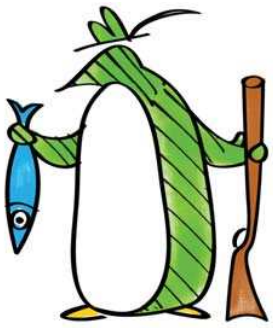
Beköszöntő

Csontos Gyula
(Csontos.Gyula@linuxvilag.hu)

Így a nyár közepén mindenki könnyed nyaralásra, pihenésre vágyik nem pedig feszített tempójú munkára. Aki teheti vízparton vagy hegyek között mulatja ideje nagyrészét, akik viszont a városokban kénytelenek maradni, azoknak is valami kellemes időtöltés után kell nézniük. Ha éppen borongós az idő, vagy nagyon meleg van a CD mellékletünkön lévő Slackware 10.0-ás változat segíthet elütni az időt. Telepítése viszonylag egyszerű, bár nem veheti fel a versenyt az egérkattingatós, magyarnyelvű, grafikus telepítőkkel, azért bátran vágjon bele mindenki.
Akik esetleg a nyár hevében zenekar alapításra adták fejüket, vagy esetleg már régebb óta van egy csapatuk, azok bizonyára hasznosnak találják majd a 14.oldalon kezdődő vezérfonalunkat, az Összefoglaló a Linux hangszerkesztőiről cikkünkben a legkülönfélébb hangfeldolgozó programokat ismerhetjük meg, a Linuxos hangstúdió cikkünk pedig mindazokhoz szól, akik szeretnének egy megbízható ámde alacsony költségvetésű programokkal működő hangstúdiót építeni.
A másik, mostanában igen szomorú téma a SPAM-ek elszaporodása. Ha valaki nem szűri a leveleit, akkor bizony előfordulhat, hogy mire a nyaralásból hazaér, a postaládája szó szerint megtelik olyan levelekkel amiket nem is kért. Ennek a megfékezésére is ajánlunk egy pár mesterfogást.

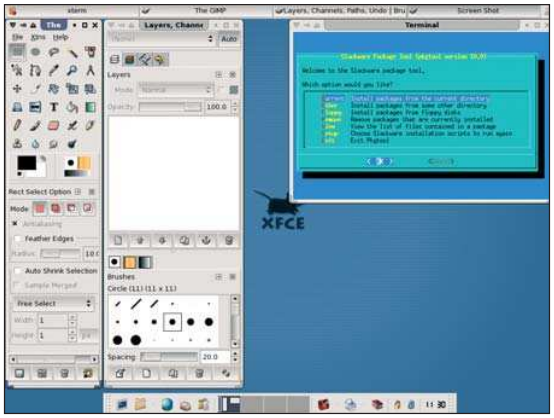
Kellemes időtöltést!



Programvadászat

Slackware 10.0

A Slackware az egyik legrégebbi linux disztribúció, immáron 12 éves múltra tekint vissza, szülőatyja ***Patric Volkerding***. Egy jól használható, stabil, átgondolt rendszerről van szó. Aki kicsit komolyabban foglalkozott már linuxszal, annak bátran merem ajánlani, de igazából mindenki megpróbálhatja telepíteni, nem egy ördöngősség. Lássuk mi is ezzel a helyzet.



Első menet: telepítés

A telepítés elkezdéséhez a szokásos CD-ről történő rendszerindítást kell engedélyezni a BIOS-ban. Ha ezzel megvagyunk helyezzük be a korongot a meghajtóba és várjuk meg amíg megjelenik a `boot:` prompt. Itt különféle indulási paramétereket adhatunk át a rendszermagnak, általában elég, ha entert ütünk. Indulás közben kiválaszthatjuk a használni kívánt billentyűzetkiosztást, majd a `root` szó begépelésével bejutunk a rendszerbe.
A rendszer felhívja a figyelmünket, ha esetleg nem színes monitor előtt ülnénk mindenképpen adjuk ki a `TERM=vt100` parancsot. Indítsuk el a `setup` programot, ez segít bennünket végig a telepítés rögös útján. A telepítő – szerintem egy zseniális program – egyszerűen kezelhető, mindent tud amire szükségünk lehet. Kiválaszthatjuk a kívánt partíciókat, a telepítendő csomagokat stb. A telepítési menet ugyan nem a legegyszerűbb, de magától értetődő, így erre nem térek ki.

Második menet: Csomagkezelés

Linux alatt, mint azt már megszokhattuk, több irányból is megközelíthejük a dolgokat, nincs ez másképpen a Slackware csomagkezelésével sem.
`Installpkg`: A parancs szintakszisa nagyon egyszerű `installpkg csomagnév`
A `-warn` kapcsolóval próbatelepítést végezhetünk, ilyenkor nem települ a csomag csak ellenőrzés történik, hogy valóban minden rendben lesz-e telepítés után. Ez akkor szerencsés, ha nem vagyunk teljesen biztosak a telepítendő csomagban, és abban, hogy az adott csomag telepítése nem okoz gondot a későbbiekben. A programok eltávolítását a `removepkg csomagnév` paranccsal tehetjük meg.
Előfordulhat, hogy frissíteni szeretnénk a rendszeren lévő csomagokat. Ebben segít a `upgradepkg` program. Ha az újabb és régebbi programcsomag neve megegyezik akkor a `upgradepkg újcsomagnév` parancs a nekünkvaló, ha különbözik a nevük akkor pedig használjuk az `upgrade régicsomagnév újcsomagnév` szintakszist.
A SlackWare csomagok kitömörítéséhez az `explodepkg comagnév` parancsot használhatjuk.
`pkgtool`: Aki nem szereti a parancssort, annak lehetősége van ezt az egyszerű menüs programot használni. Ezt a programot is tartalmazza az alaprendszer. Azonnal felbukkan a menü, első feladatunk az lesz, hogy megadjuk a telepítendő csomag forrását, jelenlegi könyvtár, más könyvtár stb..
Ha csomagot akarunk eltávolítani, akkor válasszuk a `remove` opciót. A következő lépésben válasszuk ki a telepíteni/törölni kívánt csomagokat a szóköz billentyűvel. Ezután a program magától elvégzi a többit feladatot.
A nagy kedvenc `slapt-get`:
A Debianosok biztosan felkapták a fejüket erre a megnevezésre, bizony-bizony ez egy APT-hez hasonló csomagkezelő Slackware alá. Ugyanúgy kereshetünk a program archívumokban (például a ➲ http://www.linuxpackages.net csomagjai között) mint tesszük azt a debian rendszereken.
A `slapt-get install csomagnév` paranccsal telepíthejük, a `slapt-get remove csomagnév` paranccsal pedig eltávolíthajuk a csomagokat.

Egy ideális antivírus csomag

A BitDefender AntiVírus csomag megtalálható a *CD virfilter* könyvtárban. Ez a program a szokásos Anti-vírus védelem mellett Internet tartalom szűrést és tűzfalas védelmet is nyújt, vállalati vagy otthoni környezetben. Ezzel is védve a felhasználók adatait.
A népszerű Webmin-hez is mellékelnek egy modul aminek segítségével könnyedén beállíthatjuk a rendszerünk védelmét.

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

Középút: Tenon

A Wiscore cég Tenon névvel újabb, elsősorban otthoni használatra szánt, viszonylag kevés szolgáltatást nyújtó, de kedvező árú linuxos kiszolgálót mutatott be. A webes felületen keresztül kezelhető Tenon levél-, web- és FTP-kiszolgálóként használható, továbbá forgalomirányítóként és tűzfalként is szolgál. Erőssége, hogy operációs rendszere csak olvasható adathordozóról indul, így könnyedén visszaállítható eredeti állapotába. Gyengesége, hogy helyi fájl- vagy nyomtatókiszolgálóként nem használható, már csak azért sem, mert merevlemezes meghajtóval nem rendelkezik. A leveleket, weblapokat CompactFlash kártyán tárolja, az alapkiépítés szerinti kártya 128 MB-os, de nagyobb kártyát is vásárolhatunk bele. A készülékhez ingyenes frissítési szolgáltatás jár, tehát a belső programhoz készülő biztonsági javításokat és egyéb frissítéseket mindenki szabadon beszerezheti hozzá.

➲ www.wiscore.com/en_US

Intel BIOS-utód szabadon

Az Intel bejelentette, hogy következő generációs BIOS programjának – pontosabban annak a belső programnak, mely a BIOS utóda, egyben az *Extensible Firmware Interface* egyfajta megvalósítása lesz – alapja CPL (*Common Public License*) szerződési feltételekkel lesz hozzáférhető. A BIOS több mint 20 éve biztosítja a személyi számítógépek működéséhez szükséges alapeljárásokat, a kor azonban túlhaladt rajta, sokkal több és újszerűbb szolgáltatást várnánk el tőle. Az újfajta belső program – ilyet egyébként a Phoenix is fejleszt már egy ideje – felügyelhetőség és szervizelhetőség terén egyaránt olyan lehetőségeket kínál majd, amelyek megvalósítása a jelenlegi BIOS-okkal gyakorlatilag lehetetlen volna.

Az Intel egy illesztőprogram fejlesztői készletet is kínál majd a belső programhoz, ezzel is a különféle gyártók termékei közötti együttműködést kívánja javítani. A kód a CollabNet közreműködésével lesz elérhető.

➲ www.intel.com/technology/framework

Játékkonzol-mindenes

A Sony először mutatta be PSP játékkonzoljának mintapéldányát. A konzolt ugyan már korábban bejelentették, látni azonban még senki nem látta, és a nagyközönség számára várhatóan csak év végén vagy jövőre lesz elérhető. A PSP-t, mely a Sony szerint a 21. század mobil eszköze, elsősorban a 18-34 éves korosztálynak szánják, de később a fiatalabb generációt is meg szeretnék hódítani vele. A készülék 170 x 74 x 23 mm-es, súlya 260 gramm. Belsejében egy legfeljebb 333 MHz órajelű PSP processzor és 32 MB memória rejtőzik. 16:9 képarányú kijelzője 4,3”-os, 480x272 képpont és 16,7 millió szín megjelenítésére képes, fényereje 200 dc/m^2. A PSP rendelkezik egy UMD olvasóval is, ebbe egyoldalas, kétrétegű, 1,8 GB kapacitású optikai lemezek helyezhetők; továbbá kapott egy 802.11b típusú vezeték nélküli hálózati csatolót, USB csatlakozót, infravörös kaput és Memory Stick PRO Duo aljzatot. Kiegészítő jelleggel számos eszköz megvásárolható lesz hozzá, többek közt kamera és GPS vevő is, amiből egyértelműen arra lehet következtetni, hogy sokkal több lesz egy korszerű videojátéknál.

➲ www.playstation.com

Qtopia telefonra

A Trolltech elkészült tavaly októberben bejelentett mobiltelefonos alkalmazáskészletével, a palmtopokon megismert szolgáltatásokat nyújtó *Qtopia Phone Editionnel*. Az alkalmazáskészlet gyakorlatilag minden, a Linux futtatására alkalmas processzort támogat, a billentyűzettel és az érintőképernyővel ellátott készülékeken egyaránt használható. Működéséhez legalább 8 MB Flash ROM és 16 MB RAM, valamint 16 szürkeárnyalat és 176x208 képpont megjelenítésére képes kijelző szükséges. A Qtopia Phone Edition öt összetevőből áll össze, ezek a telefonos kezelőfelület, a személyi adatkezelő alkalmazások, a telefonos alaprendszer, a szinkronizáló megoldás és a fejlesztői környezet.

Piackutatók szerint néhány éven belül a telefoneladások egynegyedét az okostelefonok teszik majd ki, és ezek között a Linux olyan szerepet játszhat, mint az asztali számítógépek körében annak idején a DOS.

➲ www.trolltech.com

Elektronikus is, meg nem is

Jie Chen, egy 18 éves fiatalember nyerte az Amerikában megrendezett második nemzetközi e-zongoraversenyt. A verseny különlegessége, hogy a darabokat Yamaha Disklavier zongorákon játsszák.

A Yamaha ezen speciális zongorái a hagyományos és a digitális megoldásokat ötvözik. Egyszerre szolgálnak normál zongoraként, digitális felvételre képes MIDI billentyűzetként és számítógépről vezérelhető gépzongoraként. A Disklavier zongorákon lejátszott darabokat rögzítés után tökéletesen – a zongora húrjainak és kalapácsainak segítségével, nem szintetizátoros megoldással – rekonstruálni lehet, akár egy interneten keresztül elküldött fájl alapján is. Sőt, a versenyre az előzetes meghallgatások után meghívott művészeket is így választották ki. Mindegyik induló a területi próbákon mutatta be tudását, amelyet mozgókép és adatfájl formájában egyaránt rögzítettek. A zsűri egy ugyanilyen zongorán játszotta vissza a darabokat, és az így hallottak-látottak alapján jelölte ki a végső versenye való részvételre érdemeseket. A meghallgatások és a verseny anyagait a rendezvény holnapjáról bárki letöltheti.

➲ www.piano-e-competition.com

Rendrakás

Linus Torvalds és a Linux rendszermagfejlesztő csoport egy hozzájárulás-követő rendszer üzembe állítása mellett döntött. A továbbiakban a fejlesztőknek digitálisan alá kell majd írniuk foltjaikat, ezzel igazolva azt a jogukat, hogy az adott kódrészlettel hozzájáruljanak a rendszermag fejlődéséhez. Minden fejlesztőnek nyilatkoznia kell arról, hogy a kódot vagy maga készítette, vagy jogszerűen vette át máshonnan, vagy harmadik féltől az előbbi feltételek valamelyike mellett szerezte be, és változatlanul adja tovább. A rendszer révén nemcsak a foltok, és egyúttal a rendszermag jogtisztaságát lehet majd garantálni, de a fejlesztők kilétét és a hozzájárulásokat is jobban követni lehet.

Jövőre a füstjelek jönnek

A Nokia 3220 jelzésű mobiltelefon – pontosabban, követve a Nokia szóhasználatát, kameratelefon – újabb kommunikációs módszerrel ismertet meg bennünket.

A 3220 nagy újdonsága az a fedlap, amely képes feliratokat a levegőbe vetíteni, miközben a telefonnal hadonászunk. A fényüzenetnek keresztelt szolgáltatást például zsúfolt helyiségekben lehet majd jól használni, ahol mit sem ér az üvöltözés, a mutogatás pedig félreérthető – elég lesz tehát beírni a telefonba, majd megfelelő mozdulattal lejátszani az üzenetünket. A telefon érzékeli a gyorsulást, ennek megfelelően jeleníti meg a szükséges karaktereket, amelyek összeolvasva a kívánt szöveget alkotják.

A telefon további érdekessége, hogy kézben tartható botkormányként is használható, így a lövöldözős, mászkálós játékokat – a készülékhez alapesetben két ilyen jár – teljesen újfajta módon futtathatjuk majd rajta. Kézenfekvő módon, az előlap fényeinek segítségével a 3220 képes arra is, hogy a hívásokat és üzeneteket fényekkel jelezze, akár a csengőhangok ritmusára villogva. A VGA felbontású kamera, a változatos testreszabási lehetőségek és a háromsávos működés már csak megkoronázza mindezt.

A Nokia 3220 hamarosan kapható lesz, ára 250 euró körül várható.

➲ www.nokia.com

Nokia – Mozilla együttműködés

A Nokia és a Mozilla Foundation együttműködésével egy új böngészőprogram fejlesztése indult meg. A Nokia támogatásával Minimo név alatt, amint az sejthető, mobiltelefonokra fog készülni egy nyílt forrású böngésző. Az évek óta futó Mozilla böngészőkre az utóbbi időben elég sok panasz volt, a Nokia megjelenésének és persze pénzének köszönhetően remélhetőleg kicsit felpezsdül az élet a tervezet környékén. A tényleges üzleti megállapodásról részleteket nem lehet tudni, a felek csak annyit árultak el, hogy a Nokia aktív szerepet vállal a nyílt forrású közösségben. Az alapítványnál remélik, hogy a közeljövőben sikerül további megállapodásokat is összehozni, amelyek alapján programjukat különféle területek, alkalmazások igényeihez igazítják, majd a létrejövő megoldásokat nyílt forrással teszik elérhetővé – egyébként 1998-ban a Mozilla.org pontosan azért jött létre, hogy félig-meddig kereskedelmi jellegű, mégis nyílt forrású fejlesztési modellt kövessen. Talán végre sikerrel.

3D asztali monitor

A Sharp LL-151D jelöléssel egy új, 15”-os, színes, különleges szemüveg használata nélkül is 3D megjelenítésre képes LCD-monitort mutatott be. A kijelző 2D módban is használható,

ekkor felbontása 1024x768 képpont, 3D módban viszont vízszintes felbontása megfeleződik. A kijelző a két szem számára eltérő képet állít elő, a különleges szemüveg elhagyásával használata is kényelmes; egyetlen hátránya, hogy a 3D kép csak a monitorral pontosan szembe nézve élvezhető. A Sharpnak már eddig is voltak hasonló kijelzői, ám azok csak hordozható gépekben jelentek meg, asztali monitorként nem. Az alkalmazási területek köre rendkívül széles, tudományos, mérnöki feladatokra és persze játékra egyaránt jól lehet használni az új monitort, amelyből a cég hamarosan nagyobb változatokat is meg kíván jelentetni.

➲ www.sharp.com

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

Mi újság a rendszermag fejlesztése körül?

A BIOS a számítógép egyik legbarátságtalanabb része. A forráskód általában titkos, frissítések pedig szinte nem is léteznek. A modern operációs rendszerek megpróbálják amint lehet maguk mögött tudni a BIOS-t, és saját hasonló feladatú kezelőfelületüket használják.
Néha a BIOS hibák miatt a OS fejlesztők kénytelenek lépéseket tenni annak kijavítására. Az M6805 eMachines esetében is éppen mostanában történt ilyesmi. A BIOS nem volt képes felismerni a CPU frissítést és az eredeti CPU által használt CPU sebességet és voltértékeket jelentette. *Tony Lindgren* készített egy Linux rendszermag foltot, amely bizonyos érvényességi teszteket végez ezeken a gépeken, és ha szükséges kijavítja a BIOS által visszaadott értékeket.

Benjamin Herrenschmidt az Open Firmware *sysfs* csatolófelületén dolgozik, hogy a PPC és PPC64 rendszereknek teljes Open Firmware támogatást biztosítson. Jól példázza a Linuxos fejlesztési stílust, hogy amint *Benjamin* bemutatta munkáját, több tervezési kérdést is alapos vizsgálat alá vettek. Például, az Open Firmware adatot ő a PCI eszközök adatainak részeként szerette volna kiadni, ami könnyen zavart okozhatna. Most akkor az összes sín és valamennyi különféle firmware kezdje el egymás adatait tárolni? És vajon a rendszer melyik más része szeretné majd máshol is ábrázolni az adatokat? *Linus Torvalds* azt javasolta, hogy a különféle szempontokat külön vizsgáljuk, ezért az Open Firmware adatokat – még ha PCI eszközökkel foglalkozik is -, az Open Firmware alkönyvtárba kell helyezni.

Dipankar Sarma készített egy API függvényt melynek segítségével bizonyos ReiserFS (és egyéb) kódot le lehet majd tisztázni. Az `rcu_barrier()` nevű függvény megvárja míg az összes RCU visszahívás befejeződik. Az RCU azaz *Read Copy Update* (Olvasás Másolás Frissítés) annak a zárolási mechanizmusnak a neve, amely segítségével az OS alacsony költséggel képes elérni a több processzor között megosztott adatterületeket. Korábban a ReiserFS-nek saját logikát kellett alkalmazni, e képesség színlelésére. *Nikita Danilov* a ReiserFS csapatból már régóta vágyott egy ilyen API kódra. Amint ezt elfogadják a fő fában, a Reiser gárdája végre kihajíthat egy halom ronda kódot.

A Linux 2.0.40 (moha-lepte teknőc – *The Moss-Covered Tortoise* – kódnevű) változatának megjelentetésével, *David Weinehall* megmutatta, hogy szándékában áll folytatni az ősi 2.0-ás sorozat karbantartását. A 2.0.40-ben többek közt kijavított számos biztonsági lyukat és fájlrendszer károsodási problémát. Bejelentésében David leszögezte, hogy semmilyen új képesség nem kerül a 2.0-ás sorozatba. Mint mondotta, A jelenlegi rendszermagok képességeit kereső felhasználóknak, a 2.4 vagy 2.6-os fára kell frissíteniük. Néhány rendszernek problémákat okozhat az átállás, ilyenek például a mindig 2.0-ás rendszermagot futtató aktív kiszolgálók, amelyeket bármilyen későbbi rendszermaghoz teljesen újra kéne tervezni. Ennek ellenére úgy tűnik nem igazán alkalmas a jelenleginél simább frissítési módszert kialakítani, tekintve, hogy a páros számú rendszermagoknak a stabilitásra kell törekedniük.

Max Asbock mostanában készítette el az IBM xSeries RSA kiszolgálóprocesszorának támogatását. Az *ibmasm* nevű meghajtó felületet biztosít a felhasználói térből érkező parancsoknak, várakozik az eseményekre és kezeli a távoli videó képességeket. Az egyetlen gubanc a jelenlegi megoldással, hogy Max csatolófelülete teljesen egyedi a Linux világában. Bár a meghajtó rendszerelléréssel foglalkozik, a `sysfs` „egy fájl, egy érték” módszerének alkalmazására Max nem lát semmilyen lehetőséget. A meghajtó a karakteres eszköz kategóriába sem préselhető be egykönnyen, ezért a rendszermag forrás-fa */drivers/misc* könyvtárába került. Figyelembe véve ezeket az eltéréseket, a meghajtó valószínűleg néhány változtatáson megy majd keresztül, mielőtt bekerülhetne a helyes rendszermagváltozatba. De még ha gyorsan el is fogadják, beletelhet egy kis időbe, mire a csatolófelület megállapodik, ahogy a kedvelt Linuxos helyek magukévá teszik.

