tune2fs -j /dev/eszk z
```

Ez a művelet semmiféle adatvesztést nem okozhat.
Az `ext2` lemezrész átalakítása után ne felejtsük el a */etc/fstab* állományt módosítani: a megfelelő bejegyzésnél a fájlrendszer típusa oszlopot (ez a harmadik) változtassuk `ext3`-ra.
Minden mást úgy hagyhatunk, ahogy volt. Nem árt még leállítanunk az önműködő `fsck`-ellenőrzést, mivel az `ext3` ezt nem igényli:

```
tune2fs -i 0 -c 0 /dev/eszk z
```

(a `-i` kapcsoló azt mondja meg, hogy az `fsck` milyen gyakran fusson le másodpercként. A `-c` kapcsolóval pedig azt állíthatjuk be, hogy hány befűzés után fusson le az `fsck` önműködően).
Ha valaki valamilyen okból kifolyólag a fájlrendszerét vissza szeretné állítani `ext2`-esre, a */etc/fstab* visszaírása után a fent említett módon állítsa be az `fsck`-ellenőrzések gyakoriságát.

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.

Teljes értékű levelezőrendszer

Építsünk több tartományt kezelő SMTP-levélkiszolgálót egyetlen gépen.

Bár e cikket útmutatónak szántuk, amelynek alapján a Postfix, az OpenLDAP és a Courier-IMAP segítségével felépíthetjük a saját teljes levelezőkiszolgáló rendszerünket, azzal nem foglalkozunk, hogyan választottuk ki épp ezeket az összetevőket, hiszen ennek kifejtése önmagában megérne egy cikket. Célunk, hogy egyetlen gépen állítsunk fel egy több tartományt kezelni képes SMTP-levélkiszolgálót, távoli IMAP-eléréssel.

Azt szeretnénk, ha nemcsak a héjprogram-azonosítóval rendelkező emberek számára kézbesítenének leveleket, hanem a héjprogram-azonosítóval nem rendelkező embereknek is lehetne IMAP-azonosítójuk. Így az azonosítókat két osztályba soroljuk: helyi és virtuális osztályba. A helyi azonosítók azok, amelyeknek van parancsértelmező elérésük. Ők a saját felhasználónevüket és jelszavukat használhatják az IMAP elérésére. A virtuális azonosítókhoz olyan felhasználónevet és jelszavat rendelünk, amely csak az IMAP-bejelentkezéshez használható. A cikk további részében a helyi és a virtuális fogalmat ilyen értelemben fogjuk alkalmazni.

Áttekintés

Az *1. ábra* bemutatja, hogyan kapcsolódik egymáshoz a Postfix, a Courier, a Procmail és az OpenLDAP. A helyi azonosítók a */etc/password* fájlban találhatók, az azonosítást pedig a betölthető azonosítómodulok (Pluggable Authentication Module, azaz a PAM) végzik. A virtuális azonosítók adatait az LDAP könyvtárban tároljuk. Az LDAP az azonosítókeresési és -hitelesítési lehetőségeket egyaránt támogatja. Ha szükséges, az LDAP könyvtárat ki lehet hagyni, így azonban jóval nehezebb lesz karbantartani a virtuális azonosítók adatait. Megfelelő beállításfájlokkal mind a Postfix, mind a Courier támogatja a virtuális azonosítókat, ezek azonban eltérő formátumúak.

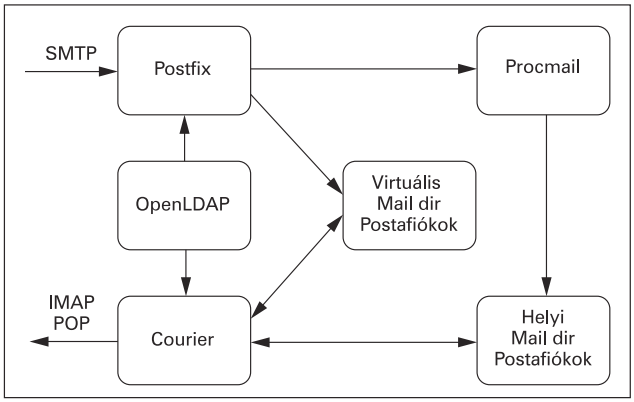
Az SMTP-ről érkező leveleket a Postfix fogadja. Az ismeretlen (helyi vagy virtuális) azonosítóról érkező leveleket elutasítja. A virtuális azonosítókhoz maga juttatja el a levelet, a helyi felhasználókhoz pedig a Procmailt használja MDA-ként.

A Courier az IMAP és a POP protokollokon keresztül távoli elérést nyújt levélládákhoz.

A levélláda helye

A helyi azonosítók leveleit Maildir formátumban a saját könyvtárukban tároljuk, a `${HOME}/Maildir/` alkönyvtár alatt. Általánosan alkalmazott megoldás, hogy a Maildir kézbesítését a */var/spool/mail* helyett az azonosító saját könyvtárába végezzük. Mind a Postfix, mind a Courier tökéletesen működik ezzel a szabványos módszerrel.

A helyi azonosítókkal ellentétben a virtuális azonosítók leveleihez nem tartozik szabványos hely. Ezért külön Unix-azonosítót készítettünk `vmail` néven, ahol az összes virtuális felhasználó leveleit elhelyezhetjük. Minden virtuális tartományhoz tartozik egy alkönyvtár a *~vmail/domains/* könyvtárban. Például ha van egy <john@pelda.hu> azonosítónk, a levelek a *~vmail/domains/pelda.hu/john/* könyvtárba kerülnek Maildir formátumban.



1. ábra Általános kiépítés

Az LDAP könyvtár megtervezése

Könyvtárunkat számtalan módon megtervezhetjük, a témát most nem elemezzük az összes lehetséges szempont szerint. Cikkünkben feltételezzük az LDAP-fogalmak és szaknyelv általános ismeretét.

A faszerkezet

Gyökérutótagként a cég tartománynevét (***myhosting.example***) választottuk. A Postfix és a Courier egyaránt az `o=hosting, dc=myhosting, dc=example` alfában keresi majd az elektronikus levél adatait. Az `o=accounts, dc=myhosting, dc=example` alfa bemutatja, hogyan tudnánk egyetlen könyvtárba beilleszteni a héjprogram-azonosítók PAM-adatait is, ez azonban a levelezéshez most nem szükséges. Minden kezelt tartomány saját szervezettel rendelkezik a hosting szervezet alatt. Minden elektronikus levélazonosító a tartományok alfájába kerül. Ennek megfelelően a <user2@domain2.example> elektronikus levélcím megkülönböztető neve:

```
mail=user2@domain2.example,o=domain2.example,
 o=hosting,dc=myhosting,dc=example
```

A fenti tervezet elég megbízható, hiszen az azonosítókat soha nem visszük át másik tartomány alá. Tervünk emellett rugalmas is, hiszen az egyes tartományfákat – amennyiben szükséges – tetszés átszabhatjuk. Minden tartományhoz tartozik egy postamester-bejegyzés, ami kettős feladatot lát el. Elsődleges célja az elérési jogosultságok szabályozása, emellett levéltovábbító elektronikus levélcímként is üzemel. Minden tartományhoz tartoznia kell egy kitiltandók listájának (abuse alias), amely a rendszergazdának továbbítja a leveleket.

Sémaválasztás

A séma mutatja meg (objektumosztályok megadásával), hogy milyen tulajdonságai (attributum) lehetnek egy bejegyzésnek. Az OpenLDAP rendszerhez szállított sémák közül egyik sem felel meg igazán olyan bejegyzésekhez, amelyeket kizárólag

1. tábla **A courierMailAccount**

Attributum	Kötelező	Leírás
mail	Igen	A teljes levélcím
homeDirectory	Igen	Az alapkönyvtár, ahol a leveleket tároljuk
uidNumber	Igen	Az üzenetek tárolására használt azonosítóhoz tartozó felhasználói ID
gidNumber	Igen	Az üzenetek tárolására használt azonosítóhoz tartozó csoport ID

2. tábla A courierMailAlias

Attributum	Szükséges	Leírás
mail	Igen	Teljes levélcím
maildrop	Igen	Levélcím, ahová továbbítani kell. Lehet helyi álnév, vagy távoli levélcím

elektronikus levelesládákhoz és levéltovábbításhoz szeretnénk használni. Ezért inkább a Courier-terjesztésben található sémát használjuk fel. Érdemes megnézni a qmail-LDAP Projekttel érkező sémát is.

A Courier-séma

A virtuálislevél-azonosítókhoz használt `courierMailAccount` objektumosztályt az *1. táblázatban* foglaltuk össze. A *2. táblázatban* látható `courierMailAlias` objektumosztályt azokhoz az elektronikus levélcímekhez használjuk, amelyek más címekre továbbítanak neveket.

A `courierMailAccount` objektumosztály sajnos nem felel meg tökéletesen a céljainknak. Az `uidNumber` és `gidNumber` számunkra felesleges, hiszen minden levelet a `vmail` azonosítóra küldünk. Valamilyen álértéket mégis be kell írnunk, mivel a séma megköveteli a jelenlétét. Figyeljük meg, hogy ha több Unix-azonosító közt osztottuk volna szét a virtuális azonosítókat, ezeknek az értékeknek is lenne értelmük. Szükségünk lesz a levélláda- (mailbox) tulajdonságra, mivel ez alapján tudjuk megállapítani a levélláda helyét a fájlrendszeren. A levélláda bejegyzésnek perjellel kell végződnie, így jelezve, hogy Maildir stílusú levelesládáról van szó. A `userPassword` kapcsolóra úgyszintén szükségünk lesz, hiszen az elektronikus levélazonosítókhoz jelszavakat kell rendelnünk, hogy IMAP- vagy POP-rendszeren keresztül elérhetők legyenek. A többi elhagyható tulajdonságot nem használjuk.

A `courierMailAlias` objektumosztály éppen megfelel nekünk. Csak a két kötelező kapcsolót fogjuk használni, az elhagyhatók közül egyikkel sem foglalkozunk. A `maildrop` tulajdonság egy másik elektronikus levélcím vagy a helyi gép egy azonosítója lehet.

Jogosultságszabályozás

Az OpenLDAP több lehetőséget is kínál a jogosultság szabályozásra. Alapértelmezés szerint a rendszergazdai azonosítónak a fa összes bejegyzésére írási és olvasási joga van. A felügyeleti feladatok egy részét érdemes kezelendő tartományonként külön azonosítókra ruházni, hogy a kisebb változtatásokat a rendszergazdai azonosító elérése nélkül is elvégezhessék. Ezt úgy érhetjük el, hogy azokban a bejegyzésekben, ahol felügyeleti előjogokat akarunk adni, a ***postmaster*** (postamester) bejegyzést `organizationalRole`-á tesszük a `roleOccupant` tulajdonsággal. Aztán az OpenLDAP-ot beállíthatjuk úgy is, hogy csak e csoport tagjai érhessék el.

Megvalósítás

Ebben a részben bemutatjuk, hogyan valósíthatunk meg egy virtuális levelezést. Minden apró részletet nem fogunk ismertetni, csak azokra térünk ki, amelyek a szabványos telepítésen felül szükségesek.

Alább felsoroltuk azokat a programokat (és változatszámokat), amelyeken az alkalmazást kipróbáltuk:

- Red Hat Linux 6.2, 7.1 vagy 7.2;
- Postfix 1.1.x;
- OpenLDAP 2.0.21;
- Courier-IMAP 1.4.1;
- Procmail 3.22.

Létre kell hoznunk a `vmail`-azonosítót, majd a *~vmail/domains/* könyvtárat. Továbbá szükségünk lesz még két azonosítóra és két csoportra a Postfixhez – a Postfix telepítési útmutatójának megfelelően.

Az OpenLDAP fordításához és telepítéséhez nem lesz szükségünk különleges ismeretekre, így az utasításokat nézzük meg a leírásban. Éles alkalmazás esetén előbb olvassuk el, hogyan lehet az OpenLDAP rendszert nem rendszergazdaként futtatni a `chroot`-környezet felállításával és másolással. Ebben a cikkben azt mutatjuk meg, hogyan kell a `slapd`-t egyetlen kiszolgálón beállítani, létrehozni az alapfaszerkezetet, illetve beszúrni néhány alapadatot az LDAP-könyvtárba.

A slapd beállítása

Szükségünk lesz a Courier sémafájlra, ezért a Courier-terjesztés ***authlib/authldap.schema*** állományát másoljuk a */usr/local/etc/openldap/schema/courier.schema* helyre. A Courier-séma a ***cosine.schema*** és a ***nis.schema*** sémáktól függ. Adjuk a következő sorokat a ***slapd.conf*** fájlhoz:

```
include
/usr/local/etc/openldap/schema/cosine.schema
include
/usr/local/etc/openldap/schema/nis.schema
include
/usr/local/etc/openldap/schema/courier.schema
```

Ezután az adatbázis-meghatározásokat a ***slapd.conf*** fájlban a következő bejegyzésekkel állítsuk be:

```
directory       /usr/local/var/openldap-ldbm
database        ldbm
suffix          "dc=myhosting,dc=example"
```

A `database` kulcsszó meghatározza, hogy milyen háttértárat használunk (itt LDBM adatbázist adtunk meg). A `directory` kulcsszó az LDBM adatbázis elérési útját adja meg. Ne feledjük, hogy az itt megadott elérési útnak a `slapd` indítása előtt már léteznie kell, és a `slapd` írási és olvasási jogokkal kell rendelkezzen a könyvtáron (és természetesen az sem árt, ha execute jogosultsága is van, különben nem fog menni – a ford.). A `suffix` kulcsszó az adatbázishoz rendelt `root` utótagot adja meg. A következő néhány sor a szuperfelhasználói, más néven `root` azonosítót adja meg:

```
rootdn
"cn=Manager,dc=myhosting,dc=example"
```

```
rootpw
{SSHA}ra0sD47QP32ASAlaAhF8kgi+8Aflbgr7
```

A `rootdn` bejegyzésnek korlátlan elérési jogai vannak a teljes adatbázis felett, ezért jelszavát a tényleges adatbázison kívül őrizzük. A `rootpw` kulcsszóval megadott jelszót mindig kódolt formában tároljuk. Soha ne írjunk be egyszerű szöveget jelszónak. A szöveges jelszó (például: secret) titkosított jelszóvá kódolását a `slappasswd` paranccsal végezzük el:

```
% slappasswd
New password: secret
Re-enter new password: secret
{SSHA}ra0sD47QP32ASAlaAhF8kgi+8Aflbgr7
```

A kiírt sort vegyük ki a `slappasswd`-ből és másoljuk a ***slapd.conf*** fájlba, ahogy a fenti példában is tettük.
A keresések gyorsítása érdekében az általánosan használt tulajdonságokhoz indexeket készíthetünk:

```
index    objectClass   pres,eq
index    mail,cn        eq,sub
```

A ***slapd.conf*** utolsó része a jogosultságszabályozás.

A könyvtárfa létrehozása

A `slapd` beállítása után ideje hozzáfognunk az LDAP könyvtár feltöltéséhez. Az OpenLDAP-pal szállított parancssoros eszközöket fogjuk használni, és LDIF fájlokat hozunk létre a könyvtár módosításához.
Az első lépés a `root` csomóponthoz tartozó alapszerkezet elkészítése: ez az otthont adó szervezet (hosting organization) és a `rootdn`-hez tartozó bejegyzés. Hozzunk létre egy fájlt ***base.ldif*** néven a következő tartalommal:

```
dn: dc=myhosting, dc=example
 objectClass: top

 dn: cn=Manager, dc=myhosting, dc=example
 objectClass: top
 objectClass: organizationalRole
 cn: Manager

 dn: o=hosting, dc=myhosting, dc=example
 objectClass: top
 objectClass: organization
 o: hosting
```

Használjuk az `ldapadd` parancsot a rendszergazdai azonosítót használva, hogy bevigyük a fenti LDIF-et:

```
ldapadd -x -D
"cn=Manager,dc=myhosting,dc=example" \
-w secret -f base.ldif
```

Tartomány felvitele

Immár felvihetjük a tartományokat a „hosting” fa alá. Minden tartománynak rendelkeznie kell legalább a postamesterrel és a kitiltandók listája bejegyzésekkel (abuse entries).
A ***domain1.example*** fa létrehozásához készítsünk egy ***domain1.example.ldif*** nevű fájlt a következő tartalommal:

```
dn: o=domain1.example, o=hosting,
dc=myhosting,
 dc=example
objectClass: top
objectClass: organization
o: domain1.example

dn: cn=postmaster, o=domain1.example,
o=hosting,
 dc=myhosting, dc=example
objectClass: top
objectClass: organizationalRole
objectClass: CourierMailAlias
cn: postmaster
mail: postmaster@domain1.example
maildrop: postmaster

dn: mail=abuse@domain1.example,
o=domain1.example,
 o=hosting, dc=myhosting, dc=example
objectClass: top
objectClass: CourierMailAlias
mail: abuse@domain1.example
maildrop: abuse
```

Figyeljük meg, hogy a `maildrop` kapcsolók mindig helyi elektronikus levélazonosítók, és a postamesterhez, illetve a */etc/**aliases* fájlban megadott álnevekre továbbítódnak. A ***postmaster*** szabályban nem adtunk meg azonosítót, így jelenleg kizárólag a rendszergazdai azonosítón keresztül lehet új azonosítókat létrehozni. Vigyük fel a tartományt a következő paranccsal:

```
ldapadd -x -D
"cn=Manager,dc=myhosting,dc=example" \
➥-w secret  -f domain1.example.ldif
```

Azonosítók felvitele

Vigyük fel a <user1@domain1.example> levélcímmel rendelkező felhasználót. Egyúttal ennek a felhasználónak adjunk postamester-előjogokat a ***domain1.example*** tartományra. Készítsük el a ***user1.domain1.example.ldif*** fájlt a következő tartalommal:

```
dn: mail=user1@domain1.example,
o=domain1.example,
 o=hosting, dc=myhosting, dc=example
objectClass: top
objectClass: CourierMailAccount
mail: user1@domain1.example
homeDirectory: /home/vmail/domains
uidNumber: 101
gidNumber: 101
mailbox: domain1.example/user1

dn: cn=postmaster, o=domain1.example,
o=hosting,
dc=myhosting, dc=example
changetype: modify
add: roleOccupant
roleOccupant: mail=user1@domain1.example,
 o=domain1.example, o=hosting,
 dc=myhosting, dc=example
```

Az első rész az azonosítóhoz tartozó bejegyzést viszi be.
A `home directory` és a `mailbox` a fájlrendszeren fizikailag

elérhető levelesládára mutat. Az `uidNumber` és `gidNumber` kapcsolók kötelezően megadandók, de mivel mi nem használjuk őket, a 101-es próbaértékkel (dummy value) töltöttük fel. A második rész módosítja a postmaster bejegyzést – hozzáadja a `roleOccupant` kapcsolót a `user1@domain1.example` DN-t használva. Hozzuk létre ezt a bejegyzést:

```
ldapadd -x -D
"cn=Manager,dc=myhosting,dc=example"
➥-w secret -f user1.domain1.example.ldif
```

Mivel az azonosítóhoz még nem tartozik jelszó, hiába rendelkezik postamesteri jogosultságokkal, nem tudjuk hitelesíteni. Az `ldappasswd` paranccsal állítsuk be a `user1` kezdeti jelszavát:

```
ldappasswd -x -D "$DN" -w $PW -s user1
➥"mail=user1@domain1.example,
o=domain1.example,
o=hosting, dc=myhosting, dc=example"
```

A további tartományokat és azonosítókat hasonló LDIF fájlokkal vihetjük fel. Az LDIF fájlok felvitele kézi módszerrel meglehetősen fárasztó, ráadásul könnyen hibázhatunk is. Később más módszereket is mutatunk a felügyeletre.

Postfix

A Postfix rendszernek csak azokat a részeit tárgyaljuk, amelyek a levélkezelésre vonatkoznak.
Töltsük le a Postfix-forrást és csomagoljuk ki. Újra kell építenünk a Postfix `Makefile`-okat, hogy figyelembe vegyék és hivatkozzanak (link) az LDAP programkönyvtárakra. Hajtsuk végre a következő parancsot:

```
make makefiles CCARGS="-I/usr/local/include
➥-DHAS_LDAP" AUXLIBS="-L/usr/local/lib -lldap
➥-L/usr/local/lib -llber"
```

Innentől követhetjük a szokásos Postfix fordítási és telepítési útmutatásokat, amelyeket az ***INSTALL*** és az ***LDAP_README*** fájlokban találunk.

A Postfix beállítása

Ha az alább bemutatott beállítási példák bármelyikéhez nem neveztünk meg kifejezetten valamilyen fájlt, akkor azt minden bizonnyal a ***main.cf*** fájlban találjuk.
A szállítási tábla (transport table) a tartományokat rendeli az (***/etc/postfix/master.cf*** fájlban megadott) üzenetkézbesítő egységekhez (message delivery transports), illetve a továbbító gazdagépekhez. Mi virtuális tartományainkat a Postfixszel érkező virtuális kézbesítő ügynököknek szeretnénk átadni. A szállítási tábla tehát valahogy így néz ki:

```
domain1.example             virtual:
domain2.example             virtual:
```

Miután egyszerű szövegfájlként elkészítettük a szállítási táblát, a `postmap` utasítással (lásd man postmap) át kell alakítanunk bináris DB állománnyá. Ha ezzel megvagyunk, mutassuk meg a Postfix rendszernek, hogy van egy szállítási táblánk, és hogy azt merre találhatja meg. Azt is tudatnunk kell a Postfix programmal, hogy ezekre a tartományokra leveleket várunk.

Ezt a `transport_maps` és `mydestination` kulcsszavakkal tehetjük meg:

```
transport_maps = hash:/etc/postfix/transport
➥mydestination = $myhostname,
➥localhost.$mydomain,
➥   $mydomain, $transport_maps
```

Könnyen megadhatunk többszörös LDAP-forrásokat is. Az LDAP-forráskapcsolók leírását a ***README_FILES/LDAP_README*** fájlban olvashatjuk, a Postfix forrásában. A kapcsolónevek `<ldapforrÆs>_kapcsol` alakúak. Az LDAP-forrás nevét a használatával adjuk meg. A ***main.cf*** fájlban keresésenként egy-egy LDAP forrásmeghatározásra lesz szükségünk.

Álnevek

Az első LDAP-forrásmeghatározást a virtuális álnevekhez hozzuk létre. Ezt az LDAP-forrást „aliases”-nek, azaz álneveknek neveztük el. Beállításunk szerint az LDAP-kiszolgáló a helyi gépen fut. A keresés alapja az LDAP-kiszolgálónkon megadott „hosting” alfa lesz. Azokat az elemeket kérjük le, ahol a `mail` elem megegyezik a levél címzettjével, illetve amelyek a `courierMailAlias` objektumosztályba tartoznak. Az álnévhez rendelt célszemélyt a `maildrop` tulajdonsága tartalmazza. A Postfix nem fog felhasználóként bejelentkezni, hanem anonim keresést végez:

```
aliases_server_host = localhost
aliases_search_base =
  o=hosting,dc=myhosting,dc=example
aliases_query_filter =
  (&(mail=%s)(objectClass=CourierMailAlias))
aliases_result_attribute = maildrop
aliases_bind = no
```

Azonosítók

Amikor az `accounts` (azonosítók) forrást használjuk, olyan bejegyzéseket keresünk, amelyek a `courierMailAccount` objektumosztályba tartoznak. Eredményként a mailbox tulajdonságot kérjük vissza:

```
accounts_server_host = localhost
accounts_search_base =
  o=hosting,dc=myhosting,dc=example
accounts_query_filter =
  (&(mail=%s)(objectClass=CourierMailAccount))
accounts_result_attribute = mailbox
accounts_bind = no
```

Az azonosítókhoz egy második, `accountsmap` nevű forrást is létre kell hoznunk, hogy az azonosítókat `catchall` (minden elem megvizsgálása) esetén is le tudjuk kérni. Enélkül az álnevekben használt `catchall` felülbírálná a tartományok virtuális azonosítóit:

```
accountsmap_server_host = localhost
accountsmap_search_base =
o=hosting,dc=myhosting,dc=example
accountsmap_query_filter =
(&(mail=%s)(objectClass=CourierMailAccount
accountsmap_result_attribute = mail
accountsmap_bind = no
```

Miután megadtuk az `aliases` és az `accountsmap` LDAP-forrásokat, tudassuk a változtatásokat a Postfix programmal, és állítsuk be a `main.cf` `virtual_maps` kapcsolóját:

```
virtual_maps = ldap:aliases
```

A példa kedvéért tételezzük fel, hogy már készítettünk egy `vmail` nevű Unix-azonosítót, amely a 125-ös UID, és a 120-as GID értékeket birtokolja, a saját könyvtára pedig a */home/vmail*:

```
:virtual_mailbox_base = /home/vmail/domains
virtual_mailbox_maps = ldap:accounts
virtual_minimum_uid = 125
virtual_uid_maps = static:125
virtual_gid_maps = static:120
```

A `virtual_uid_maps` és `virtual_gid_maps` értékeit egy különleges statikus térképhez rendeltük, belekódolva a `vmail` azonosító UID és GID értékeit. Az összes itt bemutatott kapcsoló teljes leírását elolvashatjuk a Postfix forrásával együtt kapott ***README_FILES/VIRTUAL_README*** állományban. Át kell szerkesztenünk a `local_recipient_maps` kapcsolót, hogy a `virtual_mailbox_maps`-ban keresgéljen, így a Postfix tudni fogja, hogy mely azonosítók érvényesek. Ez azért szükséges, mert a Postfix el tudja utasítani az ismeretlen azonosítókra érkező leveleket:

```
local_recipient_maps = $alias_maps
  unix:passwd.byname $virtual_mailbox_maps
```

Courier

Semmilyen különleges utasítást nem kell használnunk a Courier telepítéséhez, úgyhogy útmutatásért forduljunk nyugodtan a leíráshoz. Meg fogja találni és be fogja fordítani az LDAP-ot. Érdemes meggondolni a `--enable-workarounds-for-imap-client-bugs` kapcsoló használatát a `./configure` futtatásakor, mert enélkül a Netscape-felhasználóknak gondjaik támadhatnak, amikor a kiszolgálónkat akarják használni.
A Courier külön azonosítódémont használ, hogy az azonosítást elválassza a rendszer egyéb részeitől. Állítsuk be úgy, hogy az érvényes levélazonosítókat LDAP és PAM alatt is megtalálja. Ezt az ***authdaemonrc*** fájlban az `authmodulelist` kapcsolóval adhatjuk meg:

```
authmodulelist="authldap authpam"
```

Minden LDAP-kapcsoló az `authldaprc`-ben található. A legtöbb kapcsoló magától értetődő. A Courier-séma használatához azonban el kell végeznünk néhány változtatást. Ezenkívül az összes virtuális azonosítót a `vmail` azonosítóhoz kell rendelnünk. Összefoglalóan a következő változtatásokat kell elvégeznünk az `authldaprc`-ben:

```
LDAP_GLOB_UID           vmail
LDAP_GLOB_GID           vmail
LDAP_HOMEDIR            homeDirectory
LDAP_MAILDIR            mailbox
LDAP_CRYPTPW            userPassword
```

Három további beállítás alkalmazását érdemes megfontolnunk, ezek: a `LDAP_AUTHBIND`, a `LDAP_BINDDN` és az `LDAP_BINDPW` – mindhárom a felhasználó azonosításához tartozik. Az `LDAP_AUTHBIND` kulcsszó és a `LDAP_BINDDN`, `LDAP_BINDPW` páros kölcsönösen kizárja egymást. Mi az `LDAP_AUTHBIND` használatát javasoljuk. Az `authldaprc` egyik megjegyzése memóriaszívást említ az OpenLDAP-ban a LDAP_AUTHBIND használatakor, ezt azonban az OpenLDAP 2.0.19 változatában már kijavították. Ha az `LDAP_BINDDN` és `LDAP_BINDPW` kulcsszavakat használjuk, jelszavainkat csak a crypt, MD5 és SHA algoritmusokkal kódolhatjuk. Az SMD5 és az SSHA nem lesz elérhető. Továbbá ha `LDAP_BINDPW` adunk meg, az LDAP jelszó egyszerű szövegként kerül az `authldaprc` fájlba. Az LDAP-jelszavakat egyszerű szövegként tárolni komoly biztonsági rést jelent, így ha csak lehet, inkább az `LDAP_AUTHBIND` módszert használjuk.
Az utolsó változtatás, amit el kell végeznünk, az IMAP-kiszolgáló beindítása az `IMAPDSTART` kapcsoló YES-re történő állításával. Mostantól a `courier-imap.sysvinit` indító parancsfájlt használhatjuk az IMAP-démon indításához és leállításához.

Felügyelet

A legtöbb felügyeleti feladat, az azonosítók és álnevek felvétele, módosítása és törlése az LDAP könyvtár módosítását igényli. Ezt az OpenLDAP parancssoros eszköz segítségével vagy valamilyen általános LDAP böngésző (például a gq) alkalmazásával tehetjük meg. Jelenleg egy Jamm nevű webfelügyeleti alkalmazáson dolgozunk, amely lényegében egy Java és JSP nyelven íródott alkalmazásspecifikus LDAP-böngésző lesz. Saját LDAP-sémával is rendelkezik, amely tulajdonképpen egy kis mértékben módosított Courier-séma. A Jamm jelenleg is használható és folyamatosan fejlődik – a legfrissebb tájékoztatást a Jamm SourceForge honlapon találjuk.

Megjegyzések az azonosítókészítéshez

Amikor azonosítókat és álneveket készítünk, az LDAP adatbázisban azok azonnal működővé válnak – feltételezve, hogy a levelezőrendszer ezen felhasználja őket. A virtuális azonosítók létrehozásakor ne feledjük, hogy a hozzájuk tartozó Unix-könyvtár nem jön létre a `~vmail`-ben. Ezt azonban orvosolhatjuk, mivel a Postfix virtuális kézbesítőügynöke az első levél megérkezésekor létrehozza a szükséges könyvtárakat. Éppen ezért azt javasoljuk, hogy amint létrehoztuk az azonosítót, küldjünk egy üdvözlőlevelet.

Linux Journal 2003. február, 106. szám

Dave Dribin
1991 óta használ Unixot és 1993 óta foglalkozik Linuxszal. 1995 óta hivatásszerűen fejleszt programokat Unix-rendszereken vagy -rendszerekre. Dave jelenleg független tanácsadóként dolgozik a realtorsi National Associationnél.

Keith Garner
1994 januárja óta használja a Linuxot. 1997 óta hivatásszerűen fejleszt Unix-programokat. Keith jelenleg a realtorsi National Association alkalmazásában áll.

KAPCSOLÓDÓ CÍMEK

Courier-IMAP ➲ http://www.inter7.com/courierimap
OpenLDAP ➲ http://www.openldap.org
Postfix ➲ http://www.postfix.org
Procmail ➲ http://www.procmail.org

Kísérletezgetés a Ptrace-szel (2. rész)

Sorozatunk második részében Pradeep néhány komolyabb témát boncolgat: a töréspontok beállítását, illetve kód beszúrását a már futó folyamatokba.

Sorozatunk első részében (Linuxvilág 2003. január) láthattuk, hogyan lehet a Ptrace felhasználásával követni a rendszerhívásokat, illetve megváltoztatni a rendszerhívások jellemzőit. Most olyan összetettebb módszereket ismerhetünk meg, mint a töréspontok beállítása vagy a kódnak futó programokba történő beillesztése. A hibakeresők ezeket a módszereket töréspontok beállítására és hibakereső-kezelők futtatására használhatják. Akárcsak az előző részben, a mostani írásunkban található valamennyi kód is i386 architektúrára épül.

Csatlakozás a futó folyamatokhoz

Az első részben a követendő folyamatot gyermekként futtattuk a `ptrace(PTRACE_TRACEME, ..)` hívás után. Amennyiben csak arra vagyunk kíváncsiak, hogyan ad ki a program rendszerhívásokat, ez elegendő is. Ha azonban már futó folyamatot szeretnénk követni vagy elemezni, inkább a `ptrace(PTRACE_ATTACH, ..)` hívást használjuk.
Ha a `ptrace(PTRACE_ATTACH, ..)` függvénynek a követendő folyamatazonosítót (pidet) átadjuk, megközelítőleg azonos hatást érünk el, mintha a folyamat a `ptrace(PTRACE_TRACEME, ..)` hívást használná és a nyomkövető folyamat gyermekévé válna. A követett folyamat `SIGSTOP` üzenetet kap, így a szokásos módon megvizsgálhatjuk vagy módosíthatjuk. Miután végeztünk a módosítással vagy követéssel, a követett folyamatot a `ptrace(PTRACE_DETACH, ..)` hívással az útjára engedhetjük.
A következő kód egy kis követőprogramra mutat példát:

```
int main()
{    int i;
     for(i = 0;i < 10; ++i) {
          printf("SzÆmlÆl : %d\n", i);
          sleep(2);
     }
     return 0;
}
```

Mentsük a programot ***dummy2.c*** néven. Fordítsuk le és futtassuk:

```
gcc -o dummy2 dummy2.c
./dummy2 &
```

Ezekután az alábbi kód segítségével tudunk csatlakozni a dummy2-höz:

```
#include <sys/ptrace.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <linux/user.h>
/* Az user_regs_struct-hoz stb. */
int main(int argc, char *argv[])
{    pid_t traced_process;
     struct user_regs_struct regs;
     long ins;

     if(argc != 2) {
          printf("HasznÆlat: %s <k vetendı
                                  ➥pid>\n",
                 argv[0], argv[1]);
          exit(1);
     }

     traced_process = atoi(argv[1]);
     ptrace(PTRACE_ATTACH, traced_process,
            ➥NULL, NULL);
     wait(NULL);
     ptrace(PTRACE_GETREGS, traced_process,
            ➥NULL, &regs);
     ins = ptrace(PTRACE_PEEKTEXT,
                  ➥traced_process,
                  ➥regs.eip, NULL);
     printf("EIP: %lx vØgrehajtott utas tÆs:
            ➥%lx\n", regs.eip, ins);
     ptrace(PTRACE_DETACH, traced_process,
            ➥NULL, NULL);

     return 0;
}
```

A fenti program egyszerűen csatlakozik a folyamathoz, megvárja, amíg megáll, megvizsgálja az eip (utasításszámláló) tartalmát, majd leválik.
A követett folyamat megállása után a `ptrace (PTRACE_POKETEXT, ..)` és a `ptrace(PTRACE_POKEDATA, ..)` függvényt használhatjuk kódbeillesztéshez.

Töréspontok beállítása

Hogyan állítanak be a hibakeresők töréspontokat? Általában a végrehajtandó utasítást egy csapdautasításra cserélik le, így amikor a követett program megáll, a követőprogram (azaz a hibakereső) nyugodtan vizsgálódhat. Amikor aztán a követőprogram folytatni akarja a követett kódot, egyszerűen visszahelyettesíti az eredeti utasítást. Íme egy példa:

```
#include <sys/ptrace.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <linux/user.h>
const int long_size = sizeof(long);

void getdata(pid_t child, long addr,
             ➥char *str, int len)
```

```
{    char *laddr;
     int i, j;
     union u {
             long val;
             char chars[long_size];
     }data;

     i = 0;
     j = len / long_size;
     laddr = str;
     while(i < j) {
          data.val = ptrace(PTRACE_PEEKDATA,
                            ➥child, addr +
                            ➥i * 4, NULL);
          memcpy(laddr, data.chars, long_size);
          ++i;
          laddr += long_size;
     }

     j = len % long_size;
     if(j != 0) {
          data.val = ptrace(PTRACE_PEEKDATA,
                            ➥child, addr
                            ➥+ i * 4, NULL);
          memcpy(laddr, data.chars, j);
     }
     str[len] = ·\0·;
}

void putdata(pid_t child, long addr,
             char *str, int len)
{    char *laddr;
     int i, j;
     union u {
             long val;
             char chars[long_size];
     }data;
     i = 0;
     j = len / long_size;
     laddr = str;
     while(i < j) {
          memcpy(data.chars, laddr, long_size);
          ptrace(PTRACE_POKEDATA, child,
                 ➥addr + i * 4, data.val);
          ++i;
          laddr += long_size;
     }
     j = len % long_size;
     if(j != 0) {
          memcpy(data.chars, laddr, j);
          ptrace(PTRACE_POKEDATA, child,
                 ➥addr + i * 4, data.val);
     }
}

int main(int argc, char *argv[])
{    pid_t traced_process;
     struct user_regs_struct regs, newregs;
     long ins;
     /* int 0x80, int3 */
     char code[] = {0xcd,0x80,0xcc,0};
     char backup[4];

     if(argc != 2) {
          printf("HasznÆlat: %s <k vetendı
                 ➥pid>\n", argv[0], argv[1]);
          exit(1);
     }
     traced_process = atoi(argv[1]);

     ptrace(PTRACE_ATTACH, traced_process,
            ➥NULL, NULL);
     wait(NULL);

     ptrace(PTRACE_GETREGS, traced_process,
            ➥NULL, &regs);

     /* Utas tÆs mÆsolÆsa ideiglenes tÆrol ba*/
     getdata(traced_process, regs.eip,
             ➥backup, 3);

     /* T rØspont elhelyezØse */
     putdata(traced_process, regs.eip, code, 3);

     /* Engedj k tovÆbb a folyamatot Øs vÆrjuk
     meg, am g vØgrehajtja az int 3 utas tÆst */
     ptrace(PTRACE_CONT, traced_process, NULL,
            ➥NULL);

     wait(NULL);
     printf("A folyamat leÆllt, visszarakjuk
            ➥az eredeti utas tÆsokat\n");
     printf("FolyatatÆshoz nyomja le az
            ➥<enter> billentyıt\n");
     getchar();
     putdata(traced_process, regs.eip, backup, 3);

     /* Az eip-t visszaÆll tjuk az eredeti
     utas tÆsra, hogy a folyamat tovÆbb
     futhasson */
     ptrace(PTRACE_SETREGS, traced_process,
            ➥NULL, &regs);

     ptrace(PTRACE_DETACH, traced_process,
            ➥NULL, NULL);
     return 0;
}
```

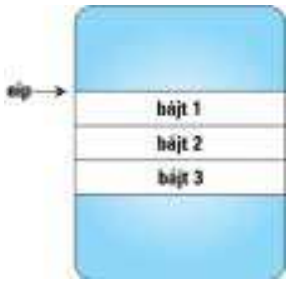
Az előbb tehát három bájtot a csapdautasítás kódjával helyettesítettünk, majd amikor a folyamat megállt, az eredeti utasítást visszahelyettesítettük, azután az eredeti helyére állítottuk vissza az eip-t. Az *1–4. ábra* segít megérteni, hogyan néz ki az utasításfolyam a fenti program végrehajtásakor.
Most, hogy már tisztáztuk a töréspont-beillesztés módszerének a hátterét, szúrjunk be pár kódbájtot a futó programba. A kódbájtok a „hello world” üzenetet fogják megjeleníteni.
Következő programunk csak egy egyszerű, de az igényeinkhez igazított „hello world” program lesz. A programot a következő paranccsal fordítsuk le:

```
gcc -o hello hello.c

void main()
{
__asm__("
       jmp forward
```

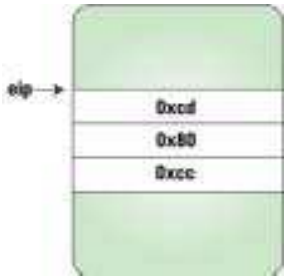
```
backward:
    popl    %esi        # A hello world sz veg
                        # c mØnek lekØrØse
    movl    $4, %eax    # ´rÆsi rendszerh vÆs
                        # kØrØse
    movl    $2, %ebx
    movl    %esi, %ecx
    movl    $12, %edx
    int     $0x80
    int3                # T rØspont. Itt a
                        # program megÆll,
                        # Øs a vezØrlØst
                        # visszaadja
                        # a sz lınek
forward:
    call    backward
    .string \"Hello World\\n\""
    );
}
```

Az előre-hátra (forward/backward címekre) ugrálásra azért van szükségünk, hogy a „hello world” szöveg címét megállapíthassuk.
A fenti assemblyhez tartozó gépi kódot lekérhetjük a GDB-ből. Indítsuk be a GDB-t, és fejtsük vissza a programot:

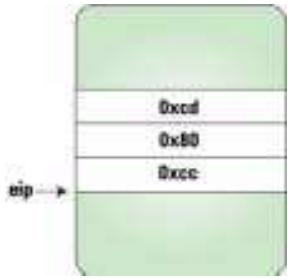
```
(gdb) disassemble main
Dump of assembler code for function main:
0x80483e0 <main>:   push   %ebp
0x80483e1 <main+1>: mov    %esp,%ebp
0x80483e3 <main+3>: jmp    0x80483fa <forward>
End of assembler dump.
(gdb) disassemble forward
Dump of assembler code for function forward:
0x80483fa <forward>:        call   0x80483e5
<backward>
0x80483ff <forward+5>:      dec    %eax
0x8048400 <forward+6>:      gs
0x8048401 <forward+7>:      insb
(%dx),%es:(%edi)
0x8048402 <forward+8>:      insb
(%dx),%es:(%edi)
0x8048403 <forward+9>:      outsl
%ds:(%esi),(%dx)
0x8048404 <forward+10>:     and
%dl,0x6f(%edi)
0x8048407 <forward+13>:     jb     0x8048475
0x8048409 <forward+15>:     or
%fs:(%eax),%al
0x804840c <forward+18>:     mov    %ebp,%esp
0x804840e <forward+20>:     pop    %ebp
0x804840f <forward+21>:     ret
End of assembler dump.
(gdb) disassemble backward
Dump of assembler code for function backward:
0x80483e5 <backward>:       pop    %esi
0x80483e6 <backward+1>:     mov    $0x4,%eax
0x80483eb <backward+6>:     mov    $0x2,%ebx
0x80483f0 <backward+11>:    mov    %esi,%ecx
0x80483f2 <backward+13>:    mov    $0xc,%edx
0x80483f7 <backward+18>:    int    $0x80
0x80483f9 <backward+20>:    int3
End of assembler dump.
```



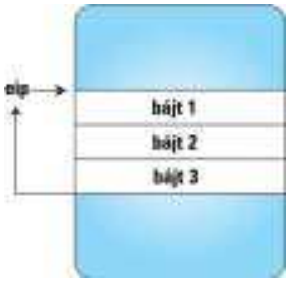
1. ábra
Miután a folyamat megállt



2. ábra Miután a csapda-utasításbájtokat elhelyeztük



3. ábra A csapda működésbe lépett, és a vezérlés a nyomkövető programhoz került



4. ábra Az eredeti utasítások visszahelyettesítése után az eip-t az eredeti helyre állítjuk vissza

Nekünk a `main+3`-tól a `backward+20`-ig van szükségünk a kódra, ami összesen 41 bájtot jelent. A gépi kódot GDB alatt az x paranccsal jeleníthetjük meg:

```
gdb x/40bx main+3
<main+3>:     eb 15 5e b8 04 00 00 00
<backward+6>:bb 02 00 00 00 89 f1 ba
<backward+14>:       0c 00 00 00 cd 80 cc
<forward+1>: e6 ff ff ff 48 65 6c 6c
<forward+9>: 6f 20 57 6f 72 6c 64 0a
```

Megszereztük a végrehajtandó utasítások bájtjait. Mire várunk akkor? Az előző példában látott módszerrel be tudjuk illeszteni őket a futó programba. A forráskód a következőképpen néz ki (most csak a `main` függvényt adjuk meg):

```
int main(int argc, char *argv[])
{   pid_t traced_process;
    struct user_regs_struct regs, newregs;
    long ins;
    int len = 41;
    char insertcode[] =
"\xeb\x15\x5e\xb8\x04\x00"

"\x00\x00\xbb\x02\x00\x00\x00\x89\xf1\xba"

"\x0c\x00\x00\x00\xcd\x80\xcc\xe8\xe6\xff"

"\xff\xff\x48\x65\x6c\x6c\x6f\x20\x57\x6f"
        "\x72\x6c\x64\x0a\x00";
    char backup[len];
```

1. lista

```
map start-mapend      protection   offset       device     inode      process file
08048000-0804d000     r-xp         00000000     03:08      66111      /opt/kde2/bin/kdeinit
```

```
    if(argc != 2) {
        printf("HasznÆlat: %s <k vetendı
            ➥pid>\n",
               argv[0], argv[1]);
        exit(1);
    }
    traced_process = atoi(argv[1]);
    ptrace(PTRACE_ATTACH, traced_process,
           ➥NULL, NULL);
    wait(NULL);
    ptrace(PTRACE_GETREGS, traced_process,
           ➥NULL, &regs);
    getdata(traced_process, regs.eip, backup,
            ➥len);

    putdata(traced_process, regs.eip,
            ➥insertcode, len);
    ptrace(PTRACE_SETREGS, traced_process,
           ➥NULL, &regs);
    ptrace(PTRACE_CONT, traced_process,
           ➥NULL, NULL);

    wait(NULL);
    printf("The process stopped, Putting back
            ➥the original instructions\n");
    putdata(traced_process, regs.eip, backup,
            ➥len);
    ptrace(PTRACE_SETREGS, traced_process,
           ➥NULL, &regs);
    printf("Letting it continue with
            ➥original flow\n");
    ptrace(PTRACE_DETACH, traced_process,
           ➥NULL, NULL);
    return 0;
}
```

Kód beszúrása szabad helyekre

Az előző példában a kódot közvetlenül a végrehajtott utasításfolyamba szúrtuk be. Sajnos az ilyesmi könnyen összezavarhatja a hibakeresőket, ezért keressünk inkább egy üres helyet a folyamatban, és ide helyezzük a kódot. Az üres helyet a követett folyamathoz tartozó */proc/pid/maps* fájl vizsgálatával találhatjuk meg. A következő függvény e térkép indulócímét keresi ki:

```
long freespaceaddr(pid_t pid)
{
    FILE *fp;
    char filename[30];
    char line[85];
    long addr;
    char str[20];

    sprintf(filename, "/proc/%d/maps", pid);
    fp = fopen(filename, "r");
    if(fp == NULL)
        exit(1);
    while(fgets(line, 85, fp) != NULL) {
        sscanf(line, "%lx-%*lx %*s %*s %*s",
               ➥&addr, str, str, str, str);
        if(strcmp(str, "00:00") == 0)
            break;
    }
    fclose(fp);
    return addr;
}
```

A */proc/pid/maps* minden sora a folyamat egy-egy lefoglalt tartományának felel meg.
A */proc/pid/maps* bejegyzést az *1. listában* láthatjuk.

A következő program a kódot az üres területre szúrja be. A szerkezete hasonlít az előző beszúróprogramhoz, de azzal az eltéréssel, hogy az új kódunk tárolásához az üres terület címét használjuk majd fel. A `main` függvény forrását az alábbiakban olvashatjuk:

```
int main(int argc, char *argv[])
{   pid_t traced_process;
    struct user_regs_struct oldregs, regs;
    long ins;
    int len = 41;
    char insertcode[] =
"\xeb\x15\x5e\xb8\x04\x00"

"\x00\x00\xbb\x02\x00\x00\x00\x89\xf1\xba"

"\x0c\x00\x00\x00\xcd\x80\xcc\xe8\xe6\xff"

"\xff\xff\x48\x65\x6c\x6c\x6f\x20\x57\x6f"
        "\x72\x6c\x64\x0a\x00";
    char backup[len];
    long addr;

    if(argc != 2) {
        printf("HasznÆlat: %s <k vetendı
            ➥pid>\n",
               argv[0], argv[1]);
        exit(1);
    }

    traced_process = atoi(argv[1]);

    ptrace(PTRACE_ATTACH, traced_process,
           ➥NULL, NULL);
    wait(NULL);

    ptrace(PTRACE_GETREGS, traced_process,
```

```
            ➥NULL, &regs);
        addr = freespaceaddr(traced_process);
        getdata(traced_process, addr, backup, len);

        putdata(traced_process, addr, insertcode,
            ➥len);
        memcpy(&oldregs, &regs, sizeof(regs));
        regs.eip = addr;
        ptrace(PTRACE_SETREGS, traced_process,
            ➥NULL, &regs);
        ptrace(PTRACE_CONT, traced_process,
            ➥NULL, NULL);
        wait(NULL);
        printf("A folyamat megÆllt,
            ➥visszahelyezz k az eredeti
            ➥utas tÆsokat.\n");
        putdata(traced_process, addr, backup, len);
        ptrace(PTRACE_SETREGS, traced_process,
            NULL, &oldregs);
        printf("TovÆbbengedj k az eredeti
            ➥folyamot\n");
        ptrace(PTRACE_DETACH, traced_process,
            ➥NULL, NULL);
        return 0;
}
```

A színfalak mögött

De valójában mi történik ezalatt a rendszermagban? Hogyan működik a Ptrace? Ez a rész önmagában megérne egy külön cikket – mégis, nézzük meg röviden, hogyan zajlanak a dolgok!
Amikor a folyamat `PTRACE_TRACEME`-mel hívja meg a Ptrace-t, a rendszermag beállítja a folyamatzászlókat, hogy jelezze, a folyamat követés alatt áll:

```
Source: arch/i386/kernel/ptrace.c

if (request == PTRACE_TRACEME) {

    /* mÆr k vetnek minketı */
    if (current->ptrace & PT_PTRACED)
        goto out;

    /* Æll tsuk be a ptrace bitet
       a folyamatzÆszl kban. */
    current->ptrace |= PT_PTRACED;
    ret = 0;
    goto out;
}
```

Amikor a rendszerhívás-bejegyzés véget ér, a rendszermag megvizsgálja a zászlót, és amennyiben a folyamatot követik, meghívja a követő rendszerhívást. A nyers assembly-részeleteket itt találhatjuk meg: arch/i386/kernel/entry.S.

Beléptünk a `sys_trace()` függvénybe, ezt a *arch/i386/kernel/ptrace.c*-ben találhatjuk. A függvény megállítja a gyerekfolyamatot, és jelet küld a szülőnek, értesítve őt arról, hogy a gyermeket megállította. Az alvó szülő így felébred, és végrehajthatja a Ptrace-varázslatokat. Amint a szülő végzett, és meghívja a `ptrace(PTRACE_CONT, ..)` vagy a `ptrace(PTRACE_SYSCALL, ..)` függvényt, a `wake_up_process()` ütemező függvény meghívásával a gyermeket is felébreszti. Más architektúrák ezt úgy oldják meg, hogy `SIGCHLD` üzenetet küldenek a gyereknek.

Összegzés

A Ptrace néhány ember számára varázslatos tudománynak tűnhet, hiszen képes egy futó program vizsgálatára és követésére. Általában hibakeresők és rendszerhíváskövető programok (például a Ptrace) használják ezt a lehetőséget, ugyanakkor érdekes távlatot nyit meg egy felhasználómódú kiterjesztés létrehozására is. Számos próbálkozás napvilágot látott már, ami az operációs rendszert felhasználószinten próbálja meg bővíteni. A ***Kapcsolódó címek*** között olvashatunk az UFO-ról, azaz a felhasználószintű fájlrendszerbővítésről (User-level extension to Filesystems). A Ptrace-t ezenkívül biztonsági rendszerek megvalósítására is fel szokták használni.
A cikkben (a jelenlegi és az előző részben) található összes kód (eredeti angol változata) tar-állományként elérhető a Linux Journal FTP lapján [ftp.ssc.com/pub/lj/listings/issue104/6210.tgz].

Linux Journal 2002. december, 104. szám

Pradeep Padala (p_padala@yahoo.com)
Jelenleg a Floridai Egyetemen diplomája megszerzésén munkálkodik. Érdeklődési területei között a rácsos kiépítésű és osztott rendszerek szerepelnek. Honlapját a ➲ http://www.cise.ufl.edu/~ppadala címen lehet elérni.

KAPCSOLÓDÓ CÍMEK

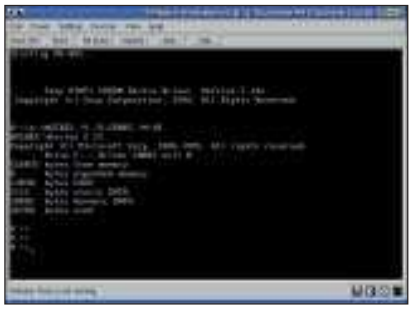
Extending the Operating System at User Level: The UFO Global File System
➲ http://www.cs.ucsb.edu/projects/ufo/97-usenix-ufo.ps

A Ptrace-kézikönyv (Secure, User-Level Resource-Constrained Sandboxing)
➲ http://csdocs.cs.nyu.edu/Dienst/Repository/2.0/Body/ncstrl.nyu_cs%2fTR1999-795/pdf

A Ptrace súgóoldala
➲ http://www.die.net/doc/linux/man/man2/ptrace.2.html

Linux alatt Windows vagy fordítva? (2. rész)

Az előző részben befejezett VMware-beállítások termését fogjuk most learatni. A választott operációs rendszer a Windows 98, ennek a telepítését követjük végig lépésről lépésre. Miután az előző részben bemutatott módon elkészítettünk egy virtuális gépet, indítsuk is el. Néhány jó tanács a virtuális gépek használatához. A telepítendő anyagot tartalmazhatja például hajlékonylemez, CD-ROM, merevlemez, azonban mindenképpen megkönnyíti a dolgunkat, ha egy rendszerindításra alkalmas CD-ROM-mal kezdünk neki ennek a sok türelmet igénylő feladatnak. Az *1. képen* egy hajlékonylemezes rendszerindítást láthatunk.



Az előkészületek

Vegyük magunkhoz a telepítőkészletünket, némi folyadékot és egy jó könyvet (na jó, ez így azért egy kicsit túlzás). A telepítés sebessége nagymértékben számítógépünk processzorának sebességétől és a rendelkezésre álló memória mennyiségétől függ, értelemszerűen ezek az adatok minél nagyobb értékeket mutatnak, annál kényelmesebb és gyorsabb lesz a telepítés. Hajdanán, amikor a 233 MHz-s Pentium II-es még gyors gépnek számított, örvendezve ült az ember fél napot a gép előtt, hogy rendszert telepítsen a virtuális gépre, manapság azonban igen nagy türelemre lenne szükségem, hogy egy ilyen napot végigüljek. Általánosságban elmondható tapasztalat, hogy legalább 256 MB memória és egy 1 GHz-s órajelű processzor kívánatos a gépbe.

Mivel az előkészített virtuális merevlemez még nem tartalmaz semmilyen lemezrészfelosztást, ezt nekünk kell létrehoznunk. Én erre az `efdisk` programot szoktam használni, ami a rendszerindító lemezemen is megtalálható. Természetesen rábízhatjuk a telepítőre is, így nem kell külön programmal bajlódnunk. A *2. képen* a merevlemez felosztását láthatjuk. Minekutána a létrehozott lemezrész 400 MB-os, nem érdemes több részre felosztani; a több gigabájtos lemezeket nyugodtan osszunk szét, mintha csak rendes merevlemezzel dolgoznánk. Miután ezzel végeztünk, a gépet újra kell indítanunk, hogy a változtatások érvénybe lépjenek. Természetesen az így létrehozott lemezrészen még egy nagyon fontos dolgot végre kell hajtani, ez pedig a formázás!

A telepítés elkezdése

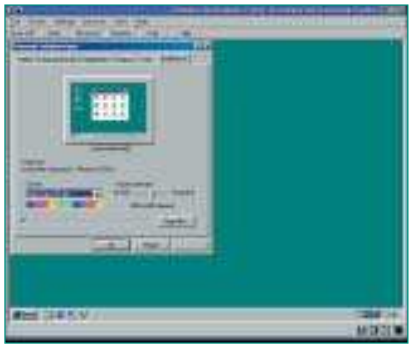
A *3.* és *4. képen* egy Windows-telepítés jellemző első képernyőképeit láthatjuk. Mint már írtam, ez a virtuális gép szinte mindenben megegyezik egy szokványos PC-vel, így semmilyen trükkhöz nem kell folyamodnunk a rendszerek – jelen esetben a Windows 98 – telepítése során. Miután elfogadtuk a felhasználói szerződést, és megadtuk a sorozatszámot, a szokványos telepítési lépéseket kell megtennünk. Az *5. képen* éppen a ***Telepítési típus***-t választom ki. Ezek után már csak a *6. képen* látható „Dőljünk hátra és ámuljunk” van hátra. Majd jön a *7. képen* látható ***Felkészülés a Windows 98 első indítására…*** képernyő. Legvégül megkapjuk a Windows „barátságos” felületét.

A telepítés

Mint fentebb írtam, nagyon megkönnyítheti az életünket, ha rendszerindításra alkalmas CD-ROM-unk van. Ezt csak behelyezzük a meghajtóba, és elindítjuk a beállított virtuális gépet, és lám, mintha csak egy igazi gép előtt ülnénk, elindul a telepítő. Ezekután a rendszer telepítése következik. Jó tanácsként viszont mindenképpen szem előtt tartandó, hogy ha a nem megfelelő beállításokkal telepítünk egy virtuális gépet, nagy sebességveszteség lesz a jutalmunk – tehát ne próbáljunk meg Windows 98-hoz beállított virtuális gépen Linuxot telepíteni.

Finomhangolás

Valószínűleg senki sincs megelégedve VMware gépének a színmélységével, mivel ez a normál VGA-felbontás 16 színnel. Ezt tudjuk kiküszöbölni a ***VMware Tools*** segítségével. Ha ezt bekapcsoljuk, máris lehetőségünk nyílik a képernyő felbontásának és színmély-



ségének a beállítására. A ***Settings*** menüpontban válasszuk ki a ***VMware Tools Install*** menüt, ekkor ez a kiegészítés telepíti önmagát. Nagyon fontos, hogy a Windowsnak futnia kell, miközben ezt a lehetőséget bekapcsoljuk, de a használatához indítsuk újra a gépet. Újraindítás után még nem történik semmi, csak annyi változik meg a rendszerben, hogy lesz még egy meghajtónk, ez az én esetemben a `D:` lett. Különféle virtuális eszközeinkhez ezen a meghajtón találhatjuk meg a meghajtóprogramokat. A ***Setup*** könyvtárban találunk egy ***setup.exe*** nevű fájlt, ezt futtassuk le, és már kész is a telepítés. A rendszer ezután szeretne újraindulni – természetesen újra is indítjuk, amikor pedig a Windows újra elindul, már 800×600-as felbontásban, 16-bites színmélységben élvezhetjük eddigi fáradozásaink gyümölcsét. Ezekután nem kell nyomkodnunk a CTRL+ALT billentyűket, hogy a virtuális gép „elengedje” az egeret: ha kihúzzuk a VMware szélére, rögtön megteszi.

A másik, általam nagyon kedvelt szolgáltatás a teljes képernyős üzemmód, amivel nagyon kellemesen csőbe tudom húzni az ismerőseimet: amikor meglátták, hogy a laptopomon Windows fut és hogy abban végzem a szövegszerkesztési feladataimat, szinte egyszerre nyúltak a telefonjukért, hogy mentőt hívjanak hozzám, mondván biztosan történt velem valami, ami semmi esetre sem nevezhető jónak! Tapasztalataim szerint a program teljes képernyős módban sokkal gyorsabban válaszol mindenre, mint az ablakosban.

Attól függően, hogy a VMware telepítésekor a hálózati kapcsolatot hogyan állítottuk be, most azt is beállíthatjuk, hogy a virtuális gépünkből is láthassuk az Internetet, a helyi hálózatot, vagy amit csak akarunk.

Nos, a Windows 98 telepítése befejeződött, ezzel is bebizonyítottuk, hogy a VMware még a Microsoft rendszereit is futtatni tudja. Természetesen nemcsak ezeket a rendszereket lehet futtatni, hanem minden olyan rendszert, ami az i386-os processzorcsaládra íródott. Mint a kapcsolódó címekből is látható találhatunk bőven olyan operációs rendszert ami, szabadon hozzáférhető, és a virtuális gépünkben is futtatható. Azért választottam azonban ezt a rendszert, mivel valószínűleg mindenki valami ehhez hasonlóval fogja kezdeni az ismerkedést.

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

KAPCSOLÓDÓ CÍMEK

Szabadon hozzáférhető operációs rendszerek:
AtheOS, könnyen kezelhető grafikus operációs rendszer
➲ http://www.atheos.cx/
Syllable, az AtheOS-ből fejlődött ki
➲ http://syllable.sourceforge.net/
A BSD-k:
➲ http://www.freebsd.org
➲ http://www.netbsd.org
➲ http://www.openbsd.org
DOS jogtiszán:
➲ http://www.freedos.org/
Solaris 8, a SUN remek operációs rendszere
➲ http://wwws.sun.com/software/solaris/binaries/
BeOS
➲ http://www.bebits.com
Linuxok a teljesség igénye nélkül (bár ezt mindenki tudja)
➲ http://www.debian.org
➲ http://www.redhat.com
➲ http://www.suse.de
➲ http://www.linux-mandrake.com
A szabad operációs rendszerek gyűjtőhelye:
➲ http://www.freeos.com

Felhasználói felület fénysebességgel (2. rész)

Sorozatunk előző részében megalkottunk egy egyszerű, de működő képnéző programot az FLTK (Fast Light Toll Kit) segítségével. Most ezt bővítjük tovább…

Legyen az első feladatunk a többképkijelölés lehetőségének a megteremtése. Azt, hogy a fájlválasztóban ezt meg tudjuk tenni, nagyon könnyen elérhetjük: a `kep_open_cb` függvényben (***nezoclass.cxx***) a `New Fl_File_Chooser` értékei között írjuk át `Fl_File_Chooser::SINGLE` bejegyzést `Fl_File_Choosr::MULTI`-ra. Így a kiválasztó már megengedi, hogy több fájlt jelöljünk ki. Hátravan azonban még ezek kezelése a programunkban.
Először is el kell döntenünk, hogy hol és hogyan tároljuk ezeket a fájlokat (eddig nem volt rá szükség, mivel azonnal megnyitottuk őket, és csak a képet tároltuk). Számomra a legegyszerűbbnek az tűnik, hogy a fájlokat – elérési útjukkal együtt – egy karaktermutatókból álló tömbön keresztül érjük el. Legyen ez egy globális változó, hogy mindenhonnan el lehessen érni. Ezenkívül pedig két változót használunk a tömb méretének, valamint az aktuális tömbindexnek a tárolására. A ***nezo.cxx*** fájlban definiáljuk, a ***nezoclass.cxx***-ben pedig utalunk rá:

```
char **fajlok;
int siz=20;
int anum;
```

illetve

```
extern char **fajlok;
extern int siz;
extern int anum;
```

A `nezo.cxx` `init()` függvényében tárhelyet foglalunk az induló tömbnek, és a tömb első elemének (`NULL`) kezdőértéket adunk:

```
::fajlok=(char**)malloc(sizeof(char*) * ::siz);
::fajlok[0]=(char *)0;
```

Ha már a ***keptar.cxx*** fájlt szerkesztjük, módosítsuk még egy helyen – így többet már nem is kell hozzányúlnunk ezen részen belül. Ha úgy állítjuk be a programot, hogy indításkor a parancssorban meg lehessen adni egy képfájl nevét, amit a logó helyett megnyit, lehetőségünk lesz rá, hogy programunkat alapértelmezett képnézőként állítsuk be más programok vagy akár a teljes grafikus felületünk számára. Ez könnyen megoldható (szebb lenne, ha részletesebben elemezné a parancssort; de mivel a legrosszabb, ami ebből adódhat, az, hogy ha a fájl nem létezik, nem nyitja meg; vagy nem kép, esetleg nem jól adtuk meg a parancssorban, nincs rá igazán szükség). Ha van parancssori érték, azt megpróbálja megnyitni, ha nincs, marad a logó. A `main` függvényt kell módosítanunk a `Kep_Nez_Wnd()` utáni és a `show()` előtti részben:

```
if(argc>1){
    ::kep_nez_wnd->loadkep(argv[1]);
    }
else {
    ::kep_nez_wnd->loadkep("logo.jpg");
    }
```

Térjünk vissza a több fájl betöltése témára! A `kep_open_cb` visszahívandó függvényt fogjuk tovább módosítani. Az első tudnivaló: az `Fl_File_Chooser` elemből a `count()` tagfüggvénnyel lehet lekérdezni, hogy a felhasználó hány fájlt választott ki. Ha ez nulla, akkor a Mégsem gombbal, vagy az ablakkezelőn keresztül csukta be az ablakot, egyébként a fájlok számát mutatja. Az értékeket a `value(int)` tagfüggvény adja vissza, egytől sorszámozva – tehát erre kell szerveznünk egy ciklust. Íme a kód (az `Fl::wait()` utáni részt kell eszerint módosítani):