Gerd Knorr írt egy meghajtót, amely ugyan felhasználói szemmel nézve önmagában semmilyen látványos dolgot nem mutat, viszont segít a többi meghajtónak az infravörös távirányító eszközök kezelésében. Az olyan meghajtók, mint a saa7134 és a bttv mostantól tiszta kóddal a szabványos Linux bemeneti réteget használhatják az eléréséhez. Érdekes csavar a dologban, hogy az alap Linux modulkezelő kód megváltozott a 2.5-ös időkben, Gerd mégis tartotta magát a régi csatolófelülethez, holott ez a meghajtó a 2.6-hoz készült. Ennek az az oka, hogy szeretné, ha kódja 2.4 és 2.6 alatt is lefordulna, a 2.4 pedig nem támogat néhány 2.6-ban megjelent funkciót. Rusty Russell, a 2.5 alatt bekövetkezett modul újratervezésért leginkább felelős fejlesztő, igen aktívnak tűnik az ilyesfajta problémák kezelésében és kijelentette, hogy együttműködési funkciókat fog készíteni a 2.4-hez amelyek lehetővé teszik, hogy a 2.6 szerkezetek mindkét rendszermag változaton helyesen lefordulhassanak.

Zack Brown

Linux Journal 2004. június, 122. szám

„A proxy marad proxy”

Beszélgetés Daczi Lászlóval, az MLDP születésnapjának alkalmából.

– Sok Linux-közeli csoport van, milyen céllal alakult a Magyar Linux Dokumentációs Projekt (MLDP) és kik alakították?

– 2001 végén 2002 elején angol nyelvtudásom fejlesztése céljából lefordítottam a Linux + XFS HOGYANt, ezt szerettem volna közzétenni egy már létező, linuxos doksikat tartalmazó webhelyen, de nem kaptam visszajelzést. Egy ideig próbálkoztam, aztán meguntam. Mivel ekkor már a fordítás során szerzett tapasztalataimat is szerettem volna megosztani másokkal, egy önálló projekt indítása tűnt célravezetőnek. Egyedül nehéz életben tartani egy ilyen vállalkozást, ezért társakat kerestem. Az FSF.hu Alapítványt megelőző baráti társaság ekkoriban vált ismertté, és mivel úgy láttam, hogy nekik nem csak a szájuk jár, ezért hozzájuk csatlakoztam. A Magyar LDP így az FSF.hu Alapítvány keretében, egy önálló projektként valósult meg. Célja a The Linux Documentation Project által készített és karbantartott dokumentumok magyarítása. E mellett a fordítások karbantartását is lényegesnek tartjuk, ezáltal a befektetett munka nem vész kárba.

– Az elmúlt két év alatt milyen nagy mérföldkövek voltak?

– Az első a projekt indulása, ez tulajdonképp a közösség tesztelése volt (2002 ápr. – 2002. júl.). Már a kezdetekben is rátaláltunk (vagy inkább fordítva) egy-két aktív emberre, akik segítettek az alapok lerakásában (fordítás, sgml konvertálás stb.). A második, az ismertség megteremtése, a fejlődés és túlélés szempontjából is fontos volt (2002. aug. – 2003. júl.). Elkészült egy szógyűjtemény, Fordítás-HOGYAN, működött a levelezőlista, és a hírportálokon is rendszeresen megjelentünk. Az első évfordulót követően (2003. aug. – 2004. jún.) elkészült egy Debian csomag, amely jelentősen megkönnyíti az sgml/xml forrás konvertálását, bővült az önkéntesek tábora, van két-három visszatérő ember. Megkezdődött a kapcsolatfelvétel a Szegedi Tudományegyetemmel, ahol a nyílt forráskódú szoftverfejlesztés a tantervben is szerepel.

– A csoport aktív tagjai milyen ellenszolgáltatásért, vagy remélt céllal vesznek részt a közös munkában?

– Pénzbeli ellenszolgáltatás nincs, a legtöbben ismereteik bővítése, vagy vizsgajegy megszerzése céljából segítenek. Mindezek mellett évente 3 önkéntes munkáját jutalmazzuk, a lehetőségeink szerint. Idén a Kiskapu Kft. jóvoltából – a szokásos oklevélen és pólón túl – Linuxvilág összes, könyvutalvány és plüsspingvin is szerepel a csomagban, melyeket Horváth Albert, Völgyi Péter és Szalai Ferenc kapnak.

– Milyen rövid- és hosszútávú terveitek vannak?

– Rövid távon egy fordítást segítő (lin/win) kliens készítése a cél, amelyen jelenleg diplomamunka keretében dolgozom. Ez remélhetőleg egy nemzetközi nyílt forráskódú projektté növi ki magát. Hosszú távon az oktatási intézményekkel való együttműködés a cél, így a rendelkezésre álló (emberi) erőforrások megsokszorozódhatnak.

– Kiknek a segítségét várjátok?

– A fordítandó dokumentumok kiválasztásához informatikai szakemberek segítségére van szükség, mivel nem területen tudom felmérni az információ aktualitását. A fordításhoz angolul tudó, és a szaknyelvet ismerő emberekre van szükségünk. A honlap csinosításában webdizájnerek segíthetnek, jó minőségű embléma elkészítéséhez pedig számítógépes ismeretekkel rendelkező grafikusok jelentkezését várjuk.

– A minap nálunk járt Richard Stallman, aki egy kérdésre azt nyilatkozta, hogy szerinte a programok honosítása illetve általában a fordítás elveszi az erőforrásokat a fejlesztéstől. Neked, mint a fordítási munkák koordinátorának mi erről a véleményed?

– Természetesen tiszteletben tartom a véleményét, szerintem viszont aki fordítani akar az úgyis fordítani fog, aki meg programozni akar, az programozni fog. Én időnként fordítok, amikor pedig megcsömörlök tőle akkor programozok. Mindenki úgy éli ki a tehetségét, ahogy tudja. Az MLDP nem egy „kényszerprojekt”, az önkéntesek a nekik megfelelő (vagy annak látszó) feladatot vállalják el.

– Minek az alapján választjátok ki, hogy mi legyen lefordítva, és mi ne?

– Alapvetően én választom ki a fordításra ajánlott dokumentumokat. Ez azt jelenti, hogy bárki választhat a TLDP dokumentumok közül egy neki tetszőt, legfeljebb megpróbálom lebeszélni róla. A saját kiválasztási szempontjaim a következők:

- lehetőleg használható információk legyenek benne, ne legyen elavult
- lehetőleg széles körben felhasználható legyen, ne túl speciális témáról szóljon (például egy adott márkájú és típusú alaplap beüzemeléséről; ez létező példa)

- mindig legyen a kínálatban kezdőknek való rövidebb dokumentum

Mivel nem mindig tudom megítélni egy dokumentum használhatóságát, ezért (kihasználva a projekt nyíltságát) ebben is az önkéntesekre támaszkodok. A Fordítás HOGYANban erről külön fejezet van.

– *A számítástechnikában különös jelentősége van a magyar szaknyelv fejlesztésének. Ti mennyire tartjátok ezt fontosnak, illetve hogyan gondoskodtok a különböző fordítóktól származó anyagok nyelvi egységesítéséről?*
– Fontosnak tartjuk. Lefordítjuk azt aminek van értelmes, érthető és (viszonylag) pontos magyar megfelelője. Például a kernel az rendszermag, viszont a proxy az marad proxy, mivel jelenleg nincs rá megfelelő magyar szó vagy kifejezés. A nyelvi egységet a szógyűjtemény (lásd Fordítás HOGYAN) illetve a lektorok biztosítják.

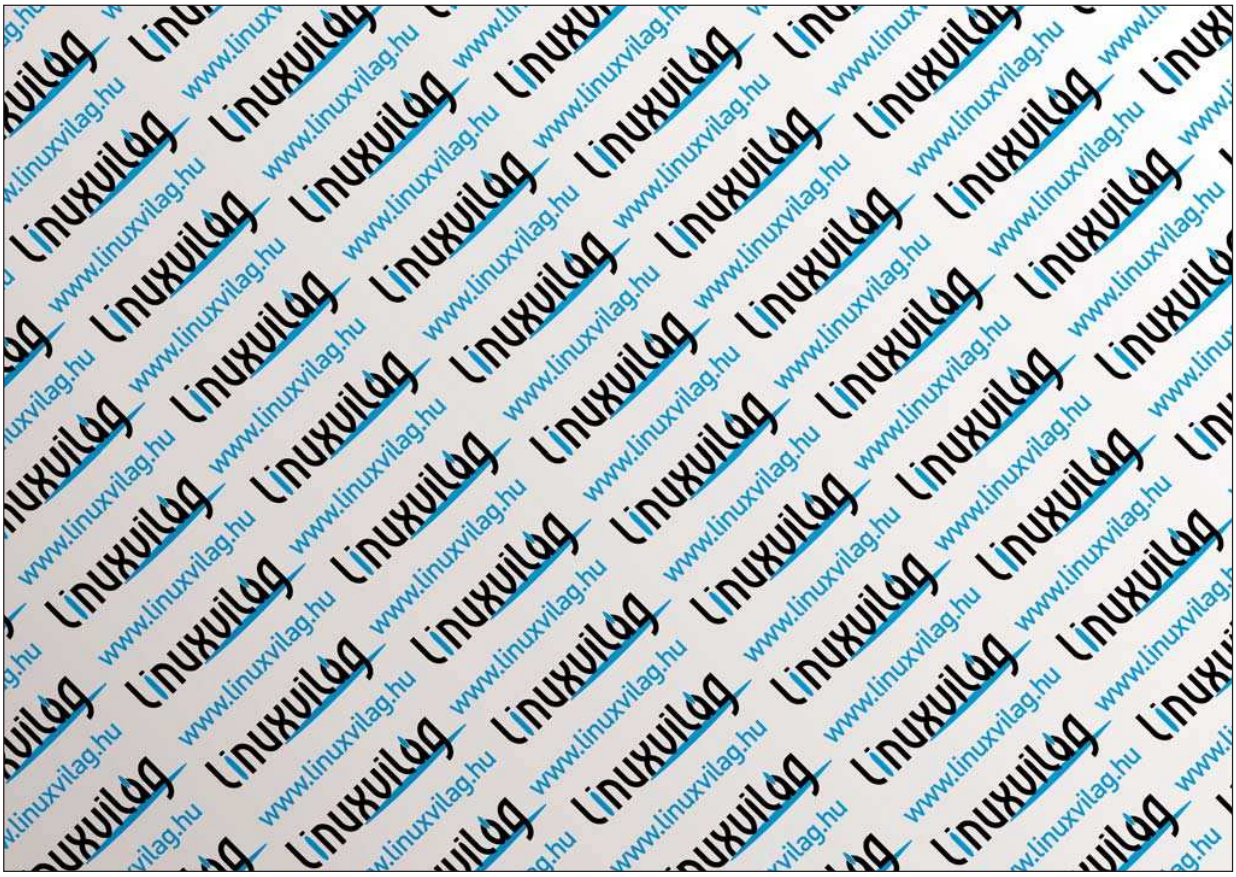
– *Kiválthatják-e az általatok magyarra fordított és ingyenesen elérhető dokumentumok a magyar nyelvű szakkönyveket?*
– Véleményem szerint nem, viszont kellő alapot nyújthatnak azok megértéséhez, illetve kiegészíthetik azokat. Egyes HOGYANokban vannak is hivatkozások nyomtatott dokumentációra.

– *Mit gondolsz, mikor éri el az MLDP a „repülési magasságot”, vagyis mikor következik be az, hogy aki fordítani akar, annak nagyítóval kell keresgélnie?*
– Ez nagyon elméleti kérdés, nem tudom. Elméletileg soha, ugyanis a dokumentációkat frissíteni is kell, márpedig minél több van, annál gyakrabban és többet kell dolgozni rajta.

– *Mit gondolsz, a jó minőségű magyar dokumentáció megléte mennyiben fogja előmozdítani a Linux elterjedését, gondolok itt az állami szférára, illetve az oktatásra?*
– Szerintem ez kulcskérdés. Bár a számítástechnikában alapvető követelmény volt az angol nyelvtudás, mára ez megváltozott. Sajnos Magyarországon eléggé csehül állunk a nyelvtudással, és mivel a számítástechnikával valamilyen szinten foglalkozók száma megnőtt, ezért az angolul értők aránya is megváltozott a „szakmában”. Egyébként is, mindenki az anyanyelvén szóló információt képes a legjobban magába szívni.

– *Bár egyesek szerint Közép-Európában a mi hálózati infrastruktúránk a legfejlettebb, egyes környező országokban a Linux nagyobb elterjedtségnek örvend, mint nálunk. Mennyire függhet ez össze a nemzeti nyelvre fordított dokumentumokkal?*
– Ezt igazából nem tudom megítélni, mivel nem ismerem a környező országok helyzetét a nemzeti nyelvre fordított dokumentumok tekintetében. Valószínűleg van valamilyen összefüggés, de biztos nem csak ettől függ. Nagyobb és maradandó előnyös változáshoz szerintem állami szerepvállalásra van szükség. Mivel azonban a magyar politikai élet olyan, amilyen, ezért jelenleg erre nem látok reményt.

Büki András – Szy György



A Linux és a protonok

Hogyan zajlik az élet a világ legnagyobb részecskefizikai kutatóintézetében?

A napokban megadatott számomra, hogy egy busznyi ember társaságában ellátogassak a Genf-i székhelyű Európai Részecskefizikai Kutatóintézetbe, CERN-be. Utazásunk célja az intézetben működő számítógépközpont meglátogatása volt, ám ehhez azonban elengedhetetlen az ott végzett kutatások alapvető ismerete, ennek következtében teljes körű ismeretanyag birtokába jutottam, amelyet ezúton szeretnék megosztani a kedves olvasókkal.

1. kép CERN legújabb épülete, mellette az irodák, géptermek villamos betápja

2. kép Életkép a központi gépterem felső szintjén

A túra leghosszabb, de nem felétlenül a legérdekesebb része természetesen az utazás volt. Genf Budapesttől 1400 km-re található, nem nehéz kiszámolni, hogy pihenőkkel együtt eltart vagy tizennyolc óráig, mire az ember autóbusszal odaér. Esetünkben sem volt ez másként, de komolyan mondom, megérte a hosszú aszalódás. Lássuk, hogy miért!

CERN

Az idén 50 éves a kutatóintézet, melynek jelenleg 20 ország – köztük Magyarország is tagja. A második világháború után, 1954-ben alakult meg a 12 alapító tagország közreműködésével. Jelenleg a francia-svájci határon terül el, egy körülbelül Margit-sziget nagyságú területen. Fő kutatási területe a részecskefizika, azaz az atommag felépítése, de más területeken is értek el jelentős eredményeket. Ilyen jelentős eredmény például a web kitalálása. Az intézetben találták ki és implementálták a világ első webkiszolgálóját, amely eredetileg a kutatási eredmények és egyéb dokumentumok automatikus megosztását hivatott megvalósítani a különböző munkákon dolgozó kutatócsoportok, egyetemek között világszerte. A dolog olyan jól sikerült, hogy mára az internet legismertebb és legelterjedtebb szolgáltatásává nőtte ki magát. Számos egyéb informatikai kutatás is folyik itt. Ennek az az oka, hogy a nagyszabású fizikai kísérletek eddig megoldhatatlannak hitt problémák elé állítják az ottani informatikai részleget, amelynek feladata, hogy kiszolgálja az intézetet, hátteret biztosítson a kísérletek számára.
Minden csoport ingyenesen ellátogathat az intézetbe, s kérhet a csoport számára kalauzolást angol,francia, német és olasz nyelveken. Mi abban a szerencsés helyzetben voltunk, hogy kiképeztek néhány ott dolgozó magyar munkatársat, hogy alkalmas vezetőnk legyen a túra során, így mi anyanyelvünkön élvezhettük az intézet bemutatását.

CERN és a fizika

Mint tudjuk, a részecskefizika az anyag legparányibb építőköveit vizsgálja módszeres alapossággal. Ezt a vizsgálatot részecskegyorsítók segítségével végzik. Az alapelv, hogy a részecskéket fénysebesség közeli sebességre gyorsítják fel,ezáltal hatalmas energia gyűlik össze a részecskékben. Ez nem a hagyományos értelemben vett mozgási energia, ugyanis a fénysebesség közelében a részecskék tömege az

Einsteini relativitáselmélet által felismert összefüggés következményeként jelentősen megnő, s mint tudjuk, a tömeg energiát hordoz.
A nagy energiával rendelkező részecskéket ezután ütköztetik más részecskékkel, anyagokkal. A hatalmas ütközés hatására felszabadul az anyagban rejlő energia, és a jelenlévő energiából újra anyagok születnek. Az ütközés segítségével tehát az ősrobbanáshoz igen közeli időpontot szimulálnak a kísérlet során.

Az részecskék ütköztetése úgynevezett detektorokban történik. Ezek vizsgálják a bekövetkező eseményt, a keletkező anyagokat.
A kísérlet eredményét ezután tovább elemzik, s következtetéseket próbálnak meg levonni az elemzés során nyert adatokból.
A kísérletek elvégzéséhez több gyorsítót is használnak párhuzamosan, amelyek leginkább méretükben különböznek egymástól.
A legnagyobb jelenlegi gyorsító egy 27 km kerületű kör alakú szerkezet, s a föld felszíne alatt 100 méterrel helyezkedik el egy alagútban. Ezen a körpályán keringetik, gyorsítják a részecskéket, melyek azután összeütköznek egymással. A részecskék gyorsítását, pályán tartását hatalmas erejű elektromágnesek végzik. Sajnos még az ilyen monumentális méretekkel rendelkező berendezések sem tudnak választ adni a kutatók minden kérdésére, ezért folyamatban van egy új gyorsító építése a régi, 27 km-es alagút nyomvonalán,melynek neve LHC (Large Hadron Collider – Nagy Hadron Ütköztető). Ez a szerkezet még erősebb, még gyorsabb, még nagyobb lesz elődjénél. Ottlétünk során benézhettük abba a csarnokba, ahol a gyorsító 300 tonnás detektorát építik, szerelik.

CERN és az informatika

Gondolom sokakban felmerült az előző bekezdés olvasása során, hogy miként is dolgozzák fel és elemzik az ütközések során nyert adatokat. A válasz természetesen az, hogy számítógéppel, azaz inkább számítógépekkel. Az elején már írtam, hogy az informatikai részleg biztosítja a kísérletek számítástechnikai hátterét, de arról fogalmam sem volt, hogy milyen méretű háttérre is van szükség. Az ütközéseket vizsgáló detektorok jelenleg is többszáz Mbit/sec sávszélességgel ontják magukból az adatokat, s ez a most épűlő gyorsító esetében már a Gbit/sec tartományba esik majd. Ennyi adatnak nem csak az szállítása, de a tárolása is igen nagy kihívás. A jelenlegi rendszer is többszáz gépből álló fürtökből épül fel, melynek eredményeképp körülbelül 310 terabájt aktív merevlemezes tárolókapacitással rendelkezik az intézet. Ide kerülnek az adatok „átmenetileg”, ám ez kevés a végleges tároláshoz, ezért az adatokat folyamatosan szalagra mentik. Ezek az érdekes szalagos tárolók valójában robotok, melyek lehetővé teszik, hogy ne kézzel kelljen a kazettákat bepakolni a leolvasóba, ha épp szükségünk van valamire.
Ezen túl természetesen szükségük van webkiszolgálókra, levelezőkiszolgálókra, terminálkiszolgálókra, útválasztókra, hálózati kapcsolókra, és minden lehetséges dologra, amelyek a meglehetősen nagyra nőtt belső hálózatot üzemeltetik, kiszolgálják. Csak ezeket a feladatokat is processzorok százai végzik. Ezen túl természetesen szükség van „némi” számítási kapacitásra a kutatási eredmények kiértékelésének céljából. A fenti rendszer természetesen össze van kapcsolva a munkatársak által használat többezer munkaállomással, amelyek így együtt alkotják CERN informatikai hálózatát.
A központi gépterem két, egymás alatt elhelyezkedő hatalmas szoba, amely összesen mintegy 2.5MWatt bemenő teljesítményt fogyaszt. Ebből sejteni lehet a gépek számát, méretét, s nem utolsó sorban a hangját. Ennyi ventillátor ugyanis már akkora hangzavart csinál, hogy egymás szavát sem lehet érteni.
Igen érdekes, hogy a mindennapi feladatok ellátására nem IBM, HP, vagy Sun nagygépeket használnak, hanem kétprocesszoros gépek százainak fürtjét. Ez ugyanis sokkal-sokkal olcsóbb, mint az azonos teljesítményű „szuperszámítógép”. Ennek ellenére bizonyos feladatokra, tesztelésre használnak klasszikus nagygépeket, de nem ez a jellemző. A munka oroszlánrészét PC-k végzik, melyeken nem is olyan meglepő módon Linux fut.

CERN és a Linux

Nem csak a kiszolgálók és a fürtök nagy részén fut szabad szoftver, de a többezer munkaállomáson is Linuxot futtatnak. Jelenleg a RedHat terjesztés 7.3.1 változatát használják – kicsit módosítva. Hozzávettek egy-két CERN-es változtatást (pl. AFS használata), és újabb rendszermagot használnak a jobb hardvertámogatás érdekében. Ezzel természetesen egy új terjesztés született, amelyet csak CEL-nek, vagyis CERN Linuxnak neveznek. Az az alapvető probléma a szabad szoftverrel egy ilyen nagy kutatóintézet esetében, hogy nincs hivatalos terméktámogatás, és az „aktív” használat következtében bizony számos hibára fény derül, ám nem biztos, hogy ezek kijavításását meg tudják várni. Ezért is használnak kiforrott, stabil változatot, mert a hibákat már nagyrészt kijavították. A hosszútávú célkitűzéseik között egyébként az szerepel, hogy a Linux használatával próbálják minél egységesebbé tenni a szoftverkörnyezetet. A következő CERN-es használatra hivatalosan ajánlott terjesztés Red Hat Enterprise Linux 3-on alapul. Vagyis annak egy újrafordított változatán – mivel a bináris kereskedelmi program, tehát nem érhető el ingyenesen, ugyanakkor a forrás nyilvános.
A belső terméktámogatásra egészen komoly erőforrásokat fordítanak, leírásokat, telepítési útmutatókat, javításokat tesznek elérhetővé, és külön személyzet van a munkatársak segítésére, hogy azok mindenképpen elboldoguljanak a gépükön futó Linuxokkal.
A CERN-nél egyértelműen látszik, hogy a szabad szoftver nem csak olcsó, de igen hatékony is. Nincs is erre jobb bizonyíték, mint az, hogy egy ilyen komoly munkát végző kutatóintézet gyakorlatilag a Linux operációs rendszerre helyezi informatikai hátterének jelentős részét, s mindezt olyan méretekben teszi, hogy közben egy új, önálló terjesztés születik.