```
if((j=kep_opener->count())){
    for(i=0;i<j;i++){
    fajlok[anum]=strdup(kep_opener->value(i+1));
    anum++;
    if(anum>=siz){
        ➥fajlok=(char**)realloc(fajlok,
        ➥sizeof(char *) * 2 * siz;
        siz= 2 * siz;
        }
    }
    ::kep_nez_wnd->actu=anum-j;
    ::kep_nez_wnd->loadkep(kep_opener->
    ➥value(1));
    ::kep_nez_wnd->redraw();
}
```

Mint észrevehető, a tömb méretét – ha betelne – a kétszeresére nagyítjuk. A fájlelérési utak és -nevek karakterei számára a `strdup` foglal memóriát. Ezt fel kell szabadítani, ha a listát töröljük; a felszabadítás a `kep_close_cb` függvénnyel történik. Mielőtt elfelejteném: a `Kep_Nez_Wnd` meghatározásánál hozzá kell adni az `int actu;` adattagot, mivel ezt fentebb használjuk. A legegyszerűbb, ha a megadott címről letöltjük a fájlokat, és az itt leírtakat csak magyarázatként használjuk. Ezzel a programnak már több fájlt meg tudtunk adni.
A következő lépés az, hogy meg is tudjuk őket nézni. A `kep_open_cb` új változata az újonnan kiválasztott képek közül az elsőt jeleníti meg (`kep_opener->value(1)`), és ezt is teszi pillanatnyilag láthatóvá. Mivel azt szeretnénk, ha lépkedni tudnánk a képek között, valamilyen eseménykezelést kell megvalósítanunk. Eddig is cél volt, hogy külön kezelőelemek nélkül, gyorsbillentyűk segítségével tudjuk használni a programot, most eszerint folytatjuk. Ennek megfelelően a `Kep_Box` leíró tagfüggvényét kell bővíteni, mivel ez kezeli az eseményeinket. Adjunk hozzá a PG_DN, PG_UP, HOME és END billentyűkhöz is a kódot a kapcsolón belül. A forrásszöveg alapján egyértelmű, hogy a hatásukra mit csinál.
Szintén itt látható egy olyan kódrészlet, ami következő célunk megvalósításához tartozik. A fenti léptetőgombok ugyanis

kikapcsolják az önműködő lejátszást. Tekintsük át ennek a megvalósítását! Az FLTK-ban az időzített feladatok elvégzésére az Fl osztály `add_timeout(double time, Fl_Timeout_Handler callback)` függvényét; eltávolításra a `remove_timeout(callback)`; újraindítására pedig a `repeat_timeout(double time, callback)`; függvényeit használhatjuk. Sejthető, hogy az első a programhoz egy időzített eseménykezelőt ad hozzá. Ez úgy működik, hogy a megadott másodperc után (mivel ez egy `double` érték, tetszőleges tört is megadható) meghívja a megadott függvényt. Elvileg egy mutatót is átadhat értékként, ezt itt nem használjuk, alapértelmezett értéke `NULL`, így nem kell megadnunk. A `remove_timeout(callback)` a megadott függvényt távolítja el az időzített feladatok közül. A `repeat_timeout` célja az, hogy magán az időzített függvényen belül meg lehessen hívni, azért, hogy a megadott idő múlva a rendszer újra meghívja. Érdekes megoldása az FLTK-nak, hogy ilyen esetben az idő mérése akkor indul el, amikor a függvény visszatér, azaz (elvileg) független attól, hogy a függvény mennyi munkát végez, vagy hogy az adott esetben milyen bonyolult a végzendő feladat. Az előző részből már ismert `menu_popup[]` tömbhöz két pontot adunk ***Lejátszás indul*** és ***Megáll*** szöveggel. Ezek visszahívandó függvényeiben indítjuk, illetve töröljük az időzített függvényt (`hajto_tocb`), amelyben elvégezzük a képcserét, a pillanatnyi tömbindex állítását, valamint az időzítő újraindítását (`repeat_timeout`). Így már működik az önműködő képváltás!

Van még néhány új képessége képnézőnknek az előző változathoz képest (ezért a változatszám fel is ugrik 0.4-re, mert annyival jobb). Ezek közül még egyet ismertetnék részletesen, a többi sorsát a forrásszövegre utalva az olvasóra bízom.

Ha a kép nagyobb, mint amit az ablakunkban meg tudunk jeleníteni, kicsinyítünk rajta – így azonban a részletek nem látszanak. Mármost megvan a lehetőségünk arra, hogy a képet nagyítsuk, de akkor jelenleg a kép kívánt részét csak a görgetősávokkal tudjuk behozni az ablak látható részébe. Ez nem kényelmes, ezért módosítunk rajta. Egyfelől a kurzormozgató billentyűk (nyilak) segítségével görgetni tudjuk a képet, ehhez a `Kep_Box` leíró tagfüggvényéhez kell a megfelelő bejegyzéseket hozzáadni, másrészről pedig azt szeretnénk, ha a képet az egérrel megragadva húzni lehetne az ablakon belül (mint például az Acrobat Readerben vagy a Gimpben a kijelölések esetében).

A dolgok egyszerűbbé tétele érdekében készítettem néhány tagfüggvényt a `Kep_Nez_Wnd` osztályhoz, ezek: a bal scroll, a jobb scroll, a fel scroll és a le scroll. Mindegyik egy egész értéket fogad, és annyival görgeti a képet a megfelelő irányba, amennyivel az lehetséges. Ha nullát adunk át, akkor nagyjából a kép 1/3 részével görget tovább (ezt használjuk a gombokkal való görgetésnél). Szintén kezeli, ha többet akarunk görgetni, mint amennyit lehetne. Így már egyszerűbb megírni az eseménykezelő részt. A gombokkal való mozgatás ezután egyértelmű.

Az egérrel való mozgatáshoz a leíró függvényt ki kell bővítenünk. A külső kapcsolóhoz hozzá kell adnunk az egérgomb-lenyomás, -vonszolás (lenyomva mozgatás) és -felengedés eseményeit. Ezek a következők: `FL_PUSH`, `FL_DRAG`, `FL_RELEASE`. Az első és utolsónál azt is megnézzük, hogy a bal gombról (`event_button1()`) van-e szó. A vonszolásnál erre nincs szükség, ugyanis az FLTK-ban egy elem csak akkor kapja meg a vonszolás eseményt, ha előtte a gomblenyomás eseményt „elvette”. Azaz, ha azt adnánk vissza az eseménykezelőből, hogy az egérgomb esemény nem érdekel bennünket (mint `Kep_Box`-ot), akkor a vonszolást meg se kapnánk. Ha az egyes egérgomb okozta az eseményt, akkor lenyomásnál beállítjuk a fog nevű, jelen esetben jelzőként (flag) működő változót, hogy benne vagyunk a mozgatási folyamatban, és tároljuk az esemény helyét. A vonszolás eseménynél ezután az új esemény koordináták alapján eldöntjük, hogy merre kell mozdulni, és meghívjuk a megfelelő scroll tagfüggvény(eke)t.

Még néhány szó arról, mivel bővült a program az előző részhez képest:

- Teljes képernyős mód – az FLTK beépített `Fl_Window::fullscreen()` módja segítségével.
- A pillanatnyi fájlnév kiírása az ablak címsorába az `Fl_Window::label()` tagfüggvény segítségével.
- Egyszerű segítségablak biztosítása az `Fl_Help_View` elem segítségével, html formátumú állomány beolvasásával.
- Nagyításnál az előző méret képközepének az ablak közepén tartása, ha a kép már nagyobb az ablaknál.

A képnéző program további fejlődését nyomon követhetik a ➲ http://demo.hmvhely.hu/ címen, ahová igyekszem feltenni az új változatot, valamint a 45. CD-mellékleten, a Magazin/FLTK könytárban.

Végezetül további kedvcsinálónak még néhány projekt, amiket az FLTK segítségével írnak: címeiket az FLTK honlapjáról (➲ http://www.fltk.org) kiindulva találhatjuk meg.

Elemek:

- `Fl_Menu_Bar` – a hagyományos, felül elhelyezett legördülő menük megvalósításához.
- `Fl_Button` – és alfajai a legkülönbözőbb nyomógombokhoz.
- `Fl_Preferences` – a Windows registryhez hasonló, szövegszerkesztővel is kezelhető adatbázis jellegű, beállítási adatok tárolására tervezett elem. Ezt használja például az `Fl_File_Chooser` a kedvencek eltárolására.
- `Fl_Roller` – a régi hangkártyákhoz vagy a legtöbb CD-olvasó elején látható forgatógombhoz hasonló jellegű elem, ami az egér segítségével vonszolással forgatható.
- `Fl_Tooltip` – a lebegő segítségekhez.

Programok, projektek:

- `Fl_Photo` – kép- és digitális kamerakezelő program. Elég sok külső könyvtár is szükséges hozzá, viszont jelenleg is hevesen fejlesztik, a 0.4 változattól rövid idő alatt eljutott a 0.9-ig. Lehet, hogy a cikk megjelenése idején már kész lesz az 1.0 is.
- Teljes ablakkezelő FLTK-ban írva: a ***Bill Spitzak*** által írt `flwm`, valamint a jelenleg is fejlesztett Equinox. Ehhez az utóbbihoz sok segédprogram is készen van már, viszont az FLTK fejlesztési változatát, a 2-est használják.
- Imview képnéző és -analizáló program (csak nézéshez nem igazán kényelmes, viszont sok mást tud).
- 3D-planetárium, valamint sok-sok kisebb segédprogram és az FLTK-ba még be nem vett elem.

Véleményem szerint érdemes megnézni az FLTK-t, mert egyrészt rendkívül logikusan megírt eszközkészlet, kicsi, gyors, rugalmas; emellett Linux/Unix, valamint Windows és Mac alatt egyaránt használható, másrészt felhasználási szerződése teljesen nyitott, még üzleti célú programot is fejleszteni lehet vele.

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

A PHP története

Hihetetlennek tűnik, de a PHP már közel tízéves múltra tekinthet vissza.

A 1994-ben ***Rasmus Lerdorf*** a saját honlapján azt kívánta megtudni, hogy kik olvassák nyilvánossá tett önéletrajzát. Erre a feladatra írt Perl nyelven egy egyszerű alkalmazást. Mivel a futtató kiszolgáló eléggé túlhajszolt volt, és a program emiatt többször feladta a harcot, Rasmus úgy döntött, hogy az egészet újraírja C-ben.
Történetesen ez a kiszolgáló, ahol ügyködött, számos más felhasználónak is otthont adott, akik felfigyeltek a munkájára. Többen megkérték, engedje meg számukra, hogy ezt a megoldást a saját lapjukon is használhassák. Rasmus hajlott a dologra, aminek rövidesen az lett a következménye, hogy egyre több kívánságot teljesíthetett programja továbbfejlesztésével kapcsolatban. Látva a dolog sikerét, munkáját leírással ellátott programcsomaggá állította össze, és levelezőlistát indított.
Ekkor kapta meg a program a nevét: Personal Home Page Tools. Ez rövidesen Personal Home Page Construction Kit névre módosult. Közben adatbázis-kiszolgálókkal is játszadozni kezdett, és összeütött egy másik alkalmazást, ami által képes volt SQL-lekéréseket összekapcsolni a hozzájuk tartozó webes űrlapokkal és listákkal. Ezt a csomagját Form Interpreter néven ismerhette meg a nagyközönség.
A dolgok felgyorsultak, 1997 végére a két program PHP/FI 2.0 néven egyesült, amihez mind a két összetevőt alaposan át kellett írni. Innentől számítható a PHP önálló programnyelvnek, olyannak, amelyet weblapokba ágyazhatóan lehetett futtatni. Mire a 2.0-s változat próbaváltozatainak a végére ért, és végleges pompájában kiadásra került, megnőtt a támogatott adatbázis-kiszolgálók száma is. E próbaváltozatok folyamán került be a MySQL-adatkapcsolat támogatása is a nyelvbe.
A PHP C nyelvű forrása ekkor még kényelmesen elfért egyetlen könyvtárban. A 2.0 végleges kiadásakor is csak a regexp-támogatás kapott kitüntetett helyet, azaz saját alkönyvtárat.

A csapatmunka eredménye

És eljött 1998 nyara, vele együtt pedig a PHP 3.0-s változata. Ez a kiadás már jelentős csapatmunka eredménye – ***Zeev Suraski*** és ***Andi Gutmans*** nagyfokú közreműködésével. Ők ketten teljesen a nulláról indulva újra felépítették a PHP parancsfájl-feldolgozó motort. Ez a mag az, amit ma Zend néven ismerünk. A PHP rövidítés ekkor nyerte el azt az értelmezését, ami jelenleg is használatos: „PHP: Hypertext Preprocessor”.
A hármas sorozat szűk kétéves fennállása során rengeteg fejlesztésen ment keresztül, de 2000 tavaszára ismét újabb nagy ugrás tanúi lehettünk.

Minőségi ugrás: a 4-es változat

A PHP 4.0 valóban ismét nagy változásokat hozott. Maga az alapnyelv is jelentősen bővült, és a rendszer magja ekkortól támogatta a különféle modulok hozzáírását. A legtöbb PHP3-ba épült támogatásból is ilyen modul készült. Egészen idáig a PHP csak az Apache webkiszolgálóval működött együtt teljes összefonódásban, míg más rendszereken csak mint CGI-alkalmazás állta meg a helyét. Egyedül a 4.0-val bevezetett egységes, webkiszolgálókkal kapcsolatot tartó alapréteg jelentett áttörést ezen

a területen. A PHP belső motorja hivatalosan ekkortól kapta meg a Zend elnevezést.

A jelen

A nagyobb változatszámugrásoknak mindig megvan az oka, miként most is. Több olyan újdonság is bekerült az új változatban, amelyek külön-külön is indokolhatták volna ezt az ugrást. Az egyik ilyen fontos lépés a CLI (parancssorbarát felület) végleges, immár nem kísérleti jellegű megjelenése. Eddig, ha bármilyen feladat ellátásához parancssorból futtatható PHP-programra volt szükség, nem létezett tökéletes megoldás. Márpedig ilyen feladat a komolyabb leterheltséggel számoló alkalmazások esetén könnyen előjöhet – elég egy komolyabb reklámcsík-kiszolgálóra gondolnunk.
A reklámok véletlenszerű elosztását ekkor összetett szempontok alapján, viszonylag bonyolultabb számításokkal határozhatjuk meg. E valószínűségeket minden egyes letöltéskor meghatározni és az alapján sorsolni túlzott és felesleges terhet jelent a gép számára. Ehelyett elegendő néhány percenként kiszámítani a megjelenési valószínűségeket, majd a következő időszakban eszerint küldeni a reklámokat, immáron letöltésenként egyetlen egyszerű kockadobással döntve. A meghatározott időközök `crontab` felállításával könnyedén létrehozhatók. Igen ám, de a PHP arra született, hogy HTTP-protokollon keresztül dolgozzon, nem pedig a héjból hívva. Arra a feladatra, hogy a PHP-kódok a `cron` segítségével mégis futtathatók legyenek, több megoldás is elterjedt:

- A Lynx szöveges böngésző felhasználásával, ami némi kivetnivalót hagy maga után, hiszen egy felesleges áttételen keresztül indítjuk be PHP-kódot.
- Az Apache-modul mellé egy CGI-változat fordításával és ennek parancssori hívogatásával. Ez már kiszolgálóbarátabb megoldás, de még mindig nem az igazi módszer.
  A 4.3.0-s változattól kezdve immár nem kell kétszer fordítanunk, ha parancssori elérést szeretnénk, és a Lynx-féle

trükközés is hamar elfelejthető. Mostantól kezdve amikor Apache-modult állítunk elő forrásból fordítással, egyben egy CLI felületű parancssorosan futtatható PHP is hadrendbe áll. Ez természetesen egy `-disable-cli` kapcsolóval kikapcsolható. Akkor sem lesz CLI felületünk, ha Apache-modul helyett a CGI-változat mellett döntünk. Lássuk, miféle kedvességekkel kényeztet minket a CLI-megvalósítás:

- Nem ad HTTP-fejléceket.
- Nem változtatja meg a pillanatnyi munkakönyvtárat a futó program könyvtárára.
- A hibaüzenetek nem kapnak HTML-formázást.
- A szabványos kimenet nem kerül átmeneti tárba, minden azonnal megjelenik a konzolon (`implicit_flush`).
- A program futási idejének nincs felső határa (`max_execution_time`).
- A meghíváskor átadott tulajdonságok a `$argc` és `$argv` változókon keresztül elérhetők. A `register_globals`-ra a CLI hasonlóképp válaszol, mintha Apache alól futna. Ennek `off` állapotba állításakor az `$argc` és `$argv` változók nem jönnek létre, viszont a `$_SERVER` tömbben továbbra is elérhetők.
- Bevezeti az `STDIN`, `STDOUT`, `STDERR` állandókat. Ezek a nevüknek megfelelő ki- és beviteli eszközökre mutatnak, használatukhoz tehát nem szükséges az `fopen()` hívása. Hasonlóképpen a bezárásukkal sem kell törődni. A CLI-t használva a PHP a Perlhez hasonlóképpen használható, azaz többféleképpen is futtathatjuk:
  1. A `php program.php` vagy a `php -f program.php` segítségével.
  2. A `-r` kapcsoló segítségével közvetlen parancsokat tudunk végrehajtatni: `php -r 'print_r(get_defined_constants());'`
  3. A bemenetére csöveken át is csatlakozhatunk.
  4. `#!/usr/bin/php` sort biggyesztve a PHP-parancsfájl elejére, azt önállóan is futtathatóvá tehetjük.

Ugyancsak most bevezetett újdonság az egységesített adatfolyamkezelés (Stream). A dolog szellemisége a Unix „minden fájl” felfogásához hasonlít. Itt arról van szó, hogy minden ki- és bemeneti művelet nagyon hasonlatos egymáshoz, legyen szó akár fájlba írásról, csővezetékből való adatfogadásról, memóriakezelésről, akár foglalaton keresztüli kapcsolattartásról. Ezeket a feladatokat rengeteg különböző, ámde mégis nagyon hasonló függvénygyűjteménnyel lehetett eddig megoldani. Foglalatokhoz példaképpen a `socket_*` függvények vannak rendszeresítve, míg FTP-re rengeteg `ftp_*` függvényt találunk. Az egységesítéshez a fájlkezelő függvények átírására, valamint jó pár `stream_*` eljárás rendszeresítésére volt szükség. A meglévő függvények egy része viszont ettől fogva már csak ezeknek az adatfolyamféle megfelelőiknek a további neveiként lelhetők fel, saját kóddal nem rendelkeznek. Így a `socket_set_timeout()` önmagában már nem létezik, csak a `stream_set_timeout()` egy álneve.
Ez nem csupán a PHP-ben fejlesztők munkáját könnyíti meg, de segíti a PHP további C-ben való fejlesztését is, hiszen így az újabb protokollok, adatforrások támogatásának bevezetéséhez nem kell teljes kezelőt írni, elég ennek a programozási felületnek a bővítményeit elkészíteni.
További újdonság, hogy a GD-támogatáshoz eleddig külön telepítendő libgd beszerzése nem szükséges, ugyanis a 4.3.0-s és későbbi kiadások már tartalmazzák. A különféle képformátumok kezelését biztosító külső könyvtárak fejlesztői változatára (libjpeg, libpng, libungif) viszont továbbra is szükség lesz. E fejlesztések mellett természetesen rengeteg hibát is kijavítottak benne, az Apache2 SAPI is közelebb került az éles kiszolgálókon való alkalmazhatósághoz. Ez véglegesen majd csak valamikor a nyárra várható, és a Zend2-motorra épülő PHP 5.0-val lesz hivatalosan is használható.

A jövő: PHP5 és Zend2

A történet 2001 nyarán kezdődött. Az akkori tervek szerint a Zend2-vel felszerelt PHP 5.0 várhatóan egyidőben jelent volna meg a 4.1-es változattal. Nos, azóta tudjuk, hogy nem így lett. A PHP 5.0-nak rengeteg elvárást kell kielégítenie, ennek megfelelően sok területen hatalmas előrelépésre lehet számítani.
A Zend-motor, ami a parancsfájlok feldolgozásáért felel, leginkább az objektumokkal kapcsolatos területen fog látványos eredményeket hozni. A PHP-t objektumtámogatottsága miatt sok vád érte, mivel a Java vagy a C++ képességeihez képest jócskán elmarad. Az egyik ilyen jelentős nehézség a jelenlegi 4-es változatokban az, hogy az „objektumváltozók” magát az objektumot tárolják, nem csak egy hivatkozást rá, mint ahogy az logikus lenne. Ennek következménye az, hogy minden egyes objektumpéldány egy másolat ahelyett, hogy ugyanazon objektumra mutatnának.
A Zend2-ben az objektumkezelés teljesen újra lett írva, ezáltal a fenti gondok megoldódnak, és ez utat nyit a komolyabb objektumalapú programozás felé. Hasonlóan nagy lépés lesz a kivételkezelés bevezetése. A `try`, `catch`, `throw` használatával immár a PHP is fel lesz vértezve azokkal az eszközökkel, amelyek segítségével „kiakadáskerülő” programokat építhetünk. ***Zeev Suraski***-nak, a Zend-motor egyik fejlesztőjének véleménye szerint a PHP ezzel a lépéssel valóban komoly, versenyképes objektumalapú nyelvvé növi ki magát. Az az előnye is meglesz a weblapba ágyazott Java lehetőségével szemben, hogy a PHP valóban pehelysúlyú parancsnyelv. Zeev mindezt röviden úgy foglalta össze, hogy a Java pont ilyen lett volna, ha parancsnyelvnek tervezték volna. Arra, hogy ténylegesen mik ezek az objektumkezelési újdonságok, egy későbbi cikkemben térek vissza.
További apróbb változások a karakterlánc-kezelésben várhatóak. Eddig is létezett az a lehetőség, hogy a karakterláncokat karaktereket tároló tömbként kezeljük, de ez egyrészt nem működik tökéletesen, másrészt a nyelvi szempontból teljesen eltérően kezelendő dolgokat jobb elkülöníteni. Ezért a jövőben a karakterláncok későbbi tömbként való kezelése figyelmeztető üzenetek megjelenését fogja kiváltani. A `$str[3]` helyett a jövőben a `$str{3}` forma használandó. A fejlesztők ennyivel természetesen nem érték be – ha már ilyen beavatkozás történik, érdemes kihozni belőle a legtöbbet.

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

KAPCSOLÓDÓ CÍMEK

- http://www.zend.com/zend/future.php
- http://www.theopenenterprise.com/story/TEO2002120400001

Fejlett modellezési eljárások (2. rész)

Sorozatunk második részében négy modellezési eljárással ismerkedhetnek meg, amelyek megkönnyítik a bonyolultabb formák létrehozását.

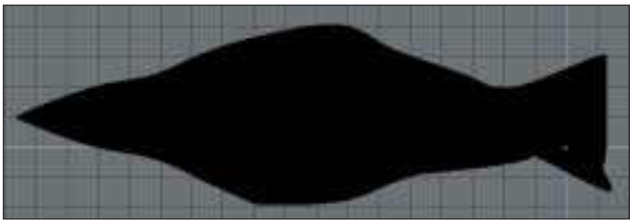
Az egyik módszer azon alapul, hogy léteznek olyan testek, amelyeket könnyedén modellezhetünk kétdimenziós alakzatnak térbelivé történő formálásával. Gondoljunk például egy csővezetékre vagy egy jövőbeli űrhajó formájára.

Térbeli kiterjesztés és formafinomítás

Az első eljárás egyúttal egy másikat tartalmaz, amit a későbbiek folyamán a feladathoz igazodva elválaszthatunk tőle. A két eljárás: az úgynevezett térbeli kiterjesztés (extrudálás) és a formafinomítás (mesh subdivision). Indítsuk el a Blendert, majd a szerkesztőnézetet a számbillentyűzet 1-es billentyűjével kapcsoljuk elölnézetbe. Első lépésben a test keresztmetszetét meghatározó alakzatot kell létrehoznunk. Az űrhajópéldánál maradva: hozzunk létre egy alakítható körvonalat (***Főmenü-Add-Curve-BezierCircle***). Húzzuk szét a két oldalsó pontját, hogy a téglalap formát lekerekítsük. Amit így létrehoztunk, már nagy vonalakban meghatározza a hajótest formáját, de nem eléggé részletes. Jelöljük ki az összes szerkesztési pontját, majd az alsó szerkesztőmezőben nyomjuk meg az F9-es billentyűt. Ezzel az alsó szerkesztőmezőt pontszerkesztő módba váltottuk át, vagyis az itt található felhasználói felület segítségével különféle műveleteket hajthatunk végre a kijelölt pontokon, görbéken és éleken. A részletek kidolgozásához e gombok közül a ***Subdivide*** gombra lesz szükségünk, amit a képernyő jobb alsó részén találunk meg. A gomb megnyomása után láthatjuk, hogy minden eddig meglévő szerkesztési pont között újabbak jelentek meg. Alakítsuk ki a hajótest részletesebb formáját, de ne hozzunk létre újabb pontokat. A forma kialakítása után ezt az alakzatot szeretnénk majd térbelivé tenni, de ehhez a görbét a Blender számára egyenes szakaszokból álló alakzattá kell átalakítanunk. Az alsó vezérlőfelület középső részén négy gombot találunk egymás alatt: ***Poly, Bezier, Bspline, Cardinal, Nurb***. A ***Poly*** gomb alkalmazásával az alakzatot szögletessé tehetjük, vagyis amellett, hogy a fő vonások megmaradnak, egyszerűbb és gyorsabban megjeleníthető formájúra alakítjuk. Ezután a TAB billentyűvel kapcsoljunk vissza tárgyszerkesztő módba, és az ALT-C kombinációval az alakzat körvonalát alakítsuk át síkfelületté.

A forma kialakítása után a következő lépés legyen a kiterjesztés. Kapcsoljunk vissza pontszerkesztő módba, és jelöljük ki az alakzat összes pontját az A billentyűvel. Ezután váltsunk át oldalnézetre, és a billentyűzeten használjuk az E-t. A megjelenő menüben megerősítjük a szándékunkat, és az egér mozgatásával az alakzatot kiterjesztjük. Mozgassuk az egeret oldalirányban, majd amennyiben szükséges, méretezzük át a keresztmetszet újonnan létrehozott részét, és a fenti két művelet ismétlésével alakítsuk ki a hajótest végleges formáját. Az én elképzelésem, ami a *képünkön* látható, csak támpontként szolgál.

A fentieket követve láthatjuk, hogy a forma még elég durva, de már alakulóban van. Formáljuk meg a szárnyakat is. Elölnézetben jelöljük ki a legszélső pontokat, az S billentyűvel húzzuk szét őket, ezután felülnézetben a szárnyak irányát és megfelelő elkeskenyedését alakítsuk ki.



A körvonalazódó űrhajó

Most már tényleg elégedettek lehetünk eddigi munkánkkal, következhet a finomítás. Szerencsére ezt a műveletet a Blender önműködő elvégzi, csak meg kell határozni a szükséges mértéket. Az alsó vezérlőfelület bal oldalán található a mérték meghatározására alkalmas gombkészlet. Itt keressük meg a ***SubSurf***-öt, majd nyomjuk meg. Ennek hatására a Blender finomítja a felületet, amelynek mértékét az alatta lévő két számmal határozhatjuk meg. A ***Subdiv*** felirat mellet található szám a szerkesztés közben látható mértéket mutatja, míg a felirattól jobbra lévő gombon a számolásnál, a megjelenítésnél szükséges finomítási fokot állíthatjuk be. A végleges részletesség beállítása után egy szép, áramvonalas űrhajót láthatunk, amelynek már csak az anyagát és a mintázatát kell meghatároznunk a sorozat folytatásának útmutatásai alapján.

A NURBS alapú modellezés

A következő modellezési eljárás alapja az, hogy bizonyos esetekben görbült felületekkel sokkal pontosabban alakíthatunk ki egy-egy formát, mintha pusztán síklapokkal határolt felületekkel tennénk. Ez az eljárás felhasználható például az emberi test modelljének kialakítására vagy más élőlény modellezésére is. A módszer neve: NURBS alapú modellezés (a rövidítés a Non Uniform Bezier Spline kifejezésből származik), és az egyik fontos tulajdonsága az, hogy a felület az úgynevezett ellenőrzőpontok (control point) változtatásával dinamikusan módosítható. Ismerkedésünk célja legyen egy egyszerű görbefelület megformálása. Elsőként a főmenüből kiindulva az ***Add-Surface-Curve*** menüpontok kiválasztásával hozzunk létre egy NURBS-görbét. Ezt a görbét alakítsuk tetszőleges formájúra, ehhez a szerkesztő eszközkészletben már korábban megismert (F9) ***Subdivide*** eszközt használjuk. Miután a megfelelő profilt kialakítottuk, készítsünk másolatokat erről a görbéről, de ne feledjük, hogy a NURBS alapú modellezés során az egyes profiloknak azonos számú ellenőrzőponttal kell rendelkezniük. Ez azt jelenti, hogy miután a másolatok elkészültek, az egyes görbékhez csak azonos számú pontot adhatunk vagy törölhetünk belőlük. A másolatokat a könnyebb szerkesztés végett egymástól csak egyetlen tengely mentén toljuk el. Így elérjük, hogy a későbbiekben is egyszerűen kiválaszthassuk őket, és az egyes görbék szerkesztésekor a pontokat is kényelmesen ki tudjuk választani. A másolatokat is egyesével módosítsuk, az elképzeléseinknek megfelelően, és amikor elkészültünk, a felületet alkotó görbéket tegyük a helyükre. Az elhelyezés után

eddigi munkánkról célszerű mentést készíteni. Ahhoz, hogy a Blender felületet tudjon feszíteni a görbékre, tudomására kell hozni, hogy ezek a különálló pontok tulajdonképpen egy tárgy

ALT-C	TSZ; átalakítás háromszögek alkotta hálóvá (úgynevezett mesh)
E	PSZ; a kijelölt pontok térbeli kiterjesztése (extrude)
CTRL-J	TSZ; tárgyak egyesítése
F	PSZ; háromszögek, felületek létrehozása
SHIFT-F7	háttérkép betöltése a szerkesztőmezőbe

pontjai. A nagyobb koordináták felől a kisebbek felé haladva jelöljük ki a létrehozott görbéket. Természetesen fordítva is csinálhatnánk, de a görbék kijelölési sorrendje befolyásolja a felület normálvektorainak az állását. Amennyiben a görbéket fordított sorrendben jelöljük ki, a felület is kifordított állapotban jelenik meg. A kijelölt görbéket ezután tehát egybe kell vonni, méghozzá a CTRL-J billentyűkombinációval. A megjelenő kérdésre megerősítő választ adva a görbék ezután már egy tárgy görbéi lesznek, még akkor is, ha ez a tárgy jelenleg csupán görbék halmaza, felület nélkül. A Blender könnyen segíthet tárgyunk e hiányosságán, miután pontszerkesztő módba váltva (TAB billentyű) a görbesereg minden pontját kijelöljük. A munka befejezéséhez a végső megoldást az F billentyű használata nyújtja, ezzel a Blendert a felület elkészítésére utasítjuk. Megjegyzendő, hogy ez a billentyű nemcsak a görbefelület létrehozására használható – ha pontszerkesztő módban kiválasztunk három pontot, szintén ezzel a megoldással alkothatunk új, az adott tárgyat alkotó háromszöget.
Az előbb ismertetett eljárás segítségével kis ráfordítással modellezhetjük például egy élő ember fejét is. Ennek megoldásához szükség lesz egy körvonalra, amelyet a földre rajzolunk, és az egyik negyedét egyenlő részekre osztjuk. A modellként szolgáló személy a kör közepén foglal helyet és egy másik személy a körvonalon azonos lépésekben mozgva felvételeket készít róla. A felvételeken ezután valamelyik rajzprogrammal kiemeljük a modell arcélét, vigyázva arra, hogy a kamerával ellenkező oldalon lévő éleket ne keverjük össze a fontosabb, közelebbi oldalon lévő formák vonalával. A profilok meghatározása után ezeket a képeket úgy másoljuk egymásra, hogy a profilt alkotó élek mindegyiken jól láthatóak maradjanak. A Gimpben például ezt az átlátszóság felhasználásával oldhatjuk meg, majd a végeredményt jpg formátumban tároljuk. A Blender lehetőséget nyújt arra, hogy a szerkesztőnézetbe háttérképet töltsünk be. A szerkesztőnézeten álló egérmutató mellett nyomjuk meg a SHIFT-F7 gombokat, majd a képet töltsük be háttérképként. A profilok vonalait követve ezek után rajzoljuk meg a görbéket, és forgassuk el őket a felvételek során követett elmozdulásoknak megfelelő szöggel. Így már nagy vonalakban láthatóvá válik a kialakuló fejforma. Utolsó lépésként a fenti módszerrel hozzuk létre a görbékre feszíthető felületet, és az egyes pontokat szerkesztve alakítsuk ki a végleges formát.

Blob objektumok használata

A harmadik modellezési eljárás már ismerős lehet azok számára, akik a PovRayról szóló (Linuxvilág 2001. június–július, a 92. oldaltól) sorozatot figyelemmel kísérték, ugyanis ebben is találkozhattunk a ***Blob*** objektumokkal. Mint az hamarosan kiderül, a Blender is képes ilyen gumilabdához hasonló tárgyak létrehozására, amelyeket a ***Főmenü-Add-Metaball*** menüpontok segítségével hívhatunk életre. Az első ilyen létrehozott objektumnál a szerkesztő eszköztár segítségével meghatározhatjuk e tárgyak általános tulajdonságait. Az F9 billentyűvel elérhető értékek közül a ***Wiresize*** határozza meg a szerkesztés során használatos számolási pontosságot, a ***Rendersize*** a megjelenítéskor alkalmazandót, míg a ***Threshold*** érték a gömböket övező erőtér erősségét. A későbbiek során ezeket az értékeket csak az elsőként létrehozott gömbnél változtathatjuk meg, és ezek a változások a többi ilyen tárgyra is hatással lesznek. Minden létrehozott „gumilabdánál" negatív hatású erőtér is beállítható, gömbforma helyett pedig hengeralakzatot is kialakíthatunk, amelyet a három tengely egyikével párhuzamosan készíthetünk el. Ebben az esetben lesz értelme a pontszerkesztő módban megjelenő ***Length*** értéknek, míg a ***Stiffness*** értékkel a vastagságot befolyásolhatjuk.
Végezetül foglaljuk össze a későbbiekben megjegyzendő és a munkánkat megkönnyítő billentyűkombinációkat (lásd *táblázatunkban*). Az ilyen összefoglalások esetében a későbbiekben is megkülönböztetem majd a tárgyszerkesztő módot (TSZ) a pontszerkesztő módtól (PSZ).
A sorozat következő részében elkezdjük az ismerkedést az anyagokkal és a felületi mintázatokkal. Tartsanak velem a későbbiekben is, és ne feledjük, hogy egy gyakorlati anyag elsajátítása során a legfontosabb segédeszköz maga a gyakorlás.

Modellezés rácsszerkezetek segítségével

Az utolsó modellezési eljárás a kiindulásul szolgáló tárgyat egy tetszőlegesen módosítható térbeli rácsszerkezet segítségével formálja át. Első lépésként hozzunk létre egy gömböt az ***Főmenü-Add-Mesh-Icosphere*** menüpontokon keresztül. A felosztás mértékét állítsuk háromra. Ezután a formát kialakító objektumot szükséges meghatároznunk, ami egy térbeli rács lesz. A főmenüből kiindulva az ***Add-Lattice*** pontokat kiválasztva létrejön egy új kocka. Ennek tulajdonságait szintén az F9 billentyű használatával előbukkanó panelen állíthatjuk be. A módosító rácsszerkezet jellemzői közül az U, a V, valamint a W értékek határozzák meg a módosításhoz használható pontok számát: ezeknek az értékeknek a változtatásával a rácspontok sűrűsége az értékeknek megfelelő tengely mentén (az U értéke az X tengelynek, a V értéke az Y tengelynek, míg a W-é a Z tengelynek felel meg) alakul át. A rács felosztását szolgáló értékek mellett meghatározhatjuk, hogy a tárgy pontjait milyen összefüggés szerint befolyásolják a rácspontok. Ezek közül a `Lin` lineáris összefüggést határoz meg, a `Card` kapcsoló hatására a módosítandó tárgy kissé sarkított lesz, a `B` pedig a Bezier-Spline közelítéssel alakítja át a tárgyat, amelyet később hozzárendelünk a módosító rácshoz. Miután létrehoztuk a rácsot, és a rácsszerkezetet elképzeléseink szerint átalakítottuk, jelöljük ki először az átalakítandó tárgyat, utána a CTRL billentyűt lenyomva tartva a rácsot is, majd a CTRL-P-vel rendeljük őket egymáshoz. Ekkor már jól látható a változtatás eredménye. Állítsuk be pontosan a rácspontokat az elképzeléseinknek megfelelően , majd a CTRL-SHIFT-A billentyűk alkalmazásával véglegesítsük a változásokat.

Fábián Zoltán (dzooli@freemail.hu, dzooli@yahoo.com) 26 éves, jelenleg programozóként dolgozik. Szabadidejében szívesen kirándul, túrázik. Emellett szeret rajzolni, érdekli a 3D-s grafika és a Linuxszal kapcsolatban minden olyan program és programnyelv, amit még nem ismer vagy nem próbált ki.

Nullmásolás felhasználói szemszögből

Lássuk, mit takar a nullmásolás fogalma, továbbá mire és hol lehet használni.

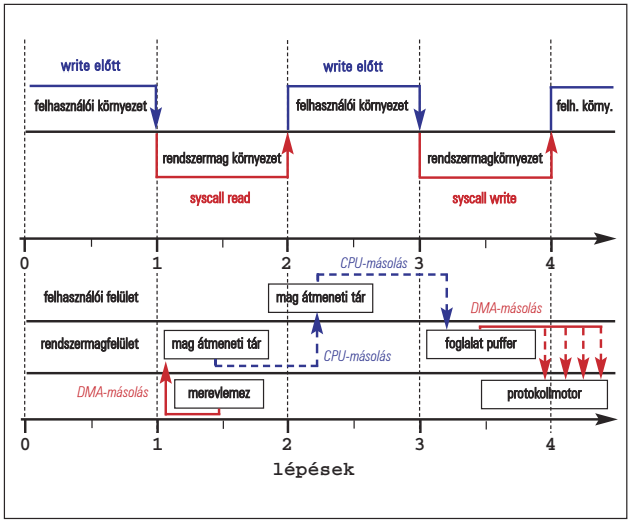
Szinte mindenki hallott már a Linux nullmásolási (zero copy) lehetőségéről, mégis sok emberrel találkozom, akik nem teljesen értik a lényegét. Ezért határoztam úgy, hogy egy cikksorozatban kicsit mélyebbre ások e témakörben, abban a reményben, hogy sikerül eloszlatnom a vele kapcsolatos bizonytalanságot. Jelen írásomban felhasználói szemszögből vizsgálom a nullmásolást, tehát a rendszermag élveboncolását szándékosan mellőztem.

Mi az a nullmásolás?

Ha egy kérdésre meg szeretnénk találni a választ, először magát a kérdést kell pontosan megértenünk. Vizsgáljuk meg, mire van szükség ahhoz, hogy egy egyszerű hálózati démon a fájlokban tárolt adatokat az ügyfeleknek átadhassa. Példakódunk:

```
read(file, tmp_buf, len);
write(socket, tmp_buf, len);
```

Elég egyszerűnek tűnik, nyilván úgy gondolod, hogy két egyszerű rendszerhívás végrehajtása nem igényel túl sok munkát. Sajnos a valóság messze nem így fest. Bár csak két hívást hajtottunk végre, az adatok másolása legalább négyszer megtörténik, és legalább ennyi felhasználói- és rendszermagkörnyezet váltásra is sor kerül. (Valójában a folyamat ennél is összetettebb, de nem akarok elmerülni a részletekben.) Az *1. ábrára* tekintve egy kicsit jobban is átláthatod a történéseket. Felül láthatók a környezetváltások, alul pedig a másolási műveletek. Első lépés: a rendszerhívás hatására a rendszer felhasználóiból rendszermagmódba vált. Az első másolást a DMA-motor végzi el, ami beolvassa a lemezről a fájl tartalmát, és egy a rendszermag címterében található átmeneti tárban tárolja.
A második lépés során a rendszer felhasználói átmeneti tárba másolja az adatokat a rendszermagéból, és visszatér az olvasási hívásból. Most, hogy az adatok bekerültek egy a felhasználói címtérben található átmeneti tárba, megkezdhetik lefelé tartó útjukat.
Harmadik lépésben az írási hívás hatására a rendszer felhasználóiból rendszermagkörnyezetbe vált. A harmadik másolás során az adatok újra egy a rendszermag címterében található átmeneti tárba kerülnek. Természetesen itt egy másik, kifejezetten a foglalatokhoz tartozó átmeneti tárról van szó.
A negyedik lépés az írási hívás visszatérése, ezzel megtörténik a negyedik környezetváltás. Ettől független és aszinkron módon egy negyedik másolás is lezajlik, amikor a rendszermag átmeneti tárában található adatokat a DMA-motor átadja a protokollmotornak. Talán felmerül benned a kérdés: „Mi az, hogy független és aszinkron módon? Hát nem történt meg az adatok átvitele még a hívás visszatérése előtt?" Az, hogy maga a hívás visszatér, önmagában még nem biztosítja a másolás tényleges elvégzését, sőt még annak megkezdését sem. Egyszerűen csak annyit jelez, hogy az ethernet illesztőprogram várólistájában voltak szabad leírók, és amit adtunk neki, azt elfogadta átvitelre váró adatként. Lehet, hogy jó néhány csomag van még a várólistában a miénk előtt. Hacsak maga az eszköz vagy az illesztőprogram nem állít fel fontossági várólistákat, az adatok küldése érkezési sorrend szerint történik. (Az *1. ábra* `fork` hívással párosuló DMA-másolása is jelzi, hogy az utolsó másolás késlekedhet.)

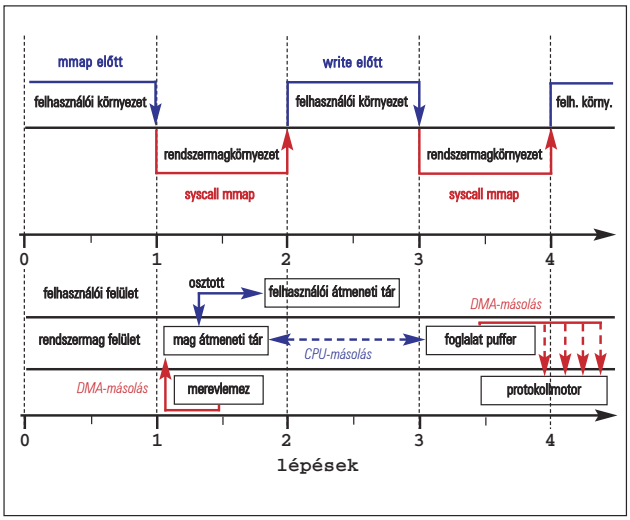


1. ábra Másolás két rendszerhívással

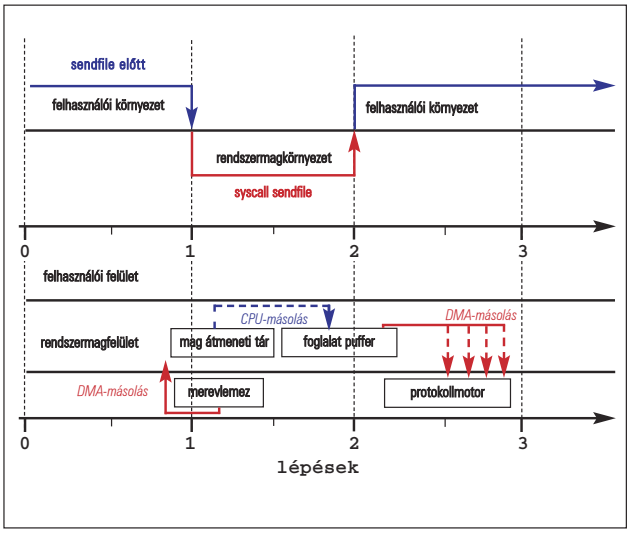
Mint láthatod, a cél eléréséhez egyáltalán nem lenne szükség ennyi adatmásolásra. Ha a másolgatások számát sikerülne csökkenteni, egyben a rendszer terhelése is kisebb lenne, így növekedne a teljesítménye. Illesztőprogram-fejlesztőként közvetlenül a vassal dolgozom, és bizony találok érdekes lehetőségeket. Egyes eszközök képesek megkerülni a központi memóriát, és közvetlenül a másik eszköznek adni át az adatokat. Így teljesen szükségtelen másolatot készíteni a központi memóriába, meg úgy általában véve nekem tetszik a dolog, de sajnos nem minden eszköz képes erre. Arra is gondolni kell, hogy a lemezről beolvasott adatokat a hálózat igényeinek megfelelő formátumba át kell csomagolni – és máris kezdünk „bonyolódni". A többletterhelés csökkentése érdekében először próbáljuk meg kiküszöbölni a rendszermag átmeneti tár és a felhasználói átmeneti tár közötti másolásokat.
Az egyik lehetőség a másolás elhagyására a `read` hívás mellőzése és az `mmap` használata. Például:

```
tmp_buf = mmap(file, len);
write(socket, tmp_buf, len);
```

A folyamatot a *2. ábra* szemlélteti. A környezetváltások változatlanok.
Elsőként az `mmap` rendszerhívás hatására a DMA-motor a fájl tartalmát egy rendszermag átmeneti tárba másolja. Ezután a rendszer az átmeneti tárat megosztja a felhasználói folyamattal,



2. ábra Az mmap használata



3. ábra A read és a write hívások kiváltása sendfile hívással

a felhasználó és a rendszermag memóriaterülete között nem végez másolást.
Másodikként a `write` rendszerhívás hatására a rendszermag az adatokat a foglalatokhoz rendelt átmeneti tárba másolja.
A harmadik másolás akkor történik, amikor a DMA-motor az adatokat átadja a rendszermagfoglalat átmeneti tárból a protokollmotornak.
Azzal, hogy `read` helyett `mmap` hívást használunk, felére csökkentettük a rendszermag által másolandó adatok mennyiségét. Már ezzel is számottevő eredményt érhetünk el, ha nagy mennyiségű adatról van szó. Csakhogy megfizetjük az árát is, az `mmap` és `write` használatának bizonyos buktatói is vannak. Az egyik lehetséges hiba az, amikor memória-hozzárendelést hozol létre egy fájlhoz, majd `write` hívást hajtasz végre, de közben egy másik folyamat ugyanezt az állományt csonkolja. Az írási műveletet egy `SIGBUS` sínhibajelzés fogja megszakítani, ugyanis hibás memóriaeléréssel próbálkoztál. Alapesetben a hibajelzés a vétkes folyamat kilövését okozza és memóriakiíratással jár, ezt viszont hálózati kiszolgálóknál általában igyekszünk elkerülni. A gondot kétféleképpen oldhatjuk meg.

Az első lehetőség egy jelkezelő (signal handler) telepítése a `SIGBUS` jelzéshez, majd egyszerű visszatérést alkalmazni a jelkezelőben. Így a `write` rendszerhívás a megszakítás előtt sikeresen kiírt bájtok számával tér vissza, az `errno` pedig sikert jelez. Hadd mutassak rá, miért rossz ez a megoldás: csak a tüneteket kezeli, de az okot nem szünteti meg. A `SIGBUS` azt tudatja velünk, hogy az adott folyamat kapcsán valami nagyon rosszul sült el, így ettől a módszertől inkább tartózkodnék.
A másik megoldás a fájl kibérlése (file leasing) a rendszermagtól, amit a Windows világában alkalmi zárolásnak is neveznek. A bérlést végrehajtva a fájlleírón a fájlt ideiglenesen bérletbe veheted. Ezt követően olvasási, illetve írási bérlést kérsz a rendszermagtól, így amikor a másik folyamat az éppen küldött fájlt megpróbálja csonkolni, a rendszermag egy valós idejű `RT_SIGNAL_LEASE` jelzést küld neked. Így tudomásodra juthat, hogy a rendszermag megszünteti a fájlra szóló ideiglenes bérletedet. Írási hívásod még azelőtt megszakad, hogy a programod érvénytelen címhez férne hozzá, és a `SIGBUS` hibajelzés miatt idő előtt elhalálozna. A `write` hívás a megszakítás előtt sikeresen kiírt bájtok számát adja vissza, az `errno` pedig sikert jelez. Az alábbi példa szemlélteti, hogyan lehet fájlt bérelni a rendszermagtól:

```
if(fcntl(fd, F_SETSIG, RT_SIGNAL_LEASE) == -1 {
        perror("rendszermag bØrlØs
                ➥beÆll tÆsa");
        return -1;
}

/* l_type ØrtØke F_RDLCK vagy F_WRLCK
   lehet */
if(fcntl(fd, F_SETLEASE, l_type)){
         ➥bØrlØst pus beÆll tÆsa");
        return -1;
}
```

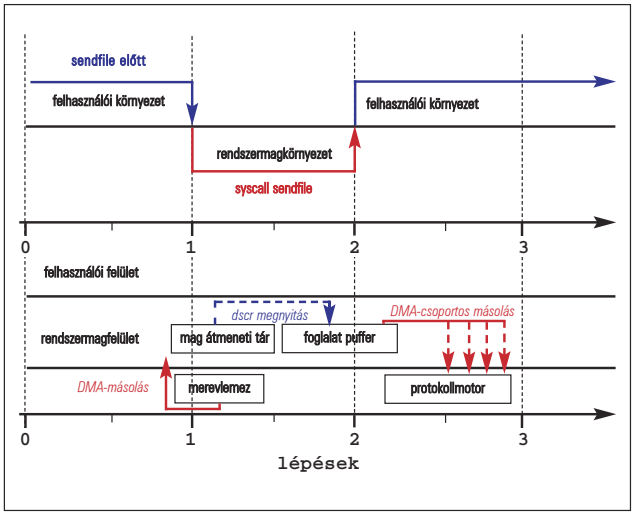
A bérlést az `mmap` használata előtt kell végrehajtanod, és csak miután végeztél, szabad megszüntetned. Utóbbit az `fcntl F_SETLEASE` hívásával érheted el, a bérlés típusát `F_UNLCK` értékre állítva.

Sendfile

A 2.1-es változatú rendszermagban jelent meg a hálózati vagy két helyi fájl közti adatátviteleket leegyszerűsítő `sendfile` rendszerhívás. A `sendfile` révén nemcsak a másolások, hanem a környezetváltások száma is csökkenthető. Használata a következő:

```
sendfile(socket, file, len);
```

A folyamatot a *3. ábra* szemlélteti.
Először a `sendfile` rendszerhívás hatására a DMA-motor a fájl tartalmát egy rendszermag átmeneti tárba másolja. Ezután az adatokat a rendszermag a foglalatokhoz rendelt átmeneti tárba másolja.
Másodszor: a harmadik másolás akkor zajlik le, amikor a DMA-motor a rendszermagfoglalatokhoz rendelt átmeneti tárában található adatokat átadja a protokollmotornak.
Nyilván eszedbe jut, hogy vajon mi történik, ha egy másik folyamat csonkolja az éppen `sendfile` rendszerhívással továbbított állományt? Ha nem jegyzünk be jelkezelőt, a `sendfile` egyszerűen a megszakítás előtt sikeresen átvitt bájtok számával visszatér, és az `errno` is sikert jelez.



4. ábra A gyűjtögetést támogató eszköz képes arra, hogy több memóriarészből olvassa ki az adatokat, így eggyel kevesebb másolásra van szükség

Ha még a `sendfile` meghívása előtt kibéreled a fájlt a rendszermagtól, változatlan viselkedést és visszatérési értéket tapasztalsz. Megkapod a már ismert `RT_SIGNAL_LEASE` jelzést is, mielőtt még a `sendfile` visszatérne.
Az eddigiek alapján sikerült elkerülnünk, hogy a rendszermag nagyszámú másolást végezzen, de egy másolás még hátravan. Vajon ezt is el lehet hagyni? Természetesen igen, ha a vas is nyújt hozzá némi segítséget. Ha el szeretnénk felejteni a rendszermag által végzett adatkettőzéseket, akkor az adatgyűjtést támogató hálózati csatolóra van szükségünk. Ez mindössze annyit jelent, hogy a továbbításra várakozó adatoknak nem muszáj összefüggő memóriaterületen lenniük, hanem több helyre is szétszórhatók. A 2.4-es rendszermagban a foglalat átmenetitár-leíróját úgy módosították, hogy teljesíti ezeket a követelményeket – Linux alatt ezt nevezzük nullmásolásnak. Az elgondolás révén nemcsak a többszörös környezetváltás, de a processzor által végzett adatkettőzések is elhagyhatók. A felhasználói alkalmazások szemszögéből nézve semmi sem változott, a kód továbbra is így alakul:

```
sendfile(socket, file, len);
```

A lezajló folyamatot a *4. ábra* szemlélteti.
Első lépésként a `sendfile` rendszerhívás hatására a DMA-motor a fájl tartalmát egy rendszermag átmeneti tárba másolja. Második lépésben az adatok másolása helyett a rendszer csak a helyüket és a méretüket megadó leírókat fűzi hozzá a foglalat átmeneti tárhoz. A DMA-motor a rendszermag átmeneti tárból közvetlenül a protokollmotornak adja át az adatokat, így az utolsó másolás is elmarad.
Mivel az adatokat a lemezről be kell olvasni a memóriába, majd ki kell küldeni a hálózatra, néhányan talán nehezményezik, hogy nem igazi nullmásolást végzünk. Ez részben így is van, de az operációs rendszer szempontjából elértük a célunkat, ugyanis az adatokat már nem többszörözzük rendszermag átmeneti tárak között. Nullmásolás használatakor nemcsak a kevesebb másolás által javítunk a teljesítményen, hanem kevesebb környezetváltásra van szükség, nem kavarjuk össze a processzor gyorsítótárát, és ellenőrző összegeket sem kell számítani.
Most, hogy tisztában vagyunk a nullmásolás lényegével, a gyakorlatban is alkalmazzuk elméleti tudásunkat. A teljes forráskódot a ➲ http://www.xalien.org/articles/source/sfl-src.tgz címen vagy a 45. CD Magazin/Zero_Copy könytárában érheted el. Kicsomagolni úgy tudod, hogy kiadod a `tar --zxvf sfl-src.tgz` parancsot. A `make` paranccsal egyrészt lefordíthatod a kódot, másrészt egy véletlenszerű tartalommal feltöltött, ***data.bin*** nevű adatállományt is létrehozhatsz.
Lássuk a kódot, a fejlécfájlokkal kezdve:

```
/* sfl.c sendfile pØldaprogram
Dragan Stancevic visitor@xalien.org 09-09-2002
fejlØc neve                    f ggvØny/vÆltoz
```

További érdekességek

Forráskód és használat

A teljes forráskódot a 45. CD Magazin/Zero_Copy könyvtárában találod. A `tar --zxvf sfl-src.tgz` paranccsal bonthatod ki, majd a `make` paranccsal hozhatod létre a futtatható állományt. A könyvtárban egy `sfl` nevű futtatható fájlnak kell létrejönnie, átadott értékek nélkül indítva a program rövid használati útmutatót jelenít meg. Mind a kiszolgáló-, mind az ügyféloldalon meg kell adnod a kiszolgáló IP-címét.

Példa:
Küldő oldal:
visitor@xalien-saucer:~/sfl-src> ./sfl s 10.0.0.2
Server binding to [10.0.0.2]
Server sent 10240 bytes.
visitor@xalien-saucer:~/sfl-src>

Fogadó oldal:
visitor@earth:~/sfl-src> ./sfl r 10.0.0.2
Client connecting to [10.0.0.2]
Client received 10240 bytes.
visitor@earth:~/sfl-src>

Megjegyzések a kódhoz

1. Én az 1033-mas kaput használtam, ha te a saját rendszereden ezt más célra használod, válassz másikat.
2. A példaprogram egyszerűsítése és rövidítése érdekében a 10 KB-os átmeneti tárat a verembe raktam. Saját kódodban a `malloc` függvénnyel foglalhatsz átmeneti tárakat.
3. A cikkben szereplő program lerövidítése érdekében elhagytam a `close` rendszerhívás visszatérési értékének vizsgálatát. Egy valódi programban természetesen el kell végezned az ellenőrzést.

Példák

Ha mélyebben is meg szeretnél ismerkedni a `sendfile` rendszerhívás használatával, pillants bele néhány olyan alkalmazásba, amelyik használja:
Apache ➲ http://www.apache.org
Samba ➲ http://www.samba.org
Mozilla ➲ http://www.mozilla.org
Pure-FTPd ➲ http://pureftpd.sf.net
Ha érdekel a rendszermag fájlbérlési szolgáltatása, nézz bele a Samba forráskódjába, az *smbd/oplock_linux.c* fájlba.

```
#include <stdio.h>           /* printf,
                                perror */
#include <fcntl.h>           /* open */
#include <unistd.h>          /* close */
#include <errno.h>           /* errno */
#include <string.h>          /* memset */
#include <sys/socket.h>      /* socket */
#include <netinet/in.h>      /* sockaddr_in */
#include <sys/sendfile.h>    /* sendfile */
#include <arpa/inet.h>       /* inet_addr */
#define BUFF_SIZE (10*1024)  /* a tmp Ætmeneti
                                tÆr mØrete */
```

Az alapvető foglalatműveletekhez szükséges `<sys/socket.h>` és `<netinet/in.h>` mellett a `sendfile` rendszerhívás törzstípusát is meg kell adnunk. Ezt a `<sys/sendfile.h>` `server` jelzőjében találjuk:

```
/* k ld nk vagy fogadunk */
if(argv[1][0] == 's') is_server++;

/* le r k megnyitÆsa */
sd = socket(PF_INET, SOCK_STREAM, 0);
if(is_server) fd = open("data.bin", O_RDONLY);
```

Ugyanaz a program kiszolgálóként (küldőként) és ügyfélként (fogadóként) is használható. Ellenőrizni kell a megfelelő parancssori átadott értéket, majd – szükség szerint – az `is_server` jelzőt beállítva küldő módba válthatunk. Az `INET` protokollcsalád egy folyamfoglalatát is megnyitjuk. Mivel a program most kiszolgálóként üzemel, adatokat kell küldenie az ügyfeleknek, ezért az adatállományt is megnyitjuk. Az adatok küldésére a `sendfile` rendszerhívást használjuk, így nincs szükség a fájl tartalmának beolvasására és a saját programunk átmeneti tárában történő tárolására. A kiszolgáló címe:

```
/* a mem ria t rlØse */
memset(&sa, 0, sizeof(struct sockaddr_in));

/* az adatszerkezet kezdı ØrtØkadÆsa */
sa.sin_family = PF_INET;
sa.sin_port = htons(1033);
sa.sin_addr.s_addr = inet_addr(argv[2]);
```

Töröljük a kiszolgáló címét tároló adatszerkezet tartalmát, majd adjuk meg a protokollcsaládot, a kaput és a kiszolgáló IP-címét; az utóbbit a program átadott értékként kapja meg. A kapu bedrótozott módon az egyébként használaton kívüli 1033-mas lesz. Azért ez, mert e kaputartomány használatához az adott rendszeren már nincs szükség rendszergazdai jogosultságra.
A kiszolgálói ág:

```
if(is_server){
      int client;  /* œj  gyfØlfoglalat */
      printf("A kiszolgÆl  a [%s] kaput
            ➥hasznÆlja.\n", argv[2]);

      if(bind(sd, (struct sockaddr *)&sa,
            ➥sizeof(sa)) < 0){
            perror("bind");
            exit(errno);
      }
```

Kiszolgálóként címet kell rendelnünk a foglalat leírójához. Ezt a `bind` rendszerhívással tehetjük meg, amely a foglalat leírójához (sd) egy kiszolgálócímet rendel (sa):

```
if(listen(sd,1) < 0){
      perror("listen");
      exit(errno);
}
```

Mivel folyamfoglalatot használunk, valahogy ki kell nyilvánítanunk kapcsolatok fogadására irányuló szándékunkat, és meg kell adnunk a kapcsolati várólista méretét is. A kapcsolati várólista (backlog queue) hosszát 1-re állítottam, de a már felépült kapcsolatok fogadása érdekében ennél nagyobb értéket is meg szoktak adni. A rendszermag régebbi változatainál a kapcsolati várólista segítségével előzték meg a syn-elárasztásos támadásokat. Mivel a `listen` rendszerhívás időközben módosult, és csak a felépült kapcsolatok beállításait adja meg, a kapcsolati várólistára többé nincs szükség; pontosabban a rendszermag `tcp_max_syn_backlog` beállítása gondoskodik arról, hogy megelőzze a rendszer ellen irányuló syn-elárasztásos támadásokat:

```
if((client = accept(sd, NULL, NULL)) < 0{
      perror("accept h vÆs");
      exit(errno);
}
```

Az `accept` rendszerhívás a függőben lévő kapcsolatok várólistájának első kérését feldolgozva új csatlakoztatott foglalatot hoz létre. A hívás visszatérési értéke az új kapcsolat leírója. Ezzel a foglalat alkalmassá vált a `read`, `write`, `poll` és `select` rendszerhívások használatára:

```
if((cnt = sendfile(client, fd, &off,
BUFF_SIZE)) < 0) {
      perror("sendfile h vÆs");
      exit(errno);
}
print("A kiszolgÆl  %d bÆjtot k ld tt el.\n",
      ➥cnt);
close(client);
```

A foglalatleíró használatával létrejött egy kapcsolat, tehát megkezdhetjük az adatok továbbítását a távoli rendszerre. Erre a `sendfile` rendszerhívást használjuk, amelynek törzstípusa Linux alatt a következő:

```
extern ssize_t
sendfile (int __out_fd, int __in_fd, off_t
          ➥*offset, size_t __count) __THROW;
```

Az első két átadott érték fájlleíró. A harmadik az eltolás, ez adja meg, hogy a `sendfile` honnan kezdje meg az adatok küldését. A negyedik a küldeni kívánt bájtok száma. Ahhoz, hogy a `sendfile` nullmásolással továbbítsa az adatokat, a hálózati csatolónak támogatnia kell az adatgyűjtést. Azoknál a protokolloknál, amelyek ellenőrző összegeket számítanak – ilyen többek közt a TCP és az UDP –, az ellenőrző összegeket is ki kell tudnod számítani. Szerencsére akkor sem kell lemondanod a `sendfile` használatáról, ha a hálózati csatolód régi, és nem támogatja ezeket a szolgál-

tatásokat. A különbség mindössze annyi, hogy a rendszermag ekkor küldés előtt összerendezi az átmeneti tárakat.

Hordozhatósági kérdések

A `sendfile` rendszerhívással általános esetben az a baj, hogy – az `open` hívással ellentétben – nincs szabványos megvalósítása. A Linux, Solaris és HP-UX alatti megvalósítások kismértékben eltérnek egymástól, és ez gondot jelenthet a hálózati alkalmazásaikban a nullmásolási lehetőséget kihasználni kívánó fejlesztőknek.

Az egyik eltérés a megvalósítások között az, hogy a linuxos két fájlleíró között teszi lehetővé az adatátvitelt, azaz fájl–fájl és fájl–foglalat párosításokat is kezel, míg a HP-UX és a Solaris megvalósítás csak fájl–foglalat átvitelekhez használható.

A másik különbség az, hogy Linux alatt nem lehet vektoros átviteleket végezni. A Solaris és a HP-UX `sendfile` megvalósítása külön átadott értékek használatával szükségtelenné teszi az átvivendő adatokhoz csatolt fejlécek miatt elvégzendő többletmunkát.

Előre tekintve

A Linux alatti nullmásolás megvalósítása még nem fejeződött be, és a közeljövőben valószínűleg módosulni fog a szolgáltatás működése – várhatóan újabb lehetőségekkel bővül. Például a `sendfile` nem támogatja a vektoros átviteleket, és a kiszolgálók – mint az Apache vagy a Samba – több, `TCP_CORK` jelzővel jelölt `sendfile` hívást kénytelenek végrehajtani. Ez a jelző arról tudósítja a rendszert, hogy újabb `sendfile` hívásokkal további adatokat akarunk küldeni. A `TCP_CORK` azonban nem fér össze a `TCP_NODELAY` jelzővel, illetve akkor is használjuk, ha az adatokhoz fejlécet akarunk csatolni.

Nagyszerű példája ez annak, amikor a vektoros átvitellel több `sendfile` hívást ki lehetne váltani, illetve a jelenlegi megvalósítással járó késleltetéseket el lehetne hagyni.

Ugyancsak kellemetlen megkötés, hogy a jelenlegi `sendfile` csak 2 GB-nál kisebb fájlok továbbítására használható. Manapság egyáltalán nem szokatlanok az ekkora állományok, viszont ekkora adatmennyiséget megkettőzni küldés közben több mint kellemetlen. Mivel ilyen esetben az `mmap` és a `sendfile` egyaránt használhatatlanok, egy `sendfile64` megvalósítás igencsak jól jönne az újabb rendszermagváltozatokban.

Összegzés

Korlátai ellenére a nullmásolás hasznos szolgáltatás, és remélem, írásomban elegendő adatot találtál ahhoz, hogy te is elkezdd használni a saját programjaidban. Ha érdekel a téma, olvasd el a hamarosan megjelenő második részt, amelyben a rendszermag szempontjából járom körül a nullmásolás kérdéskörét.

Linux Journal 2003. január, 103. szám

Dragan Stancevic

Húszas éveinek végén járó rendszermag- és eszközfejlesztő. Szakmája szerint programmérnök, de az alkalmazott fizika is érdekli, szabadidejében különlegesen magas elektromos feszültségekkel szeret kísérletezni.

Gyorsrendezés elemzése DDD segítségével

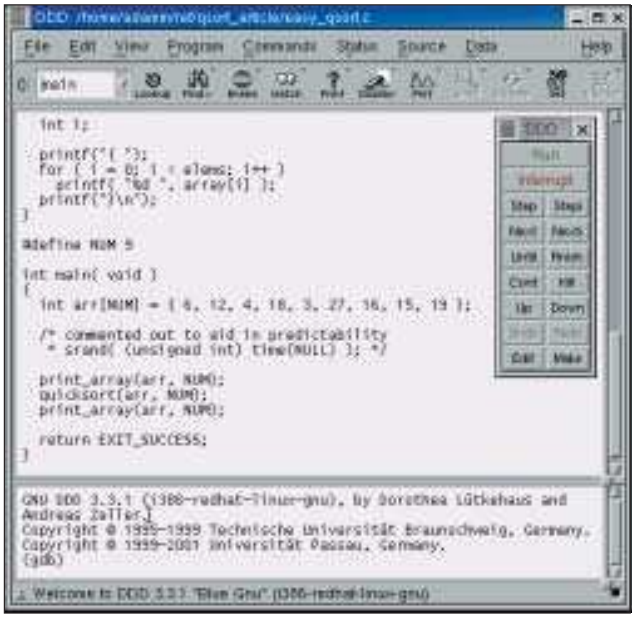
Ismerkedjünk meg egy jó hibakeresővel, és értsük meg a gyorsrendezést!

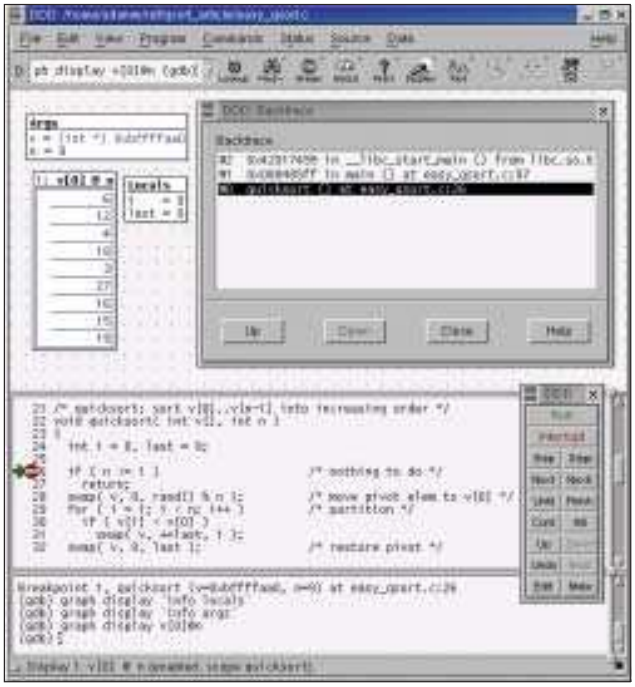
A rendezés a számítógépek által az egyik leggyakrabban végrehajtott művelet, és erre a gyorsrendezés az egyik leghatékonyabb módszer. Írásomban egyrészt bemutatom, mennyire hasznos lehet egy grafikus hibakereső, másrészt ismertetem a gyorsrendezés elvét.
A DDD (Data Display Debugger) egy szabad grafikus felület, amely számos hibakereső felett használható. A továbbiakban a DDD segítségével vizsgáljuk meg a gyorsrendezés egyik egyszerű C-megvalósítását.
Először is be kell szerezni a DDD egy példányát, majd telepíteni kell. Bináris és forráscsomagokat egyaránt találhatunk, ebben RPM alapú terjesztések esetében az rpmfind.net lehet a segítségünkre, Debian-csomagokat pedig a debian.org segítségével keríthetünk. Cikkem írásakor 3.3.1-es DDD-t és Red Hat 7.3-as terjesztést használtam. A továbbiak során három dolgot feltételezünk:

1. GNU/Linux alapú számítógéppel rendelkezünk, ami be van kapcsolva;
2. ismerjük az alapvető C-fogalmakat, mint tömbök, ciklusok és önhívás (recursion);
3. megfelelő C-fordítóval rendelkezünk, például a GNU GCC-vel. Még ha fogalmad sincs a programozásról, próbáld meg áttanulmányozni a kódot – hasznát látod a későbbiek folyamán.

C. A. R. Hoare 1962-ben vázolta fel a gyorsrendezés algoritmusát, amit negyven év elteltével még mindig széles körben használnak. A gyorsrendezés talán „oszd meg és uralkodj” felfogása miatt lett „gyors”, mivel a nagyobb elemeket kisebbekre osztva szükségtelenné teszi nagy számú összehasonlítás elvégzését. Ellentéte például a legkisebb kiválasztásának elvére épülő rendezés, amely minden elemet összehasonlít az összes többivel. Természetesen ez nem feltétlenül jelenti azt, hogy a gyorsrendezés mindig gyorsabb vagy ez a legjobb rendezési mód, egyszerűen csak jó ismerni. Az írásomban szereplő megvalósítás nem javított és nem bővíthető, kizárólag egész számokból álló tömbökkel működik.
A kód jelentős része ***Brian W. Kernighan*** és ***Rob Pike*** „The Practice of Programming” című művéből származik.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

/* Brian W. Kernighan Øs Rob Pike The
Practice of Programming
* c mß k nyvØnek 32-34. oldalÆr l
*/

/* swap: v[i] Øs v[j] felcserØlØse */
void swap(int v[], int i, int j)
{
       int tmp;
tmp = v[i];
v[i] = v[j];
v[j] = tmp;
}

/* quicksort: a v[0]..v[n-1] elemek n vekv1
   sorrendbe rendezØse */

void quicksort(int v[], int n)
{
       int i = 0, last = 0;
       if (n<=1)
              return; /* semmit nem csinÆl */
       swap(v, 0, rand() % n);
/* az elvÆlaszt  elem mozgatÆsa v[0]-ba */

for(i = 1; i < n, i++)    /* kettØosztÆs */
              if ( v[i] < v[0])
                     swap(v, ++last, i);

swap(v, 0, last);
/* az elvÆlaszt  elem visszaÆll tÆsa */

quicksort(v, last);
/* a kisebb Ørtøkek rendezØse */

quicksort(v+last+1, n-last-1);
/* a nagyobb Ørtøkek rendezØse */

}


void print_array(const int array[], int elems)
{
       int i;
       printf("{");

for(i = 0; i < elems; i++)
              printf("%d ", array[i]);

printf("}\n");

}


#define NUM 9

int main(void)
{
       int arr[NUM] =
           ➥{6, 12, 4, 18, 3, 27, 16, 15, 19};

/* megjegyzØsbe raktuk, hogy elıre lÆtni
   lehessen a rendezØs menetØt
   srand( (unsigned int) time(NULL) ); */

print_array(arr, NUM);
```





```
quicksort(arr, NUM);
print_array(arr, NUM);
return EXIT_SUCCESS;
}
```

Lépésről lépésre

Mentsd a kódot egy ***easy_qsort.c*** nevű állományba. Ezután fordítsd le:

```
$ gcc -Wall -pedantic -ansi -o qsortof -g
  ➥easy_qsort.c
```

A GCC-nek megadott kapcsolók közül a `-g` a legfontosabb, amely hibakeresési hivatkozással (symbol) látja el a kódot. Próbáld lefuttatni a programot, így kiderül, minden rendben van-e:

```
$ ./qsortof

{ 6 12 4 18 3 27 16 15 19 }

{3 4 6 12 15 16 18 19 27 }
```

A kimenet első sora a rendezetlen tömb, a másik pedig a gyorsrendezés futtatása utáni helyzetet tükrözi. Vajon hogyan működik? Hívjuk segítségül új barátunkat, a DDD-t:

```
$ ddd qsortof
```

Ezzel elindul a DDD. Zárd be a felbukkanó ***Tipp*** vagy ***Névjegy*** ablakot, és az első ábrán láthatóhoz hasonlók tárulnak eléd. Nem árt, ha bekapcsolod a sorok számozását. Jelöld be a ***Display Source Line Numbers*** (Forráskódsorok sorszámának megjelenítése) négyzetet, ezt az ***Edit, Preferences, Source*** (Szerkesztés, Beállítások, Forrás) menüpont alatt találod.
Most már csak egy töréspontot kell hozzáadnunk, és indulhat a hibakeresés.
Jelöld ki a semmittevő sort úgy, hogy az oldalt megjelenő sorszámára kattintasz. Ezután a () gombnál kattints a ***Set/Delete Breakpoint*** (Töréspont hozzáadása/törlése) parancsra, és a parancsablakban kattints a ***Run*** (Futtatás) gombra.
Ekkor egy piros stopjel jelenik meg a töréspontot tartalmazó sorban, illetve egy zöld nyíl is felbukkan ugyanitt (ez jelzi az éppen végrehajtandó kódrészletet). Jelenítsünk meg néhány belső adatot a DDD segítségével. Először válaszd a ***Data, Display Local Variables*** (Adat, Helyi változók megjelenítése) menüpontot. A második legyen a ***Data, Display Arguments*** (Adat, Átadott értékek megjelenítése), majd következzen a ***Status, Backtrace*** (Állapot, Visszakövetés). Végül gépeld be a `graph display v[0]@n` parancsot a konzolablakba, majd nyomd le az ENTER billentyűt. Ekkor felbukkannak a `v[]` tömb elemei, 0-tól n-ig (lásd a második ábrát).

Vajon mi történik odabent?

A töréspontot úgy adtuk meg, hogy ha az átadott tömb csak egyetlen elemet tartalmaz vagy üres, akkor önhívással folyik, a vizsgálat feltétlenül szükséges, hiszen ha egy vagy nulla elemet adunk át, akkor le kell állítani az önhívást (erről később még lesz szó).
A vizsgálat után fogunk egy elválasztó elemet, és a tömb elejére mozgatjuk. Kattints a ***Next*** (Tovább) gombra, míg a zöld nyíl a `swap` függvény hívásáig nem ugrik. A ***Next*** gombra kattintva sorról sorra haladhatsz, az alfüggvények meghívása nélkül; a ***Step*** (Lépés) gombra kattintva viszont az alfüggvényekbe is belépsz. Kattints még egyszer a ***Next*** (Tovább) gombra, és figyeld, hogy mi történik.
Az én gépem felcserélte `v[]` nulladik és első elemét, vagyis a `rand() % n` hívás 1-et adott vissza. Ha többször is megvizsgálod a programot, észre fogod venni, hogy a `rand() % n` mindig 1-et ad. Véletlen létére elég egyhangú, nemde? Ebben a példában az `srand()` hívás megjegyzésbe tételével elmarad az álvéletlenszám-generátor megbolygatása, ezért a `rand()` mindig megjósolható eredményt ad.
Az így meghatározott elválasztó elemet használjuk fel arra, hogy a számokat nála kisebbekre és nagyobbakra osszuk. Az elválasztó elem azért került a `v[0]` helyre, mert amíg a teljes tömböt át nem vizsgáltuk, addig nem ismerjük a pontos helyét. A kettéosztást végző ciklus végiglépked a tömb összes elemén, és összehasonlítja őket az elválasztó elemmel; ez esetemben a 12 volt. A `last` elem növelése előzetesen történik (preincre-

ment), így az egyes cellában lévő elemet önmagával cseréljük fel. Igen, én is tudom, feleslegesen. Az algoritmus hatékonyabbá tétele kellemes feladat az önmaguk sanyargatását élvező olvasók számára. Még meg akartam említeni, hogy az ***Interrupt*** (Megszakítás) gombra kattintva a program futását bármikor megszakíthatjuk, majd a ***Run*** (Futtatás) gombbal újraindíthatjuk. Kattintgass a ***Next*** (Tovább) gombra, amíg az `i` értéke 3, a `last` változóé pedig 2 lesz; ehhez figyeld a ***Locals*** (Helyi változók) ablak tartalmát. A kettéosztó hurok `if` elágazása most a 18-at és a 12-t hasonlítja össze. A vizsgálat sikertelenül zárul (kattints), ezért az `i`-t növeljük (kattints), a `last` változó értéke 2 marad.
Maradjunk szoros kapcsolatban a ***Next*** gombbal, amíg az `i` értéke 9 nem lesz. A tömb most a következőképpen néz ki:

```
{ 12 6 4 3 18 27 16 15 19 }
```

Újra kattintás, és a 12, amely eddig a kisebb számok között tévelygett, helyet cserél a 3-mal.
Miután az elválasztó elem az eredeti helyére kerül, önhívással újra meghívjuk a `quicksort` függvényt, miközben átadunk neki egy mutatót `v[0]`-ra, és közöljük vele, hogy egy háromelemű – a kisebb számokat tartalmazó - tömböt fog kapni. Ezután újra meghívjuk, de ezúttal `v[4]`-re mutatunk és ötelemű tömböt adunk neki, most a nagyobb számokkal. Újra és újra meghívjuk a `quicksort` függvényt, egészen addig, ameddig az átadott tömbökben csupán egyetlen elem marad. Ekkor megindul a hívások visszatérése – elsőként az utolsó –, és mire a `main` függvénybe érkezünk, rendezett tömböt kapunk. Kifújhatjuk magunkat!

Ami jól jöhet

Ha elég mélyen elmerülsz az önhívású `quicksort` hívások mocsarában, egy ponton a `v[0]@n` ablak el fog tűnni. Egy megfelelő gombot létrehozva egy pillanat alatt újra előcsalogathatod. Gombot a ***Commands, Edit Buttons..., Data Buttons*** (Parancsok, Gombok szerkesztése, Adatgombok) menüpont segítségével hozhatsz létre. A szövegbeviteli mezőbe gépeld be az alábbiakat:

```
graph display v[0]@n // Varray
```

Egy ***Varray*** nevű gombnak kell megjelennie a ***Data*** (Adat) ablak tetején. Ha a `v[0]@n` ablak olvas (letiltódik), kattints rá az egér jobb gombjával, és válaszd az ***Undisplay*** (Elrejtés) parancsot. Ezután kattints a ***Varray*** gombra. Ha az ablak továbbra sem jelenik meg, próbálj néhányszor rákattintani a ***Next*** (Tovább) gombra, és próbálkozz újra.

A múlt bűvöletében

Korábban bekapcsoltattam veled a ***Backtrace*** (Visszakövetés) ablakot. Miközben `quicksort()` hívások tornyosulnak feletted, a ***Backtrace*** ablak soraira kattintva érdekes lehet belekukkantani a verem tartalmába. Láthatod, hogyan jutottál el a pillanatnyi függvényhívási környezethez, és a futás más időpontjaiban milyen adatok voltak a veremben.

A gépi kód ablak

Ha eléggé elkábultál, tégy egy kísérletet a ***View, Machine Code*** (Nézet, Gépi kód) paranccsal, amellyel megjelenítheted az éppen folyó műveletek assembly utasításait.

További érdekes algoritmusok és adatszerkezetek

A DDD csodálatos látvány, amikor láncolt listákat vagy más adatszerkezeteket jelenít meg. Próbáld ki!

Linux Journal 2003. január, 105. szám

Adam Monsen
Érdeklődő természetű orvostanhallgató, jelenleg a washingtoni classmates.com kezdeményezés programozója. Szeret zongorázni, szörfözni, amatőr artista. GNU/Linux Red Hat 7.3 alapú gépén Perl, Java és néha C nyelven kódolgat.

KAPCSOLÓDÓ CÍMEK

A Limit Theorem for „Quicksort", *Uwe Rösler*, 1999. május 2.:
➲ http://citeseer.nj.nec.com/rosler99limit.html
DDD-honlap ➲ http://www.gnu.org/software/ddd
Quicksort, *C.A.R. Hoare*, 1962. április
➲ http://www3.oup.co.uk/computer_journal/hdb/Volume_05/Issue_01/050010.sgm.abs.html
Brian W. Kernighan és *Rob Pike* "The Practice of Programming" 32-34. oldal (ISBN: 0-201-61586-X)
GDB honlap ➲ http://www.gnu.org/software/gdb

Linux alapú gőzturbina-próbapad

Vajon a szentpétervári Központi Kazán és Turbina Intézet hogyan garantálja a biztonságot és a pontos vezérlést az erőműturbinák kipróbálásakor?

Annak ellenére, hogy a matematikai modellek és a számítógépek teljesítményének elképesztő növekedése révén szinte bármit lehet szimulálni vagy ki lehet számolni, bizonyos területeken a gyakorlati tapasztalatok nem vesztettek a jelentőségükből, és nem pótolhatók számítógépes modellezéssel.
Az ilyen területek egyike az alacsony nyomású gőzturbinák (LPMT) tervezése. Az LPMT-k fontos részei a gőz vagy kombinált gáz–gőzciklusban üzemelő erőműveknek, és az ilyen erőművek által előállított energiának akár a 20 százalékát is adhatják. A nagy- és közepes nyomású turbinákkal ellentétben – amelyeknél a gőz jól ismert jellemzőkkel rendelkezik –, az LPMT-k strukturálatlan, nem szimmetrikus, nedves gőzzel dolgoznak. Az ilyen jellegű áramlásokra jelenleg még nem léteznek teljesen megbízható matematikai modellek. A valós kísérletek nélkülözhetetlenek a turbinák áramlási útvonalainak tervezéséhez, illetve a turbinák számítógépes modelljeinek továbbfejlesztéséhez.
Az egész világon mindössze néhány próbapad létezik. Az egyik Szentpéterváron található és a Központi Kazán és Turbina Intézet tulajdona – jómagam hét éve dolgozom itt. Képzelj el egy 18 méter magas, 700 négyzetméter alapterületű, csövekkel, vezetékekkel és mérőberendezésekkel teli csarnokot. Középen egy gigászi építmény áll, ez a modellturbina háza, belőle két hatalmas cső vezet ki. A próbák idején óránként 40 tonna friss gőzt használ el, amelynek nyomása 30 bar, a hőmérséklete 400°C a próbapad bevezetésein, körülbelül 4 bar és 200°C a modellturbina bevezetésén, a kivezetéseken pedig mindössze 30 mbar nyomáson távozik.

A számítógépes és mérőberendezések felépítése

Az Alstom Powerrel indított közös „Tanja” tervezetünk keretében megújítottuk a próbapad számítógépes háttérrendszerét. A rendszer most három részből áll:

1. Egy nagy pontosságú tudományos mérőrendszerből, ami az áramlási útvonalak adatait gyűjti (a továbbiakban DAS-Flow).
2. Egy technológiai mérőrendszerből, ami a működtető személyzet számára gyűjti az adatokat (a továbbiakban DASOP).
3. A kutatók és mérnökök által használt munkaállomásokból.

A DAS-Flow rendszert eredetileg főként az ügyfeleink adták. Az áramlási útvonalon több mint 200 helyen méri a nyomást, illetve ötven helyen a hőmérsékletet. A rendszer egy különálló részével, 12 darab mozgatható érzékelővel a nyomás turbinán belüli eloszlását is nyomon követhetjük. Minden érzékelő tengelyirányban mozgatható, illetve forgatható – erről távvezérelhető léptetőmotorok gondoskodnak. A nyomásmérésekhez a PSI, Inc. PSI-9000 sorozatú érzékelőit használjuk. Ezek rendkívül pontos adatokat szolgáltatnak: az alapnyomások esetében 0,01, a többinél 0,1 százalékos pontossággal mérnek. A rendszer csak a megadott időpontokban, üzembiztos állapot mellett végez méréseket, dinamikus nyomások mérésére alkalmatlan.
A DASOP rendszert magunk építettük, a pad próbák alatti folyamatos vezérlésére alakítottuk ki. Valós idejű módban működik, az adatokat több mint 150 nyomást, hőmérsékletet, fordulatszámot és rezgést mérő érzékelőtől gyűjti be. Az adatokat két képernyőn jeleníti meg a működtető személyzet számára, illetve egy soros terminál is helyet kapott a vezérlőhelyiségben. Segítségével a személyzet figyelemmel követheti a próbapad víz-, gőz- és olajkezelő rendszereinek pillanatnyi állapotát. A DASOP biztonsági ellenőrzéseket is végez, illetve figyelmeztetési és vészjelző határértékeket is kezel.
Számítógépeink – az egy darab IBM RISC munkaállomástól eltekintve – teljesen átlagos személyi számítógépek, processzoruk a 386-ostól a Pentium 4-ig és az Athlonig terjedő teljes vonalat lefedi.

Miért éppen Linux?

Amikor – 1995-ben – bekapcsolódtam a Tanja tervezetbe, már jókora tapasztalatot szereztem adatgyűjtő és -értékelő

DASOP adatmegjelenítő modulok

rendszerek építésében a Digital PDP-11-es gépek RT-11 vagy RSX-11 operációs rendszert futtató orosz másolataival. Mike, a testvérem 1994 körül vette meg nekem az első Linux-terjesztésemet, ami egy – ha jól emlékszem – nagyjából 0.99-es Linux-rendszermagra épülő Slackware volt. Hamar rájöttem, hogy szinte minden feladatomat meg tudom oldani a hasonló programok forrásának tanulmányozásával, illetve kiindulópontként használva őket. Első adatgyűjtő rendszeremmel 1994-ben készültem el. Az `ncurses` volt az alapja, korábbi munkahelyemen azóta is működik, hozzá sem kellett nyúlnom.
A Tanja tervezet kezdetekor csak az ügyfeleink által adott DAS-Flow rendszer állt rendelkezésünkre. Operációs rendszerek sokasága volt itt, egyaránt volt benne MS-DOS, Microsoft Windows 3.11 és NT, QNX és AIX. Hogy valami jót is mondjak: a számítógépek TCP/IP alapú helyi hálózaton át kapcsolódtak egymáshoz.
A teljes rendszer fejlesztési irányvonalán gondolkodva, illetve az addigi tapasztalataimra alapozva úgy döntöttünk, hogy technológiai mérőrendszerünk, a DASOP, illetve hálózatunk központi részének az alapja a Linux lesz. A fő okok a következők voltak:

- Használatra kész alkalmazások széles választéka állt rendelkezésükre, amelyeknek tanulmányozás céljából a forráskódját is el lehetett érni, illetve alapul lehetett használni őket további programok fejlesztéséhez.
- Megbízható működés olcsó személyi számítógépeken – esetünkben ez a legfontosabb követelmények egyike.

- Költségvetésünk szűkös, így a Linux ingyenessége fontos szempont volt.

Mivel dolgozunk most? Számítógépes rendszerünk most hat linuxos PC köré épül. Több mint hat év alatt soha nem volt meghibásodás, a gépek éveken át folyamatosan üzemeltek, újraindítás csak két okból fordul elő: a gépek bővítésekor, illetve a hosszabb áramkimaradások alkalmával, amikor már a szünetmentes tápegységek is kifulladtak.
Red Hat terjesztéssel kezdtünk. Később a KSI nevű, orosz nyelvre honosított, RPM alapú terjesztést kezdtük el használni, tavaly pedig a Debian jutott szerephez. Jelenleg a központi kiszolgálónk és az én fejlesztői gépem Debian/Woody rendszert futtat. Nagyon elégedettek vagyunk a Debiannal, és úgy vélem, idén az összes linuxos gépünkre ez fog kerülni.

A hálózat

Minden számítógépünk rendelkezik helyi hálózati kapcsolattal; a hálózat három részből áll. Az első szegmensen a DAS-Flow számítógépek, a másodikon a DASOP és az irodai számítógépek találhatók, a harmadik pedig a külvilág felé vezet egy bérelt vonalon keresztül. A harmadik szegmensen csupán egy számítógép van, ezt használjuk átjáróként az Internet felé, illetve IP Chains alapú tűzfal és levelezőkiszolgáló is fut rajta. A hálózat „közepén” helyezkedik el a központi kiszolgálónk. Fájl- és nyomtatókiszolgálóként szolgál a többi számítógép számára, de a legfőbb feladata nem ez. A próbák alkalmával nagy mennyiségű adatot gyűjtünk össze. Ezek az adatok – nyers formában – mérési eredményekként, illetve később kiértékelt értékekként önműködően bekerülnek egy MySQL-adatbázisba. Egy Apache webkiszolgáló HTTPS protokollon keresztül az összes felhasználó számára – a kutatóktól kezdve egészen az ügyfelekig – sokoldalú felületet biztosít az adatbázishoz.
A bejegyzett felhasználóknak csak egy böngészőre van szükségük az adatbázis eléréséhez, az adatok közötti kereséshez és az eredmény grafikus vagy szöveges formában való megjelenítéséhez.
A rendszer PNG, CGM és PDF formátumokat használ. Az adatok megjelenítésére szolgáló oldalakat elsősorban – Apache `mod_php` modulként futó – PHP segítségével hozzuk létre. Szinte az összes grafikont valós időben állítjuk elő, `gnuplot` és Perl CGI-parancsfájlok segítségével, amelyek kiválasztják a megfelelő értékeket az adatbázisból, átadják őket a `gnuplot`-nak, majd a létrejött képet továbbítják az Apache felé.
Több mint ötven különböző CGI-parancsfájlt írtunk, így a felhasználók rengeteg különféle megjelenítési módot használhatnak, illetve választási lehetőségük gyakorlatilag mindenre kiterjed: mely értékeket szeretnénk megjeleníteni, mik legyenek a keresési feltételek, milyen jellemzőket akarnak kirajzoltatni, önműködő vagy kézi legyen-e a tengelyek méretezése, milyen legyen a simítás és a közelítés stb.
Szeretném külön kiemelni a `gnuplot`-nak a munkánkban betöltött szerepét. Számomra ez a legnagyszerűbb tudományos rajzolóalkalmazások egyike, hatalmas tudással és kimeneti formátumok széles választékával. A kiválóan tervezett `JpGraph PHP` osztályokat is használom bizonyos megjelenítési feladatokra, főleg a gyorskeresések eredményének kirajzolására.
Az általunk kifejlesztett programrendszer másik fontos része a DASOP technológiai adatgyűjtő rendszer. Moduláris felépítésű: adatgyűjtő, kiértékelő, foglalatalapú adatcsere és alkalmazói modulokat tartalmaz.
Az adatgyűjtő modul egy a számítógéphez RS232 felületen csatlakozó programozható adatvezérlővel (Programmable Data Controller – PDC) működik együtt. Másodpercenként 150 értéket vesz át a PDC-től, illetve szükség esetén digitális be- és kiviteli műveleteket végez vele. Minden mérési eredmény egy osztott memóriaterületre kerül, egy kétdimenziós tömbbe, ahol az egyes oszlopok nyers formában az összes jellemző értékét tartalmazzák. Az oszlopok száma rögzített, így visszamenőleg mindig adott számú adatsor áll rendelkezésre a memóriában.
A kiértékelő modul, amely jelzők segítségével marad összhangban az adatgyűjtő modulokkal, beolvassa az utolsó mérési eredménysort az osztott memóriából, elvégez néhány azonnali számítást, majd a kiértékelt adatokat ugyanabba az oszlopba írja vissza, egyben meg is hosszabbítva azt.
A foglalatalapú adatcseremodul a távoli alkalmazói modulok számára biztosítja a hozzáférést a megosztott memóriaterülethez. Alkalmazói modulból sokféle létezik. Egyesek helyben, a mért és kiértékelt adatokat tartalmazó osztott memóriarész közvetlen elérésével futtathatók, mások távoli memória-hozzáférését az adatcseremodul teszi lehetővé. Az alkalmazói modulok között adatnaplózó, biztonsági vezérlő és adatmegjelenítő modulokat találunk.
Az adatmegjelenítő modulok valós időben, különféle szempontok szerint jelenítik meg az adatokat. Néhány ábrázolási példa: az értékek változása az idő függvényében, oszlopdiagram (amelynél az oszlopok színe adja meg a hozzájuk tartozó jellemző állapotát – normál, figyelmeztetés, vészhelyzet), valódi műszerek kijelzőjét utánzó rajzok.
Ütemezési elvárásaink viszonylag enyhék, nem váruk szigorúan valós idejű működést. Számunkra a programok által megvalósított valós idejű működés is megfelelő, tehát gépeinken normál Linux-rendszermag fut. Az adatgyűjtő, kiértékelő és adatcseremodulok kizárólag C nyelven íródtak, és egyazon számítógépen futnak. A biztonsági, a naplózó és a megjelenítő modulok némelyike szintén ezen a számítógépen fut.
A megjelenítő modulok egy része egy másik számítógépre került, amely az előbbi X termináljaként üzemel. Mindkét számítógép és a hozzájuk tartozó két monitor a próbapad vezérlőhelyiségében található, így a működtető személyzet minden adathoz hozzá tud jutni.
Egyes megjelenítő modulok a kutatók gépein futnak, ezek az adatcseremodulon keresztül kapják az adatokat.
A megjelenítő modulok viszonylag hosszabb idő alatt tisztultak le. Eleinte `ncurses` alapú, a szöveges linuxos konzolon dolgozó programjaink voltak. Később az X is képbe került, eleinte szabványos X11 és Xt könyvtárakkal. A következő lépés a SuSE-ből vett Motif kipróbálása volt. A nyílt forrású GTK egy vagy két évvel később jelent meg, és én azonnal áttértem rá. Az utóbbi két évben szinte az összes megjelenítő és egyéb modul Tcl/Tk alapú, illetve erőteljesen támaszkodnak a BLT kiterjesztésre.

Összegzés

Éles ipari környezetben történő sok év programfejlesztés és -használat alapján arra a következtetésre jutottunk, hogy minden szempontból a nyílt forrású megoldások a leghatékonyabbak. A következő lépésünk az lesz, hogy a megmaradt kereskedelmi programokat is nyílt forrásúakra cseréljük le.

Linux Journal 2003. február, 103. szám

Alexandr E. Bravo
Szentpéterváron él. 1982-ben a Politechnikai Egyetemen automatizálás és telemechanika szakon szerzett diplomát.

A vörösszemhatás eltávolítása GIMP-pel

Belefáradtunk azokba a fényképekbe, amelyekről a családtagjaink és barátaink „sátáni” szemekkel néznek vissza ránk?

Ahogy napjainkban egyre inkább elárasztják a piacot az olcsó digitális fényképezőgépek és lapolvasók, úgy választja egyre több Linux-felhasználó a népszerű nyílt forráskódú GNU Image Manipulation Programot (GNU képszerkesztő program, rövidebb nevén: The Gimp), amikor digitális képeiket kívánják szerkeszteni. Írásunkban egy egyszerű eljárást ismertetünk arra vonatkozóan, hogy pillanatfelvételeinkről miként távolíthatjuk el a Gimp segítségével a rettegett „vörös szemeket”.

Az általam ismertetésre kerülő eljárással megmenthető a pupilla létfontosságú tonalitása, vagyis a különböző részein megjelenő sötét és világos árnyalatok változatossága. Ugyanígy változatlanul hagyja a pupillát fedő szaruhártyáról visszatükröződő fényeket, amelyek a fényképezett személy életteliségének érzetét keltik.

A Gimp legtöbb menüpontja a kép ablakán történő jobb egérkattintással hívható elő. Leírásomban a jobb egérkattintást JK-val fogom rövidíteni. Ha egy alkalmazandó Gimp-tevékenységet szeretnék leírni, zárójelek közé tett menüpontsorozatot vagy billentyűkombinációkat olvashatunk majd. Például egy kép megnyitásakor a (JK>***File***>***Open***) formát használom. Ha célszerűbbnek tűnik a billentyűkombinációs megoldás, listát közlök azokról a gombokról, amiket meg kell nyomnunk – például ilyen a kép másolására szolgáló CTRL-C.

Következzen az eljárás. Indítsuk el a Gimpet, majd nyissuk meg a vörösszem-hatásban szenvedő képet (JK>FILE>OPEN), ahogy az *1. képen* látható. Ha betöltöttük a képet, nagyítsuk ki a szem részletét (néhányszor megnyomva a + billentyűt és szükség esetén görgetve a képet), így a vörös pupillák egy szép nagy nézetéhez juthatunk.

Következő lépésként a fotóalany pupilláját alkotó képpontokat szeretnénk kijelölni. Számos megoldás létezik erre, tapasztalatom alapján a legjobban ezek közül a fuzzy select (képlékeny kiválasztás, másik elterjedt nevén „magic wand” – varázspálca) eszköz használható. Az eszköz a hasonló színárnyalatú pontok tartományának elvén működik. Ezen olyan területet értünk, amelynek pontjai a pillanatnyi ponthoz viszonyítva csak meghatározott színeltérést mutathatnak (itt jelentkezik a „fuzzy” – képlékeny körvonalú – tulajdonság: beállíthatjuk ennek az eltérésnek a küszöbértékét). Minden RGB rendszerben tárolt kép (ez a legtöbb képformátum normál tárolási rendszere) három színcsatornából áll: vörös, zöld és kék (red, green, blue), amelyek minden képpont esetén meghatározott értéket veszznek fel. Ha ezeket a csatornákat egyenként megvizsgáljuk, rendszerint azt vesszük észre, hogy a pupillák fuzzy selecttel való kiválasztásához a zöld csatorna rendelkezik a legnagyobb kontraszttal.

Hívjuk elő a rétegek (***Layers***) párbeszédablakát (CTRL-L), és kattintsunk a ***Channels*** (csatornák) fülre. Töröljük a kék (***Blue***) és vörös (***Red***) csatornák kiválasztását. A Layers párbeszédablaknak a *2. képén* látható képet kell mutatnia, csak a ***Green*** (zöld) csatorna legyen kiemelve. A két csatorna láthatóságát nem kapcsoltuk ki, így a kép ablakában nem észlelhető semmi változás, de azzal, hogy csak a zöld csatornát választottuk ki, a fuzzy select eljárás a szomszédos képpontok kiválasztásakor csak a zöld csatorna értékét veszi figyelembe.

1. kép Aranyos kisgyerek vagy ördögfióka?

Most az eszköz beállítási lehetőségeinek megjelenítése céljából kattintsunk kétszer a fuzzy select (magic wand) eszközre a Gimp eszköztárában. A megfelelő küszöbérték (Threshold) beállításához egy kis gyakorlatra van szükség, de általában az alapértelmezett beállítás növelésére van szükség. Érdemes azzal az értékkel próbálkozni először, ami a *3. képen* látható. A határvonal lágyságának (Feather) beállítását szintén érdemes megváltoztatni, valamilyen kis értéket adva neki, ahogy az ábrán látható.

Most kattintsunk a képen a pupilla vörös tartományára. Látnunk kell a „menetelő hangyákat” a terület körül, ahogy a pupilla nagy részének kijelölése megtörténik. Ha nem megfelelő, töröljük a kijelölést (SHIFT-CTRL-A), finoman növeljük a küszöbértéket (Threshold), és próbálkozzunk újra. Ugyanígy ha a pupillán kívüli részek is kijelölésre kerültek, a küszöbérték csökkentésével próbálkozhatunk. További lehetőség a ***Grow selection*** (növekvő kiválasztás, JK>Select>Grow) és a ***Shrink selection*** (csökkenő kiválasztás, JK>Select>Shrink) párbeszédablak – ha nagyjából jó az eredmény, csak néhány pont nem találta meg a helyét a kijelölés során. Ha megtörtént a pupilla megfelelő pontjainak kiválasztása, tartsuk lenyomva a SHIFT billentyűt és kattintsunk a másik pupilla vörös részére (a SHIFT nyomva tartása mellett történő kiválasztás a tartományt hozzáadja a már kijelölt részhez). Most már mind a két pupillának az *4. képen* látható módon kijelölt állapotban kell lennie.

Egy tipp a Gimp gyakorlottabb felhasználói számára: ha ismerjük a ***quick mask*** (gyorsmaszk) eszközt, ezzel is javíthatjuk a hibás kiválasztást. Kattintsunk a ***quick mask*** feliratú gombra,

2. kép A színcsatornák a régteg megjelenítőben

3. kép Csökkentjük a határértéket

egy kisméretű lágy ecsettel végezzük el a megfelelő színezést, ezután térjünk vissza a kijelölésre.
Most térjünk vissza a rétegek (***Layers***) párbeszédablakra, jelöljük be a vörös (Red) csatornát és vegyük le a jelölést a zöldről (***Green***). Miután ellenőriztük, hogy csak a vörös van kiválasztva, színtelenítjük (desaturate) a kijelölt területet (JK>Image>Colors>Desaturate).
Itt az idő, hogy szemügyre vegyük az eredményt. Kapcsoljuk ki a kijelölés láthatóságát (CTRL-T), ezzel a pupilla körül eltűnnek a „masírozó hangyák" – jobban látszik, mit műveltünk. Azt nem árt tudatosítani, hogy a kijelölés még él, csak nincs látható jele.
Ha elégedettek vagyunk az eredménnyel, kapcsoljuk vissza a kijelölés láthatóságát (CTRL-T), töröljük az összes kijelölést (CTRL-SHIFT-A), csökkentsük a nagyítást (-), hogy jobban szemügyre vehessük munkánk eredményét, és további változtatásokat hajtsunk végre. Ha nem tetszik az eredmény, kapcsoljuk vissza a kijelölés láthatóságát (CTRL-T) és a CTRL-Z billentyűkombinációval lépkedjünk vissza addig a pontig, ahonnan módosíthatjuk a kijelölést, vagy válasszunk más megközelítést a hiba kijavítására.
Meg kell említenem ennek az eljárásnak egy olyan változatát, ami szerintem egy árnyalattal jobb eredményhez vezet. Ennek feltétele az, hogy Gimp példányunkra telepítve legyen a ***Channel Mixer*** (csatornakeverő) bővítmény. A ***Channel Mixer*** nagyszerű eszköz arra, hogy a kép részleteit vagy az egész képet fekete-fehérre változtassuk; jóval több beavatkozást enged a folyamatba, mintha csak egyszerűen a ***Desaturate*** szolgáltatást vagy az RGB>Grayscale átalakítást használjuk. A Channel Mixer nem része a Red Hat rendszerrel kapott GIMP 1.2.3 csomagomnak, de a Gimp-bővítmények nyilvántartásában (The Gimp Plugin Registry, registry.gimp.org) megtalálható. Egyszerűen fordítsuk le és másoljuk a saját könyvtárunkban lévő ***.gimp-1.2/plug-ins*** könyvtárba.
Az eljárásnak ebben a változatában minden a fent említett módon zajlik, de ahelyett, hogy a vörös csatornát színtelenítjük, minden csatornának töröljük a kiválasztását, és indítsuk el a ***Channel Mixert*** (JK>Filters>Colors>Channel Mixer). Az eszköz lehetővé teszi, hogy az RGB értékeket tetszőleges százalékban keverjük. Jelöljük ki a Monochrome négyzetet, csökkentsük jelentős mértékben a vörös színt, és növeljük a zöldet. Én a *6. képen* látható arányokat használtam: 10%-nyi vörös, 60%-nyi zöld és 30%-nyi kék színt. Kell egy kis gyakorlat ahhoz, hogy kiválaszthassuk, melyik arány adja a pupillák leginkább valósnak tűnő képét, de kiindulásként ez biztosan megfelel.
Amikor az arányokat megfelelőnek találjuk, kattintsunk az ***OK***-ra. Ha nem vagyunk biztosak benne, hogy az eredmény megfelelő, és más arányokkal szeretnénk próbálkozni, lépjünk vissza (CTRL-Z), kapcsoljuk vissza a kijelölés láthatóságát (CTRL-T), és futtassuk újra ugyanazt a szűrőt (SHIFT-ALT-F). A ***Channel Mixer***

4. kép Mindkét pupilla kijelölve

5. kép Az elkészült kép

használatával elért eredmény a *7. képen* látható: a pupillák szép sötétek, enyhe tónusbeli változás figyelhető meg a határvonal mentén, kellemes megjelenésű fényvisszaverődés látható a szemben. Ha az eredménnyel elégedettek vagyunk, akkor ahogy korábban is tettük, kapcsoljuk vissza a kijelölés láthatóságát (CTRL-T), töröljük az összes kijelölést (CTRL-SHIFT-A), és csökkentsük a nagyítást (-), hogy eredeti méretben vizsgálhassuk meg munkánk eredményét. A *5. kép* a ***Channel Mixer*** használatával elért végeredményt mutatja.
Talán most egy kicsit bonyolultnak tűnik a módszer, de ha ráérzünk az ízére, néhány percnél nem vesz több időt igénybe. A legjobb az egészben, hogy az eredmény kitűnő minőségű, amit különösen nagyfelbontású tintasugaras nyomtatóval készített papírkép esetén tapasztalhatunk.

Linux Journal 2003. február, 106. szám

Eric Jeschke (eric@redskiesatnight.com)
Az Indiana Egyetemen Számítástechnikából szerzett PhD-fokozat birtokosa; dolgozott programmérnökként, egyetemi tanárként és szabadúszó tanácsadóként. Hawaiin él a feleségével, gyermekeivel és túlsúlyos macskájával. Szabadidejét legszívesebben a családjával, szabadtéri kalandokkal, fotózással és a Linuxszal tölti.