Komáromi Zoltán

Összefoglaló a Linux hangszerkesztőiről

Akár a munkaasztalunk eseményeihez készítünk menő hanghatásokat, akár zenealbumainkat szeretnénk digitalizálni, mindenképpen szükségünk lesz a most következő programok valamelyikére.

A zenefájl-szerkesztő program minden hangokkal dolgozó felhasználó nélkülözhetetlen eszköze, hiszen ezzel lehet a rögzített hangokat szerkeszteni, kozmetikázni és finomítani. Ezen műveletek némelyike a szövegszerkesztésben is megszokottakhoz hasonlóan folyik – ilyen például a kivágás/másolás/beillesztés műveletcsoport –, mások viszont csak a hangszerkesztés területére jellemzőek.

Ez a cikk egy gyors kirándulásra invitálja az olvasót a linuxos hangszerkesztő programok világában. Nincs szükség arra, hogy bármilyen speciális dolgot tudjunk a digitális hangról vagy a digitális hangfeldolgozás elméletéről, és ha a bemutatott programok bármelyikét kedvünk szottyanna kipróbálni, azt is könnyen megtehetjük. Hiszen nincs másra szükségünk, csak egy működő hangrendszerrel rendelkező gépre. Mielőtt azonban nekivágnánk felfedező utunknak, vizsgáljuk meg, mire is jó egy átlagos hangszerkesztő és hogyan kell használni.

A közös tulajdonságok

A hangszerkesztés egyes műveletei legkönnyebben grafikus felületen hajthatók végre. A hangadatok láthatóvá tételével könnyen kikereshetők a problémás részek, mint például a szünetek vagy az amplitúdó-tüskék. Azzal, hogy a módosításra váró részeket gyorsan meg tudjuk keresni, nagymértékben felgyorsul az egész szerkesztési folyamat. A fájl egyes területeit pontosan ki tudjuk jelölni, a közelítési/távolítási lehetőségek pedig bármely ponton lehetővé teszik a nagyítást vagy kicsinyítést. Ez azért hasznos, mert így könnyen és gyorsan tudunk nagy fájlrészeket szerkeszteni, vagy egy-egy nagyon pontosan behatárolt, apró részt érintő műveletet végrehajtani.

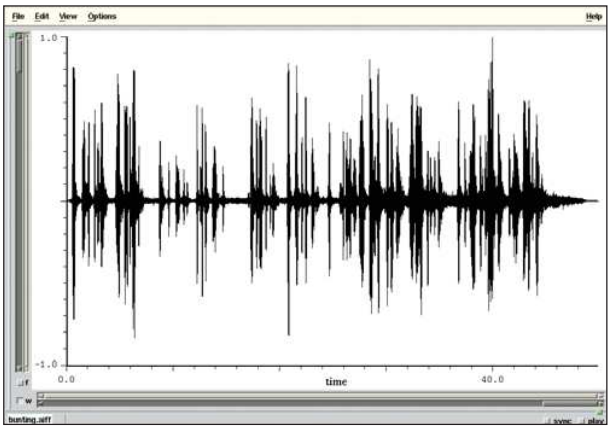
Egy jól megtervezett hangszerkesztő programnak mindenféleképpen rendelkeznie kell az alábbi műveletek lehetőségével:

- Kivágás/másolás/beillesztés
- Összeillesztés/beszúrás/csere

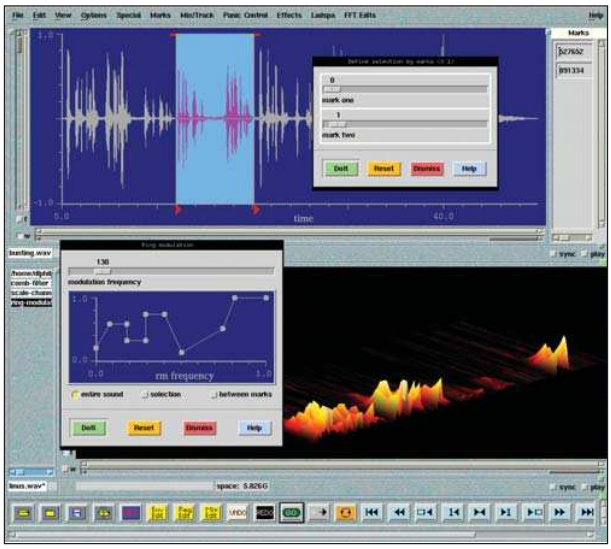
1. táblázat *A zeneszerkesztők képességei*

Szerkesztőprogram	ALSA	JACK	LADSPA	GUI	Méretkorlát	Licenc
Snd	i	i	i	Motif, GTK [1]	lemezhely	GPL
MiXViews	n [2]	n	n	InterViews	memória	[3]
DAP	n [2]	n	n	XForms	memória	GPL
Audacity	i	i	i	wxGTK	lemezhely	GPL
ReZound	i	i	i	FOX	lemezhely	GPL
Sweep	i	i	i	GTK	lemezhely	GPL
GLAME	i	n	i	GTK	lemezhely	GPL
LAoE	n [2]	n	n	Java	lemezhely	GPL
Swami	i	i	i	GTK	lemezhely	GPL
WaveSurfer	i	i	i	Tcl/Tk	lemezhely	[4]
GNUsound	i	n	i	GTK	memória	GPL
KWave	n [2]	n	n	Qt	memória	GPL

[1] Grafika nélkül is lefordítható. [2] Együttműködik az ALSA's OSS/Free emulációval. [3] Nem üzleti felhasználásra ingyenes, szabadon terjeszthető. [4] Korlátozás nélkül felhasználható és terjeszthető.



1. kép Az alapértelmezett Snd



2. kép A cikkíró Sdn-je

- Minták áthelyezése
- Sávok keverése
- Fájlok/sávok szinkronizálása
- Az időskála tömörítése/kiterjesztése
- Csúcsérték-eltolás
- Kiegyenlítés/szűrés
- Mintavételi frekvencia váltása
- Különböző időformátumok megjelenítése
- Egy fájl többféle nézetének a megjelenítése
- Több fájl egyidejű megjelenítése
- Független X/Y tengely szabályozás
- Maximális mintavételi érték megkeresése
- Az amplitúdó-burkológörbék különböző ábrázolásmódban (db, peak, RMS) való megjelenítése
- Csúcspont és amplitúdó burkológörbék szerkesztése
- Változó sebességű visszajátszás
- A minták kinyomtatásának lehetősége
- Spektrum-analízis

Amint azt látni fogjuk, az ebben a cikkben bemutatandó szerkesztőprogramok teljesítik ezeknek az alapvető elvárásoknak a túlnyomó részét, sőt, gyakran további egyedi lehetőségeket, eljárásokat is lehetővé tesznek.
Az *1. táblázat* a tulajdonságok egy másik sorát foglalja össze, amelyek nagy része – bár nem mindegyik – csak a Linuxra jellemző. A táblázatból azt is láthatjuk továbbá, hogy az itt bemutatott egyes szerkesztőprogramok milyen módon teszik ezeket elérhetővé.

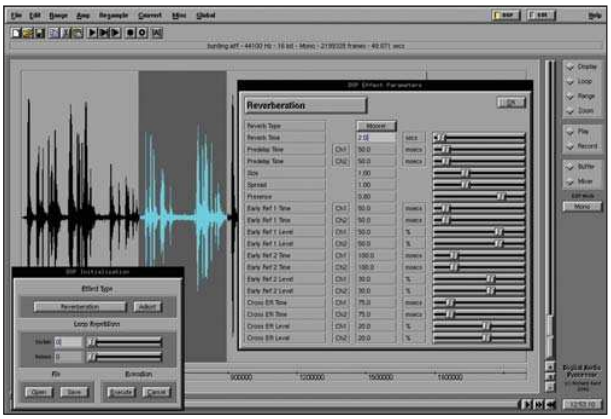
A mindennapi használat

Most pedig gondoljuk végig a hangszerkesztő programok néhány felhasználási lehetőségét. Az alábbi lista semmi esetre sem teljes, csupán azt mutatja meg, hogy én általában hogyan hasznosítok egy ilyen szerkesztőprogramot a saját munkámban:

- Felesleges szünetek eltávolítása a felvételekből.
- Nagyméretű fájlok feldarabolása kisebb részekre.
- Normalizálás.
- Olyan hanghatások létrehozása, mint a zengés (***reverb***), kórus vagy duplázás (***flanging***)
- A lejátszás sebességének csökkentése a hangmagasság megváltozása nélkül.
- Recsegések és kattogások eltávolítása a felvételekből.
- A mintavételezési sűrűség megváltoztatása.
- A fájlformátum megváltoztatása.
- A frekvencia-összetétel megváltoztatása, szűrése, azért hogy csillogóbb, vagy mélyben gazdagabb legyen.
- Zeneszerzés.

A lista még több tucatnyi művelettel kiegészíthető, minden felhasználó talál majd valamilyen különleges alkalmazási módot. Oktatói munkámban például nagyon hasznosnak bizonyult visszajátszás sebességének megváltoztatása úgy, hogy a hangmagasság közben ne változzon. A diákok gyakran hoznak hozzám olyan felvételeket, amelyek nehezen érthetőek, ha eredeti sebességen játsszuk le azokat. Ilyenkor az eredeti cd vagy mp3 fájlt wav formátumba alakítom át, beolvasom az Snd szerkesztőprogramba, majd az eredeti hangmagasság megtartása mellett egészen addig csökkentem a lejátszási sebességet, amíg minden egyes hangot tisztán nem lehet érteni. Így sokkal könnyebbé válik a pontos átirat elkészítése. Néhány szerkesztőprogram lehetővé teszi, hogy mindezt valós időben végezzük el, sőt, olyan is akad, amellyel a beállított ismétlendő szakaszt a lejátszás közben tetszőlegesen megváltoztathatjuk, ami rendkívül hasznos.
A normalizálás egy fájl amplitúdóit emeli a viszonylagos csúcspontjaira, így az összes amplitúdóérték a csúcsponthoz viszonyítva emelődik meg. A projektfájlok CD-re írás előtti normalizálásával az egyes részek közti hangerőkülönbségeket tudom kiegyenlíteni. A normalizálás a professzionális felvételek előállításának is elterjedten alkalmazott előfeldolgozó művelete.
A hangszerkesztő programoknak jó hasznát vehetjük a rosszul tömörített hangfájloknál is. Néhány szerkesztőprogram képes az mp3 és Ogg formátum közvetlen beolvasására, más programok viszont először átalakítják ezeket az állományokat, és az eredményként kapott fájlt olvassák be. Ezekkel az eszközökkel képes vagyok a felesleges szünetek eltávolítására és a felvétel megsérült helyeinek kijavítására. A hullámforma-nézetben ezek a helyek sérült vagy hiányos görbeként jelennek meg. A fájl eredeti formátumba történő

3. kép A MiXViews

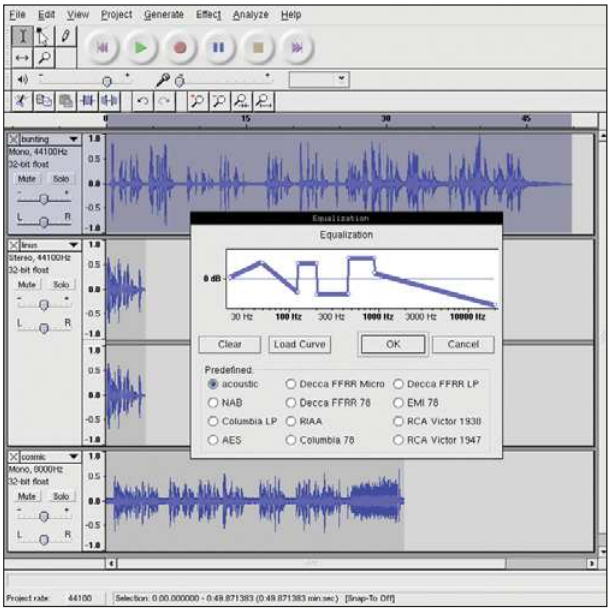


4. kép A DAP

visszaalakítása előtt normalizálást és frekvenciakiegyenlítést is szoktam végezni. Egy veszteséges tömörítéssel átalakított fájl hangfájllá alakítása majd újbóli tömörítése nagyfokú minőségromlással jár, ezért a szerkesztőprogram kiegyenlítő eszközével újra visszaállítom a frekvenciaegyensúlyt.
A rendelkezésre álló helyen a bemutatásra kerülő programoknak csak a legfontosabb vonásait tudom ismertetni, ezért minden program esetén csak a kiemelkedő szempontokat veszem figyelembe, a mélyebb megismeréshez saját kipróbálással juthatunk majd el. Kezdjük a körsétánkat néhány, Linux alá már régóta elérhető zeneszerkesztő programmal.
A zeneszerkesztő programok első hulláma akkoriban érte el a Linuxot, amikor az OSS/Free volt a rendszer hangfelülete, a Motif pedig egy vonzó grafikus eszközkészletnek számított. Ezek a programok nemcsak Linux alatti fordításra és futtatásra készültek, hanem a különböző UNIX rendszerekre is.

Az Snd 7.0

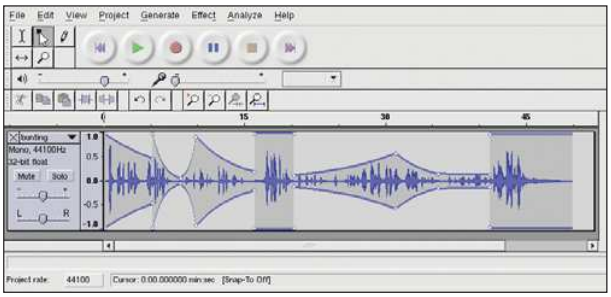
Az Snd-t ***Bill Schottstaedt*** ügyeskedte össze, a fejlesztést már a PDP miniszámítógépek korában elkezdte. Bár a program tudomásom szerint 1996 óta létezik, a Linux támogatása csak 1997-ben született meg. Az Snd-t tekinthetjük úgy is, mint egy különlegesen hatékony hangfájl-szerkesztő programot, egy határtalanul rugalmasan programozható hangszerkesztő eszközkészletet, vagy a Common hang- és zenekörnyezet grafikus komponensét. A zenei programok Common nevű családja tartalmazza ***Bill Schottstaedt*** Common Lisp Music (szoftveres hangszintetizáló) és Common Music Notation, valamint *Rick Taube* Common Music (egy zeneszerzésre alkalmazható metanyelv) programjait. Ezek olyan Lisp-alapú programok, amelyek beállításaik révén összetett interaktivitásra képesek. Az Snd hatékonysága leginkább a Guile felhasználói felületében rejlik, amelynek az alapja a Lisphez hasonló Scheme programozási nyelv. Az Snd felhasználói felülete tartalmaz egy Listener nevű ablakot, amelybe a felhasználó Guile-parancsokat írhat be, ily módon testre szabva a programot vagy megváltoztatva a megjelenését.
A képernyőképekből látható, hogy az Snd milyen változatosan beállítható. Az *1. képen* az Snd Motif-változata látható az alapértelmezett megjelenéssel. A *2. képen* a felhasználói felület már egy nagymértékben a saját beállításokra épülő képet mutat. Megtörtént az új menük létrehozása a beépített DSP-modulokhoz és az LADSPA bővítményekhez, a színek és a háttér is egyéni beállítást tükröz, és néhány összetett eszköz megépítése is megtörtént az Snd hanghatás-feldolgozójához. A 2. képen látható az Snd által ábrázolt amplitúdó-hullámforma, az OpenGL-megjelenítővel együtt, amely a hang spektrumát, a frekvenciaösszetevőit ábrázolja. A különböző kijelzők saját, helyzetérzékeny (***Context-sensitive***) előugró menükkel rendelkeznek, csakúgy, mint a fájl kijelölt és nem kijelölt tartományai.
A hangszerkesztő programok közül az én kedvencem az Snd, de biztosan sokan más véleményen lesznek. Ha egy kicsit megismerkedünk a Lisp nyelvvel, az Snd képességeibe is nagyobb bepillantást nyerhetünk. Szerencsére az Snd részletes dokumentációval segít át a tanulási folyamat nehézségein. Előre elkészített beállítófájlok is rendelkezésünkre állnak a testreszabás gyorsabb és egyszerűbb elvégzéséhez.



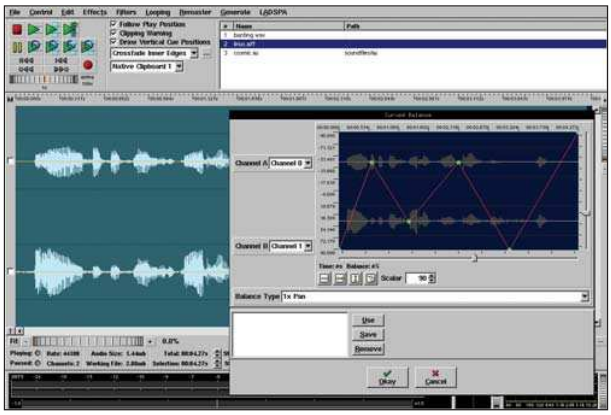
5. kép Az Audacity

A MiXViews 1.30

Doug Scott MiXViews programjának 1.0 változata 1995-ben jelent meg és ezzel az itt ismertetett programok legrégebbi-



6. kép Az Audacity Envelope (burkológörbe) eszköze



7. kép A ReZound

ke címet szerezte meg magának. A MiXViews a kezdetektől egy egyemberes projekt volt, melynek célja, hogy a UNIX és Linux rendszerekhez jó minőségű zeneszerkesztő programot biztosítson. A projekt továbbra is ebben az egyszemélyes formában folytatódik, és – ennek ellenére, vagy éppen ezért – magas színvonalon teljesíti a szerkesztőprogramtól elvártakat.

A MiXViews megbízható programcsomagot kínál az alapvető zeneszerkesztői teendők ellátására, és ezen felül van néhány olyan szolgáltatása is, amelyet nem találunk meg más Linuxra írt szerkesztőprogramban. A ***phase vocoding*** (*beszédtömörítésre kifejlesztett kódoló eljárás – a ford.*) és a lineáris prediktív kódolás (LPC) olyan digitális analizáló-szintetizáló eljárások, amelyek inkább az olyan programok kellékei szoktak lenni, mint amilyen a ***Csound*** vagy a ***Common Lisp Music***. Ezek az eszközök megvizsgálják a hang frekvenciáit és amplitúdóértékeit majd egy különleges vizsgálati fájlformátumban tárolják azokat. A fájlok egy Csound-hoz hasonló programmal beolvashatóak, s ez a program egy független eszközt kínál a fájlban tárolt frekvencia- és amplitúdó-összetevők szabályozásához, mielőtt a kapott értékek alapján a program újra előállítaná a hangfájlt. A MiXViews a saját LPC és a phase vocoder eszközeivel teljes körű szolgáltatást nyújt, ráadásul olvasni és szerkeszteni is képes a Csound phase vocoder kódolójával előállított fájlokat. A *3. képen* láthatunk néhányat a MiXViews grafikus eszközeiből a phase vocoder és LPC-vizsgálati adatok szerkesztése közben. Noha a működés hátterében álló elgondolás és a matematikai megoldások meglehetősen riasztóak lehetnek, a MiXViews felülete kísérletezésre ösztönzi az embert és magukat az eszközöket könnyen kezelhetővé és érdekessé teszi.

Ha ki szeretnénk próbálni a MiXViews-t, erre az előre lefordított bináris állományt javaslom. A program lefordítása ugyanis egy kicsit trükkös, és egy kevésbé elterjedt grafikus eszközkészletre (InterViews) is szükség van hozzá, érdemesebb a bináris fájlt letölteni, amit azonnal használatba is vehetünk.

A DAP 2.1.4

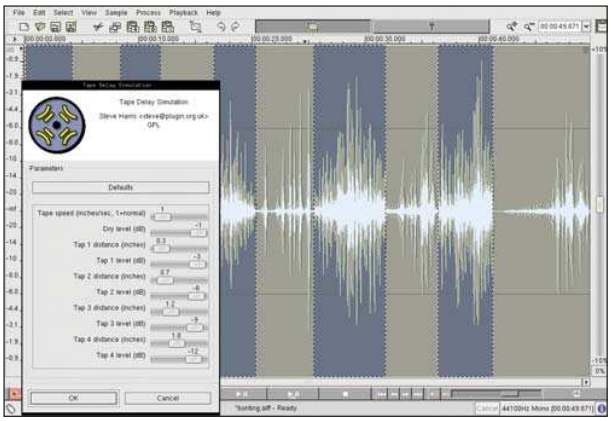
A DAP (Digital Audio Processor) ***Richard Kent*** programozó hozzájárulása a több operációs rendszeren futtatható hangszerkesztő programokhoz. A MiXViews-hoz hasonlóan a DAP grafikus felülete is egy viszonylag régi grafikus eszközkészletre, az ***XForms*** programkönyvtárra épül. A program a MiXViews-ra hasonlít abban is, hogy a szerkeszthető fájl méretének felső határa a rendszer memóriájának méretétől függ. Másrészről viszont az AIFF-hangfájlok szerkesztéséhez a DAP rendelkezik néhány kivételesen jól kivitelezett hullámszerkesztő eszközzel. A program szolgáltatásai közt megtaláljuk a DSP-modulok egy jól összeválogatott gyűjteményét (amely ***Kai Lassfolk*** SPKit kódjának a kiterjesztése), és egy jól használható mono-sztereo illetve sztereo-kvadro átalakítót.

A DAP néhány szerkesztőeszköze külön is említést érdemel, főleg a ***Resample*** és az ***Edit/Mix*** menük alatt találhatóak. A Resample menü az idő megnyújtásával vagy anélkül kínál hangmagasság- és mintavételifrekvencia-változtatási lehetőségeket, míg az Edit/Mix párbeszédablakok (Mix és Mix Range) egy csinos grafikus szabályozóeszközt biztosítanak a kevert fájl amplitúdójának kiegyensúlyozásához. Az AIFF fájlformátum és ennek a ciklusok támogatására gyakorolt hatása az egész programon érezhető. Például ha egy hanghatást alkalmazunk egy fájlon, a DSP párbeszédpanele egy eszközt biztosít a ciklusok fenntartásának és felszabadításának finomítására (4. kép). Bár a DAP felépítése kicsit elfogult az AIFF formátum javára, azért képes RAW és WAV formátumú fájlok importálására és exportálására is. Sajnos a DAP fejlesztése nem következetes. A XForms-ra épülő grafikus felület felett már eljárt az idő, és a fájlméret-korlát is komoly hátrány. A szerző elismeri a DAP korlátait, de ha beágyazott hurokpontokkal rendelkező AIFF-fájlokkal kell dolgoznunk, a DAP még mindig hasznos eszköznek bizonyulhat.

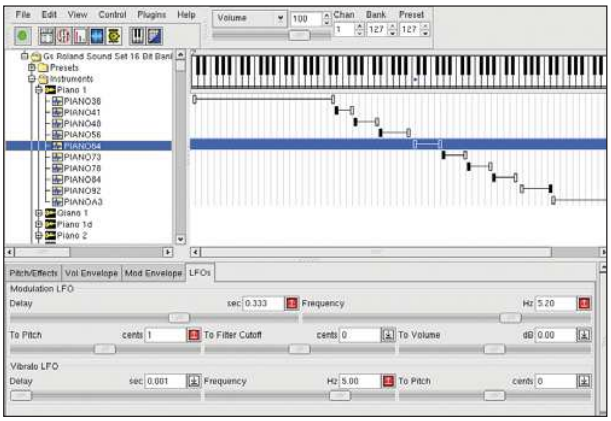
A szerkesztőprogramok következő csoportja a Linux hanggal kapcsolatos fejlesztéseinek újabb hullámához tartozik. Ezek természetes környezetéhez tartozik a korszerű grafikus felülettel ellátott eszközkészlet és az olyan Linuxos hangrendszer-komponensek, mint az ALSA, a JACK és a LADSPA. Fogalmilag is egységesebbek az elődjeiknél és sok hasonlóságot mutatnak a Windows és Macintosh felhasználók számára megszokott eszközökkel.

Az Audacity 1.1.3

Az Audacity az első képviselője ezeknek az újhullámos Linuxos hangszerkesztő programoknak. A program C/C++ nyelven íródott, a wxWindows felületfüggetlen grafikus eszközkészletet alkalmazza, és támogatja a natív LADSPA jelfeldolgozó bővítményeket. Az újabb kiadásai emellett megfelelnek a JACK követelményeinek is, s ezzel az



8. kép A Sweep



9. kép A Swami

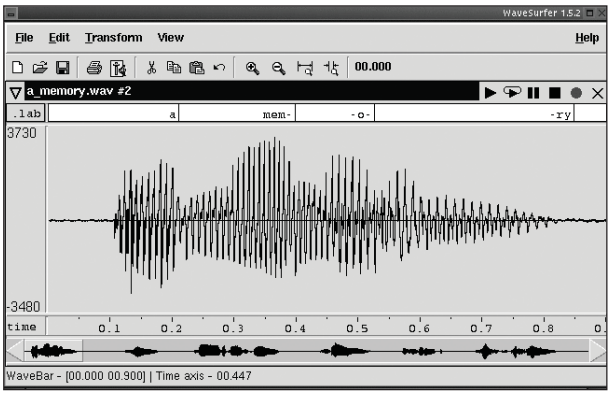
Audacity felhasználói számára lehetővé válik, hogy a program bemenetét és kimenetét más, JACK-képességekkel ellátott program felületével kapcsolják össze.
Az ***5. kép*** az Audacity-t mutatja három, egyidejűleg megnyitott állománnyal, amelyek egyike egy mono WAV-fájl, a másik egy sztereo AIFF-fájl, a harmadik pedig a Sun AU formátumában lévő fájl. Az Audacity képes MP3 és Ogg fájlok importálására is. Hála az ***Ogg/Vorbis*** programkönyvtáraknak az Ogg exportálás közvetlenül is működik, az MP3 exportálás feltétele azonban egy felhasználó által biztosított kódoló program. Az *5. képen* működés közben láthatjuk az Audacity natív frekvenciakiegyenlítő bővítményét is.
Az Audacity grafikus szerkesztőeszközeit élvezet használni. A *6. képen* a burkológörbe eszközt (***envelope tool***) láthatjuk az 5. képen látott egyik hangfájlra alkalmazva.
A különálló minták szintjén az Audacity rajzeszközei nagymértékben megkönnyítik az amplitúdótüskék és egyéb folytonossági hibák eltávolítását vagy kijavítását.
Az Snd-hez hasonlóan az Audacity is rendelkezik egy Lisp-alapú programozást lehetővé tévő felülettel, amely ***Roger Dannenberg*** Nyquist nevű alkotása. A Nyquist egy hangszintetizálásra és jelfeldolgozásra kifejlesztett nyelv, az *Audacity Effects* menüje pedig ehhez kínál egy olyan promptot, amely lényegében a Snd Listener-jéhez hasonlóan működik. A felhasználó beír egy Nyquist-kifejezést a prompt párbeszédablakába, megnyomja az OK gombot és amennyiben a kifejezés értelmezhető, az Audacity végrehajtja az aktív hangfájlon a kívánt műveletet.
Sokkal többet el lehetne mondani az Audacityről, mint amennyire itt lehetőségem van. Szerencsére a program kezelése könnyen elsajátítható, nyugodtan kipróbálhatjuk önállóan is a képességeit.

A ReZound 0.9.0 béta

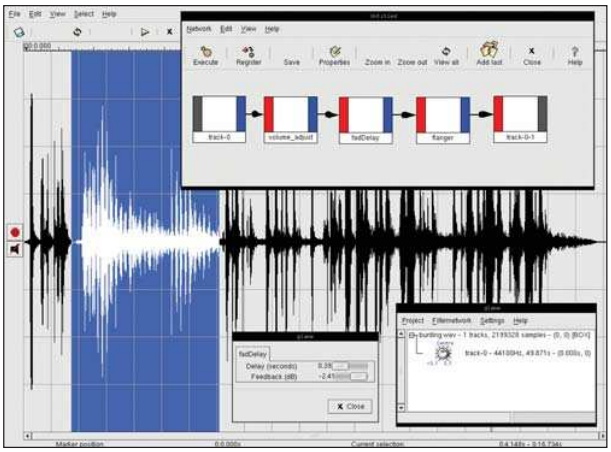
A színes felhasználói felület és kitűnő elrendezés teszi ***Davy Zurham*** ReZound programját kellemes vizuális élménnyé és könnyen használhatóvá. De a pofás külcsín a legkevesebb, emellett ugyanis megtaláljuk a szerkesztéshez szükséges teljes eszközkészletet, kitűnő átviteli szerkezeteket, néhány lenyűgöző beépített szűrőt, az LADSPA bővítmények támogatását és egy egyedülálló remastering/CD-író szerkezetet.
A *7. képen* látható a ReZound három beolvasott fájllal, amelyek egyikén éppen a Curved Balance eszköz dolgozik.
A Curved Balance a ReZound egyik remastering-eszköze, amelyek közt találhatunk zajkaput, dinamikasűrítőt, erősítő és normalizáló eszközöket. A ReZound egyéb szerkesztői ínyencségeivel együtt ezeket az eszközöket használhatjuk arra, hogy a tökéletességig gyúrjuk a hangot, mielőtt CD-re kiírnánk. A ReZound még ahhoz is biztosít egy egyszerű párbeszédablakot, hogy a kész munkát (a ***cdrdao*** program segítségével) közvetlenül a File menüből írjuk CD-re.
A ReZound LADSPA-támogatása a ***Kjetil Matheussen*** által írt LADSPA VST bővítmény (***vst.so***) alaptámogatásáig terjed. Ennek a bővítmények támogatására írt bővítménynek a használata egy működő, WINE-alapú felületet biztosít a VST/VSTi bővítmények Linux alatt történő futtatásához. A ***vst.so*** bővítmény jelenleg még a fejlesztés egy korai szakaszában van, ezért a gazdaprogrammal való együttműködése nagyon ingadozó lehet.
A ReZound legfrissebb változata támogatja a JACK zenei kiszolgálót, amellyel a ReZound kapcsolódhat az egymással párbeszédre képes hangfeldolgozó alkalmazások JACK-hálózatához. További JACK-fejlesztések is várhatók, ilyen többek között a hatás-előnézet, a zajeltávolító eszközök, natív idő/hang skálázás és még sok egyéb szolgáltatás és fejlesztési lehetőség.

A Sweep 0.8.2

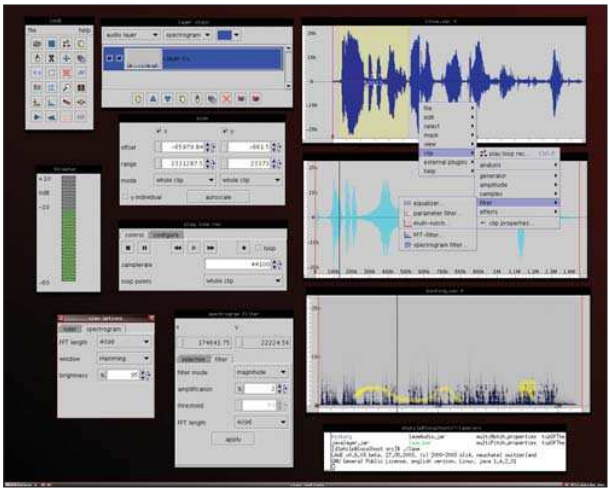
Első pillantásra ***Conrad Parker*** Sweep programja nagyon hasonlít a többi itt bemutatott újabb szerkesztőprogramra. Az ALSA-képességek biztosítják a megbízható alapvető szerkesztőfunkciókat, támogatja az LADSPA bővítményeket és tetszetős és korszerű GTK-felülettel bír. Azonban a Sweep két szokatlan eszközt is kínál, amelyek különleges értékkel ruházzák fel a programot. Az egyik a nemlineáris szerkesztés számára a többszörös tartomány kijelölésének a lehetősége, a másik pedig egy érdekes kis eszköz, amelyet Scrubby-nak neveztek el. Egy hangfájlban a kijelölés megadása rendszerint úgy történik, hogy a kurzort a kijelölendő terület kezdőpontjára visszük, megnyomjuk a bal egérgombot, majd azt nyomva tartva elhúzzuk a kurzort a szakasz végéig. Ez a kijelölési módszer általánosan elterjedt, az eddig bemutatott szerkesztőprogramok mindegyike ezt alkalmazza. Ezt teszi a Sweep is, de lehetővé teszi többszörös kijelölések meg-



10. kép A WaveSurfer



11. kép A GLAME



12. kép A LaoE

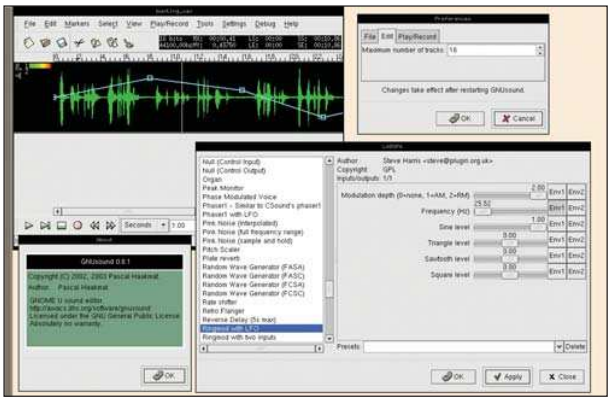
adását is. Tartsuk nyomva a CTRL billentyűt a kijelölés folyamán, és voil□, máris több szakaszunk van kijelölve várva a további feldolgozást.
Az ***Invert Selection*** (kijelölés felcserélése) ügyes megoldást kínál arra, hogy egy hangfájlban a hatások párbeszédjét hozzuk létre. A *8. kép* mutatja ennek az átalakításnak a következményeit: megadunk egy többszörös kijelölést, alkalmazzuk a reverse hatást ezeken a szakaszokon, felcseréljük a kijelölést (a kijelöltek válnak jelöletlenné és fordítva), majd alkalmazzuk az LADSPA hatást az újonnan kijelölt részekre. Élvezetes, hatékony és kreatív szolgáltatás.
A Scrubby a Sweep virtuális lemezjátszótűje, amely képességeit tekintve egy szabadon mozgatható lejátszófejnek felel meg, és olyan szolgáltatásokat nyújt, amelyeket leginkább egy lemezlovas lemezjátszója esetén szoktak emlegetni. A Scrubby előadóeszközzé változtatja a Sweep-et, ami egy hangfájl-szerkesztő programnál meglehetősen furcsa tulajdonság. Egy képernyőképpel nem érzékeltethetjük eléggé mindezeket a képességeket, ki kell próbálnunk a programot és a saját szemünkkel és fülünkkel meggyőződni az igazságról.

A Swami 0.9.1a

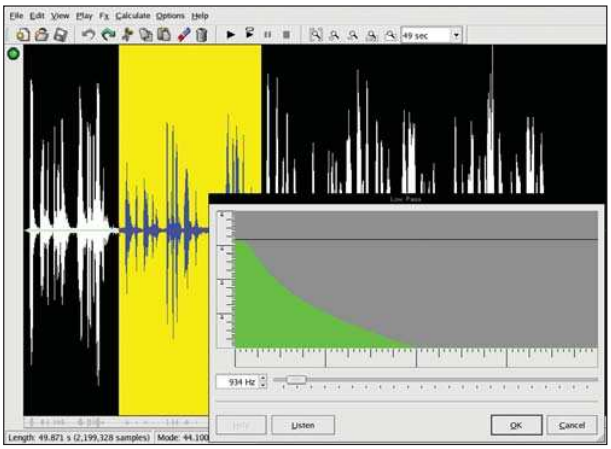
A soundfont (SF2) formátum egy olyan összetett hangfájlformátum, amelyben nem kizárólag a hangadatok tárolódnak, hanem a különböző hanghatásokra és az előadásra vonatkozó szabályok is. A soundfont formátum megkerülhetetlenné vált a számítógépes hang világában, olyan alkalmazásokban lelve meg otthonát, mint a Csound, jMax, Fluidsynth és még sok másik.
Amennyiben soundfont formátumban lévő anyagokkal szeretnénk dolgozni, kétségtelenül az eszközeink közé kívánkozik ***Josh Green*** Swami nevű programja. A Swami egy kizárólag a soundfont formátumot kezelő szerkesztőprogram, amely jól megtervezett grafikus felülettel, és egy sereg hasznos szolgáltatással bír. Szerkeszthetünk már létező hangfájlokat, vagy létrehozhatjuk a saját soundfont formátumú fájljainkat a különálló minták szintjétől kezdve az összetett hangszerekig. Arra is lehetőségünk van, hogy megtervezzük és összeállítsuk saját soundfont-csoportjainkat. Rendelkezésre állnak az eszközök sebességét, átviteli görbéjét, billentyűzetkiosztását, és modulátor-útvonalát beállító eszközök is. A Swami pontosan meghatározott célterülete miatt már nincs igazán mit elmondani róla, első osztályú, melegen ajánlott Linuxos programról van szó.

A WaveSurfer 1.5.7

A WaveSurfer-t ***Kåre Sjölander*** és ***Jonas Beskow*** alkotta meg azért, hogy a beszédkutatásban megalkossák a legjobban használható eszközt. A WaveSurfer egy tökéletesen használható általános célú hangfájl-szerkesztő, de különleges erényei akkor mutatkoznak meg igazán, amikor a beszéd világában kell a hangot elemezni, ábrázolni vagy szerkeszteni.
A WaveSurfer a népszerű Tcl/Tk parancsnyelven és eszközkészlettel íródott, s így az arra késztetést érző felhasználók számára teljes hozzáférést enged a program belső világába. A WaveSurfer hangfeldolgozással kapcsolatos műveleteit a szintén ***Kåre Sjölander*** által írt SNACK hangfüggvénykönyvtár kezeli. A SNACK maga is kiegészíthető C/C++ nyelven írt felhasználói bővítményekkel.
A *10. kép* a WaveSurfer egy egyszerű alkalmazását mutatja a beszédelemzésben és -ábrázolásban. A főpanel a teljes hullámban kiemelt részt jeleníti meg, a felirat-kijelző pedig a hangzó fonémát jelzi. A felirat-sáv csak egy a WaveSurfer beszédközpontú jellegzetességei közül. A többi közt találunk spektografikus megjelenítőket, hangmagasság-görbe kivonást, és a támogatott hangfájl- és hangrögzítési formátumok széles palettáját.



13. kép A GNUsound



14. kép A KWave

A GLAME 1.0.1

A GLAME fejlesztői a szerkesztőprogramjukban egy szokatlan tervezési filozófiát valósítottak meg. A GLAME (GNU/Linux Audio MEchanics) fel van szerelve mindazokkal a hangszerkesztő képességekkel, amiket elvár az ember, ezen felül egy hatékony hangszintetizáló és -feldolgozó egységet is tartalmaz, melynek neve filternetwork (szűrőhálózat). A filternetwork egy rajztáblát tartalmaz, amelyen ikonok képviselik a hangszintetizáló alapegységeket, ezeket összekapcsolva jön létre a hangszintetizáló vagy -feldolgozó lánc. Pillanatnyilag ezek az alapegységek lehetnek oszcillátorok, burkológörbe-generátorok, szűrők, I/O-modulok és LADSAP-bővítmények. Ha elkészült egy ilyen szintetizáló hálózat, akkor futtatható, és valós idejű hang vagy pedig kimeneti fájl állítható elő a segítségével további feldolgozásra (természetesen a GLAME-ben). A hullámforma kijelzőjén jobb egérgombbal kattintva egy felugró menü jelenik meg, amelyben megtalálható az Apply Custom menüpont. Ezt kiválasztva a szűrőhálózatunkat alkalmazhatjuk az éppen aktív hangfájlra, ami érdekes feldolgozási lehetőségeket kínál számunkra.
A *11. képen* egy egyszerű példát láthatunk erre. A hullámforma-kijelzőben kijelölt részt egy olyan szűrőhálózattal dolgoztuk fel, amely egy erősítővezérlőt, egy LADSPA késleltető bővítményt és egy flanger-effektet tartalmazott.
A sáv-modulok és az alapértelmezett I/O-kapuk is a hálózat részei, amelyek az eredeti bemenetet és a feldolgozott kimenetet képviselik.

LAoE 0.6.03 béta

Oliver Gäumann Layer-based Audio Editorja (LAoE, réteg-alapú hangszerkesztő) egy újabb egyedülálló tervezési filozófia hordozója. A szerkesztési munkafolyamat a LAoE programban a következőképpen épül fel: létrejön egy verem a hangfájlokból, majd ezután következik a kívánt szerkesztési és feldolgozási eszközök megnyitása a veremben lévő egy vagy több réteg (hangfájl) számára.
Első alkalommal nagyon furcsának tűnt a munkának ez az újszerű megközelítése, de miután megértettem a program szervezési módját, élvezni kezdtem az elrendezést és egy gyors munkamódszert sikerült kifejlesztenem a segítségével.
A LAoE külön jutalompontokat kap az eredetiségéért, amely a spektrumkijelzőjén való közvetlen szerkesztési lehetőségben nyilvánul meg. Egy, a felhasználó által definiált ecsettel lehet az FFT-szűrés feletti területeket lefesteni, maga a szűrő pedig finomabb felbontásra is beállítható. Az itt bemutatott szerkesztőprogramok nagy része biztosítja a frekvenciaösszetevők tulajdonságainak megjelenítését, de csak a LAoE teszi lehetővé a közvetlen spektrális szerkesztést.
A LAoE az egyetlen Java-alapú szerkesztőprogram az itt bemutatottak közül. A 800 MHz órajelű gépemre – amely nem igazán gyors gép a mai mércével mérve – feltelepítettem a Sun JDK 1.4 változatát, s a LAoE felhasználói felületét minden szempontból gyorsnak és rugalmasan reagálónak találtam.

A GNUsound 0.6.1

Pascal Haakmat GNUsound programja megjelenésében szerény, tartalmában azonban annál gazdagabb. Itt is megtaláljuk az alapvető szerkesztőeszközök teljes skáláját, a LADSPA bővítmények támogatását, és néhány különleges eszközt a hangfájlok kijelölésére, kiválasztására és megtekintésére. A GNUsound is alkalmazza a sávos szerkesztési lehetőséget, ami azt jelenti, hogy egy folyamat során kijelölhetünk fájlokat és azokat egy sok sávos magnó sávjaihoz hasonlóan használhatjuk a keverésre.
A GNUsound egy másik ügyes megoldása a hanghatások feldolgozásakor használt burkológörbe megvalósítása. A két felhasználó által megadott burkológörbe közül az egyik kijelölhető, mint a hozzá rendelt feldolgozási paraméter vezérlőgörbéje, ezzel még dinamikusabb körvonalat adva a hanghatás feldolgozásunknak.
Bár a GNUsound eredeti szándék szerint a GNOME alatti futtatásra készült, minden gond nélkül sikerült egy Planet CCRMA Red Hat 9.0 rendszer alatt lefordítanom és a BlackBox ablakkezelővel futtatnom.

A KWave 0.7.0-1

A KWave már 1999 óta fejlesztés alatt áll, így valószínűleg inkább a tiszteletre méltó öregfiúk közé kellett volna sorolni. Azonban a fejlesztőcsapat lépést tartott a célkörnyezetével, a KDE-vel, aminek korszerűbb megjelenés és néhány érdekes továbbfejlesztés lett az eredménye.
Az új KWave is megtartotta az eredeti program törekvéseit arra vonatkozóan, hogy a fájlok feldolgozása elsősorban

grafikus eszközökkel folyhasson. A *14. képen* láthatjuk a KWave aluláteresztő szűrőjének szerkesztőablakát kiegészítve egy feldolgozó-előnézeti lehetőséggel. A Listen gombbal ciklikusan lejátszható a fájl vagy a kijelölt részlet, miközben valós időben állítgathatjuk a szűrő paramétereit, ami egy igen jól használható szolgáltatás a hanghatások teszteléséhez. A kedvenc eszközeim némelyike, mint például az összegző szintézisgenerátor, nem került az eredeti KWave-ből újraírásra. Ezek a szolgáltatások most szürkén jelennek meg a menükben és nem elérhetőek, de a fejlesztők tervezik ezeknek a funkcióknak a visszaállítását, további tulajdonságokkal kiegészítve. A GNUsound-hoz hasonlóan a KWave által kezelt fájl méretének is korlátot szab az elérhető memória mennyisége, de ezt leszámítva egy remek kis szerkesztőprogramról van szó, amely nagyszerűen megfelel a KDE környezetben mindennapi használatra.