Útmutató GPRS-Bluetooth használatához

Internetezés Bluetooth-kapcsolaton keresztül Linux alatt.

A Linuxvilág 21. számában már szóltunk a Linux alatti GPRS csomagkapcsolt adatátvitelről („GPRS Linux alól is” címmel, 70. oldal). Részletesen tárgyaltuk, miképpen használjuk a soros kapura kötött telefont internetezésre. De arról nem esett szó, mi történjen akkor, ha új típusú alaplapot vettünk vagy laptopot használunk, és nincs rajta soros kapu. Előfordulhat az is, hogy kényelmi okokból nem ragaszkodunk a kábelrengeteghez, esetleg belefáradtunk abba, hogy nagyjából ötóránként lemerül a telefonunk. Támogatja készülékünk a Bluetooth-ot? Szerezzünk be egy USB-s Bluetooth-adaptert és máris használhatjuk kedvenc operációs rendszerünk alatt!
Írásomban részletesen ismertetem, hogyan sikerült Debian Woody 3.0 és SuSE 8.1-es alatt egy Widcomm v.1.1 ESB-s Bluetooth-adapterrel, Nokia 6310-es telefonnal, valamint Vodafone kártyával kiválóan működő GPRS-kapcsolatot megvalósítani. Célkitűzésünk eléréséhez elsőként egy 2.4.x-es rendszermag beszerzése szükséges. A telepítéshez kövessük az alábbi lépéseket!

1. lépés – a rendszermagfordítás

A következőket adjuk hozzá a */etc/modules.conf*-hoz:

```
#Bluetooth kapcsolat
options ppp_async flag_time=0
# Bluez
alias net-pf-31 bluez
alias bt-proto-0 l2cap
alias bt-proto-2 sco
alias bt-proto-3 rfcomm
alias bt-proto-4 bnep

#Bluez - UARTs
alias tty-ldisc-15 hci_uart
```

2. lépés – a DNS beállítása

A DNS-kiszolgálók beállítása következik – amennyiben ez már megtörtént, ugorjunk tovább. A beállításhoz a `usepeerdns`-t használjuk (sajnos némelyik telefon nem támogatja). Amennyiben nem működne, akkor kézzel kell beállítani a */etc/resolv.conf*-ban. Valami ilyesmit kell a szövegszerkesztőben látnunk:

```
nameserver xxx.xxx.xxx.xxx
nameserver xxx.xxx.xxx.xxx
```

Természetesen az x-ek helyére egy valódi DNS-kiszolgáló IP-címét kell begépelni. Ezután ellenőrizni kell, hogy a */etc/host.conf* tartalmazza-e az `order hosts,bind` sort. Miután mindezt végrehajtottuk, egy gyors rendszermagfordításra lesz szükségünk. A Bluetooth-támogatást tegyük modulba.

```
# grep -i blue /usr/src/linux-2.4.19/.config
CONFIG_USB_BLUETOOTH=m # Ezt a modult le kell
majd t r lni!
```

Ezt a modult azért kell törölni, hogy ne a `bluetooth.o` modult töltse be a rendszer, hanem a `hci_usb.o` -t – bár lehet, hogy van szebb megoldás is, nekem ez tűnt a legegyszerűbbnek!

```
# Bluetooth support
CONFIG_BLUEZ=m
CONFIG_BLUEZ_L2CAP=m
CONFIG_BLUEZ_SCO=m
# Bluetooth device drivers
CONFIG_BLUEZ_HCIUSB=m
# CONFIG_BLUEZ_USB_FW_LOAD is not set
# CONFIG_BLUEZ_USB_ZERO_PACKET is not set
# CONFIG_BLUEZ_HCIUART is not set
# CONFIG_BLUEZ_HCIDTL1 is not set
# CONFIG_BLUEZ_HCIVHCI is not set
```

Megjegyezném, hogy a SuSE 8.1-es rendszermagját szükségtelen volt újrafordítani, elegendőnek bizonyult a már említett `/lib/modules/2.4.19/kernel/drivers/usb/bluetooth.o` modul törlése.
Nagyon fontos még, hogy a */usr/src/linux* hivatkozásnak a saját rendszermagforrásunkra kell mutatnia. Ekkor állunk készen arra, hogy telepítsük a Bluetooth-kapcsolathoz szükséges fájlokat. Ezek a ➲ http://bluez.sourceforge.net/download/download.html oldalon érhetők el, de a Linuxvilág 45. CD-mellékletének Magazin/Bluetooth könyvtárában is megtaláljuk őket:

bluez-kernel-2.3.tar.gz (A `create_dev` parancsfájl csak a Bluez CVS-ben található meg.)

```
bluez-libs-2.2.tar.gz
bluez-utils-2.1.tar.gz
bluez-sdp-0.8.tar.gz
bluez-pan-1.1-pre1.tar.gz
bluez-hcidump-1.3.tar.gz
bluez-hciemu-1.0.tar.gz
bluez-bluefw-0.7.tar.gz
```

A fájlokat ebben a sorrendben csomagoljuk ki, és egyszerűen a `./configure` majd `make` és végül `make install` parancsokkal fordítsuk le, azután telepítsük őket. Futtassuk le a `create_dev` parancsfájlt, és a */lib/modules/2.4.19/*-es könyvtárban adjuk ki a `depmod -a` parancsot, és máris elkészültünk a telepítéssel.

3. lépés – a Bluetooth-kapcsolat beállítása

A telepítést követően állítsuk be a Bluetooth-kapcsolatot. Mivel már telepítettük a programokat, rendelkezésünkre áll `hciconfig`, `hcitool`, `rfcomm` és `/etc/init.d/bluetooth` programunk, tehát használjuk is őket! (Ha esetleg SuSE alatt a `/etc/init.d/bluetooth` parancsot nem leljük, a `# cp bluez-utils-2.1/scripts/bluetooth.rc.rh /etc/init.d/bluetooth` paranccsal másoljuk a helyére.) Adjuk ki a `/etc/init.d/bluetooth`

`start` parancsot, mire a */var/log/messages*-ben valami hasonlót kell látnunk:

```
Jan 13 07:45:08 debian kernel: BlueZ Core ver
2.2 Copyright (C) 2000,2001 Qualcomm Inc
Jan 13 07:45:08 debian kernel: Written
2000,2001 by Maxim Krasnyansky
<maxk@qualcomm.com>
```

Adjuk ki az `lsmod` parancsot, és azt látjuk, hogy az alábbi modulok töltődtek be:

```
l2cap                   17408   1
(autoclean)
bluez                   35048   1
(autoclean) [l2cap]
```

Ha mindez megvan, adjuk ki a `modprobe hci_usb;modprobe usb-uhci` parancsot, és már állíthatjuk is be az adaptert a `hciconfig` parancs segítségével. Ha ekkor az adapter a 00:00:00:00:00:00 címet írja ki, sajnos nem működik, ezért adjuk ki a `hciconfig hci0 down` és `hciconfig hci0 up` parancsokat. Ezután a rendszernek már látnia kell az adapterünket. Ezt követően kapcsoljuk be a telefonon a Bluetooth-hozzáférést, és próbáljuk meg pingelni:

```
# hcitool inq
Inquiring ...
        00:02:EE:42:34:2B        clock
        ➥offset: 0x3465     class: 0x502204

# l2ping 00:02:EE:42:34:2B
Ping: 00:02:EE:42:34:2B from 00:10:60:29:17:28
➥(data size 20) ...
0 bytes from 00:02:EE:42:34:2B id 200
➥time 34.10ms
0 bytes from 00:02:EE:42:34:2B id 201
➥time 26.16ms
2 sent, 2 received, 0% loss
```

4. lépés – a GPRS beállítása

Elérkezett az ideje, hogy GPRS-hozzáférésünket is beállítsuk. Amennyiben a Bluez-csomagokat rendben lefordítottuk és telepítettük, a */etc* könyvtárban egy ***bluetooth***/ nevű könyvtárat kell látnunk.

```
# ls -l /etc/bluetooth/
total 20
drwxr-xr-x     2 root      root
➥4096 Oct 30 00:27 firmware
-rw-r--r--     1 root      root
➥1371 Oct 29 19:24 hcid.conf
-rw-------     1 root      root
➥7 Oct 30 00:07 pin
-rw-r--r--     1 root      root
➥329 Nov  5 17:44 rfcomm.conf
```

Az ***rfcomm.conf***-ban a telefon címét át kell írni, mely így fest:

```
        # Bluetooth address of the device
        device 11:22:33:44:55:66;
```

ez most nálam így néz ki:

```
        # Bluetooth address of the device
        device 00:02:EE:42:34:2B;
```

5. lépés – a telefon beállítása

A telefonbeállításnál engedélyeztem a láthatóságot, valamint biztonsági okokból a pinkód (személyi azonosító kód) használatát is beállítottam. A */etc/bluetooth/pin* fájl tartalmazza a pinkódot, ami alapbeállításként 1234. Itt most egy pillanatra álljunk is meg, mert a ***hcid.conf***-ban található egy olyan lehetőség, ami pinkódválasztáshoz nélkülözhetetlen, és ennek a futtatásához a ***python-gtk*** csomag szükséges.

```
# PIN helper
pin_helper /bin/bluepin;
```

Úgy gondoltam, hogy ez nem okozhat gondot, mert mindkét fent említett terjesztés tartalmazza. Ám tévedtem, mert nekem ez a szolgáltatás nem működött. A telefon „Eszköz nem elérhető" vagy „Hibás kód" hibaüzenetet adott. Szerencsére ez a nehézség is legyőzhető, ehhez a kódot a következőképpen írjuk át:

```
# PIN helper
pin_helper /bin/btpin;
```

Továbbá a */bin* könyvtárban hozzunk létre egy `btpin` nevű parancsfájlt a következő tartalommal:

```
#!/bin/bash
echo "PIN:1234"
```

Ezután már csak futtathatóvá kellett tenni, és amikor a telefon kéri, megadni a kódot. (Természetesen bármilyen pinkódot választhatunk!)

6. lépés – a telefon csatlakoztatása

Elérkezett az ideje, hogy most már csatlakoztassuk a telefonunkat!

```
# rfcomm bind /dev/rfcomm0 00:02:EE:42:34:2B
```

A folytatáshoz még szükségünk lesz a `gprs`, `gprs-connect-chat`, `gprs-disconnect-chat` parancsfájlokra. Ezeket másoljuk be a */etc/ppp/peers/* könyvtárba. A ***gprs*** fájlban a következő módosításokat kell végrehajtani: a `/dev/ttySx` bejegyzést `/dev/rfcomm0`-ra kell cserélni, a sebességet 115200-ra, a forgalomvezérlést `crtscts`-re kell állítani. Most már semmi sem állíthat meg minket, irány az Internet! Nincs más dolgunk, mint megfelelő jogosultságok birtokában kiadni a `# pppd call gprs` parancsot egy konzolban, hogy a kimenetét is követni tudjuk (45. CD Magazin/Bluetooth/kimenet.txt). Természetesen PC-nket a Bluetooth segítségével nemcsak a telefonnal tudjuk összekapcsolni, hanem egy másik PC-vel is, ha például helyi hálózatot szeretnénk létrehozni. Ennek bemutatása azonban már túlmutat e cikk keretein.

Steinhausz Gábor (steing@freemail.hu)
27 éves, nős, végzős informatikushallgató, hobbija a hegymászás és természetesen a Linux mint az asztali PC-k operációs rendszere.

A proftpd beállítása

Egy sokoldalú FTP-kiszolgáló Apache-szerű beállítóállománnyal.

Az FTP (File Transfer Protocol) egy igen régóta méltán népszerű protokoll, amelyet elsősorban hatékony állományátvitelre terveztek. Mind belső hálózaton, mind az Interneten rengeteg felhasználási területe létezik. Te is könnyedén felállíthatsz egy FTP-kiszolgálót, viszont biztonságos beállításához több kell egy deb-, vagy egy RPM-csomag telepítésénél. Az FTP igen összetett protokoll, ugyanakkor a vonatkozó rfc elolvasásával közelebb kerülhetsz a megismeréséhez. Ajánlom az rfc959-t, amely Debian alatt a `proftpd-doc` csomag része, és `/usr/share/doc/proftpd-doc/rfc/rfc0959.txt.gz` néven érhető el. Mint láthatod, a `proftpd` telepítését Debian Woody alatt mutatom be, 2.4.19-es rendszermaggal és a Netfilter használatával. A parancsok más terjesztésekben eltérhetnek ettől, de a telepítés menete nagy vonalakban ugyanaz.

Active és passive mód

A lustábbak kedvéért a fontosabb fogalmakat itt ismertetem, de ez nem jelenti azt, hogy felesleges lenne elolvasni a fellel-hető leírásokat. Ne felejtsd el, hogy a különböző fórumokon (levelezőlistákon, IRC-n stb.) nem szívesen foglalkoznak olyan kérdésekkel, amelyekre valahol már léteznek válaszok.
Egy jellemző FTP-kapcsolat a következőképpen fest: az ügyfél csatlakozik a kiszolgáló egy megadott kapujára (alapesetben ez a `21/tcp - ftp`). Ha a hitelesítés sikeres volt, a kapcsolat további része az eggyel alacsonyabb számú kapun folytatódik (`20/tcp - ftp-data`). Ezt követően kétféleképpen történhet a fájlátvitel. Active módban a kiszolgáló csatlakozik az ügyfél egyik kapujára, míg passive módban az ügyfél kapcsolódik a kiszolgálóhoz. A passive mód az alapértelmezett – ez a tűzfalazásnál nyerhet nagy jelentőséget. A témától elszakadva: egy szigorú tűzfal esetén jó ötlet lehet a belső hálóról active módban elérni egy FTP-kiszolgálót az Interneten, ha passive módban nem sikerül állományokat letölteni. Láthatod, hogy az active, illetve passive mód a kiszolgáló oldaláról nézve működő vagy tétlen.

A proftpd telepítése

A Debian Woodyban található `proftpd` csomag egy megbízható változatot tartalmaz (1.2.5), így szerény véleményem szerint a kiszolgálót felesleges forrásból telepíteni. Ha valakinek mégis ez a rigolyája, az 1.2.7 változatszámút az ➲ ftp://ftp.proftpd.org/distrib/source/proftpd-1.2.7.tar.bz2 címről töltheti le, illetve a 45. CD Magazin/Proftpd könyvtárában található. A deb-csomagot `dselect`-tel vagy egy hasonló `apt`-előtéttel célszerű telepíteni. A forrás lefordítása nem különösebben nehéz feladat:

```
# mv proftpd-1.2.7.tar.bz2 /usr/src
# cd /usr/src
# bzcat proftpd-1.2.7.tar.bz2 | tar xv
# cd proftpd-1.2.7
# ./configure --prefix=/usr
# make
# make install
```

A deb-csomag telepítése során a `debconf` felteszi azt a kérdést, hogyaz FTP-kiszolgálót ***standalone*** (önálló) vagy `inetd` módban szeretnéd-e futtatni. Önálló módban az `ftp` démon folyamatosan fut, és foglalja a számára kijelölt kaput, várva az ügyfelekre. Az `inetd` esetén az Internet Superserver fogja a kaput, és csak szükség esetén, azaz egy új ügyfél érkezésekor indítja el a démont. Az `inetd`-s megoldás kevesebb erőforrást emészt fel, ugyanakkor minden új ügyfelet megvárakoztatsz, hiszen türelmesnek kell lenniük, mire a démonfolyamat elindul. Általánosságban elmondható, hogy húsz felhasználó után már mindenképp érdemes önálló `ftp` démont futtatni. Ez a választás természetesen nem végleges, utólag könnyedén megváltoztathatod a döntésedet.
Debconf esetén további kérdést jelent, hogy szeretnéd-e engedélyezni az úgynevezett anonim elérést. Ez azt jelenti, hogy regisztrált felhasználónév, illetve jelszó nélkül is el lehessen-e elérni a kiszolgálót, vagy sem. Anonim eléréskor az ügyfél felhasználónévként anonimoust (névtelen) ad meg, jelszóként pedig többnyire egy elektronikus levélcímet. Érdemes nemmel válaszolni, s ezáltal kitiltani a vendégfelhasználókat. A döntés nem végleges, hiszen utólag könnyű őket engedélyezni, de legalább a kiszolgáló beállítása alatt nem érhető el jelszó nélkül a rendszer.

A /etc/proftpd.conf állomány

Mint említettem, a `proftpd` beállítóállománya az Apache ***httpd.conf*** mintáit követi. Ennek megfelelően álljon itt egy lista a fontosabb irányelvekről:

- `ServerName {string}`
  A kiszolgáló nevét határozza meg. A kapcsolódást követően alapértelmezésben ezt a szövegfüzért látják a parancssori `ftp`-t használók:
  `"ProFTPD 1.2.5rc1 Server (Debian) [inter.net]"`
  Itt a Debian a `ServerName` által meghatározott név, míg az `inter.net` a számítógép úgynevezett FQDN-je (Fully Qualified Domain Name), vagyis a tartománynév.
- `ServerType {"inetd" | "standalone"}`
  Itt a kiszolgálófolyamat futtatásának módját határozod meg. Az `inetd`, illetve `standalone` közötti különbség leírását lásd fentebb.
- `User {string}`
  Annak a felhasználónak a neve, aki a démon tulajdonosa. Debian alatt alapértelmezésben ez ***nobody***, viszont ezt a felhasználót más folyamatok is használják, így érdemes egy külön ***ftpd*** nevű felhasználót létrehozni `/bin/false` héjjal, és azt adni meg itt.
- `Group {string}`
  Annak a csoportnak a neve, aki a démon tulajdonosa. Szintén ajánlott a ***nogroup*** helyett egy ***ftpd*** nevű felhasználó létrehozása.

- `DefaultRoot {string} [string]`
  Ez a parancs az angol szakszókincs szerint egy ***jail root***-ot hoz létre, ami valójában egy ketrec a felhasználóidnak. Az első értékként megadott könyvtárból nem tudnak feljebb lépni, vagyis számukra ez az új gyökér. A könyvtárnév az egyes felhasználókra nézve lehet relatív. A második érték nem kötelező, ezzel egy vagy több csoportra engedélyezheted avagy tilthatod ezt a megszorítást. Például:
  `"DefaultRoot ~ download,!upload"`
  Ennek a segítségével eléred, hogy minden felhasználó, aki tagja a ***download*** csoportnak, de nem tagja az ***upload*** csoportnak, be legyen szorítva a saját könyvtárába (~).
- `AllowForeignAddress {"on" | "off"}`
  Engedélyezi az ügyfélnek, hogy a `PORT ftp` parancs használatakor a sajátján kívül más címét is használhassa. Ha nem érted, hogy ez mit jelent, akkor nem olvastad el az rfc-t, erre viszont most nem térnék ki. A lényeg, hogy ha ezt bekapcsolod (alapértelmezésben tiltva van), akkor a felhasználók igénybe vehetik az FXP nyújtotta lehetőségeket. Az FXP egy olyan módszer (nem külön protokoll), amellyel két kiszolgáló között anélkül mozgathatsz állományokat, hogy a saját gépedre letöltenéd őket.
- `AuthUserFile {string}`
  Lehetőség van a `proftpd`-ben más forrásokból is meríteni a felhasználói adatbázist, nem kötelező a */etc/passwd*-t használni. Ez azért remek, mert az FTP-felhasználóknak nem kell feltétlenül létezniük a rendszerben. E források között szerepel az LDAP-, az SQL-kiszolgálók, illetve egy másik ***passwd*** állomány. Ezt az utóbbi forrást használja ez az irányelv, ami már elég nagyfokú biztonságot tesz lehetővé. Gondolj csak arra, hogy így könnyen megakadályozhatod az FTP-felhasználók távoli belépését `telnet`-en vagy `ssh`-n keresztül. Mivel azonban itt nem létezik az árnyékjelszó fogalma, óvatosan állítsd be a jogosultságokat arra az állományra, amit itt értékként megadsz. A legjobb, ha a tulajdonosa és a csoportja az, amit a ***User***, illetve a ***Group*** meghatározásnál megadtál, és csak a tulajdonos és a csoport tudja írni és olvasni, a többieknek pedig semmilyen jogosultságuk nincs. Az állományt az `ftpasswd` parancs segítségével lehet létrehozni, illetve karbantartani, közvetlen írása nem ajánlott.
- `AuthGroupFile {string}`
  A hasonló nevű `AuthUserFile`-hoz hasonló, csak ez a */etc/group* állományt helyettesíti. Szintén az `ftpasswd` paranccsal illik írni.

Az ftpasswd

Ez egy olyan könnyen használható segédeszköz, mely az `AuthUserFile` vagy az `AuthGroupFile` által megadott állományt módosítja. Kapcsolóit két egyszerű példán keresztül mutatom be. Létrehozok egy andras nevű felhasználót, és a ***download*** nevű csoportba teszem bele.

```
# ftpasswd --passwd --name andras --uid 1000
➥--gid 100 --home /home/andras --shell /bin/sh
```

A `--passwd` határozza meg, hogy a felhasználók adatbázisát szeretném módosítani, nem pedig a csoportokét. Lehetőség nyílik egy `--file` kapcsoló használatára, így meg lehet határozni az adatbázis nevét. Ha elhagyod, alapértelmezés szerint *./ftpd.passwd*, azaz a pillanatnyi könyvtárban egy ***ftpd.passwd*** nevű állomány. A többi kapcsoló magától értetődő. A `--gid`-et elhagyhatod, ekkor egy a `uid`-del megegyező csoportazonosítót vesz alapul. Senkit ne tévesszen meg, hogy létezik egy `--shell` kapcsoló, és egy valós héj van megadva utána! Ez nem jelenti azt, hogy az adott felhasználó be tudna lépni távolról `telnet`-en vagy `ssh`-n, hiszen nem is létezik a rendszerben. Mindössze a PAM (Pluggable Authentication Modules) miatt van itt szükség egy valós héj megadására. Többek közt így lehet egy FTP-felhasználót „hibernálni”: `/bin/false`-t adsz meg héjnak, legyen szó akár `AuthUserFile`-ról, akár nem. A felhasználó jelszavát ezután kétszer egymás után kell begépelned, akárcsak a `passwd` parancsnál. Ez a jelszó az adatbázisban titkosítva tárolódik.

```
# ftpasswd --group --name download --gid 100
```

A `--group` azt mondja meg, hogy a csoportok adatbázisát módosítom. A `--file` elhagyása miatt az állomány neve *./ftpd.group*. Paranoiások figyelmébe ajánlom a `--enable-group-passwd` kapcsolót, ami szerintem ebben az esetben teljesen felesleges.

A Netfilter beállítása

FTP-kiszolgálót tűzfallal ellátni eléggé összetett feladat. A Netfilter `ip_conntrack_ftp` modulja ugyanakkor jelentősen leegyszerűsíti ezt a feladatot. Ez nyomon követi az `ftp` csomagokat, így azt sem szükséges tudnod, mi az az `ftp-data` kapu. Ha van ilyen modulod, egy szempillantás alatt beállítható (lásd a 45. CD Magazin/proftpd könyvtárában).
Ha a tűzfaladnak nincs `ftp`-nyomkövető modulja, egy kicsit nehezebb lesz a dolgod. Először is engedélyezned kell az `ftp (21)`, az `ftp-data (20)` kapukat, illetve a passzív átvitelhez mindazokat a kapukat, amik szóba jöhetnek. Ez érthető, hiszen az ügyfélnek el kell tudna érni azt a kaput, ahol az átvitel folyik. A gond csak az, hogy a kapu az 1024-65 535 tartomány bármelyik eleme lehet. Ha ezt mind engedélyezni akarod, ne is telepíts tűzfalat a kiszolgálóra. Azzal lehet segíteni az ügyön, hogy megmondod a `proftpd`-nek, hogy egy szűk tartományból válasszon magának passzív kaput, és csak azt engedélyezed a tűzfalban. Az utóbbi varázslat a `PassivePorts` meghatározással hozható létre.

PassivePorts {int} {int}

Az első egész szám az intervallum alsó, míg a második a felső határát jelöli (közöttük csak szóköz van). Fontos, hogy ez a tartomány elég nagy legyen ahhoz, hogy az elvárt számú kapcsolatot ki tudja elégíteni. Ennek megfelelően egy `ip_conntrack_ftp`-t nem használó IP Tables tűzfalas FTP-kiszolgáló a 2. listán látható módon nézhet ki (lásd a 45. CD Magazin/proftpd könyvtárában).

Végezetül

A `proftpd` szolgáltatásait tekintve az egyik leggazdagabb FTP-kiszolgáló. A virtuális kiszolgálók létrehozásáról még szót sem ejtettünk. Sok szerencsét a kísérletezéshez, és írjatok bátran, ha valami kérdésetek van.

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

Mindaz, amit a memóriáról tudni érdemes

Elérkeztünk következő nagy témakörünkhöz: a memóriakezeléshez.

Vessünk egy pillantást az *1. ábrára*, ami egy jellemző folyamatrendszerű operációs rendszer felépítését mutatja. Eddig a két legalsó réteggel foglalkoztunk, a folyamat- és eszközkezelővel – ezeket az operációs rendszer magjának nevezzük. Ezen a szinten közvetlenül tarthatunk kapcsolatot a számítógép alkatrészeivel, és gyakorlatilag bármit megtehetünk. A továbbiakban az ábrán világoskékkel jelölt elemekről lesz szó, amelyet kiszolgálóknak (server) szokás nevezni, és a felhasználói alkalmazások számára kínálnak bizonyos szolgáltatásokat, olyanokat, amelyeket ők maguk képtelenek elvégezni. A kiszolgálók nem a rendszermag részei! Sokkal kevesebb jogosultsággal bírnak (például közvetlenül nem érhetik el a beviteli–kiviteli kapukat), mint például az eszközkezelők. A három legfontosabb kiszolgáló: a memóriakezelő, a fájlrendszer és a hálózatkiszolgáló. Az utóbbival nem foglalkozunk, mivel részletes ismertetése meghaladná e sorozat kereteit. A fájlrendszerre azonban részletesen vissza fogunk térni, ugyanis ez lesz a következő, egyben utolsó nagy témakörünk.

Mindenekelőtt azonban van még néhány fontosabb összefüggés a kiszolgálók és az operációs rendszer további részei között, amelyekre érdemes rávilágítanunk. Már sokszor kitértünk rá, hogy az operációs rendszerek két legfontosabb feladata az erőforrás-kezelés és a gép szolgáltatásainak kiterjesztése (az úgynevezett rendszerhívások bevezetésével). Láthattuk, hogy az erőforrások kezelésének oroszlánrészét a rendszermag végzi. Vajon mi foglalkozik a rendszerhívásokkal? A választ talán már sejtjük is: a kiszolgálók. Ők azok, akik értelmezik a beérkező rendszerhívásokat és az alsóbb rétegeket utasítják a kért feladat elvégzésére.

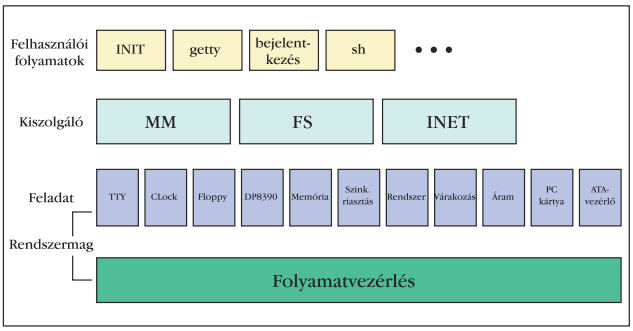
Fontos még megemlítenünk, hogy a kiszolgálók maguk is folyamatok, és alig különböznek valamiben a közönséges felhasználói folyamatoktól. A két lényegbevágó különbség az, hogy egyrészt magasabb szinten futnak, tehát több mindenhez van jogosultságuk (már ha az adott gép processzora támogatja a védelmi szinteket, például ilyen a Pentium). Másrészt a kiszolgálófolyamatok a rendszer indulásától kezdve egészen a leállításig folyamatosan futnak.

Ennek a kiépítésnek az az előnye, hogy a kiszolgálókat akár egy másik, távoli gépen is futtathatjuk. Ennek köszönhetően – apró módosítások árán – a fájlrendszert akár távoli fájlkiszolgálóként is használhatjuk.

Memóriarendszer

Az operációs rendszerek esetében a memória pazarlása megengedhetetlen fényűzés. Főleg, ha figyelembe vesszük azt az örök igazságot (amelyet gyakran Parkinson törvényeként is emlegetnek), amely szerint a programok mindig kitöltik a rendelkezésre álló memóriát. Könnyen elképzelhető, hogy nemsokára 1 GB memória már egy kiló kenyér árával fog vetekedni, ugyanakkor abban is biztosak lehetünk, hogy a jövő alkalmazásai már nem fogják annyival beérni, mint a maiak.

Világunkra az is jellemző, hogy többféle memóriát forgalmaznak. Akad nagyon gyors, ám rendkívül kicsi, felejtő és drága. Ugyanakkor nagyon olcsó, rettentő nagy kapacitással bíró,



1. ábra A folyamatstrukturált operációs rendszer elvi felépítése

tartalmát áramszünet esetén is megtartó, ellenben borzalmasan lassú is található közöttük. Az az egy azonban, amelyre mindenki áhítozik, a határtalan kapacitással és sebességgel bíró, mindenki számára megfizethető és tartalmát örök időkre megjegyző memória egyelőre sajnos még csak a vágyálmok birodalmában létezik.

Ha létezne, bizonyára minden gépbe ezt szerelnék, és nem lenne ennyi gond a memóriakezeléssel. Mivel nem ez a helyzet, a legtöbb számítógépben található memóriák a következő módon szerveződnek rendszerbe: legfelül a kicsi, ám rendkívül gyors és drága gyorstár (cache), alatta a már nem annyira gyors, de nagyobb kapacitású és olcsóbb központi memória (ez a RAM), végül a legalsó rétegben az akár több gigabájtos, olcsó, lassú, de nem felejtő lemezes tároló található. (A központi memóriát gyakran operatív tárnak is nevezik. Ez arra utal, hogy maga a futtatott program is itt foglal helyet.)

A hierarchikus memória kezelése nem egyszerű feladat: nemcsak azt kell megszerveznünk, hogy a szabad memóriarészeket nyilvántartsuk, és hatékonyan osszuk ki őket a futó programok között, hanem figyelni kell arra is, hogy mit melyik memóriában tárolunk. Ha egy folyamat például blokkolt állapotban van, akkor a központi tárból nyugodtan kitehetjük a lemezre, ezzel is felszabadítva némi helyet a többi program számára.

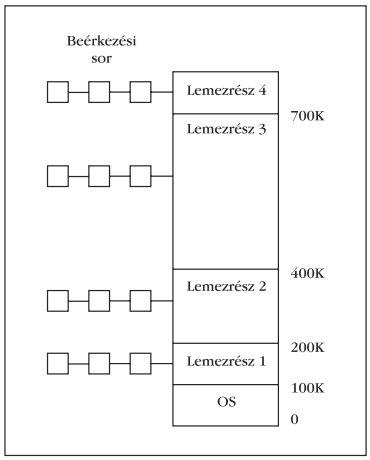
Az operációs rendszernek azt a részét, ami ezeket a feladatokat megvalósítja, memóriakezelőnek nevezzük.

Memóriakezelés lemez nélkül

A legegyszerűbb memóriakezelők nem használják a lemezt, csak a központi memóriával gazdálkodnak. Ezek közül is a legalapvetőbb az, amikor a memóriában egyszerre csak egy programot tárolunk – ezt a taktikát alkalmazta az MS-DOS is. Az operációs rendszer a lemezről betölti a kívánt alkalmazást a központi tárba (tekintet nélkül arra, hogy mi volt benne előzőleg), majd végrehajtja. Amint a program lefutott, kezdődhet minden elölről. Ennek a módszernek a megvalósítása rendkívül egyszerű, viszont nem teszi lehetővé több program egyidejűleg való futtatását, mivel az egész memóriát egyetlen programnak adományozta.

Egy többfeladatos operációs rendszerben azonban a memóriát egynél több programmal is meg kell osztani. A már legendásnak

számító OS/360-as rendszerben a következő módszert alkalmazták: a memóriát több különböző méretű részekre, úgynevezett lemezrészekre osztották fel, amelyek méretét az operátornak a rendszer indulásakor előre meg kellett határoznia. Amikor a program elindult, a rendszer megkereste magának azt a legkisebb lemezrészt, amelyikbe a program még belefért. Miután megtalálta, berakta a lemezrész várakozási sorába, és várta, hogy felszabaduljon. Miután felszabadult, a program betöltődhetett a kiválasztott memóriarészbe, és elkezdődhetett a végrehajtása. Ennek a módszernek MFT (Multiprogramming with Fix Tasks, lásd *2. ábrán-kon*) volt a neve.



2. ábra
Memóriakezelés az OS/360-asban

Az MFT-vel rengeteg gond akadt. Pazarló módszernek bizonyult, hiszen előre megadott méretű lemezrészek voltak, így hiába használtuk ki csak a felét a számunkra kiosztott memóriaszeletnek, akkor is elveszett az egész. Az operátornak ezért mindig mesterkednie kellett, hogy alkalmas méretű lemezrészeket hozzon létre, figyelembe véve olyan szempontokat is, hogy például az adott munkanapon inkább nagyobb vagy kisebb memóriaigényű programokat fognak-e futtatni.
A másik baj az volt, hogy könnyen előfordulhatott olyan eset, amikor egy-egy kisebb lemezrészért több program is sorban állt, míg a nagyobbak senkinek sem kellettek. Ekkor egy olyan módosítást találtak ki, hogy csak egy várakozási sor legyen, és ha egy lemezrész kiürül, akkor a rendszer azt a programot tölti be a megüresedett helyre, amelyik a legjobban ki tudja használni a lemezrészt és amelyik ezek közül legelöl áll a sorban. Ezzel a megoldással az volt a gond, hogy kirekesztő módon viselkedett a nagyon kicsi alkalmazásokkal szemben, mert ha kizárólag a méretüknél sokkal nagyobb lemezrészek ürültek ki, csak akkor töltődhettek be az üres helyre, ha nem várakozott másik, náluk nagyobb alkalmazás egy üres memóriaszeletre.
Az MFT a legkönnyebben megvalósítható olyan memóriakezelő módszer, amelyik azt támogatja, hogy egyszerre több program is a memóriában legyen. Sok évig sikerrel alkalmazták a nagygépes rendszereken – fent említett hátrányai ellenére. Ma már azonban sehol sem alkalmazzák, és egyetlen rendszer sem támogatja.