Záró megjegyzések

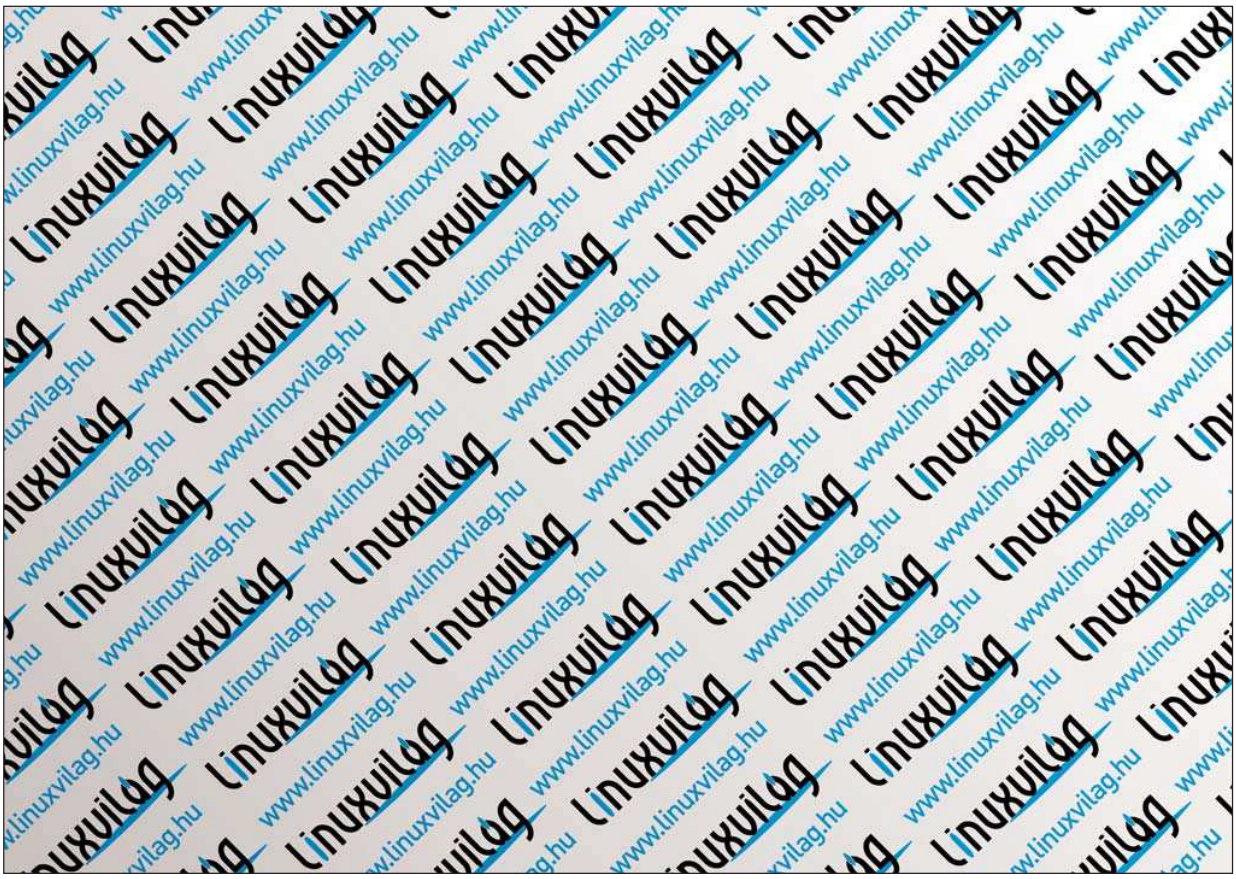
Remélem, cikkemmel sikerült felkeltenem az érdeklődést a bemutatott programok némelyike iránt. Hiszitek vagy sem, ezeken kívül is rengeteg hangfájl-szerkesztő programot lehet találni, itt csak a legnépszerűbbekre összpontosítottam. A Linux Sound & MIDI Applications (Linuxos hang- és MIDI-programok) honlap Soundfile Editors (hangfájl-szerkesztők) szakaszában teljes listát találhatunk a Linux alatt elérhető szerkesztőprogramokról (lásd a hálózaton elérhető Kapcsolódó címek szakaszt).

Melyik a legmegfelelőbb? Ezt nagyon nehéz eldönteni. Személy szerint az Snd iránt érzek nagy rokonszenvet a programozhatósága miatt és a ReZound iránt a grafikus felületének jólszervezettsége okán, de mindenkinek saját magának kell kipróbálnia néhányat ahhoz, hogy megtalálja az igényeinek leginkább megfelelőt. A legfontosabb, hogy ne ijedjünk meg ha első ránézésre túl bonyolultnak tűnik is némelyik program közülük. Közelítsünk úgy hozzájuk, mintha a GIMP-pel tennénk: próbálgassuk véletlenszerűen a szolgáltatásokat, amíg rá nem jövünk, mit is csinálnak, és ne féljünk a visszalépés gombot használni ha szükséges. Az ilyen típusú programok próbálgatása jó szórakozást nyújt, és kreatív csatornákat nyithat meg az emberben. Ha pedig létrehoztok valamilyen megosztásra érdemes zenét, ne habozzatok a tudomásomra hozni. Gyerünk, dobjatok össze néhány élvezetes zajt!

Linux Journal 2004. június, 122. szám

Dave Phillips

Az Ohio állambeli Findlayben élő zenész, tanár és író. 1995, a Linuxszal való első próbálkozása óta aktív tagja a Linux zenei közösségének. Ő a szerzője a The Book of Linux Music & Sound (A Linux hangjának és zenéjének könyve) könyvnek, valamint a Linux Journalban megjelent számos cikknek.



Linuxos hangfelvevő stúdió

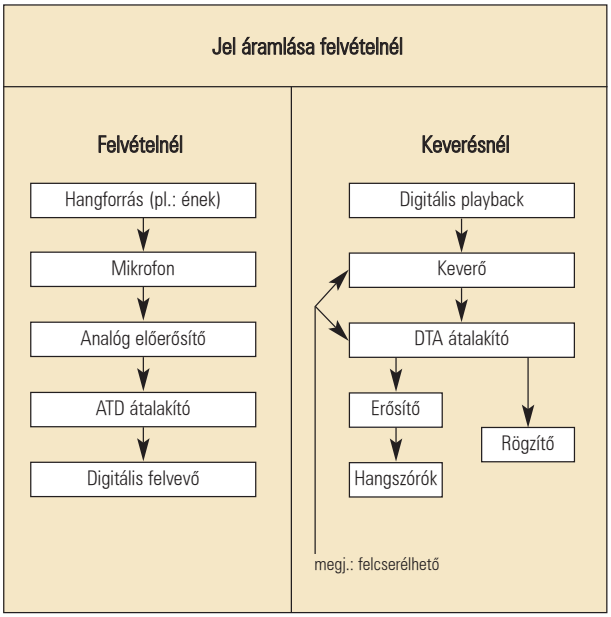
Linux-alapú merevlemezes rögzítő segítségével saját, olcsó stúdiót hozhatunk létre. Mostantól közted és a készítendő nagyszerű album közt semmi más nem áll, mint a gyakorlás, barátom, csakis a gyakorlás.

Billentyűzet mellett nőttem fel. A hidegháborús grey TRS-80-assal kezdtem, aztán jött a zöld-képernyős Apple II, IBM klónok, 8088-as, 286-os, PC-DOS, majd Windows (amelyben nem volt parancssor) és végül a UNIX parancssor. Később a rögzítés kezdett izgatni, ami a parancssortól a stúdiók felé vonzott. Továbbra is PC-s srác voltam, de soha meg nem fordult volna a fejemben, hogy PC-t vigyek a stúdióba. A megfizethető merevlemezek és memóriák a túl kicsik, a hangkártyák ócskák, a processzorok pedig lassúak voltak.
Aztán megérkezett a Linux. Persze meg kellett várnom míg a merevlemezek megnőttek, a lapkák felgyorsultak, de a fizetős programok továbbra is kívül álltak az elérhetőség körén. Fejlesztettem a stúdiómat, miközben sok mindent megtanultam a Linuxról. Ideje elmesélnem mit alkottam a stúdiómban és elmondanom hogyan készíthetünk Linux-alapú stúdiót. A Linux hangrögzítés általános leírása túl nagy lenne, így ahol szükséges külső forrásokra fogok hivatkozni (lásd az hálózati Források szakaszt).

Hogyan hozzunk létre Linuxos Stúdiót

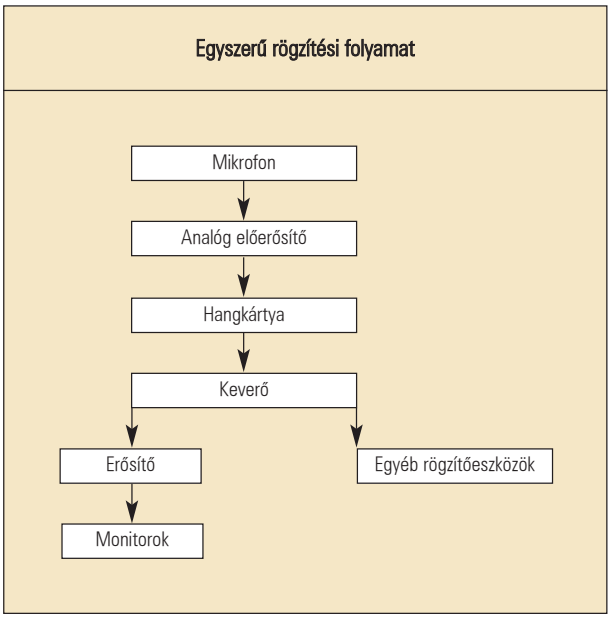
Mint általában, a stúdióhoz vásárolandó dolgok és azok beállítása attól függ hogyan döntünk néhány kulcsfontosságú kérdésben. Az alkatrészpark olyan egyszerű mint a 3.1415. Bármi ami képes Linux-ot futtatni, Linux audio-alkalmazásokat is tud működtetni, de azért tartsuk sem előtt a következőket:

- A hangfelvétel cd minőség mellett (44.1KHz, 16 bit) 5MB-ot használ sávonként, azaz három perc zene sztereóban rögzítve 30MB-ot foglal el a merevlemezen. A Többsávú felvétel (***Multitracking***) kettőnél több sávot használ. Egy átlagos, három perc hosszú, 24 sávos projekt 360MB-ot használ el, nem számolva a felhasznált, rögzített hangot.
- Kisebb fejlesztések a RAM méret és a CD-ROM sebesség terén mindig jól jönnek ha öregebb felszereléssel dolgozunk. A CD író szintén jó barátunk, mint az bizonyára mindenki tudja.
- Néhány hibás videokártya zajt visz a hangba.
- Linux alatt néha nehéz hozzájutni a meghajtókhoz, ezért olvasunk utána és kérdezősködjünk vásárlás előtt, különösen a hangkártyák esetében.

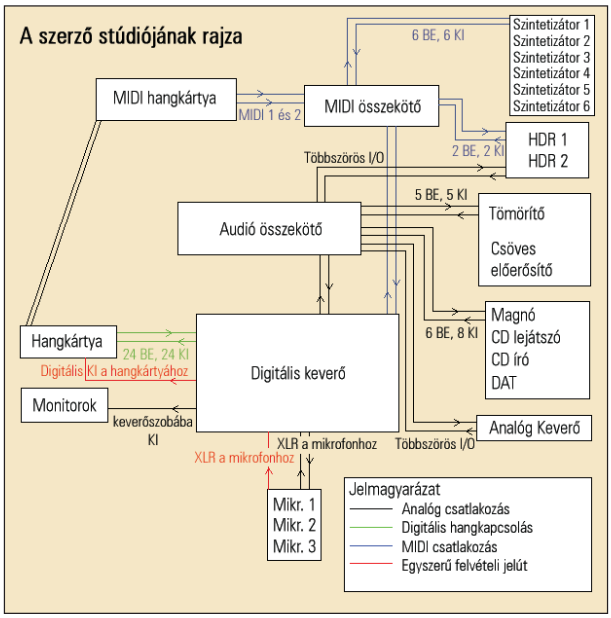


1. ábra Alap Rögzítési Folyam egy egyszerű projekthez

A program beszerzése majdnem ugyanilyen könnyű.
A latencia alacsony kell legyen, ezért a rendszermagot kicsit meg kell buherálnunk egy alacsony lappangási folt segítségével. A merevlemezt megfelelően kell beállítanunk. Ezek a témakörök nem férnek el itt, de utánanézhetünk a dolognak a Források közt és a weben. Továbbá készítsünk Microsoft Windows-os kettős rendszerindítási lehetőséget hátha hibakeresésre lesz szükség. Elképzelhető, hogy le kell tesztelnünk az alkatrészeket egy másik operációs rendszeren, hogy a problémát leszűkíthessük a Linux meghajtóra, illetve lehet egyéb feladatunk is, például frissítenünk kell a gyári kódot (***firmware***), amelyet Windows gépen kell elvégeznünk.
Most hogy megvan a gépünk, ideje eldöntenünk milyen stúdió alkatrészekre van szükségünk. Én szeretem az adott projekt jelfolyamatát figyelni mert abból megtudom mire van szükség. Az *1. ábra* mutatja be az alapokat, hogy merre megy a jel egy rögzítési projektben. Vessünk egy pillantást



2. ábra Egyszerű Egy-utas jelfeldolgozás a stúdióban



3. ábra A Szerző stúdiója

a *2.* és *3. ábrára* is. A második egy egyszerű stúdió drótozási szerkezete míg a harmadik az én stúdióm szerkezetét mutatja be. A ***lynchpin***-el kezdem, ami tulajdonképpen néhány gyengítés a jelláncon.

Analog-Digital átalakítás

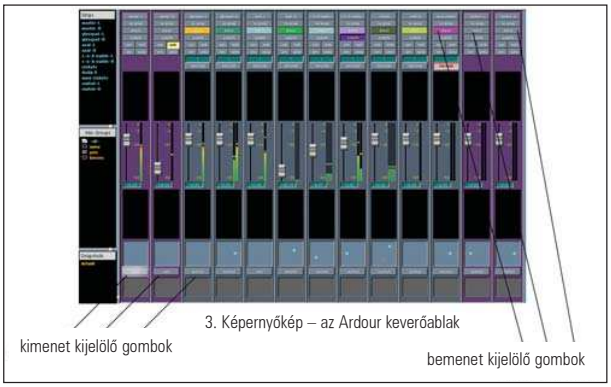
A digitális rögzítés kulcskérdése az analóg-digitális és a digitális-analóg átalakítás (ADC és DAC). Más szavakkal, a hangnak valahogy be kell jutni a és ki kell jönni e a számítógépből. Mindkét irányban választanunk kell több lehetséges megoldás közül.
Rögzítéshez szükségünk lesz ADC átalakításra. Ez történhet a hangkártyában, a digitális keverőn, vagy önálló ADC-n.
A hang kijövetele (DAC) két részből áll, hallgatásból (vagy monitoring-ból amiről alább többet olvashatunk) és a keverésből. Keveréskor jobb ha nem váltunk vissza analóg módba. Digitálisan keverhetünk a gépen belül és kívül is, a keveréket kimentve ***.wav*** állományként vagy digitálisan elküldve a digitális rögzítőnek. Azt mindenképpen jegyezzük meg, hogy ahhoz, hogy a hangot hallani lehessen, DAC átalakítást kell végezni. Ha mindent digitálisan oldunk meg, CD-t készítünk és az autónkban lejátsszuk azt, akkor ott történik a DAC.

A keverő: Analóg, Digitális vagy Program?

Mint azt nyilván mindenki tudja, a lehetőségek korlátlanok. Tetszőleges összeállításokat választhatunk, például analógból digitálissá alakíthatjuk a hangot hangkártyával, míg digitálisból analóggá azon kívül, és vica versa. Valamivel egyszerűbb, ha minden átalakítást egy helyen végzünk, legyen az a hely akár a hangkártya, akár egy másik eszköz. Az, hogy ezzel leegyszerüsítjük-e a dolgot, attól függ, hogy van-e külső keverőnk.
A legegyszerűbb összeállítás esetén, ahol – tegyük fel – tömegcikként kapható hangkártyát használunk egyetlen sztereó kimenettel, a keverést végezhetjük teljes mértékben a számítógépen is, és akkor nem sokat kell törődnünk azzal, milyen ADC és DAC átalakítás történik a hangkártyában. Ilyenkor nincs külső keverőnk (lásd az alábbi Előerősítő részt).
Ha viszont rendelkezünk külső keverővel, az mindenképpen analóg vagy digitális. Mindkét esetben profi hangkártyára van szükségünk, amely képes szétválasztani a csatornákat, szemben a tömegtermékekkel, amelyek egyetlen sztereó keveréket küldenek ki, így kénytelenek vagyunk a gépen keverni.
Ha analóg keverőt használunk, a DAC és ADC részek a hangkártyában történnek. Ennek megfelelően olyan hangkártyára vagy hangkártya/breakout box párosra van szükségünk, amely analóg be és kimenettel rendelkezik. Ilyen például az RME PCI kártyasíne és a Multiface kombinációs kártya (combination card). Nagyon jó bevezetést találunk erről a kártyáról a LJ Weblapján (lásd a hálózati forrásokat).
Ha digitális keverő mellett döntünk a DAC és ADC magában a keverőben lesz. Ennek megfelelőn olyan hangkártyára lesz szükségünk, amely a keverőnkkel együttműködő digitális ki- és bemenettel rendelkezik. Az én esetemben ez történetesen az RME HDSP 9652-ese.
Sok digitális keverő rendelkezik egyébként programcsomagokban megtalálható beépített hatásokkal (effektekkel) és feldolgozással, például útózengéssel, tömörítéssel és zajkapukkal. Az analóg keverők közül csak kevés rendelkezik ilyen képességekkel, így ha hagyományos analóg keverést hajtunk végre kicsit többet kell fordítanunk a kimenő hatásokra és feldolgozásra. Ha van rá pénzünk, érdemes beruházni egy ilyenbe, hiszen még ma is sok előnyük van, de ha olcsón akarjuk megúszni a dolgot, a digitális keverőt javaslom.
Néhány kérdés a legjobb hangkártya kiválasztásához:

- Zajos?
- Képes felvenni miközben lejátszik (duplex mód)?
- Hány csatornán képes lejátszani egyszerre?
- Hány csatornát képes rögzíteni egyszerre?
- Milyen típusú fizikai I/O csatlakozói vannak?

4. ábra Az Ardour Szerkesztőablaka



3. Képernyőkép – az Ardour keverőablak

5. ábra Az Ardour mix ablakban kiválaszthatjuk a bemenetet, a kimenetetés valamennyi csatorna szintjét

- Van beépített MIDI rendszere?
- Van hozzá Linux meghajtó?

Ha az utolsó kérdésre keressük a választ, az ALSA Sound Card Matrix-ban érdemes körülnéznünk (lásd a forrásokat).

Mikrofonok

Ha akusztikus forrásokat rögzítünk, például hangot vagy dobokat, mikrofonra is szükségünk lesz. Döntésünket ismét a pénzkeret illetve a rögzítendő anyag határozza meg. Például ha közepesen nagy keretünk van és akusztikus gitárt és énekest szeretnénk rögzíteni, két darab AKG 414-est javaslok (darabja körülbelül 1,000 dollár). Ha ősi hangfelvételeket kell rögzítenünk és bőven van keretünk, egy darab Neumann U87-est javaslok (körülbelül 3,000 dollár). Persze az is lehet hogy szűkös a keret, mégis szeretnénk gitárzenét és vokált felvenni. Ez esetben két Shure SM58-assal próbálkoznék, melynek darabja 100 dollár. Természetesen ha soha nem rögzítünk akusztikus forrásokat, kizárólag közvetlenül csatlakoztatott szintetizátort használunk, mikrofonra és előerősítőre sincs szükségünk.

Előerősítők

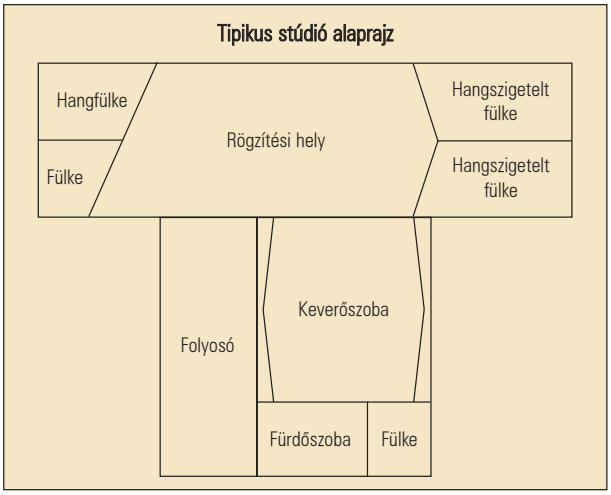
A mikrofonból érkező jelet erősíteni kell, hogy elég erős legyen a megfelelő rögzítéshez vagy sugárzáshoz. Ha számítógép-mikrofont csatlakoztatunk egy populáris hangkártya mikrofon bemenetéhez, valójában előerősítőt használunk, így elegendő erős jelet kell kapnunk. Ha azonban a line input bemenetre csatlakoztatjuk, aligha fogunk bármit is hallani. A profi hangkártyáknak nincs is 1/8”-as mikrofon bemenetük, mivel feltételezik, hogy van külső előerősítőnk.
A kérdés az, hogy külön erősítőt használjunk, vagy inkább a keverőbe építettet. A legtöbb embernek általában tökéletesen megfelel a keverőbe épített erősítő. Az egyetlen probléma ilyenkor az lehet, ha egy időben többre van szükségünk, mint amennyit a keverőnk elbír, vagy ha esztétikai okokból szeretnénk külső keverőt használni.
Az előerősítő szükségessége jó indok a külső keverő vásárlására, hiszen ha több analóg bemenettel rendelkező profi hangkártya mellet nincs keverőnk, akkor egy rakat külső előerősítőre lesz szükségünk, ami általában drágább és kevésbé rugalmas.