Relokáció és védelem

Még mielőtt belemerülnénk a ma használatos memóriakezelési eljárásokba, nem árt, ha előtte megemlítünk két olyan nem elhanyagolható nehézséget, amelyet minden memóriakezelési eljárásnak meg kell tudnia oldani.
Amikor egy programot lefordítunk, az utolsó lépésben, amit összeépítésének vagy szerkesztésnek nevezünk, a főprogramot, az eljárásokat és az osztott könyvtárakat (erről bővebben a következő részben szólunk) egy közös címtérben helyezzük el. A szerkesztést végző alkalmazásnak (linkernek) tudnia kell, hogy a főprogram hol fog kezdődni, és merre helyezkednek el majd azok az eljárások, amelyek a végrehajtás során meghívásra kerülnek.
Nehézség abból adódik, hogy a betöltéskor minden program más címtartományba kerül, és sosem tudhatjuk előre, hogy a mi programunkat melyik memóriarész fogja megkapni. Amikor a program végrehajtása elkezdődik és egy olyan utasítást kap, hogy például hívjuk meg az 50-es címen lévő eljárást, gond van. Az 50-es cím az operációs rendszer területéhez tartozik – a program nyilvánvalóan nem arra a címre kíván ugrani, hanem a saját lemezrészének kezdetéhez viszonyított 50-es címre. A megoldás az lenne, hogy a lemezrész kezdeti címéhez hozzáadjuk a program által megadott címet, így megkapjuk a meghívandó eljárás valódi helyét. Ezt a műveletet nevezzük eltolásnak (relocation).
Az operációs rendszerek különböző módon oldják meg a fenti feladatot, azaz a áthelyezést. Az előbb már említett OS/360-as például a program betöltésekor önműködően módosította az utasításokat, úgy, hogy minden címhez hozzáadta a kiválasztott lemezrész kezdőcímét. Ehhez azonban az kellett, hogy a szerkesztőprogram egy térképet helyezzen el, ami a rendszernek megadja, hogy mely értékeket kell úgymond „relokálni". Ez a módszer még ma is használatos, főleg a mikroszámítógépek világában.
Az áthelyezéssel szorosan összefügg a védelem kérdése. Az imént bemutatott módszerben a programok abszolút címekkel dolgoznak, ezáltal némi ügyeskedés segítségével könnyen írhatunk olyan programot, amelyik képes belepiszkálni más folyamatok memóriájába, vagy akár az operációs rendszer számára fenntartott területre is.
A védelem megvalósításának módja szorosan összefügg az áthelyezéseknél alkalmazott módszerrel, sőt az is fontos, hogy az adott rendszer milyen gépen fut. A legtöbb eszköz ugyanis segít e két dolog megvalósításában az operációs rendszernek. Maradjunk a szokásos példánál az OS/360-assal. Itt a memóriát 2 KB-os blokkokra osztották, és minden blokkhoz hozzárendeltek egy négybites kulcsot, amelyeket csak az operációs rendszernek áll módjában megváltoztatni. Amikor egy program hozzá akart férni egy memóriablokkhoz, a programállapotszó-regiszterbe (PSW) be kellett tölteni az adott blokkhoz tartozó kulcsot. Az ellenőrzés eszközszinten történt, és ha a kód nem egyezett, a művelet nem hajtódott végre.
Sokkal elterjedtebb volt egy másik módszer, amelyik egyszerre kínált megoldást az áthelyezésés a védelem kérdésére. Az eszköz két egyedi regiszterrel rendelkezett: a bázis- és a háttérregiszterrel. Amikor egy folyamat megkapta a futási lehetőséget, a bázisregiszterbe betöltődött a program memóriarészének kezdőcíme, a háttérregiszterbe pedig annak mérete. Innentől kezdve az eszköz minden memóriacímet a bázisregiszterhez viszonyított, így az operációs rendszernek többé nem kellett a kód módosításával foglalkoznia. A háttérregiszter jelenléte a védelem kérdését is megoldotta, mivel ha a program egy olyan címre akart hivatkozni, amelyik saját memóriarészén kívül volt (azaz a háttérregiszternél egy nagyobb értékre hivatkozott), a rendszer azonnal gyilkolt.
Ezt a módszert sok helyen alkalmazták, még az első IBM PC-kben található Intel 8088-as processzorok is ezen alapultak (habár itt mellőzték a háttérregisztert, így a védelmi részét gyakorlatilag kiiktatták). A 286-os processzoroktól kezdve azonban egy teljesen más módszer jelent meg, amit védett módnak (protected mode) nevezünk. De minderről majd a következő részben szólunk.

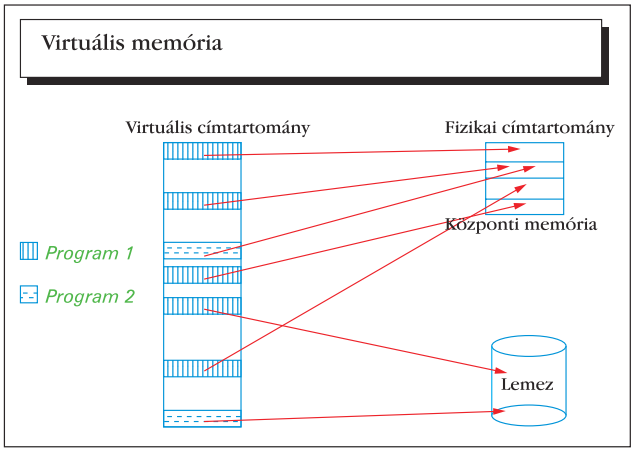
Csereterület és virtuális memória

Az emberiségnek hamar szembesülnie kellett azzal a ténnyel, hogy a memória sosem elég, és az összes futó program egy-

szerre általában nem fér el a memóriában. Ezért a központi tárat ki kell egészíteni a lemezzel. A lemezes tárolók a memóriához képest ugyan lassúak, viszont sokkal nagyobb kapacitással rendelkeznek, tehát nyugodt szívvel odapakolhatunk minden olyan dolgot, amire az adott pillanatban nincsen szükség, például a blokkolt folyamatok memóriarészeit.
A lemez és az operatív tár közötti ide-odapakolás megvalósítására kétféle megközelítés is létezik. Az első módszer a csere. Ilyenkor a programokat csak teljes egészükben mozgathatjuk a lemez és a központi memória között.
A csere azonban nem jelent megoldást arra a helyzetre, ha a futtatni kívánt program meghaladja a rendelkezésre álló szabad memória méretét. Ilyenkor hajdanán az volt a bevett szokás, hogy a programokat rétegekre darabolták, és ezeket a rétegeket megszámozták. Először a 0. réteg töltődött be a memóriába és kezdett el futni. Amikor a végrehajtás befejeződött, betöltötték a következő részt, és folytatódott tovább a program végrehajtása. Ezt overlay módszernek nevezik. Csak egy gond volt vele: programjának részekre való bontását a programozónak saját magának kellett megoldania. (Szerencsére a rétegek cseréjét az operációs rendszer maga is el tudta végezni). Ez nem volt igazán szórakoztató elfoglaltság, ezért egyre inkább felmerült az igény egy olyasfajta megoldás megalkotására, amely a programozót megkíméli az effajta lélekölő tevékenységektől.
Ekkor jött a nagy ötlet: a virtuális memória. Lényege az, hogy a programnak mindig csak azok a részei tartózkodnak a memóriában, amelyekre éppen szükség van – a többit a lemezen tároljuk. A virtuális memória másik vonzó tulajdonsága, hogy több program együttes futtatásánál is alkalmazhatjuk. Ebben az esetben több program darabjai tartózkodnak a memóriában.
Háromféle megvalósítását ismerjük a virtuális memóriának: a lapozást, a szegmentálást és a két módszer egyszerre való alkalmazását.

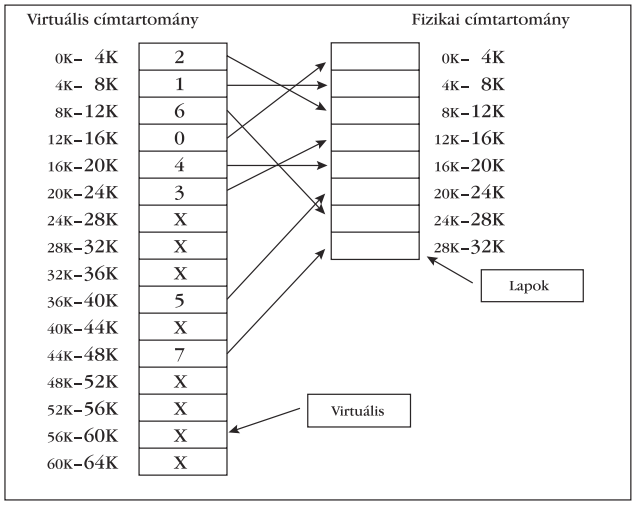
A lapozás

A virtuális memóriának az a kiinduló gondolata, hogy kettéválasztjuk a címtartomány és a memóriarekesz fogalmát. Hogy ez mit is jelent? Vegyünk példának egy olyan számítógépet, amely mondjuk 4 KB (4096 bájt) fizikai memóriával rendelkezik. A gép regiszterei 16 bitesek, így pontosan 65 536 bájt megcímzésére nyílik lehetőségünk. Az, hogy egy számítógép mekkora memóriát tud kezelni, kizárólag címregiszterei hosszától függ. Azoknak az értékeknek a halmazát, amelyet a címregiszterek felvehetnek, a számítógép címtartományának nevezzük. Jelen esetben ez a 0, 1, ... , 65 535. Ez azt jelenti, hogy legfeljebb 64 KB memóriát tudunk címezni (tehát hiába van a gépben mondjuk 128 KB memória, akkor is csak 64-et lehet belőle használni).
Amikor nem létezett még a virtuális memória, a memóriacímeket két részre osztották: a hasznosakra és a haszontalanokra. Az utóbbi csoportba nyilván azok a címek sorolódtak, amelyekhez már nem tartozott valódi memóriarekesz, azaz jelen példánknál maradva nagyobbak voltak, mint 4095.
Itt az ideje kettéválasztani a címtartomány és a memóriarekesz fogalmát. Számítógépünk egyszerre csak 4096 bájtot tud a memóriájában tartani, de mondhatjuk azt, hogy ezeknek a címei ne feltétlenül a 0 és 4095 közé essenek. Mondhatjuk azt is, hogy a fizikai memória első rekeszéhez rendelt cím legyen inkább a 4096, a második rekeszéhez a 4097, és így tovább. Ha úgy tetszik, egy leképezést (függvényt) határoztunk a címtartományból (0 ... 65 535) a valódi memóriacímekre (0 ... 4095).
Mi ebben a tudomány? Ha a gépben nincs virtuális memória, akkor úgymond állandó leképezésünk van a 0 és 4095 közötti címekre. Ha valamelyik program e tartományon kívül címezne, bizony csúnya véget ér a történet (például egy hibaüzenet kíséretében megszakad a futása). Ha viszont a virtuális memória működik, és tegyük fel, a 8192 és 12 287 közötti címet szeretnénk elérni, akkor a következő történik: először is mindaz kiíródik a lemezre, ami eddig a memóriában volt. Ezután meg kell keresni a 8192 és 12 287 közötti tartományt a lemezen, majd be kell tölteni a memóriába. Legvégül egy új leképezést kell létrehozni a virtuális és a fizikai memóriacímek között (*3. ábra*).

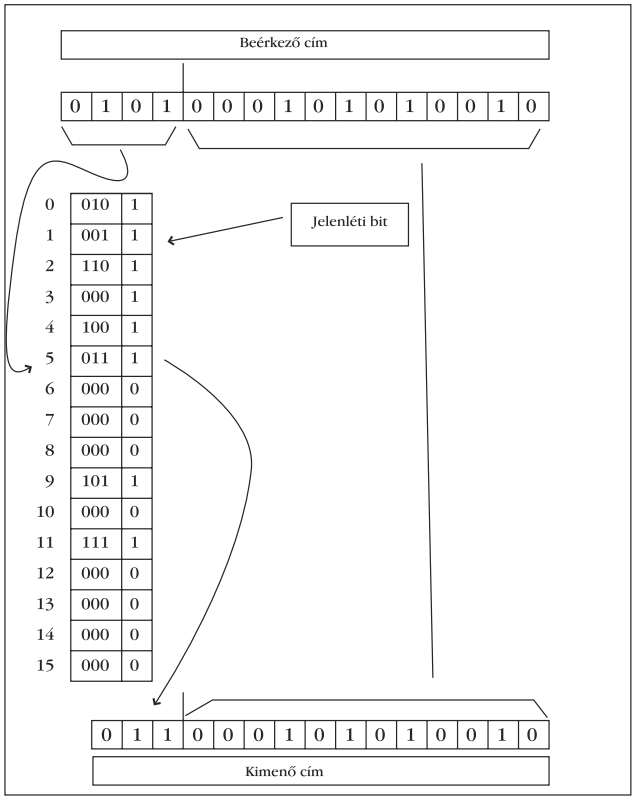


3. ábra A virtuális és a fizikai címtartomány kapcsolata

Ezt az eljárást lapozásnak nevezzük, a lemezről beolvasott memóriadarabkákat pedig lapoknak. A fent említett leképezés a virtuális és a fizikai címek között tulajdonképpen egy táblázat, amelyet laptáblának vagy memóriatérképnek hívunk.
(A fenti példában feltételeztük, hogy van elegendő hely a lemezen ahhoz, hogy az egész címtartományt lefedhessük. A való életben ez nincs mindig így, de az elv akkor is ugyanaz marad).
A virtuális és a fizikai címtartományt mindig azonos méretű blokkokra bontjuk (ezek lesznek a lapok), a méret mindig a kettő valamelyik hatványa. Manapság az általános lapméret 512 bájt és 64 KB közé esik, de ez mindig az adott feladattól függ. Így nem ritka a több megabájtos lapméret sem. A fizikai címtartomány egy-egy blokkjába fogjuk betölteni a lapokat, és ezeket lapkereteknek nevezzük.
Fel szeretnénk hívni a figyelmet arra, hogy sem a programozó, de még maga a futó program sem tud semmit minderről.
A programozónak úgy kell megírnia a kódját, mintha nem is lenne lapozás. Ő a memóriát irtózatosan nagynak, folytonosnak és lineárisnak látja. Például nem kell olyan dolgokkal foglalkoznia, hogy az adott memóriaszelet most a központi tárban vagy a lemezen van-e. A virtuális memória másik megvalósítása esetén a következő részben tárgyalt szegmentálásnál már egy kicsit más a helyzet, ott már nem árt, ha a programozó is tud a szegmensek létezéséről.
Mielőtt egy példán részletesen bemutatnánk a lapozás működését, a könnyebb megértés végett sokat segítene, ha a lemezt úgy képzelnénk el, mint azt a dolgot, ami az adott program memóriájának az eredeti változatát tartalmazza, és ami a fizikai memóriában helyezkedik el, az annak a másolata. Természetesen, ha a memóriában megváltozik a tartalom, a változásnak a lemezen is végbe kell mennie.
Lássuk, miként fest ez a gyakorlatban! Tegyük fel, hogy van 32 kilobájtos fizikai memóriánk és ehhez 64 kilobájt virtuális memória tartozik. A lapméret legyen 4 KB, ami azt jelenti, hogy a fizikai memóriában egyszerre nyolc lapot tárolhatunk (más-



4. ábra A laptábla



5. ábra Az MMU címátalakításának folyamata. Jelen gép processzora 15 bites címekkel dolgozik, de a virtuális címtartomány mérete 16 bit. Az MMU feladata az, hogy a 16 bites virtuális címből 15 bites valódi címet készítsen

ként fogalmazva: nyolc lapkeretet tartalmaz). Mivel a virtuális memória 64 kilobájtos, ezért ott kétszer annyi, azaz 16 lap raktározása lehetséges. Ne feledjük, hogy lemezeinken mindig az összes lap tárolódik, még azok is, amelyek éppen a memóriába is be vannak töltve. Tehát a példánkként szolgáló számítógép összes memóriája együttesen 64 KB és nem 32 + 64 KB! A 64 kilobájtos memóriánkat tehát felosztottuk egyenlő részekre, jelen esetben 4 kilobájtos lapokra, így összesen 16 lappal gazdálkodhatunk. A fizikai memóriába ezek közül be van töltve valamelyik nyolc. Hogy melyik nyolc, azt a laptáblából olvashatjuk ki. Ez tulajdonképpen egy táblázat, ami pontosan annyi bejegyzést tartalmaz, ahány lapunk van. Minden bejegyzéshez egy úgynevezett jelenléti bit tartozik, ami azt mondja meg, hogy az adott lap éppen be van-e töltve a fizikai memóriába vagy sem. Ha be van töltve, akkor azt is kiolvashatjuk a táblázatból, hogy melyik lapkeretében találhatjuk (*4. ábra*).

Nézzük meg, mi történik, ha a program végre szeretne hajtani egy `MOV REG,20818` gépszintű utasítást (firmware). Ez az utasítás jelentse azt, hogy valamelyik regiszterbe be szeretnénk tenni az 5000-es címen található értéket (és tegyük fel, hogy a címregiszter 16 bit hosszúságú). Természetesen ez egy virtuális cím, amivel a processzor nem igazán tud mit kezdeni, ezért majd át kell alakítani fizikai címmé.

Erről az úgynevezett MMU (Memory Management Unit, azaz a memóriakezelő egység) gondoskodik. Ez egy olyan eszköz, ami általában a processzorlapkán foglal helyet, de van olyan változata is, amelyben külön van választva a processzortól. Akárhogy is, a feladata ugyanaz.

Az MMU tehát megkapja az 20 818-os virtuális címet, és egyből egy 4 bites, úgynevezett virtuális lapszámra és egy 12 bites offsetre bontja. Az, hogy a címből hány bit a lapszám, és mennyi az offset, csakis attól függ, mekkora méretű lapokkal dolgozunk. (Mivel mi 4 KB-osakkal munkálkodtuk, ezért 12 bit az offset, mert 2^{12} az pontosan 4096). Ezzel meg is állapítottuk, hogy az 5. virtuális lapon van a keresett adat (a számozást a nullával kezdjük).

A következő feladat, hogy megkeressük a laptáblában az 5. laphoz tartozó bejegyzést, és megnézzük, hogy a memóriában található-e. Ha a jelenléti bit egyre van állítva, akkor igen, és kiolvassuk, hogy melyik lapkeretben lelhető fel – mondjuk az ötödikben. Ekkor már gyerekjáték megadni a fizikai címet, csupán ki kell számolnunk, hogy hol kezdődik az ötödik lapkeret, és hozzá kell adni az offsetet. Az így kapott értéket a processzor már fel tudja dolgozni.

Igen ám, de mi történik, ha az adott lap mégsincs a fizikai memóriában (tehát a jelenléti bit nulla). Ebben az esetben laphibáról beszélünk. Ilyenkor az operációs rendszernek meg kell keresnie a merevlemezen a kért lapot, majd be kell töltenie valamelyik lapkeretbe. Ha ez megtörtént, az egész művelet kezdődhet elölről.

Nem mindegy, hogy melyik lapkeretbe töltjük be az új lapot. Azt az elvet, ami alapján az operációs rendszer eldönti, hogy melyik lap helyére tegye be az újat, lapcserélési algoritmusnak nevezzük.

Vegyük észre, hogy a feladat oroszlánrészét az eszközök végzik el, az operációs rendszernek csak akkor kell intézkednie, ha a kívánt lap nem tartózkodik a fizikai memóriában. A mai korszerű processzorok szinte kivétel nélkül támogatnak valamely virtuális memóriát kezelő módszert. Pentium processzorokban például ez a képesség nagyon fejlett, a lapozás mellett a szegmentálást és a kettő együttes kombinálást is támogatják. Az UltraSparc processzorok esetében is a virtuális memória lapszervezésű. Mindezek ellenére egyre jobban kezd elterjedni a lapkezelés programon keresztüli megoldása. A mai RISC-processzorokat támogató gépek, például az Alpha esetében ez már így működik. Hogy miért ezt a módszert használják, arról a következő részben fogunk szólni.

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.

Eszközválasztás

Érvek és ellenérvek négy webfejlesztő eszköz – a mod_perl/Mason, a J2EE, a Zope és az OpenACS – mellett és ellen.

Ha valaki arra kérne bennünket, hogy nevezzük meg a legjobb autót a piacon, valószínűleg azt válaszolnánk, hogy ez attól függ, ki fogja használni azt az autót. Végül is egy manhattani nyolcfős család valószínűleg más típusú járművet szeretne, mint egy agglegény kódlovag Észak-Dakota faluvidékén. Ugyanez igaz a programozási nyelvek és a fejlesztőeszközök estében is. Mindegyiknek megvan a maga helye és alkalmazási területe.

Bár ez nyilvánvalónak tűnhet, számos programozó úgy gondolja, hogy az általa használt eszközkészlet vagy nyelv bármikor bármely feladat megoldására alkalmas. Mint a régi mondás tartja: ha egyetlen munkaeszközöd egy kalapács, minden teendő olyan, mint egy szög. Egyetlen programozási nyelv sem tökéletes az összes feladathoz. Ezért ismernek a tapasztalt programozók több különféle nyelvet és tanulnak meg folyamatosan újakat.

Egészen az utóbbi pár évig a programozók programjaikat többnyire sebességre és alacsony memóriaigényre igazították. Ez érthető, hiszen a processzorok viszonylag lassúak, a memóriák pedig elég drágák voltak, így aztán amelyik program nem a legtöbbet próbálta meg kipréselni az alkatrészekből, az vacaknak tűnt. Manapság azonban már az olcsóbb, gyors számítógépek és a kedvező árú, méretes memóriák áldásait élvezhetjük. Ez azt jelenti, hogy a programtervezőknek lehetőségük nyílik olyan nyelveket használni, amelyek a gyors fejlesztést és a hosszú távú programkarbantartást részesítik előnyben. Ez nem azt jelenti, hogy ellenzem a memória vagy sebesség javítását, egyszerűen csak kevésbé tartom fontosnak ahhoz képest, hogy üzembiztos, karbantartható programokat fejlesszünk gyorsan és egyszerűen.

Közel két évvel ezelőtt elhatároztam, hogy egy hosszú cikksorozatot szentelek négy alapvető webfejlesztő módszernek: a mod_perl/Mason (Linuxvilág 1., 3–4. szám), a J2EE (Linuxvilág 13–14. szám), a Zope (Linuxvilág 15–19. szám) és az OpenACS rendszereknek (Linuxvilág 22–25. szá,m). Ezek a rendszerek nemcsak érdekesek és hasznosak, de egyben gondolatébresztők, továbbá új távlatokat nyitnak a webfejlesztők előtt. Bár az egyes közösségek között néha fellángolnak a viták arról, hogy melyik termék a legnagyszerűbb, a valóság az, hogy mindegyik egy kicsit más típusú feladatot próbál megoldani.

E hónapban arra szánunk egy kis időt, hogy megpróbáljuk összefoglalni az elmúlt pár évben megismert keretrendszereket és ötleteket. Természetesen nem várom azt, hogy aki a cikket elolvassa, azonnal az összes itt megnevezett alkalmazást használja is; mindössze abban reménykedem, hogy a legelfogultabb rajongónak is egy kis gondolkodnivalót tudok nyújtani.

mod_perl/Mason

Az Apache megérdemelten a nyílt forrású kezdeményezések egyik mintapéldánya. Megbízható, jól beállítható, kitűnően leírt, üzembiztos és bővíthető. Lenyűgöző dolgokat vihetünk végbe az Apache-csal, és sokféle módon igazíthatjuk hozzá saját igényeinkhez. Ha olyan webalkalmazást kell írnunk, aminek a lehető leggyorsabban le kell futnia, új modulokat írhatunk C-ben, amelyek aztán zökkenőmentesen illeszkednek az Apache-hoz.

Bár a C-programok gyorsan végrehajtódnak, és az Apache-könyvtárak (amelyeket manapság Apache Portable Runtime néven ismerhetünk) igencsak széles körű szolgáltatásokat és támogatást adnak a modulok alkotóinak, a fejlesztés C alatt lassabb és hibákra hajlamosabb, mintha valamilyen magasabb szintű nyelv, például Perl vagy Python alatt dolgoznánk. Így aztán talán nem is meglepő, hogy léteznek olyan Apache-modulok, amelyek ezeket a nyelveket ágyazzák be az Apache kiszolgálóba. A `mod_perl` lehetővé teszi, hogy C helyett Perl nyelven készítsünk Apache-modulokat – a Perl sebességét és rugalmasságát meglovagolva –, miközben közel korlátlan uralmat kapunk kiszolgálónk felett.

Tulajdonképpen ha valaki nagy teljesítményű webalkalmazást kér tőlem, különösen ha szövegfeldolgozás vagy relációsadatbázis-kezelés is a feladat részét képezi, a `mod_perl` mindig előtérbe kerül. Megírhatnám a modult C-ben is, de minek bajlódjak vele? Előfordul, hogy valóban érdemes C alatt dolgozni, de a legtöbb esetben a `mod_perl` sebességét még a nagy teljesítményű alkalmazások esetében is kielégítőnek találtam. Természetesen a `mod_perl` csodájának fénye némileg megkopik, amint a grafikus tervezők is színre lépnek. A tervezők nem igazán szeretnek kódot szerkeszteni, valahányszor megváltoztatják a stílust (vagy a tartalmat) az adott lapon, és ha rászabadítjuk őket a Perl-modulok forráskódjára, abból nem sok jó származik. Így aztán különféle sablonrendszerek tucatjai születtek, amelyek mindegyike valamilyen módon képes keverni a Perl és HTML nyelvet. Az egyik legnépszerűbb közülük a Mason, amely tudását az évek során már számos komoly terjesztő honlapon keresztül bizonyította.

A Mason valóban csodálatos eszköz, ami kitűnő egyensúlyt teremt a gyors fejlesztés (a Perlnek köszönhetően), a könnyű karbantarthatóság (a Masonnek hála) és (a `mod_perl` érdemeként) a gyors végrehajtás között. A Mason-levelezőlistától előnyös tájékoztatást és jó támogatást várhatunk, a csomag fejlesztőinek munkája pedig csodálatra méltó, mivel az idők során folyamatosan tökéletesítették programjukat. A Mason korszerű változatainak beállítását, használatát és hibakeresését látva azok az első változatok, amelyeket pár évvel ezelőtt először használtam, kezdetlegesnek tűnnek.

A Mason egyben háttér és keretrendszer, amelyben saját alkalmazásainkat készíthetjük el. Igaz ugyan, hogy az `Apache::Session` felhasználásával könnyedén készíthetünk sütiket és egyszerűen rendelhetünk a felhasználókhoz egyedi azonosítókat, de a kiforrott alkalmazásokban megtalálható felhasználói feliratkozás, a csoportok és a jogosultságok rendszerét mind magunknak kell megvalósítanunk. Néhány projektben ez a megfelelő megoldás, hiszen a szükséges rugalmasságot kínálja. Ha azonban már ötödjére kapjuk azon magunkat, hogy ismét egy felhasználókat és jogosultságokat kezelő rendszert készítünk, esetleg úgy dönthetünk, hogy egy kicsit több hátteret nyújtó lehetőséget választunk.

Java és J2EE

A Sun már jó néhány éve hirdeti kiszolgálóoldali megoldásként a Javát. A J2EE (Java 2, Enterprise Edition) elnevezés egy olyan gyűjtőfogalom, amelybe több, a fejlesztőket ilyen megoldások kialakításában segítő különféle módszer tartozik. A servletek olyan osztályok, amelyek a kiszolgálón valósítanak meg kódot; a JavaServer Pages (JSP-k) olyan Java/HTML-sablonok, amelyeket futásidőben fordítunk servletté. A JDBC-illesztő adatbázisok elérését teszi lehetővé, míg az Enterprise JavaBeans a tranzakciókat és az önműködő relációsobjektum-átalakítást kezeli. Ahhoz, hogy a Java világába beléphessünk, rengeteg rövidítést és módszert kell megismernünk, valamint több különféle szabvány számos változatát elsajátítanunk.

Amióta először megjelent, többször dolgoztam már Javával, s szinte minden esetben arra jöttem rá, hogy hiába szeretném, hogy érdekeljen, képtelen vagyok rávenni magamat a megkedvelésére. A Java önmagában nem rossz, és a különféle módszerek, amiket letesz az asztalra, mind igen lenyűgözők. A servleteket könnyű elkészíteni; a JSP-k (különösen a JSP-khez készíthető egyedi tagok) érett és figyelemre méltó sablonrendszert alkotnak, végül a JDBC mindent biztosít számunkra, amit egy adatbázis-kezelő felülettől valaha is elvártunk. Igaz ugyan, hogy az EJB használata a legtöbb projektben kétségtelenül túlzás, ugyanakkor hihetetlen hasznos lehet azokban a nagy fejlesztői csoportokban, amelyeknek a Sun eredetileg szánta. Továbbá számos jó megvalósítás, köztük kitűnő nyílt forrású alkalmazáskiszolgálók és eszközök születtek, ami mindenképpen lenyűgöző és ösztönző erőt képvisel.

Úgy tűnik, hogy a webfejlesztés világának „nagycége" a Java lett. A dolgokat megbízhatóan oldja meg, rengeteg képesség rejlik a tarsolyában, és számos szabványt támogat, fejlesztőeszközök seregei állnak a rendelkezésünkre – és meglehetősen sok ember használ Javát. Csak éppen a Java-projektekkel járó pluszmunka egy kicsit sok az én ízlésemnek. Már az is hosszú időt vesz igénybe, ha csak azt szeretnénk megállapítani, hogy melyik szabvány melyik változata melyik Jakarta alprojekt melyik változatához való. Ahogy sokkal érdekesebb kis cégnek dolgozni, mint nagynak, ugyanúgy sokkal érdekfeszítőbb szerintem Perl vagy Python alatt programozni, semmint Javában. Ráadásul a J2EE hasonló gondoktól szenved, mint amelyeket a `mod_perl` és a Mason esetében leírtam, nevezetesen, hogy tisztán csak hátteret ad, anélkül, hogy bármi figyelmet szentelne a beépített alkalmazásoknak. A fejlesztők csodálatos dolgokat hozhatnak létre, de minden projektben mindent újra ki kell fejleszteniük.

Talán a legjobban azt kedvelem a Java világában, hogy olyan nagy figyelmet fordít a karbantartható és megbízható programokra. Jó néhány próba- és fejlesztőeszköz, többek közt az Ant, a Cactus, a JUnit és a log4j lehetővé (sőt talán magától értetődővé) teszi, hogy a programozók teljes körű teszteket végezzenek a program kiadása előtt.

Tehát akkor a Java jó választás a webfejlesztéshez? Véleményem szerint minél nagyobb projekten dolgozunk, annál inkább érdemes elgondolkodnunk a Javán mint lehetőségen. De a klasszikus kis webalkalmazások esetében, amelyekkel a kis műhelyek dolgoznak, a fejlesztéshez kapcsolódó elkerülhetetlen többletmunka túlságosan komoly tényező, amit már nem hagyhatunk figyelmen kívül.

Zope

A Zope ékes bizonyítéka annak, hogy a nyílt forrású programok nem csak másolják kereskedelmi versenytársaikat. A Zope az objektum-adatbázist ötvözi a többprotokollú kiszolgálóval, továbbá gazdag objektumkészlettel és gördülékeny webalapú kezelőfelülettel ruházza fel őket. A Zope újító jellegű, okos, öröm vele dolgozni, és azon ritka nyílt forrású programok közé tartozik, amelyet a felhasználóknak terveztek, tehát nem csak a kódlovagoknak szól. A grafikusok örömmel hallják, hogy lehetőségük nyílik a dokumentum bármely korábbi állapotához visszatérni, a webalapú kezelőfelület „vissza" (undo) képességével.

A Zope több programozói felülettel is rendelkezik, amelyek a hatékonysággal szemben az egyszerűséget részesítik előnyben. Egyszerű DTML-sablonokat és Python-parancsfájlokat készíthetünk, használhatjuk az elbűvölő ZPT sablonokat, amelyek teljesen elválasztják egymástól a programot és a megjelenítési logikát, vagy végigjárhatjuk a teljes utat és egy új Zope-terméket készíthetünk. Az igazi erő a Zope-termékekben rejlik. Mivel minden termék egyben egy osztály is, különféle címeken egyetlen termékből több példányt is létrehozhatunk. Mivel az objektumok saját objektumrendszereiken kívül a címrendszer alapján is örökölnek (szerzeményezés), a példány jogosultságai, viselkedése vagy a kinézete címenként eltérő lehet.

No, ez eddig úgy hangzott, mintha a HTTP óta a Zope lenne a legjobb dolog, ami a Weben megjelent. Valóban, a Zope-kóderek növekvő száma egyben azt is jelenti, hogy egyre több ingyenes letöltés lesz elérhető és egyre több kereskedelmi termék használja alapjául a Zope-ot.

Sajnos a Zope-nak is megvannak a maga hibái, az első és legnagyobb az, hogy a tanulási görbéje igencsak meredek lehet. Mégha tapasztalt webfejlesztő is valaki, a Zope megköveteli, hogy majdnem az összes elgondolást az alapoktól újra megtanulja, megváltoztatva szinte minden szokást, amit az évek során összeszedett. Ez nem feltétlenül káros dolog, hiszen a Zope-ban elég jó megoldásokat találunk, de meglepetést okozhat és óvatosságra késztethet, egyszerűen azért, mert a Zope felhasználása a kezdeti, induló időszakban kétségtelenül lelassítja a dolgokat.

A másik gondom a Zope-pal az objektum-adatbázis alapja. Az objektum-adatbázisoknak hagyományosan több nehézségük is akad, és a ZODB szépen követi is ezt a hagyományt. Ugyanakkor a relációs adatbázisok még mindig elég általánosak, az emberek gyakran erre számítanak (általában meg is követelik). Elméletben ez sem gond. A Zope beépített ZSQL tagfüggvényei lehetővé teszik, hogy mindenféle különösebb erőlködés nélkül látványos dolgokat műveljünk a relációsadatbázis-lekérdezésekkel. A gond akkor válik nyilvánvalóvá, amikor az adat két különféle hely között oszlik meg: a ZODB és a relációs adatbázis között. Én például az összes adatomat egyetlen központi helyen szeretem tartani, így aztán ez a fajta megosztás számomra eléggé kiábrándító.