Megfigyelés

A hallgatáshoz azt használhatunk amit csak szeretnénk, a számítógép hangszórótól kezdve a fejhallgatón és az otthoni hangfal/erősítő kombináción keresztül a stúdió referencia monitorokig. A hangfalak és a monitorok közt azonban van egy nagy különbség. A hangfalakat úgy tervezték, hogy feljavítsák a felvétel minőségét, a monitorok célja az eredeti pontos és a lehetőségekhez képest változatlan visszaadása. Amennyiben pontos munkát szeretnénk végezni, monitorokra lesz szükségünk.
A stúdió monitorok többféle változatban kaphatók. A legfontosabb amit tudnunk kell, hogy a készlet erősített-e. Ha nem, akkor erősítőre is szükségünk lesz, az otthoni hangfalainkhoz hasonlóan. A hagyományos hangfalaink lecserélése a stúdió monitorokra majd rácsatlakoztatása a meglévő erősítőre nem nagy feladat.

A digitális rögzítő

A digitális rögzítőnk a Linuxos gép lesz. Az alternatívák ez esetben a program kiválasztásánál jelentkeznek. Gyakorlatilag nyílt forráskódú audio-alkalmazások százai állnak rendelkezésünkre Linux alatt, a merevlemezes rögzítőktől kezdve a MIDI szekvencerekan át az MP3 rögzítőkig. Valamennyit lehetetlen lenne most ismertetni, ezért az általam használt Ardour nevű stúdió-eszközre fogok koncentrálni. (A program beszerzéséről a „Hogyan kezdjünk hozzá” szakaszban olvashatunk bővebben.)
A legtöbb programig el is Google-ozhatunk, van jó néhány nagyszerű csomag. Jómagam Red-Hat használó lévén a Planet CCRMA-t használom. A Planet a ***Stanford Center for Computer Research in Music and Acoustics*** projektje,melyet egy nagy tudású srác, ***Fernando*** vezet. Nando Red Hat RPM csomagokat készít a legtöbb audio és videó alkalmazáshoz, meghajtóhoz, eszközhöz sőt, még egyedi rendszermagokhoz is, ráadásul kiterjedt telepítési útmutatót is írt a rendszermag, az ALSA meghajtó és program, valamint a gépünk teljesítmény növelési lehetőségeiről.
Egy idézet az Ardour honlapról, „Az Ardour több csatornás merevlemez rögzítő (HDR) digitális audió munkaállomás (DAW). Egy időben 24 vagy még több, 32-bit szélességű csatornát képes rögzíteni 48KHz-en....” Az Ardour használatához 2.4-es vagy későbbi, alacsony latenciájú rendszermag, 0.9-es vagy újabb sorozatú ALSA hangmeghajtó valamint



6. ábra A tipikus stúdió alaprajzon megtaláljuk az fő rögzítő teremtől elkülönített elhatároló fülkéket (ISO), de ennél egyszerűbb szerkezet is elképzelhető

a JACK (Jack Audio Connection Kit) szükséges. Ezeken túl ablakkezelőre is szükségünk lesz, ugyanis sok más Linuxos audió-alkalmazáshoz hasonlóan parancssorból nem fut. Magam részéről az Ardourt Fluxbox, néha KDE alól futtatom, de szionte bármelyik kezelő megteszi.
Az Ardour bármely, ALSA által támogatott hangkártyával jól kell működjön. Az egyik ok amiért a HDSP-t választottam, éppen az, hogy az Ardourt az RME kártyákra gondolva tervezték. Az Ardour nagyon hasonlóan néz ki és működik, mint a Digidesign Pro Tools programja.
A program indítása mindössze annyiból áll, hogy beindítjuk a JACK-et, majd a JACK futása közben futtatjuk az Ardour-t. A legjobb ha rendszergazdaként tesszük mindezt, ugyanis csak a így érhetünk el valós idejű prioritási szintet. A jack általános indítóparancsa a következő:

```
jackd -d alsa -d hw:0
```

Így a JACK kiszolgáló induláskor az ALSA-t használja eszközként, és az alapértelmezett hangkártya az ALSA eszköze lesz. A parancssori kapcsolókról többet megtudhatunk a JACK felhasználói Dokumentációjából. A Pro Tools-hoz hasonlóan, az Ardour is sokat tud. Annyi hangsávot készíthetünk amennyit a gépünk bír, rögzíthetünk sávokat, keverhetünk belsőleg, futtathatunk bővítményeket amelyeket olyan módon kapcsolunk össze ahogy csak a hangkártyánk és mi el tudjuk képzelni. Nálam egy átlagos folyamat körülbelül 20 Ardour sávot vezérel 20 különféle kártyakimenetre, nyolc további sávot kever az Ardour-on belül, valamint további két sávot a kimenetre küld, mindezt a digitális keverőn keverve. Viszonylag egyszerű egy ilyet létrehozni. Egyszerűen ráböktem az ***Out*** gombra a ***mix*** ablakban minden egyes sáv vége felé (*5. ábra*), majd a listából kiválasztottam az ***output channel***-t. Másik lehetőségként keverhetünk mindent az Ardouron belül is, majd a kimenetet .wav fájlként küldhetjük ki. A ***mix*** ablak grafikus potméterekkel is fel van szerelve, pontosan úgy ahogy a Pro Tools-ban. Vannak továbbá bővítményei és automatái is. Az Automatizálás használata annyiból áll, hogy rákattintunk az ***arec***-re, beállítjuk a beállítani valókat, lekapcsoljuk az ***arec*** rögzítést majd az automatizálás visszajátszásához rábökünk az ***aplay***-re. Amint láthatjuk, az Ardour használata éppoly magától értetődő mint bármely profi szintű DAW rendszeré. Ami annyira azért nem magától értetődő, de nem is tart túl sokáig megtanulni. Mivel a program egyelőre béta állapotban van, a dokumentációra még várni kell, de a Pro Tools kézikönyvének elolvasása átfogó képet ad a működéséről. A hálózaton nagyon jó Hogyan-okat találhatunk (lásd a forrásokat). A cikk születésének időpontjában az Ardour 0.9beta8-1 állapotban volt. Nem árt ha ezt észben tartjuk, gyakran mentünk és nem riadunk meg ha hirtelen összeomlik. Hibajelentéseinkkel segíthetjük az 1-es verziószám elérését (lásd a Forrásokat).

Terem

A stúdiókban általában vezérlőtermet, rögzítőtermet és elszigetelt termeket találunk. Ha van elég helyünk, mindet összehozhatjuk, de ha nincs, saját vezérlőtermünkre kell szorítkoznunk. A *6. ábrán* egy tipikus stúdió alaprajzát láthatjuk.
Vannak akik drága berendezéseket vásárolnak, beteszik egy irodába és aztán profi stúdiónak nevezik, ami igen messze van az igazságtól. A legtöbb amit a felvételeink minőségéért tehetünk, nem az, hogy 5,000 dolláros mikrofont vásárolunk, nem is az ha 77-húros gitárt használunk amit maga Belzebub készített, hanem, hogy stúdiónk akusztikáján javítunk. Két dolgot kell figyelembe venni, a rögzítési teret és a hallgatói környezetet. Könnyű mellényúlni a rögzítési hely kiválasztásánál, de még könnyebb teljesen rossz helyre pakolni a hallgatói helyet.

Gyakorlat

A jó stúdió gyakorlat több egy számítógépnél és a hozzátartozó csinos nyílt forrású alkalmazásoknál. Például, ne feledjünk el a számítógép világán kívüli sávokat is felvenni. Használhatunk csöves előerősítőt, klasszikus utózengőt vagy kimenet-feldolgozót. Kísérletezzünk, ne féljünk keverni a régit az újjal, és nézzük el, ha a programunk nem egészen azt adja amit szeretnénk. A dobgép soha nem helyettesíti a dobost. Nem árt ha körültekintőek vagyunk kábelezés és az általános stúdió karbantartás terén is.
A hangkábeleink távol legyenek az áramvezetékektől, kizárólag derékszögben keresztezzék egymást, és csak akkor, ha feltétlenül szükséges, a csatlakozóinkat pedig tartsuk tisztán.

(*Megjegyzés: A téma iránt érdeklődő olvasóinknak szíves figyelmébe ajánljuk a Kiskapu Kiadó gondozásában 2002-ben megjelent* LINUX Zene és hang *című könyvet – a szerk.*)

Linux Journal 2004. május, 121. szám

Aaron Trumm 14 évesen kezdett hip-hop zenét rögzíteni. Azóta hét albumot és számtalan kiegészítő projektet adott ki. Megalapította és azóta is tulajdonosa a NQuit Records-ak, majd létrehozta Techno/Classical/ Poetry Project Third Option-t, melyben klasszikus zongorajátéka és költeményei, valamint Tamara Nicholl költeményei hallhatók.

Egyszerű USB illesztőprogram írása

Az egész szobát beragyogó, színes fényű linuxos gép, és egyszerű USB illesztőprogram írása a következő szerzeményed számára. Hogy mi köze a kettőnek egymáshoz?

Cikksorozatom eddigi részeiben áttekintettem, hogyan készíthetünk különféle típusú rendszermag-illesztőprogramokat, és ismertettem a rendszermag-illesztőprogram felületeit – TTY, soros és I2C – valamint az illesztőprogrammag működését. Itt az ideje, hogy továbblépjünk, és valódi eszközökhöz írjunk valódi illesztőprogramokat. Első célunk egy egyszerű, USB-s lámpa Linux alatti beüzemelése lesz.
Szerkesztőnk, ***Don Marti*** egy jópofa apróságot szemelt ki: ez a Delcom Engineering által készített USB Visual Signal Indicator. Don feladatomul azt tűzte ki, hogy bírjam működésre a készüléket Linux alatt.

Az eszközprotokoll

Ha egy eszközhöz illesztőprogramot szeretnénk írni, akkor először azt kell kitalálni, hogyan tudjuk vezérelni azt.
A Delcom Engineering egészen nagyvonalú, termékeihez teljes USB protokoll leírást mellékelnek, amely a hálózatról is ingyenesen letölthető. A leírásban megtaláljuk, hogy az USB vezérlőlapka milyen parancsokat fogad, és hogy hogyan kell azokat használni.
Egy Microsoft Windows DLL-t is adnak, amelynek alapján más operációs rendszerek alá is könnyebben elkészíthető a készüléket vezérlő program.
Az eszközhöz mellékelt leírás ugyanakkor csak a benne található USB-vezérlő működését tárgyalja. A különféle színűLED-ek kapcsolgatását nem ismerteti, ezt magunknak kell kitalálnunk. A készülék kinyitása után – vigyázzunk, nehogy a csavarok eltávolításakor elrepüljön a belül lévő kis rugó – vizsgáljuk meg a belsejében található nyomtatott áramkört.
Ellenállásmérő vagy bármilyen más, zárt áramkör felismerésére használható műszerrel kideríthetjük, hogy a három LED a fő vezérlőlapka 1-es kapujának első három lábára csatlakozik-e. Ha megnézzük a leírást, akkor megtudjuk, hogy az 1-es kapu lábain megjelenő szintek vezérlésére a `Major 10`, `Minor 2`, `Length 0` parancs szolgál. A parancs az USB parancscsomag legkisebb helyiértékű bájtját írja ki az 1-es kapura, amelynek alapállapotba hozatalkor az alapértelmezett szintje magas. Megvan tehát, hogy a különféle LED-ek beállításához milyen USB parancsot kell küldenünk a készüléknek.

Melyik LED melyik?

Már tudjuk, hogy a kapu lábait melyik paranccsal lehet engedélyezni, de azt még nem, hogy melyik színes LED melyik lábhoz csatlakozik. Egy egyszerű programmal ezt is kideríthetjük, amivel a kapu lábaihoz az összes lehetséges értékkombinációt előállítjuk, majd kiküldjük ezeket a készüléknek. Egy ilyen program segítségével állt elő az *1. táblázat*, amely az értékek és a LED-színek párosításait tartalmazza. Tehát, ha a kapu minden lába engedélyezve van (0x07 hexadecimális érték), akkor egyik LED sem világít. Ez összeillik a leírásban talált állítással, amely szerint alapállapotba hozatal után az 1-es kapu alapértéke magas. Valóban, nem sok értelme lenne annak, ha az első bekapcsoláskor bármelyik LED is világítana. Most már tudjuk, hogy adott LED bekapcsolásához a neki megfelelő lábat alacsony (kikapcsolt) szintre kell állítanunk. A táblázatból kiderül, hogy a kék LED-et a 2-es, a pirosat az 1-es, míg a zöldet a 0-s láb vezérli.

Rendszermag illesztőprogram

Újonnan szerzett információinkkal felvértezve gyorsan dobjunk is össze egy rendszermag illesztőprogramot!
Az már nyilvánvaló, hogy USB illesztőprogramot készítünk, de vajon milyen felületet vegyünk igénybe a felhasználói tér felé ? Blokk-eszközként elég furcsa volna egy ilyen lámpát kezelni – itt ugyanis nem adatok fájlrendszerbeli tárolásáról van szó –, ezért karakteres eszköz mellett döntünk.
Ha viszont karakteres illesztőprogramot készítünk, akkor egy fő- és egy mellékazonosítót kell fenntartanunk neki. Sőt, gondoljuk át azt is, vajon hány mellékazonosítót igényel az illesztőprogram? Mi történik, ha valaki 100 darab USB-s lámpát akar rákötni a gépére? Ha előrelátók akarunk lenni, akkor legalább 100 mellékazonosítót le kell foglalnunk, még ha ez súlyos pazarlásnak is tűnik mindazok szemében, akik egyszerre csak egy lámpát akarnak használni. Ha karakteres illesztőprogramot írunk, akkor arra is ki kell találnunk valamilyen módszert, hogy a különféle színeket külön-külön tudjuk kapcsolgatni. A karakteres illesztőprogramoknál ez hagyományosan különböző `ioctl` parancsokkal történik, de nekünk van egy jobb ötletünk is, minthogy új `ioctl` parancsokkal tűzdeljük meg a rendszermagot.

1. táblázat *A kapuértékek és a LED-színek párosításai*

Kapuérték (hexadecimális)	Kapuérték (bináris)	Bekapcsolt LED-ek
0x00	000	piros, zöld, kék
0x01	001	piros, kék
0x02	010	zöld, kék
0x03	011	kék
0x04	100	piros, zöld
0x05	101	piros
0x06	110	zöld
0x07	111	Egyik LED sem világít

Minden USB-s eszköz saját könyvtárral rendelkezik a *sysfs* fában, miért ne használjuk tehát a *sysfs*-t, és miért ne hozzunk létre három fájlt az USB eszköz könyvtárában, például ***blue(kék)***, ***red(piros)*** és ***green(zöld)*** névvel? Így bármilyen felhasználói térben futó programmal át tudjuk kapcsolni LED-es eszközünk fényeit, legyen szó C programról vagy héjparancsfájlról. Egyúttal megszabadulunk a karakteres illesztőprogram megírásának nyűgétől is, és mellékazonosítókat sem kell szerezni az eszközhöz.
USB-s illesztőprogramunkkal szemben alapvető elvárás, hogy biztosítsa az USB alrendszernek a következő öt dolgot:

- Egy mutató az illesztőprogram tulajdonos moduljára. Így az USB mag helyesen kezelni tudja az illesztőprogram modulhivatkozási számlálóját.
- Az USB illesztőprogram neve.
- Az illesztőprogram által kezelt USB azonosítók listája. Ezt egyrészt az USB mag használja annak meghatározására, hogy az adott eszközhöz melyik illesztőprogram társul, másrészt a felhasználói térben futó parancsfájlok ennek alapján töltik be az illesztőprogramot, amikor a felhasználó csatlakoztatja az eszközt a géphez.
- Egy `probe()` függvény, amelyet az USB mag hív meg, ha az USB azonosítók táblázata alapján egyező eszközt talált.
- Egy `disconnect()` függvény, amelynek meghívására az eszköz rendszerből való eltávolításakor kerül sor.

Az illesztőprogram mindezeket az adatokat az alábbi kódrészlettel teszi hozzáférhetővé:

```
static struct usb_driver led_driver = {
      .owner =          THIS_MODULE,
      .name ="usbled",
      .probe =          led_probe,
      .disconnect =     led_disconnect,
      .id_table =       id_table,
};
```

Az id_table változó megadása a következő:

```
static struct usb_device_id id_table [] = {
       { USB_DEVICE(GYÁRTÓAZONOSÍTÓ,
       ➥TERMÉKAZONOSÍTÓ) },
       { },
};
MODULE_DEVICE_TABLE (usb, id_table);
```

A `led_probe()` és a `led_disconnect()` függvényekről később lesz szó.
Az illesztőprogram moduljának betöltésekor a fenti `led_driver` adatszerkezetet be kell jegyezni az USB magnál. Ehhez mindössze az `usb_register()` függvényt kell meghívni:

```
retval = usb_register(&led_driver);
if (retval)
       err("usb_register sikertelen. "
           "Hibakód: %d", retval);
```

Amikor pedig az illesztőprogramot eltávolítjuk a rendszerből, akkor törölni kell bejegyzését az USB magnál:

```
usb_deregister(&led_driver);
```

A `led_probe()` függvény meghívására akkor kerül sor, amikor az USB mag megtalálja a lámpát. Ilyenkor csak annyit kell tennie, hogy elvégzi a készülék kezdeti értékadását és a megfelelő helyen létrehozza a három *sysfs* fájlt.
Mindezt az alábbi kód végzi el:

```
/* A helyi eszköz adatszerkezet kezdeti értékadása */
dev = kmalloc(sizeof(struct usb_led), GFP_KERNEL);
memset (dev, 0x00, sizeof (*dev));

dev->udev = usb_get_dev(udev);
usb_set_intfdata (interface, dev);

/* A három sysfs fájl létrehozása az USB
eszköz könyvtárában */
device_create_file(&interface->dev,
                  ➥&dev_attr_blue);
device_create_file(&interface->dev,
                  ➥&dev_attr_red);
device_create_file(&interface->dev,
                  ➥&dev_attr_green);

dev_info(&interface->dev,
    "Az USB LED eszköz csatlakoztatása
    ➥megtörtént.\n");
return 0;
```

A `led_disconnect()` függvény ugyanilyen egyszerű, hiszen csak fel kell szabadítanunk a lefoglalt memóriát és el kell távolítanunk a *sysfs* fájlokat:

```
dev = usb_get_intfdata (interface);
usb_set_intfdata (interface, NULL);

device_remove_file(&interface->dev,
                  ➥&dev_attr_blue);
device_remove_file(&interface->dev, &dev_attr_red);
device_remove_file(&interface->dev,
                  ➥&dev_attr_green);

usb_put_dev(dev->udev);
kfree(dev);

dev_info(&interface->dev,
```

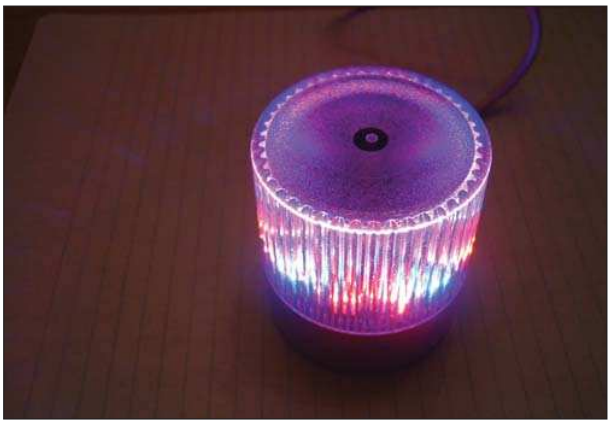
```
        "Az USB LED eszköz leválasztása
        ➥megtörtént.\n");
```

A *sysfs* fájlok olvasásával lekérdezhetjük a LED-ek aktuális állapotát, írásukkal pedig be- és kikapcsolhatjuk az egyes LED-eket. Ennek érdekében az alábbi makró mindegyik LED-hez két függvényt hoz létre, és megad továbbá egy *sysfs* eszközjellemzőfájlt:

```
#define show_set(value)
static ssize_t
show_##value(struct device *dev, char *buf)
{
   struct usb_interface *intf =
   ➥to_usb_interface(dev);
   struct usb_led *led = usb_get_intfdata(intf);

   return sprintf(buf, "%d\n", led->value);
}

static ssize_t
set_##value(struct device *dev, const char *buf,
          ➥size_t count)
{
   struct usb_interface *intf =
   ➥to_usb_interface(dev);
   struct usb_led *led = usb_get_intfdata(intf);
   int temp = simple_strtoul(buf, NULL, 10);

   led->value = temp;
   change_color(led);
   return count;
}

static DEVICE_ATTR(value, S_IWUGO | S_IRUGO,
                 ➥show_##value, set_##value);
show_set(blue);
show_set(red);
show_set(green);
```

Összesen hat függvényünk lesz, `show_blue()`, `set_blue()`, `show_red()`, `set_red()`, `show_green()` és `set_green()` névvel, továbbá három jellemző-adatszerkezet, `dev_attr_blue`, `dev_attr_red` és `dev_attr_green`. Mivel a *sysfs* fájlvisszahívók elég egyszerűek, illetve mindhárom érték esetében (blue-kék, red-piros és green-zöld) ugyanazt a műveletet kell elvégezni, makrót használtunk – így kevesebbet kell gépelni. Ez a megoldás nem szokatlan a *sysfs* fájlfüggvények körében, például a rendszermag forrásfájának I2C lapka illesztőprogramokat tartalmazó ágában, a *drivers/i2c/chips* könyvtárban is láthatunk ilyesmit. Nos, eljutottunk vére oda, hogy ha a felhasználó be szeretné kapcsolni a piros LED-et, akkor csupán egy 1-est kell írnia a *red* nevű *sysfs* fájlba. Ennek hatására meghívódik az illesztőprogram `set_red()` függvénye, amely viszont a `change_color()` függvényt hoza működésbe. A `change_color()` függvény így néz ki:

```
#define BLUE    0x04
#define RED     0x02
#define GREEN   0x01
   buffer = kmalloc(8, GFP_KERNEL);
```



A készülék a piros és a kék LED bekapcsolása után

```
color = 0x07;
if (led->blue)
   color &= ~(BLUE);
if (led->red)
   color &= ~(RED);
if (led->green)
   color &= ~(GREEN);
retval =
   usb_control_msg(led->udev,
               usb_sndctrlpipe(led->udev, 0),
               0x12,
               0xc8,
               (0x02 * 0x100) + 0x0a,
               (0x00 * 0x100) + color,
               buffer,
               8,
               2 * HZ);
kfree(buffer);
```

A függvény első lépésként a color (szín) változó összes bitjét 1-esre állítja. Ezután, ha valamelyik LED-et be kell kapcsolni, akkor átállítja a hozzá tartozó bitet. Következő lépésként USB vezérlőüzenetet küldünk a készüléknek, amiben kiírjuk rá a kívánt színbeállító értéket. Elsőre furcsának tűnhet, hogy kisméretű, mindössze nyolc bájtot tároló buffer változónkat `kmalloc` hívással hozzuk létre. Hogy miért nem adjuk meg egyszerűen a veremben, miért vesszük a nyakunkba a dinamikus helyfoglalás és felszabadítás nyűgét?
Nos, azért, mert a Linux futtatható néhány olyan géptípuson is, ahol a rendszermag-veremben létrehozott adatokat nem lehet USB eszköznek elküldeni, vagyis minden, USB-eszköznek szánt adatot dinamikusan kell létrehozni.

LED-ek működés közben

Megírtuk, lefordítjuk és betöltjük illesztőprogramunkat, és már gyönyörködhetünk is abban, ahogyan az kezelésbe veszi a gépünkhöz csatlakoztatott készüléket. Az illesztőprogramhoz kötődő USB eszközök mindegyike az illesztőprogram *sysfs* könyvtárában lelhető fel:

```
$ tree /sys/bus/usb/drivers/usbled/
/sys/bus/usb/drivers/usbled/
`-- 4-1.4:1.0 ->
```

```
../../../../devices/pci0000:00/0000:00:0d.0/usb4/
➥4-1/4-1.4/4-1.4:1.0
```

Az ebben a könyvtárban található fájl egy közvetett hivatkozás az USB eszköz valódi helyére a ***sysfs*** fában. Ha benézünk ebbe a könyvtárba, láthatjuk az illesztőprogram által a LED-ekhez létrehozott fájlokat:

```
$ tree /sys/bus/usb/drivers/usbled/4-1.4:1.0/
/sys/bus/usb/drivers/usbled/4-1.4:1.0/
|-- bAlternateSetting
|-- bInterfaceClass
|-- bInterfaceNumber
|-- bInterfaceProtocol
|-- bInterfaceSubClass
|-- bNumEndpoints
|-- blue
|-- detach_state
|-- green
|-- iInterface
|-- power
|   `-- state
`-- red
```

Ha most nullát vagy egyet írunk a blue, a green vagy a red fájlba, akkor megváltoznak a készülék fényei:

```
$ cd /sys/bus/usb/drivers/usbled/4-1.4:1.0/
$ cat green red blue
0
0
0
$ echo 1 > red
[greg@duel 4-1.4:1.0]$ echo 1 > blue
[greg@duel 4-1.4:1.0]$ cat green red blue
0
1
1
```

Ezzel a képen látható állapot áll elő.

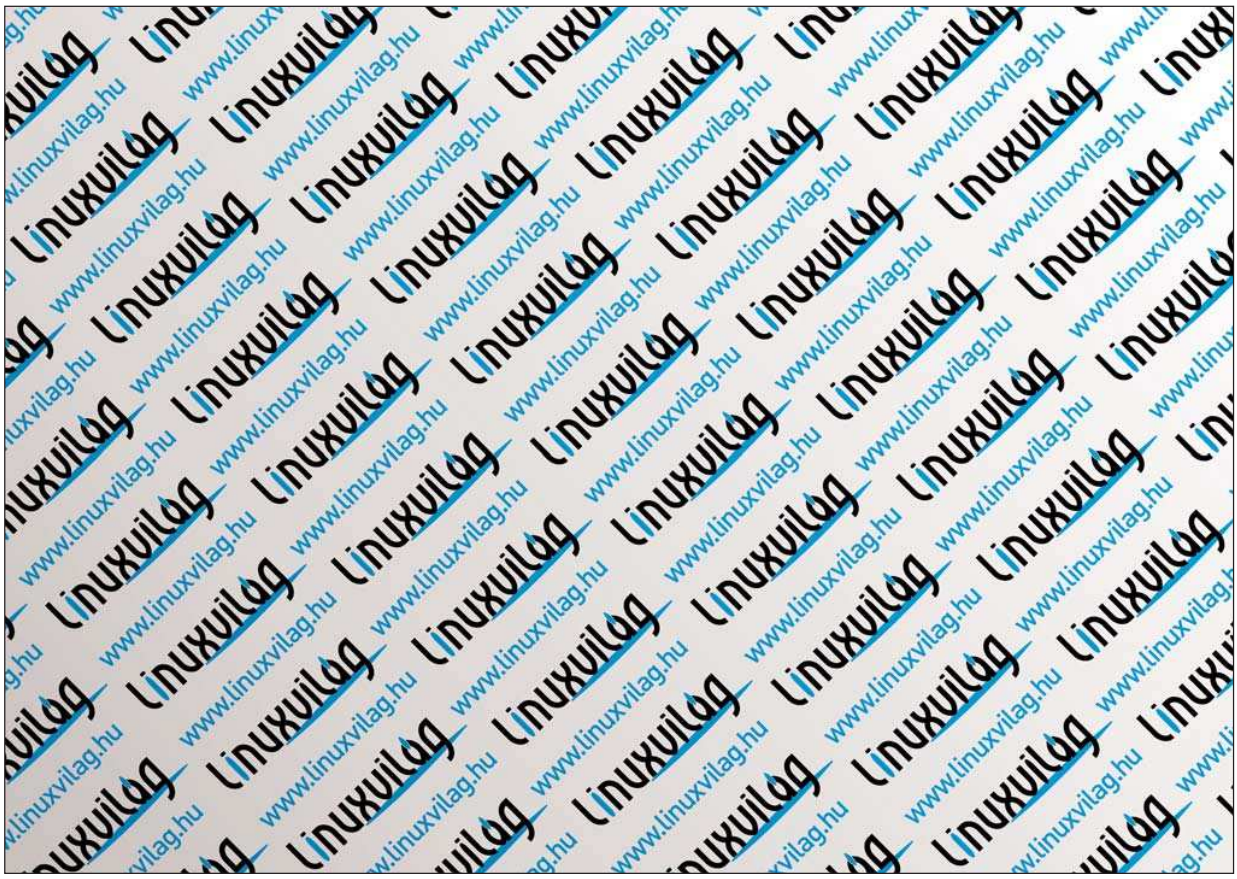
Van másik út is?

Rendben, írtunk egy egyszerű rendszermag-illesztőprogramot (amely egyébiránt megtalálható a 2.6-os rendszermagfában, a ***drivers/usb/misc/usbled.c*** fájlban.) ehhez az aprósághoz, de vajon ez a legjobb módja a készülékkel való kapcsolattartásnak? Mi van az ***usbfs*** vagy a ***libusb*** használatával? Hiszen ezekkel külön illesztőprogram nélkül is vezérelhetjük készülékünket a felhasználói térből! Cikksorozatom következő részeiben erről lesz szó. Megmutatom, hogyan működik mindez, és néhány egyszerű héjparancsfájllal hogyan lehet vezérelni az USB-s lámpát.

Ha valaki látni szeretné, hogyan kell rendszermag illesztőprogramot írni valamilyen készülékhez, akkor ne habozzon, bátran tudassa velem.

Linux Journal 2004. április, 120. szám

Greg Kroah-Hartman



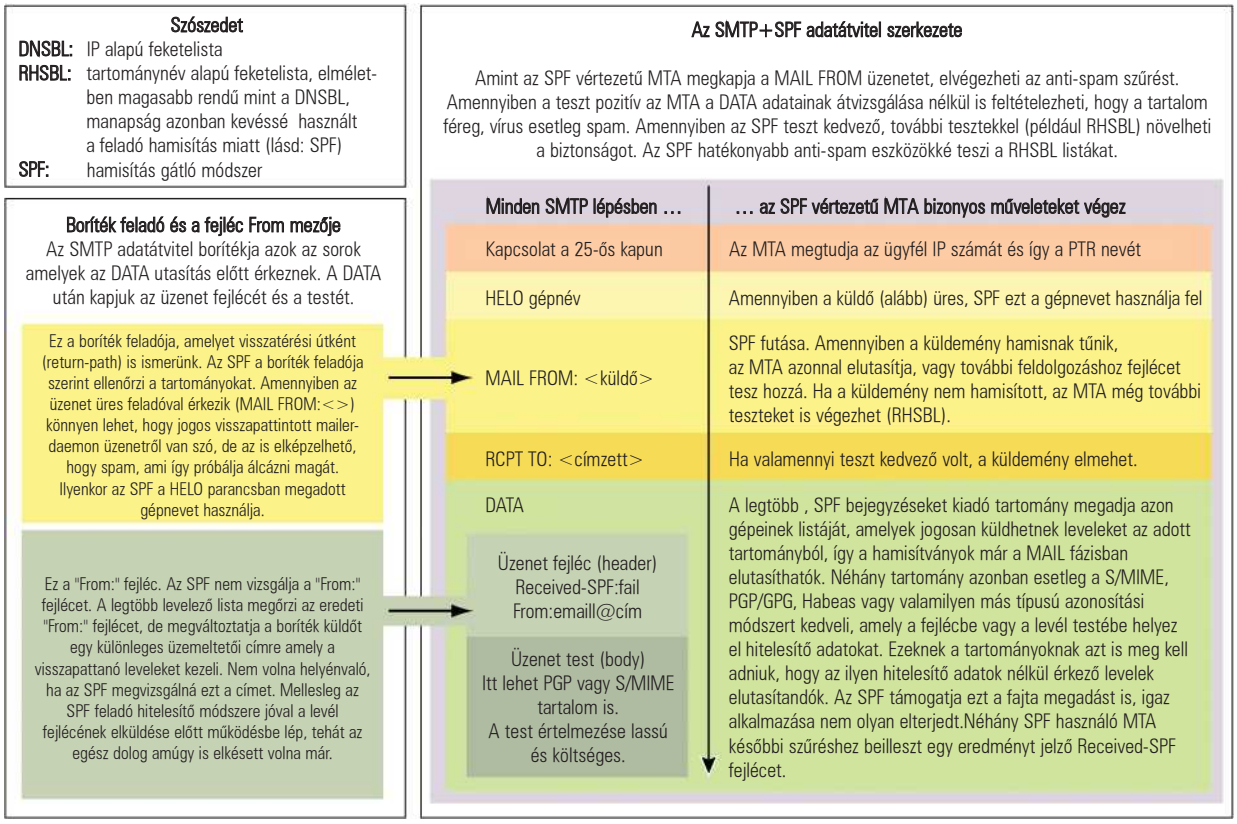
SPF bemutató

A szemétlevelek (spam) által okozott problémák kezelésében nagy segítséget nyújthat, ha időben felfedezzük a hamisításokat. Őrizzük meg e-mail címünk jó hírét egy egyszerű DNS technika segítségével.

Az SPF (Sender Policy Framework) egy új és egyre népszerűbb hamisítás-ellenes szabvány, melynek célja, hogy a különféle férgek, vírusok és szemétlevelek ne tudjanak tetszőleges e-mail címet írni az SMTP levélboríték „küldő” mezőjébe. Két fő részből áll: a tartománygazdáknak a DNS-ükön elérhetővé kell tenniük az SPF bejegyzéseket, az e-mail gazdáknak pedig SPF kézelérésre képes MTA-t kell alkalmazniuk, amelyek felhasználják ezeket az adatokat. Az SPF bejegyzések azt a gépet azonosítják ahonnan a tartomány kimenő levelei érkeznek. Bármilyen más helyről érkező levél hamisítottnak számít. E kétrészes cikk első részében az SPF alapjaival és kompromisszumaival ismerkedhetünk meg, valamint megtudhatjuk, hogy a DNS gazdáknak miképpen kell beállítaniuk az SPF bejegyzéseket. A második cikk inkább az e-mail rendszergazdák számára készül, és azt mutatja be, hogyan kapcsolhatják be az MTA SPF védelmet. A cikk 2004 január elején született, ennek megfelelően az Internet akkori állapotát tükrözi.

Férgek, Vírusok, Joe-Job módszerek és boríték feladó hamisítás

Ma saját magamtól kaptam spamet. Én alapítottam a ➲ http:\\www.pobox.com-ot, és értek az e-levelekhez. Gyorsan rá is csaptam a H billentyűre, hogy lássam a fejlécet és megvizsgáljam a Received sorokat. Ahogy gyanítottam: mint az általam kapott legtöbb spam, ez is egy szélessávú hozzáféréssel rendelkező gépről jött. Valószínűleg egy javítatlan Windows 2000-et futtató, játékra és mp3 hallgatásra használt öreg PIII-ról lehet szó, mely koszos alsóneműk halma alatt csendesen zümmög valakinek az ágya mellett. De az is lehet, hogy egy Idaho-i krumplifarmon zörög, vagy esetleg a Central Park-ra van szép kilátása. Akárhogy is, valószínűleg megfertőzte a spammelőkkel együttműködő Sobig vírus valamelyik variánsa. A gép jogos tulajdonosának fogalma sincs róla, hogy gépét megfertőzték, nem tudja, hogy gépe óránként jó néhány ezer levelet és vírust küldözget szerte azóta a rég elfelejtett nap óta, amikor az a különös csatolmány nem akar megnyílni, hiába böködte.
A spam üzenetek elrejtik valódi származási helyüket. A spammelők megrontott gépeket használnak az üzenetek szétküldésére. Hamisítják a levélfejléceket. Átírják a Received fejléceket, hogy becsapjanak. Hibás Tárgy-at fabrikálnak, hogy átejtsék a statisztikai szűrőket és meghamisítják a `>From` sorokat, eljátszva, hogy leveleiket a PayPal vagy az eBay küldte.
A spammelők gyakran a visszatérési utat is meghamisítják. Amikor a levél kézbesíthetetlen, visszapattan az eredeti feladóhoz, akinek a címe a visszatérési útban található. Itt nem a levélfejléc From: címéről van szó, hanem az SMTP boríték visszatérési címéről, az RFC2821 MAIL FROM mezőről. A spammelő program gyakran használ régi címeket, esetleg egyszerűen csak megpróbál kitalálni néhány gyakran használt nevet, vagy szótártámadást indít. Az eredmény rengeteg hibás és visszapattanó levél.
A spammerek nem szeretnék ezeket megkapni. Sokkal jobban tetszik nekik, ha valaki más kapja meg őket. Ezért véletlenszerűen kiválasztanak egy címet, vagy egyszerűen a címzettet írják ide. Ez az oka, hogy úgy nézett ki, mintha magamtól kaptam volna spamet. Néha valamelyik gyűlölt ellenségük címét írják be, készakarva behamisítják a címét így aztán elárasztják majd a visszapattanó levelek ezrei.
1997-ben valaki a levelek visszatérési címébe a ➲ http:\\www.joes.com címét hamisította, amit aztán annyi levél árasztott el, hogy tíz napra leállt – innen a művelet neve: joe-job. A Hotmail és az AOL minden nap küzd ezzel a problémával: rengeteg spammer hamisít a levélbe AOL címet, de valójában persze nem az ő kiszolgálóikat használja. Hagyományos SMTP eszközök segítségével az AOL semmit sem tehet ez ellen. Ha egy pólóra nyomnánk az AOL logót és megpróbálnánk eladni azt, egy szemvillanás alatt a nyakunkon volna az AOL összes ügyvédje. A spammelők ugyanakkor naponta hamisítják a @aol.com címet. És ezzel nekik együtt kell élniük, mert SMTP-t használnak. Az Simple Mail Transfer Protocol (SMTP) több mint húsz éve született – egy barátságosabb, kedvesebb világban. Az egész Internet Maroknyi kutatási intézményből állt. Az SMTP azóta is jól szolgál minket, de már mutatkoznak rajta az öregség jelei. Az SMTP nyílt és hiszékeny. Szabályai viszonylag kötetlenek. Bármilyen levélküldőt beilleszthetünk, és tetszőleges fejlécet komponálhatunk. Felmerül a kérdés, hogy manapság egy olyan protokoll, ami lehetővé teszi a joe-job féle cselekedeteket, vajon nem túl nyitott és hiszékeny-e. Ezért kellett kitalálni a feladó-azonosítást: Az SPF kicsit komolyabb szabályokat vezet be.



1. ábra Az SPF segítségével a levelező kiszolgáló ellenőrizni tudja, hogy a másik kiszolgáló ténylegesen azt a címet használja, amelyet a levélben állít

SPF alapú feladó azonosítás

Amikor levelünket elküldjük valamelyik tartománynak, MTA programunk egy DNS keresés (MX lekérdezés) segítségével találja meg, hogy melyik kiszolgálóra kell továbbítani a levelet. Az ilyen kiszolgálót levélcserélőnek hívjuk (mail exchanger, röviden MX). Az apró tartományoknak többnyire csak egy MX kiszolgálójuk van, a nagy tartományok általában többet tartanak fenn. A tartományra érkező levelek annak MX kiszolgálójára érkeznek be.

Lássuk akkor a nagy ötletet. Az esetek 99%-ában, amikor a tartomány elküldi a levelet, a levél a tartomány által irányított viszonylag kis számú kiszolgálóról származik. Tekintve, hogy a tartomány ezeknek a kiszolgálóknak nyújt DNS szolgáltatást, bármely, más helyről érkező levelet valószínűleg hamisnak tekinthetünk. Ezt a módszert célzott feladó-sémának nevezzük (designated sender scheme, *1. ábra*).

A célzott feladó séma fő előnye, hogy segít elhárítani a hamisítókat, ráadásul nagyon egyszerű megvalósítani. Végtére is a tartománygazdák már eleve tudják, milyen kiszolgálók küldenek leveleket az adott tartományból. Amikor azt mondom „levelet küld a tartományból" , az SMTP küldemény forrását értem alatta, azaz a boríték MAIL-FROM feladó mezőjében található kiszolgálót. Nem a From: fejlécről van itt szó. Ez nagyon fontos különbség.

A tartományból érkező levelek általában kis számú kiszolgálóról érkeznek. Ez a kis és nagy tartományokra egyaránt igaz. Az aol.com levelei az AOL kiszolgálóiról jönnek. A saját tartományom levelei természetesen a saját tartományom kiszolgálóiról. Mindenesetre biztosan nem jönnek valami koszos alsóneműkkel fedett gépről.

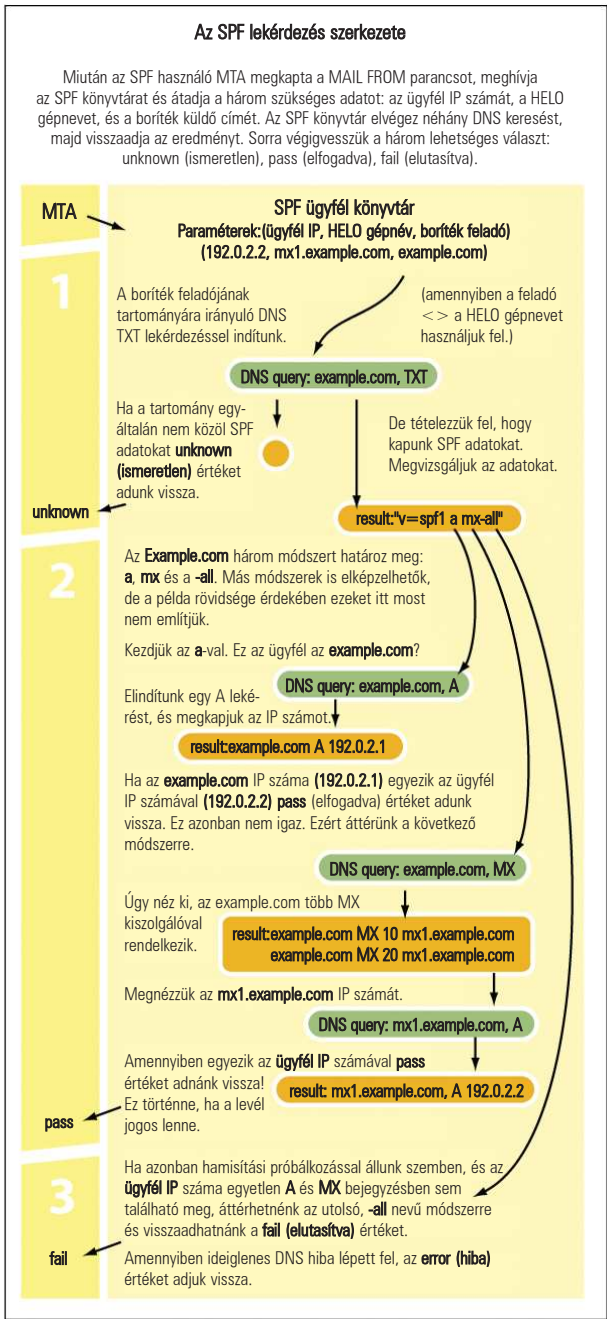
Sok ISP már ma is alkalmaz hasonló szabályokat, bár eléggé esetlegesen és gyakran kicsit hibásan. Az a gond ugyanis, hogy az egyik ISP nem igen ismeri a másik ISP belső világát, így gyakran hoz rossz döntést. Például az aol.com levelező-kiszolgálói elképzelhető, hogy az aol.net tartományba is tartozhatnak, vagy fordítva. Nem lenne sokkal egyszerűbb, ha az AOL saját maga adná meg a kiszolgálói nevét valamilyen egyszerű, rugalmas, bővíthető nyílt formátumban, amit bárki felhasználhat?

Nos, pontosan ezt teszik. Az SPF szabványos, rugalmas, bővíthető és nyílt formátum, amit bárki használhat. Az AOL pedig épp mostanában kezdte el közzétenni saját SPF bejegyzéseit. Az MTA-k értelmezhetik ezt a bejegyzést, melynek segítségével már el tudják dönteni, hogy a levél valóban a @aol.com tartományból érkezik, avagy hamisítvány. Ez az egész szigorítás teljesen önkéntes: azok a tartományok, melyek nem tesznek elérhetővé SPF bejegyzéseket, továbbra is ugyanúgy elküldhetik a leveleiket. Néhány szokatlan tartomány esetleg jobban működik, ha nem ad ki ilyen bejegyzéseket; rajtuk áll mit szeretnének. A legtöbb tartomány valószínűleg szívesen használná az SPF előnyeit.