Azután ott van a sebesség kérdése: a Zope kifinomult jogosultság- és szerzeményezésrendszere valószínűleg gyorsabb, mint ahogy te vagy én magunktól meg tudtuk volna írni, de még így is elég lassúcska. A nehézség hivatalos Zope-megoldása a ZEO (Zope Enterprise Objects), amely lehetővé teszi, hogy egyetlen objektumadatbázist egyszerre több Zope-kiszolgáló is elérhessen. Jelenleg napi egymillió találatot bír el, ami a legtöbb lapon , amin dolgoztam, több mint elég. A különlegesen nagy webhelyek készítői azonban már aggódhatnak a Zope gyorsasága miatt, vagy másik megoldásként tervbe vehetik egy felsőkategóriás vas megvásárlását a ZODB kiszolgálóhoz.

Végül: a Zope-termékek többnyire függetlenek. A jó hír ebben az, hogy a fejlesztők párhuzamosan, a másik akadályozása nélkül tudnak dolgozni. A rossz hír, hogy a dolgok nincsenek annyira egységesítve, mint kellene: egységes irányítás nélkül a szolgáltatások ismétlődnek. Lehet, hogy egy ilyen méretű

nyílt forrású projekt esetében ez már elkerülhetetlen, mindenesetre kicsit kiábrándító.
Az utóbbi egy-két évben a Zope Corporation az alkalmazásfejlesztés helyett a tartalomkezelés területén kezdte el reklámozni a Zope-ot. Természetesen minden tartalomkezelő rendszert újra kell programozni és módosítani kell a vásárló igényeinek megfelelően, így a különbség nem annyira szembetűnő. Bár a Zope nem az egyetlen nyílt forrású tartalomkezelő rendszer a piacon, kétségtelenül az egyik legkifinomultabb és egyben az egyik legkiforrottabb is.
Saját munkáim esetében akkor választom a Zope-ot, ha az ügyfél projektje jelentős mértékű trükkös fejlesztést igényel, vagy ha viszonylag egyszerűen használható felhasználói felületet szeretne, illetve ha tartalomkezelésről van szó. Továbbra is nagyra tartom, és valószínűleg az elkövetkező években is elég sokat dolgozom majd a Zope-pal.

OpenACS

Az OpenACS a születésekor közösségi weblapokhoz tervezett kifinomult adatmodell volt, számos Tcl nyelven írt web-, illetve adatbázissablonnal kiegészítve. Idővel aztán egy sokkal nagyobb, sokoldalúbb projektté nőtte ki magát: független csomagokkal rendelkezik, amelyek a programot és az adatmodellt egyaránt képesek frissíteni, zökkenőmentesen képes együttműködni az Oracle vagy a PostgreSQL rendszerekkel, és kifinomult sablonozórendszert tartalmaz, amely a kódot elválasztja a HTML-kimenettől. Továbbá az OpenACS rengeteg előre elkészített alkalmazással érkezik, amelyek szinte bármit képesek megoldani, amire csak egy közösségi weboldalon szükségünk lehet – a webnaplóktól kezdve a gyakran ismételt kérdéseken át egészen az e-boltig. Igen rövid idő alatt elkészíthetünk egy weboldalt, s ehhez semmi mást nem kell használnunk, kizárólag a böngészőnket.
Mindig az OpenACS-t ajánlgatom azoknak a nonprofit szervezeteknek, akik hálózati közösségeket akarnak létrehozni, el akarják érni a tagjaikat, le akarják bonyolítani a tagok közti vitákat, vagy egyszerűen adatokat szeretnének közölni a módszer mélyebb ismerete nélkül.
Azt mondják, az OpenACS több gonddal is küszködik. Az első és legfontosabb a tanulás. A Zope megismerése elég nehéz, hiszen oly sok módszert szükséges megérteni hozzá. Az OpenACS sokkal egyszerűbb modellel bír, de az égvilágon mindent relációs adatbázisban tárol. Noha ez azt jelenti, hogy minden adat egy helyen van, a relációs adatbázisok hírhedten rosszak a rendszerkezelésben és a világ összes okos OpenACS-fejlesztője sem képes ezt kiküszöbölni.
Így aztán, az OpenACS-csel való munkához meg kell tanulnunk, hogyan készíthetünk egyszerű objektumöröklési modellt és a hozzátartozó teljes körű felületet, ami elérhetővé teszi.
Ha még soha nem készítettünk tárolt eljárásokat, vagy nem dolgoztunk olyan adatbázissal, ami táblák tucatjait vagy százait tartalmazza, lehet, hogy az OpenACS-hez szükséges háttértudás végül letaglóz minket.
Az OpenACS másik nehézsége, hogy csak kevés leírás áll a fejlesztők rendelkezésére és jóformán semmi a végfelhasználók számára. Az OpenACS kétségkívül összetett rendszer, amelyet egy szakmán kívüli ember számára elég nehéz leírni, de meglehetősen bosszantó, ha semmilyen segítséget sem találunk a témában. Becsületükre legyen mondva, a fő openacs.org honlapot mostanában modellezték és írták újra, röviddel azelőtt, hogy ezt a cikket megírtam volna, és úgy tűnik, e tekintetben is akad néhány kedvező változás.
Végezetül kicsit ironikusnak találtam, hogy az OpenACS egyre lomhábbá vált az idők folyamán. Talán elfogadható érv, hogy ez amiatt történt, mert a legfrissebb változat (e sorok írásakor a 4.x) sokkal többet tud a felhasználók, csoportok és engedélyek témakörében, mint az elődei, és ezek ellenőrzése minden HTTP-kérelemnél időt vesz igénybe. Ezen felül a fejlesztők is tudják, hogy sok javítást még végre lehet hajtani ahhoz, hogy ne igényeljen minden kérelem olyan sok adatbázis-lekérdezést.

Összegzés

Néhány nappal a cikk megírása előtt megjelent a Weben egy új hír arról, hogyan ültették át a Yahoót PHP webprogramozási környezet alá. Én személy szerint más nyelvekkel szeretek dolgozni, és nem éreztem volna nagy kedvet, hogy a teljes Yahoót új nyelven írjam újra. De a Yahoo egyedi igényeinek megfelelően úgy tűnik, hogy a PHP tényleg jó választás.
Kapnak egy jó pontot tőlem, amiért végiggondolták az összes lehetőséget, mérlegelve az előnyöket és hátrányokat, míg végül megkapták azt, ami leginkább illik az igényeikhez.
Ahogy már a cikk elején is írtam, mélyen hiszek abban, hogy mindig meg kell keresni azt a módszert, ami az adott feladat szerint igényekhez a legjobban igazodik. Nekem, a fejlesztőnek ez azt jelenti, hogy folyamatosan új nyelveket, módszereket kell megismernem és elsajátítanom; ugyanakkor azt is magában foglalja, hogy az ügyfeleim olyan megoldásokat kapnak, amelyek illeszkednek a feladataikhoz, és én mint programtervező széles körű és mélyebb tudást gyűjtök.
Az a tény, hogy ezek a rendszerek az Interneten ingyenesen hozzáférhetők, azt mutatja, hogy az egyetlen akadályt csak a ráfordított idő és erőfeszítés képezheti. Mindenkit arra buzdítok hát, hogy szorítson egy kis időt a kipróbálásra – te is és a veled dolgozó emberek is kétségtelenül értékelni fogják az eredményt.

Linux Journal 2003. február, 103. 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

A `mod_perl`-ről a ➲ http://perl.apache.org lapon, a Masonről pedig a ➲ http://www.masonhq.com lapon olvashatunk.

A J2EE témával mélyebben a ➲ http://java.sun.com foglalkozik. Az Apache Software Foundation Jakarta Projektje a nyílt forrású Java programhoz a ➲ http://jakarta.apache.org címen lelhető fel.

A Zope főlapja a ➲ http://www.zope.org. Akik újak a Zope-témában, nézzenek el a ➲ http://www.zopenewbies.net lapra; a nagyobb tudású felhasználók a ➲ http://www.zopezen.org-ot keressék. Végül az OpenACS közösséget az ➲ http://www.openacs.org címen érhetjük el.
A PostgreSQL adatbázist, amelyet az OpenACS-hez használhatunk, a ➲ http://www.postgresql.org címen találjuk.

Az I. és II. évfolyam egységára lapszámonként 500 Ft

#1. 1. szám 2000. november, 2 CD _____ pld.
#2. 2. szám 2000. december, 2 CD _____ pld.
#3. 1. szám 2001. január, 2 CD _____ pld.
#4. 2–3. szám 2001. február–március 3 CD _____ pld.
#5. 4. szám 2001. április, 2 CD _____ pld.
#6. 5. szám 2001. május, 2 CD _____ pld.
#7. 6-7. szám 2001. június–július 3 CD _____ pld.
#8. 8. szám 2001. augusztus, 2 CD _____ pld.
#9. 9. szám 2001. szeptember, 2 CD _____ pld.
#10. 10. szám 2001. október, 2 CD _____ pld.
#11. 11. szám 2001. november, 2 CD _____ pld.
#12. 12. szám 2001. december, 2 CD _____ pld.

A III. évfolyam 7. számig a magazin egységára lapszámonként 1000 Ft

#13. 1–2. szám, január–február, 2 CD _____ pld.
#14. 3. szám, március, 2 CD _____ pld.
#15. 4. szám, április, 2 CD _____ pld.
#16. 5. szám, május, 2 CD _____ pld.
#17. 6. szám, június, 2 CD _____ pld.
#18. 7. szám, július, 2 CD _____ pld.
#19. 8. szám, augusztus, 2 CD _____ pld.
#20. 9. szám, szeptember, 2 CD _____ pld.

Az akció keretében a megrendeléseket csak postai utánvéttel tudjuk teljesíteni. Az utánvét költsége egységesen 300 forint minden csomagra.

A kitöltött megrendelőlapot küldje be szerkesztőségünkbe faxon vagy postán.

Cím: 1081 Budapest, Népszínház u. 29.
Fax: 303-16-19

Megrendelő neve: _____

Cég neve: _____

Cím: _____

Telefonszám: _____

Fax: _____

E-mail: _____

_____

megrendelő aláírása

Grafikonon a vállalkozás

A bemutatásra kerülő nyílt forráskódú eszközök segítségünkre lehetnek abban, hogy nyomon követhessük bármilyen vállalkozásunk alakulását.

Itt van, François, a Collins-szótár 509. oldalán. A vállalkozás – ami e havi beszélgetésünk témája – a francia entreprise szóból származik. Látod, mon ami, úgy van, ahogy mondtam neked: a meghatározás szerint ez olyan projektet vagy vállalkozást takar, amely merészséget vagy erőfeszítést igényel. Manapság legtöbbször valamilyen üzleti tevékenységet folytató szerveződést értünk alatta. Ettől függetlenül a meghatározás így is megállja a helyét.

Az emberek naponta vágnak bele újabb és újabb feladatok megoldásába (különféle projekteket indítanak, ha így jobban tetszik). Ezeknek a vállalkozásoknak a sikere gyakran szoros kapcsolatban áll azzal, hogyan irányítják őket, mennyire körültekintő a tervezés, és milyen gondosan követik a folyamatokat. Mon Dieu, François! Miért nem szóltál, hogy már ilyen késő van? Már meg is érkeztek a vendégeink. Bonjour, mes amis, isten hozott titeket Chez Marcelnél, a kitűnő Linux-konyha és az ellenállhatatlan borok otthonában. François! Hozd fel, kérlek az 1996-os South Australia Coonawarra Shirazt, immédiatement!

Ahogy épp François-nak magyaráztam az imént, mai menünk középpontjában az új, vakmerő projektek indítása, vagyis a vállalkozás áll. Egy vállalkozás elindításának nyilvánvaló tervezési és irányítási szempontjai mellett izgalmas feladatnak tűnik egy nagy projekt pillanatnyi állapotának a feltérképezése is. Ezt általában Gantt-grafikonokkal végzik, amelyet ***Henry L. Gantt*** fejlesztett ki 1917-ben. A jól ismert vízszintes oszlopgrafikont fejlesztette tovább könnyen használható termelésirányítási eszközzé, vizuális eszközt biztosítva egy projekt állapotának ábrázolására, és ezzel nagymértékben egyszerűsítve a munkafolyamatok nyomon követését és az irányítást. Nézzük, hogyan is működik. A Gantt-grafikon vízszintes tengelye a folyamatra szánt időt ábrázolja. Ez feloszthatóra napokra, hetekre, vagy bármilyen más időegységre, ami az ábrázolás szempontjából ésszerű. Gondoljunk arra, hogy egy projekt rettentő hosszú ideig is eltarthat. A függőleges tengelyen a munkafolyamatot felépítő feladatok vannak felsorolva. Például nemrégiben fel kellett töltenem a raktárkészletemet itt, Chez Marcel borospincéjében. A munka ebben az esetben leltározásból (François), egy kis polcátszervezésből (François és Chef Marcel), szállításból (Henri of Henri's Fine Wines) és végül a raktár újrafeltöltéséből állt (François).

Ha már François-nál tartunk, az én nagyrabecsült pincérem visszaérkezett a pincéből. François, tölts kérlek a vendégeinknek! Ez a Coonawarra Shiraz biztosan ízleni fog, mes amis. A Shiraz hagyományos zamata mellett észrevehettek egy kis csokoládé ízt a bukéjában, és talán egy kis mentát is. És ez az íz… Bocsánat, elkalandoztam egy kicsit, a Gantt-grafikonokról volt szó. Az egyes feladatokat különböző hosszúságú (esetleg más-más színű) vízszintes sávok jelölik, ahol a hosszúság az adott feladathoz szükséges időt mutatja. A projekt bármelyik pontján húzható egy függőleges szakasz a grafikon tetejétől kezdődően, amely jelentésül szolgál arról, hogy a munkafolyamat éppen milyen állapotában van. Egyszerű, nem?

Az egyszerűség az a gondolat, ami a Graphical UI Gantt Chart Generator (az eredetileg ***Jason R. Govig*** és ***Seth Goldstein*** által karbantartott, pillanatnyilag ***Glen Stewart*** fejlesztése alatt álló program) alapeszméje. Ez a Gantt-grafikonok előállítására hivatott webalapú rendszer egyszerűen CGI parancsfájlok gyűjteménye. Tökéletesen megfelel a célra, ha könnyen használható, hálózati munkára alkalmas ábrázolóprogramot keresünk. A működéséhez szükségünk van az Apache-ra és két Perl-modulra, ezek a `CGI.pm` és a `Date::Manip.pm`. Ezeket a Perl-modulokat legegyszerűbben a CPAN paranccsal telepíthetjük, a parancssorból:

```
perl -MCPAN -e "install Date::Manip"
perl -MCPAN -e "install CGI"
```

Hozzá kell tennem, hogy mindezt rendszergazdai jogosultsággal kell elvégeznünk. Ha korábban még soha nem használtuk a CPAN parancsot Perl-telepítésre, akkor több kérdésre is felelnünk kell majd a beállításra vonatkozóan. Ezt a beállítást csak egyszer kell megtennünk, onnantól kezdve a telepítés teljesen zökkenőmentes folyamat. A csomag használatához böngészőprogramunkba írjuk be a a megfelelő címet. Az én rendszeremen ez valahogy így fest: ➲ http://webserver/gantt/. A Gantt-grafikon csomag forrása az ➲ http://associate.com/gantt honlapon érhető el. Először csomagoljuk ki a forráskódot webkiszolgálónk dokumentumkönyvtárának gyökerébe:

```
tar -xzvf gantt-1.0.tar.gz
mv gantt-1.0 gantt
```

Az egyszerűbb kezelhetőség kedvéért én rögtön átneveztem a könyvtárat. Bármilyen nekünk tetsző nevet választhatunk. Vessünk egy pillantást a ***distribution*** könyvtár alatt lévő könyvtárakra, különös tekintettel a ***/users***-re! A felhasználónak, ami alatt az Apache-ot futtatjuk (az én rendszeremen ez a www), a teljes könyvtárra írási joggal kell rendelkeznie. Keressük meg a felhasználót és a csoportot a ***httpd.conf*** fájlban. Néhány változtatásra is szükségünk lesz, ami leginkább két állományt érint. Az első a ***variables.pl*** nevű fájl. A megváltoztatandó sorok a saját oldalunk beállításai, nevezetesen a dokumentumgyökér (amelyről fentebb már volt szó), a grafikonkészítő címe, a rendszert felügyelő neve és elektronikus levélcíme:

```
# full path to site on server
$docroot = '/usr/local/apache/htdocs/gantt/';
# URL of site
$wwwroot =
➥'http://webserver.yourdomain.dom/gantt/';
# Name of site administrator
$admin = 'Your Name';
# Email of site administrator
$adminEmail = 'your_email@yourdomain.dom';
```

Egy kis módosításra lesz szükség a ***dbhelp.pl*** fájlban is: meg kell adnunk a ***variables.pl*** elérési útvonalát:

```
# Edit this to point to the location of your
# variables.pl file
require
'/usr/local/apache/htdocs/gantt/variables.pl';
```

Végül az Apache kiszolgáló ***httpd.conf*** állománya kíván meg egy kis szerkesztést. Engedélyeznünk kell a CGI parancsfájlok futtatását a ***gantt*** könyvtárából (ami a dokumentumok könyvtárából nyílik), amihez a következő szakaszhoz hasonló bejegyzésre van szükség:

```
<Directory "/usr/local/apache/htdocs/gantt">
    Options ExecCGI
</Directory>
```

A webkiszolgáló újraindítása után (apachectl graceful) a rendszer működésre kész.
A Gantt grafikonkészítő használatához gépeljünk be egy nevet a bejelentkezési mezőbe – bár az ablak elektronikus levélcímet vár, bármilyen egyedi név megteszi. Ha ez az első alkalom, hogy a programot futtatjuk, egy párbeszédablakot kapunk, amelyben a nevünket és elérhetőségünket kell megadnunk. A ***Submit*** (elküld) gomb megnyomása után megadhatjuk a projektünk adatait és a benne résztvevő tagokat.
Ezután felsoroljuk azokat a feladatokat, amelyek végrehajtása elvezet a vállalkozás sikeres befejezéséhez. Minden sor külön színkóddal rendelkezik. Bármikor lehetőségünk van további feladatok felvételére. Az oldal minden módosítása a feladat kezdő és befejező hetének és a végrehajtásért felelős személynek a meghatározásából áll. Amikor befejezésképpen a Submit gombra kattintunk, a program a grafikonot önműködően létrehozza.
Egy másik, a figyelmünkre érdemes projekt, a MrProjekt (➲ http://mrproject.codefactory.se). A Gnome Office részét képezve ez már a munkaasztaloz kötődő alkalmazás, így lehet, hogy semmit sem kell tennünk, ha a Gnome felhasználói felületként már telepítve van a gépünkön (esetleg akkor sem, ha nincs). A MrProject megtalálható a legutóbbi Mandrake, Red Hat, SuSE és a többi Linux-rendszercsomag telepítőlemezein. Amikor az `mrproject &` paranccsal első alkalommal indítjuk el a MrProjectet, egy üres projektet nyitunk meg. A ***New*** (új) gombra kattintva egyszerre több projektet is megnyithatunk. A bal oldalon találhatjuk meg a ***Resources*** (erőforrások), a ***Gantt Chart*** (Gantt-grafikon) és a ***Tasks*** (feladatok) gombokat. Az egyik nézetből a másikba való átkapcsolás egyszerűen ezeknek a gomboknak a megnyomásával történik.
Egy új folyamat elindításakor a menüsor ***File*** (fájl) menüjének ***Project Properties*** (a projekt jellemzői) menüpontot kell kiválasztanunk. Írjuk be a folyamat nevét, a kezdés dátumát (ezt a MrProject egy barátságos legördülő naptárral segíti), a vezető nevét és a szervezetre vonatkozó adatokat. Kattintsunk a ***Close*** (bezár) gombra, és mentsük a projektet valamilyen tetszőleges értelmes néven.
A következő lépésünk a folyamat során rendelkezésünkre álló erőforrások rögzítése lehet. Ezt az erőforrások megjelenítésével (***Resources***), majd az ***Insert*** (beszúrás) gombra való kattintással érhetjük el. Ezzel egy alapértelmezett erőforrásrekordot veszünk fel a listára, ami egy jobbegérgomb-kattintással szerkeszthető. Az erőforrás lehet valamilyen anyagi természetű dolog vagy időráfordítás. Költséget is itt tudunk megadni.
A feladatok rögzítéséhez az oldalsó gombsorról a ***Tasks*** feliratút kell választanunk, ezután az ***Insert***-et. Az erőforrásokhoz hasonlóan itt is egy általános feladatot kapunk, amit be kell állítanunk. Ezek a feladatok tetszőleges módon leírhatók, majd erőforrásaink egyikéhez rendelhetők. A feladatok beírásának sorrendjével nem kell foglalkoznunk, a végrehajtási sorrend utólag bármikor megváltoztatható oly módon, hogy a feladatot a listából kiválasztva a ***Move up*** (mozgatás felfelé) vagy a ***Move down*** (mozgatás lefelé) menüpontra kattintunk. Az egy feladathoz rendelt idő napokban kerül kifejezésre, de törtnapok beírására is lehetőségünk nyílik. A feladat végrehajtásának százalékos értékét is itt rögzíthetjük.
Bármelyik munkafázisban átválthatunk a Gantt nézetre. Elegáns megoldás, hogy a feladatok idejét a vízszintes sávra való kattintással és húzással módosíthatjuk. (Azt hiszem, több időt kell a „Borkóstolás és minőségellenőrzés” feladatához rendelnem.) Feladatfüggőségek szintén bármikor hozzáadhatók, hiszen könnyen előfordulhat, hogy egy feladat elkezdéséhez egy másiknak előbb be kell fejeződnie.
Ez csak két példa azokból a programcsomagokból, amelyekkel lehetőségünk nyílik a projektek kezelésére, nyomon követésére és ábrázolására. Ha kíváncsiak vagyunk arra, hogy milyen egyéb eszközök elérhetők el a Világhálón erre a célra, ajánlom a „Call Center, Bug Tracking and Project Management Tools for Linux” (Telefonos ügyfélszolgálati, programhiba-kereső és projektkezelő eszközök Linuxra) című oldal meglátogatását (➲ http://www.linas.org/linux/pm.html).
Keressük meg az oldalon a ***Project Management*** (folyamatkezelés), vagy akár a ***Schedulers*** (ütemező programok), ***Planners*** (tervezőeszközök), ***Gantt Chart Tools*** (Gantt-grafikon készítők) című részeket, ha ezekhez hasonló csomagokat szeretnénk további vizsgálatoknak alávetni. Az ajánlatok között egyformán szerepel szabadon felhasználható, illetve kereskedelmi termék, és minden csomaghoz megtalálható annak tömör leírása, valamint a honlapjára mutató hivatkozás.
Most látom csak, hogy a poharaitok majdnem üresek. Kérjük meg François-t, hogy gyorsan orvosolja ezt a nehézséget.
A következő havi viszontlátásig ürítsük poharunkat egymás egészségére. A votre santé! Bon appétit!

A cikkben szereplő programok a 45. CD Magazin/Fogado könyvtárában találhatók meg.

Linux Journal 2003. február, 103. szám

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

KAPCSOLÓDÓ CÍMEK

Call Center, Bug Tracking and Project Management Tools for Linux ➲ http://www.linas.org/linux/pm.html
Graphical UI Gantt Chart Generator ➲ http://associate.com/gantt
MrProject ➲ http://mrproject.codefactory.se
Marcel's Wine Page ➲ http://www.marcelgagne.com/wine.html

Veled vagy nélküled?

A Treo 180g mobiltelefon és zsebtitkár (Siemens *45).

Mostanában a mobiltelefonokat már nem az alapszolgáltatásukért vesszük meg, hanem elsősorban a kinézetük, másodsorban pedig a pluszszolgáltatásaik miatt. Az már fel sem merül bennünk kérdésként, hogy vajon az alapszolgáltatásokat (hívásfogadás és -kezdeményezés, netán SMS-küldés és -fogadás) megbízhatóan (fagyásmentesen) képesek-e számunkra biztosítani. A válasz 98 százalékban „nem tudja" lenne. Ki ne találkozott volna olyan Nokia 6210-essel, ami három hónapos korában fogja magát, és egyszerűen kikapcsolgat. Egy átlagos telefon többet van garanciális szervizben, mint kellene. Én még emlékszem olyan esetre is, amikor Nokia 2110-es telefonomat beszélgetés közben elejtettem a lépcsőházban. A telefon komótosan, minden lépcsőfokot érintve leballagott a második emeletről a földszintre, és amikor felvettem, még a beszélgetőpartnerem is a vonalban volt. Nos, az a telefon szinte sose fagyott le. Igényeink azóta nagyot fejlődtek, manapság egy átlagos tízezer forintos akciós telefonnal nagyobb sávszélességgel lehet internetezni, mint annak idején egy internetes gerincen ülve. Bevallom, engem egy kicsit mindig is zavart, hogy a telefonom, a Palmom és a linuxos gépem között folyamatosan össze kell hangolnom a telefonszámaimat és egyéb adataimat. Így amikor a Handspring cég megjelent az első olyan Palm operációs rendszert futtató gépével, amely 900/1800-as telefon is egyben, nagyot dobbant a szívem. Nekem kell egy ilyen! Be is szereztem egyet azonnal, és bár ne tettem volna!

Treo 180g

A Treo 180g szolgáltatásait tekintve egy 3.5.1-es Palm OS-t futtató tenyérgép, amely 16 MB belső RAM-mal rendelkezik, és kapott egy 900/1800-as Visor phone nevű bővítményt. A régi változatokkal ellentétben itt egybeépítve található a telefon, így egy igen mutatós flippes telefonnál „alig" nagyobb. Palmos és a linuxos kapcsolódási lehetőségének előnyeit kihasználva a már meglevő Palmomról olyan módon tudtam a programokat az új gépre áttelepíteni, hogy egyetlen programot sem kellett újra beállítanom. Ahol a Palmon elengedtem a tollat, a Visoron ott került újra elő. A Treo 180g-hez a gyári csomagolásban rendkívül gyakorlatias módon nem egy dokkoló állomást adnak, hanem egy olyan USB-csatolókábelt, amelybe az úti töltő külön is becsatlakoztatható. Nézzük, mit is kell tennünk, hogy Linux alatt munkára bírjuk a Treót!

A Treo beállítása

A 2.2.2x-es, illetve a 2.4.x-es rendszermagok mindegyike tartalmaz az USB menüpont alatt egy ***USB Serial Converter Support*** menüt. Nos, itt a következőket kell beforgatni:

```
CONFIG_USB_SERIAL=y
CONFIG_USB_SERIAL_GENERIC=y
CONFIG_USB_SERIAL_VISOR=m
```

A legtöbb asztali felhasználásra szánt Linux-terjesztést olyan „gyári" rendszermaggal szállítják, amihez hozzá sem kell nyúlni. Jobb esetben, amikor az eszközt bedugjuk az USB-csatlakozóba, betölti a megfelelő modult és már használható is. A memóriaadapterekkel ellentétben a soros USB-eszközök akkor láthatók Linux-oldalról, amikor az eszköz kapcsolatot kezdeményez. Ha tehát a Visor oldaláról elindítjuk az összehangolást, jobb esetben a */var/log/messages* állományban hasonlót fogunk látni:

```
Jan 22 12:26:29 timeout kernel: usbserial.c:
Handspring Visor converter detected
Jan 22 12:26:29 timeout kernel: visor.c:
Handspring Visor: Number of ports: 2
Jan 22 12:26:29 timeout kernel: visor.c:
Handspring Visor: port 1, is for Generic use
and is bound to
 ttyUSB0
Jan 22 12:26:29 timeout kernel: visor.c:
Handspring Visor: port 2, is for HotSync use
and is bound to
 ttyUSB1
```

Egy előző cikk kapcsán (Linuxvilág 21. szám, 70–71. oldal) rengeteg megkeresést kaptam a Siemens Me 45, illetve a Siemens S45-ös készülékek Linuxszal való összeköthetőségéről. Jó hírem van mindenki számára: már két gyári linuxos program is letölthető hozzá. Az első program `scmxx` névre hallgat, és a legegyszerűbben az `apt-get install scmxx` paranccsal telepíthető (Debian GNU/Linux alatt), ez azonban csak néhány lehetőséget tartalmaz. Ennél sokkal profibb a `gnokii`-hoz hasonló program, az `sl45c`, amely minden 45-ös sorozatú készülékhez használható – ez a ➲ http://www.sl45i.nl/sl45c/ címről tölthető le. Tudja írni és olvasni a telefon memóriáját, küldhetünk vele SMS-t, bekapcsolhatjuk a netmonitort (csak haladóknak!), valamint a telefon 320 KB méretű dinamikus memóriáját mintegy hajlékonylemezként használva állományokat is tárolhatunk benne. További nagyon hasznos szolgáltatás, hogy a gép óráját összehangolhatjuk a telefonéval, tehát a gép óráját érdemes például egy magyar időkiszolgálóhoz, a time.kfki.hu-hoz állítani, majd a telefon óráját egy mozdulattal a géphez; de ugyan ilyen egyszerűséggel tölthetünk fel képeket is a telefonra. A programot egyelőre még csak forrásból lehet telepíteni, viszont nem igényel különösebb hozzáértést.

Hangoljuk össze óráinkat!

Amennyiben nem ezeréves alaplapunk van, mely folyamatosan elfelejti az órát, abban az esetben elegendő naponta egyszer összehangolni az atomórához. Ennek legegyszerűbb módja: rakjuk fel az `rdate` programot, a Debiant futtatóknak `apt-get install rdate`. Majd az `rdate -s time.kfki.hu` paranccsal el tudjuk (rendszergazdai módban) fogadtatni a géppel a pontos időt. Ezek után érdemes a hwclock -uw utasítással visszaírni a BIOS-ba. Ezt követően a gnokii vagy sl45c programmal már el tudjuk küldeni a pontos időt a telefonjainknak is.

Ha mégsem ezt látjuk, célszerű megpróbálkoznunk a `visor.o` modult betöltésével, amit az `insmod visor` paranccsal tudunk megtenni. Ha ez nem sikerülne, jöhet a rendszermagfordítás.
A kapcsolattartás a Linux oldaláról – a Linuxvilágban már többször bemutatott – `pilot-link`, illetve a `jpilot` (grafikus) programok segítségével lehetséges.
A Visor oldalán a ***system panel/preferences/Connection*** alatt célszerű egy új csatlakozást készíteni, ugyanis a gyári beállításokkal az USB sebessége csak 56 K-ra van levéve, és ezt nem is engedi megváltoztatni. Ha azonban felveszünk egy csatlakozást, annak akár 230 400 b/s-t is beállíthatunk. Fontos azonban, hogy a Linux oldalán is ugyanaz a csatlakozási sebesség legyen beállítva, mert eltérő sebességű adatátvitelnél igen érdekesen működhet. Mindezek beállítása után a már meglévő adatainkat, illetve az újakat is egy mozdulattal telepíthetjük.
Nézzük a „telefont”: a beépített kétnormás (900/1800) GSM szabványú telefon első használatra elég fapadosnak mondható. Mivel azonban a legtöbb dobozos Treo alapesetben amúgy sem tudja a GPRS-t, a GPRS-frissítést érdemes letölteni a ➲ http://www.treogprs.com címről. Ez aztán nemcsak GPRS-képessé teszi majd kézi mindenesünket, hanem egy csomó további szolgáltatást is elérhetővé tesz a GSM terén. Ezenkívül nagyon fontos, hogy a 2,5 órás beszélgetési időt a programfrissítés körülbelül 3–3,5 órára tolja ki. Mindenképpen érdemes feltenni a javítást, ugyanis rengeteg SMS-sel kapcsolatos gyerekbetegséget is javít. A frissítésről annyit érdemes tudni, hogy mindenképpen windowsos felületű gép szükséges hozzá, ugyanis egy böhöm nagy ***.EXE*** állományba van minden belepakolva. A frissítés felülírja a ROM egy részét, és a használható rendszer memóriából is elvesz nagyjából 1 MB-ot. Ennek ellenére az alapprogram-felszereltségről elmondható, hogy szegényes. Az ember azt hinné, hogy itt egy remek eszköz, amelynél a GSM program csatolón keresztül érhető el, és a fejlesztők ezer programot írnak rá. De nem. Ugyan a telefonkönyv szolgáltatás több lépésben is elérhető, valamint az AdressBookból és a SIM memóriából is egy mozdulattal tárcsázni tudunk, ennek ellenére például az SMS-küldési lehetőségeink meglehetősen kezdetlegesek. A Palm operációs rendszerre íródott FUNSMS rengeteg hasznos SMS-küldési, illetve -fogadási lehetőséget kínál számunkra, például a Flash SMS küldését. Ez olyan SMS, amely a megérkezés pillanatában „megnyílik” a fogadó félnél, tehát nem kell megnyitni, illetve szinte korlátlanul menthetjük az SMS-eket. A FUNSMS programnak is létezik olyan változata, amelyben a Treo már próbaváltozatban támogatott, de sajnos még nem tökéletes, infratelefonok és Palm-eszközök között viszont gyönyörűen működik.

GPRS

Azért, hogy ne csak rosszat írjak a Treo 180g-ről, el kell mondanom, hogy a frissítés után nagyon kényelmesen lehet róla internetezni vagy WAP-oldalakat nézegetni a beépített böngésző, illetve a letölthető Wapman program segítségével. Nagyon nagy hibája azonban, hogyha például programokat töltünk fel a PC-ről a Treóra, a telefonon nem lehet hívást fogadni, mivel a telefon-csatolófelület nem tud középpontba kerülni, és ilyenkor a nem fogadott hívások listájában sem jelenik meg a hívó, továbbá a készülék foglalt jelzést ad.

Összegzés

A Treo 180g egy Handspring Edge-tudású, 16 MB memóriával rendelkező hasznos segítőtárs, de számolni kell azzal a lehetőséggel, hogyha a Treo meghibásodik, akkor se telefon, se titkár.

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

KAPCSOLÓDÓ CÍMEK

➲ http://wwwhandspring.hu
➲ http://wwwtreogprs.com
➲ http://www.sl45i.nl/sl45c/
Telefonok összekötése Linuxszal (régebbi linuxvilágos cikkek) ➲ http://www.guska.hu/publication/cikkek/