A SPF bejegyzések felajánlásához a tartománynak mindössze egyetlen bejegyzést kell a zóna állományhoz adnia. A bejegyzés szöveges sor, akár ma is összerakhatjuk. Nézzük meg, hogyan is néz ki ez a bejegyzés.

SPF példa

Tegyük fel, hogy a ➲ http:\\www.example.com SPF bejegyzéseket akar felajánlani. Szeretné, ha a világ MTA-i elolvasnák az SPF bejegyzéseit és ennek alapján elutasítanák a ne-



2. ábra Az SPF minden bejövő levélre egyszerű DNS-alapú keresést alkalmaz

vében hamisított leveleket. Reméli, hogy az SPF segítségével csökkentheti a viaszpattanó joe-job levelek és hibás figyelmeztetések számát. Ezért zóna állományához a következő sort adja hozzá:

```
example.com.  IN TXT  "v=spf1 a mx ptr -all"
```

A `v=spf1` verzió azonosító mutatja, hogy SPF bejegyzésről van szó. A `-all` jelentése: alapértelmezés szerint minden levelet utasítsunk el. Azok a tartományok, amelyek egyáltalán nem küldenek levelet, az egyszerű `v=spf1 -all` alaknál meg is állhatnak. Ha azonban a tartomány szeretne leveleket is küldeni, meg kell adnia a módszert, amellyel eldönthető hogyan kell kinéznie egy hiteles levélnek. A módszer középre kerül, az `-all` kapcsoló elé. Az SPF lekérdezés eredménye az első egyezést mutató módszer lesz, mivel a `-all` mindennel megegyezik, a végére kell helyeznünk.

A módszereket balról jobbra értelmezzük. A `v=spf1` a `mx ptr -all` sorozat alkalmazásakor először megvizsgáljuk, hogy a kapcsolódó ügyfél megtalálható-e a tartomány A bejegyzései közt, avagy ha itt nem találjuk meg, az MX kiszolgálók listájában. Ezek után az MTA leellenőrizheti, hogy az ügyfél gépneve egyezik-e a tartománnyal. Ha egyik módszer sem ad eredményt, a `-all` értékelődik ki, és az MTA jogosan elutasíthatja a levelet.

Az A, MX, PTR és IP4 kulcsszavak a tartományok nagy többségének bőségesen elegendőek. A `spf.pobox.com/wizard.html` beállítás-varázslója segít nekünk megalkotni a saját tartományunkhoz tartozó SPF bejegyzést. Amennyiben a mi problémánk összetettebb, kipróbálhatjuk a „Advanced SPF” széljegyzet alatt leírtakat.

Egyszerű SPF

A: Az A módszer lényege, hogy az example.com IP címéről szabad az example.com-ról érkező leveleket küldeni. Ha azt akarjuk megadni, hogy a valami-más.com IP címét is engedélyezzük, a következőket írjuk: `a:valami-más.com`. Annyi A bejegyzést alkalmazhatunk amennyit csak akarunk.

MX: Az MX módszer jelentése, az example.com minden MX kiszolgálója jogosan küldhet levelet az example.com tartományból. Amennyiben a valami-más.com MX kiszolgálóit is engedélyezni szeretnénk, adjuk ki a `mx:valami-más.com` utasítást. Annyi MX bejegyzést használhatunk, amennyit csak szeretnénk.

PTR: a PTR módszer lényege, hogy amelyik a gép example.com-ra végződő PTR bejegyzéssel rendelkezik, jogosan küldhet levelet az example.com tartományból. Jó választás lenne ez például a Yahoo-nak, melynek minden kiszolgálója yahoo.com-ra végződik. Ugyanakkor nem túl jó választás a Comcast Borland szolgáltatónak. Amennyiben azt szeretnénk, hogy a valami-más.com végződésű kiszolgálók is küldhessenek leveleket az example.com tartományból, adjuk meg a `ptr:valami-más.com` kifejezést. Tetszőleges számú PTR mechanizmust használhatunk.

IP4: Tegyük fel, hogy a `192.0.2.0` számú C osztályú hálózat jogosult küldeni levelet az example.com tartományból, ilyenkor az `ip4:192.0.2.0/24` kifejezést kell bevinnünk.

Bővíthetőség

Az SPF számos beépített lehetőséggel rendelkezik. A legalapvetőbbek segítségével megadhatjuk a tartományunkból levelet küldő gépek címeit. Ez a funkció szinte minden tartomány számára elegendő, hiszen minden tartomány levelei gépek viszonylag kis csoportjától érkeznek. De ha a tartományunkról érkező leveleket valamilyen más módon külön-

Továbbfejlesztett SPF

Exists: az Exists módszer tartománynevekre ráhúzható kifejezéseket használ, ahol makrókat is használhatunk. Például, az `exists:%{ir}.%{l}._spf.example.com` ráhúzható a `2.2.0.192.ceo._spf.example.com` névre. Az SPF ügyfél A lekérdezést fog végrehajtani a ráhúzott tartománynévre, és ha visszakap valamilyen A bejegyzést (például 127.0.0.2) az Exists megadja az engedélyt. Ezt a technikát alkalmazhatjuk például, amikor szeretnénk, hogy a ceo@example.com különleges felhasználó egy adott gépről, mondjuk a 192.0.2.2-es címről leveleket tudjon küldeni, miután létrehoztuk a fenti tartománynévnek megfelelő A bejegyzést. Vannak, akik saját DNS kiszolgálót készítenek az összetett Exists lekérdezések kezelésére. Az Exists felhasználásával a határ a csillagos ég.
Include: ha leveleinket egy másik szervezet kiszolgálóin keresztül küldjük, az Include segítségével adhatjuk meg a tartományt, így a megfelelő SPF bejegyzés hívódik és helyettesítődik be. Például, ha a vanity tartomány az ISP.com levelező kiszolgálóin keresztül küldi a leveleit, használhatja az include:isp.com bejegyzést. Bármely kiszolgáló, amely leveleket küldhet az ISP.com tartomány nevében, ezt követően jogosult lesz a vanity tartományt is használni. Az include-dal több tartományt is megadhatunk.
A Redirect és az Exp: módosítók az eddig látott egyéb megoldásoktól némiképp eltérőek; kettőspont helyett egyenlőség jelet használnak. Bár a módszerek megismételhetők, egy SPF bejegyzésben csak egy módszerünk lehet. A `Redirect` módosító hasonlóképpen működik, mint az `Include`, azzal a különbséggel, hogy az eredeti kérelmet teljes egészében helyettesíti az új bejegyzés. Az `Exp` megadásával leíró szöveget adhatunk meg. Ha az MTA elutasítja a hamisítási kísérletet, az itt megadott leíró szöveg jelenik meg az eredeti feladónak visszaküldött SMTP hibaüzenetben. Lehetnek jogos felhasználóink, akik nem az általunk megadott SMTP kiszolgálókat használják, az SPF hamar kideríti kik is ők. A leíró szöveget egy URL-re is állíthatjuk, ahol a levelező ügyfél helyes beállításaival kapcsolatos további információkat helyezünk el. Az itt bemutatott módszerek részletes leírását az ➲ http:\\www.spf.pobox.com/mechanisms.html címen találjuk.

SPF és hagyományos Antispam módszerek

A DNS feketelisták vagy gátlólisták (blocklist, DNSBL): Az IPv4 tér 32 bit széles; 2^{32} körülbelül 4.2 milliárd – 4.2 milliárd homokszem megtöltene egy dömpert. Képzeljük el, hogy valamennyi homokszemet feketére vagy fehérre festjük. Az IP-alapú feketelista derék próbálkozás, de túlságosan alacsony szinten próbálja kezelni a problémát. A jó DNSBL rendszernek el kell döntenie, hogy az adott IP cím spammelő vagy sem, csakhogy helyesen kell döntenie mind a 4.2 milliárd IP cím esetében. Ne csodálkozzunk rajta, hogy a DNSBL listák jönnek-mennek. A fenntartóik idővel belefáradnak és feladják.
Jobboldali feketelisták: az RHSBL tartománynéveket használ, a DNSBL IP címeket. A tartománynevek sokkal alkalmasabbak az internetes objektumok azonosítására, azonban a RHSBL listák még nem olyan népszerűek, mint a DNSBL táblák. Vajon miért? A spam soha nem jön a spammer.net-ről. Sokkal valószínűbb, hogy a yahoo.com címet hamisítják. Ez az ahol az SPF segíthet: ha a spammelők valódi nevükről küldik a leveleket, feltartóztatásuk is egyértelművé válik.
Címellenőrzés: a MAIL fázisban leellenőrizhetjük a boríték feladóját, ha megpróbálunk levelet küldeni neki. Ha a próbaüzenet címzett ismeretlen üzenettel tér vissza, nem fogadjuk el az üzenetet. Ez azért hasznos, mert a spammelők gyakran választanak véletlenszerű címeket. Azonban ahogy a címellenőrzés egyre általánosabbá válik, a spammelők is várhatóan létező címeket fognak hamisítani-még egy érv az SPF mellett.
Aláírás alapú megoldások: a PGP/GPG és S/MIME felhasználók aláírják üzeneteiket. A fogadó a kulcs kiszolgálóról letöltött kulcsokkal ellenőrizheti az üzenet hitelességét. Olyan szerekezeteket is javasoltak, ahol maga a DNS működne a nyilvános kulcsok tárházaként. Ezek a megoldások azért hasznosak, mert a .forward állományok módosítás nélkül működhetnek tovább. Ugyanakkor hátrányuk, hogy a hitelesség ellenőrzéséhez a levélnek keresztül kell haladnia a csővezetéken, így sávszélességet és CPU időt használ el. Mindenesetre a fenti módszerek bármelyikét alkalmazó tartomány használhatja az SPF módszert annak megadására, hogy minden aláírás nélküli üzenetet el kell utasítani.
Válasz: Nyilván nem szeretnénk válaszolni egy spamre, különösen nem egy hamisított spamre. Ha az SPF megmutatja nekünk, hogy a küldő címe egyértelműen hamis volt, nyugodtan törölhetjük a levelet anélkül, hogy válaszolnánk rá.

böztetjük meg, mondjuk mindig S/MIME jellel látjuk el őket, a megszokott `a` vagy `mx` helyére az `smime` kódot kell írnunk. A küldő azonosítás egyik megközelítési módja a dedikált feladó mechanizmus (designated sender mechanisms) kialakítása (A, MX, PTR és IP4). Újfajta küldő-azonosítási módszerek is kialakulóban vannak. Az SPF bővíthető, így majd ezekkel is együtt tud működni. A jövendőbeli mechanizmusokat értelmező SPF bővítmények már helyesen fogják tudni értelmezni őket. Azok az SPF bővítmények amelyek nem értették meg az új mechanizmust, ismeretlennek fogják tekinteni, és úgy veszik, mintha a tartományunknak egyáltalán nem lenne SPF bejegyzése.

Az Altartományok és az MX kiszolgálók védelme

Manapság a spammelők tartományneveket hamisítanak. Lehet, hogy hamarosan gépneveket hamisítanak. Lehet hogy megpróbálják joe-job módszerrel célba venni a laptopunkat, kipróbálva az username@ibook.example.com címet. Ezért nem árt ha az altartományunkat is megvédjük. Kezdjük az MX kiszolgálókkal, majd térjünk át az összes A bejegyzéssel rendelkező gépre. Lássuk miért.

A visszapattanó levelek küldő mezőjében a `MAIL FROM: <>` üzenetet találjuk. A nullfeladó-cím biztosítja, hogy visszapattanó levelek ne pattanjanak vissza újra, és ne hozzanak létre ciklust. Amikor az SPF üres feladó címet lát, visszatér a `HELO` parancsban megadott címhez. Amikor a saját MTA-nk küld visszapattanó levelet, a gépnevet megadja az elküldött `HELO` parancsban. Ha ez a gépnév megtalálható az SPF rendszer listájában, az üzenet átmehet. Az SPF tehát egyúttal a `HELO` hamisítást is meggátolja.

Utazó Postás (Mailman) és továbbítási problémák

Az SPF célja: legtöbb hasznot hozni a lehető legolcsóbban. Megszigorítja a szabályokat, így a huligánok nehezebben tudnak rossz dolgokat művelni, mindeközben nem zavarja a jó embereket, akik jó dolgokat tesznek. Néhány komolyabb felhasználó, aki ki szeretné használni a SMTP laza szabályait, elképzelhető, hogy kényelmetlennek találja az SPF rendszerét. Ebben a szakaszban bemutatjuk, milyen problémákat okozhat az SPF a komoly felhasználóknak, valamint megnézzük hogyan kerülhetjük ki őket.

A legtöbb felhasználó a leveleit az ISP SMTP kiszolgálóin keresztül küldi ki. A legtöbb modern ügyfél támogatja a SASL azonosítási módszert vagy, ahol a felhasználóknak kívülről kell betárcsázniuk az ISP hálózatára, a POP-before-SMTP eljárást. Azok a felhasználók akik miden levelüket az ISP SMTP kiszolgálóin keresztül küldik, automatikusan SPF-kompatibilisek, azaz semmit sem kell tenniük.

Csakhogy néhány komoly felhasználó, aki a laptopján saját MTA-t futtat véletlenszerű IP címekről is küldhetik a leveleket, és így teljes mértékben kikerülik az ISP kiszolgálóit.

Az SPF ezen felhasználók igényeihez is alkalmazkodik: a továbbfejlesztett mechanizmus (lásd a „Továbbfejlesztett SPF” széljegyzetet) lehetőséget ad bizonyos felhasználóknak, hogy ne kelljen az ISP SMTP kiszolgálóit használniuk. Továbbra is azt tehetnek, amit akarnak.

Vannak komoly felhasználók, akik tucatnyi vagy még több címet használnak, amelyeket aztán `/etc/aliases` bejegyzésekkel vagy *.forward* állományokkal irányítanak egy helyre. A klasszikus átirányítás elvei szerint a boríték küldője változatlan marad és csak a címzett címe íródik át. Ez azonban problémát jelenthet, amikor az üzenet megérkezik a rendeltetési helyére – ugyanis továbbra is az eredeti feladó címet tartalmazza, így az SPF teszt hamisnak minősíti.

A megoldás szerencsére nem nehéz, egyszerűen csak át kell váltanunk újrapostázásra, ahol a küldő címe is átíródik. Ezt rengeteg módon megtehetjük. A nekünk legmegfelelőbb megoldást az SPF FAQ (spf.pobox.com/faq.html#forwarding) leírásából választhatjuk ki. A legtöbb végfelhasználónak nem sokszor akad dolga a továbbítással, inkább csak a komolyabb felhasználóknak van szükségük ilyen megoldásokra. Amennyiben harmadik fél által nyújtott szolgáltatást használunk az alumni, vanity tartományon vagy egyéb üzleti továbbító szolgáltató (pobox.com) gépén keresztül, elvárhatjuk, hogy meg tudják oldani nekünk az újrapostázást.

Spam feltartóztatása: Ez is a megoldás része

Az SPF elsődleges célja a hamisítás megakadályozásának további előnyei is vannak. Nem szeretnék több spamet kapni saját magamtól, és nyilvánvalóan nem szeretném, ha te kapnál olyan spamet, ami azt állítja, hogy én küldtem. A férgek és vírusok általában szintén hamisítják a küldő címét, így ezeket is megállíthatjuk az SPF segítségével. A hamisítás megakadályozása egy további előnnyel is bír. Ha a spammelők kénytelenek a saját nevüket használni, meg tudjuk állapítani mely tartományok jogosultak és melyek spammelők. Vannak, akik már most is ezt teszik: A jobboldali blokkolási listák (RHSBL) a DNS blokkolási listák (DNSBL) névkiszolgáló alapú változatai. Azok a spammelők, akik nem félnek saját tartományukat használni hamar az RHSBL listákon találják magukat, így könnyedén megállíthatjuk őket. Az SPF alapú világban az RHSBL sokkal fontosabbá és hatékonyabbá válna.

Miért használjuk az SPF-et?

A nagy tartományok, például az ISP-k, bankok, jól ismert cégek szeretik saját maguk irányítani a cégjelüket. Szinte kötelességük megvédeniük a nevüket. Az ➲ http:\\www.altavista.com éppúgy terjeszt SPF bejegyzéseket, mint az AOL és az Oxford. Minden nap új és új tartományok lépnek be az egyre bővülő körbe. A kisebb tartományok egyszerűen azért használnak SPF rendszert, mert nem szeretnének joe-job áldozatokká válni.

A fogadói oldalon az ISP-k MTA fejlesztésekbe kezdenek, és persze szívesen használják az SPF-et hiszen így kevesebb a hamisítás, ezzel együtt kevesebb spam, vírus és féreg. A sávszélesség költségük is lecsökken, hiszen az SPF segítségével még az adat elküldése előtt lecsaphatnak a spammelőre. Nincs szükség semmilyen titkosításra, ellenőrzésre vagy aláírásokra. Az SPF pénzt takarít meg.

Adaptálás

Mire ez a cikk megjelenik, az SPF támogatás már valószínűleg beépített része, vagy letölthető modulként elérhető lesz a legfrissebb SpamAssassin, Postfix, Sendmail, Exim és qmail terjesztésekben. Az üzleti antispam-terjesztők az SPF támogatása mellett döntöttek; egyikük, a Declude JunkMail, jelentette, hogy az SPF éles környezetben is sikeresen megállítja a kéretlen leveleket.

Ha minden jól megy, az SPF szabvány a közeljövőben RFC szabvánnyá válik. Több ezer tartomány, köztük néhány igazán nagy, azonban már most is terjeszti az SPF listákat. Nincs okunk várni; már ma érdemes kiadni saját SPF bejegyzéseinket.

Linux Journal 2004. április, 120. szám

Meng Weng Wong a pobox.com e-mail továbbító cég alapítója és CTO-ja, amely tizedik évfordulóját ünnepli ez évben. Jelenleg science-fiction novellasorozatán dolgozik, amely olyan bolygón játszódik, ahol Clarke híres elképzelése nyomán, nanotechnológiával valósították meg a fantasy mágiát.

SPF, MTA-k és SRS

Az előző cikkben áttekintettük, hogy DNS segítségével hogyan jelölhetjük meg eredetiként kimenő elektronikus leveleinket. Most eljött az ideje annak, hogy a bejövő levelek ellenőrzéséről is gondoskodjunk, és megvédjük felhasználóinkat a hamisított, kéretlen levelektől és a férgektől.

A Sender Policy Framework (SPF, küldő házirend-keretrendszer) a válaszútvonal hamisítása ellen segít védekezni – amit általában férgek, vírusok és levélszemét terjesztők alkalmaznak. Az SPF életre hívása két szakaszból áll. Először a rendszergazdák SPF-bejegyzéseket tesznek közzé a DNS-ben. Ezek a bejegyzések az egyes tartományok által a kimenő levelek kezelésére használt kiszolgálókat adják meg. Az SPFre képes MTA-k (***mail transport agent***, levéltovábbító ügynök) később ellenőrzik a bejegyzéseket. Ha egy levél nem az SPF-ben megadott kiszolgálóról érkezik, akkor bátran hamisnak nyilváníthatjuk.

A továbbiakban – kapcsolódva előző írásomhoz – felvázolom, hogyan ruházhatjuk fel SPF képességekkel a levélkiszolgálónkat. Szó lesz arról is, hogy az elektronikus leveleket továbbító, vagy weben előállító szolgáltatások a küldő módosításával hogyan működtethetők tovább az SPF bevezetése után is.

Az MTA bővítése SPF-képességekkel

A linuxos világ legfontosabb levéltovábbító ügynökei (***MTA***) a Sendmail, a Postfix, a Qmail és az Exim. Noha a legtöbb levélszemétszűrő megoldást kínáló cég már megoldotta az SPF támogatását (vagy legalábbis tervezi), az MTA-k esetében ez a feladat ránk vár. Az SPF-MTA együttműködést kétféle módon valósíthatjuk meg.

Aki szereti maga lefordítani a programokat, az az SPF letöltési oldalán kezdjen, itt ugyanis mindenki megtalálja a saját MTA-jához készült SPF modult és a hozzá tartozó telepítési útmutatót. Aki inkább csomagkezelővel telepíti a programokat, az nagy valószínűséggel talál olyan előre fordított MTA-változatot, amely eleve támogatja az SPF használatát. A legtöbb beépülő modulnak szüksége van a ***Mail::SPF::Query Perl*** könyvtárra.

A könyvtár a ***CPAN***-ról telepíthető a legkönnyebben, de csomag formájában is megpróbálhatjuk előkeresni. Lényegében egy egyszerű programot biztosít az SPF-lekérdezések parancssorból való futtatására. Egy egyszerű démont is tartalmaz, amely UNIX-tartományon vagy ***inet*** foglalaton keresztül kezeli az SPF-kéréseket.

A beépülő modulok nagy része alapesetben az SPF alapú ellenőrzésen megbukó üzenetek elutasítására szólítja fel az MTA-t, a többihez pedig egy `Received-SPF` fejlécet fűz. Ha kissé mértéktartóbbak akarunk lenni, akkor elutasítás helyett a `Received-SPF: fail` sorral is bővíthetjük a fejlécet. Ezt a lehetőséget a beépülő modulok leírása ismerteti bővebben.

Sendmail

A Sendmail beépülő modulok fogadására szolgáló felületét Milternek nevezzük. (Lásd az internetes forrásokat.) Az újabb Sendmail-változatok alapesetben is támogatják a Milter használatát. A Sendmail foglalat alapú felületen keresztül tartja a kapcsolatot a Milterrel. Értesíti azt a befelé irányuló SMTP-tranzakciókról, a Milter pedig megmondja a Sendmailnek, hogy mit kell tennie. A Milter démonként fut, indítása is külön történik. Az SPF weboldalon két Milter érhető el, egy Perl és egy C alapú. A Perl alapú változat kifinomultabb, ha viszont gyorsabb működést szeretnénk, akkor a C alapú változatot válasszuk. A Milter és a Sendmail együttműködéséhez néhány sorral bővíteni kell a sendmail.mc fájlt, újra kell fordítani a ***sendmail.cf***-et, majd újra kell indítani a Sendmailt. Ha inkább nem akarjuk használni a Miltert, a libspf-hez tartozik egy olyan folt is, ami lehetővé teszi az SPF közvetlen beépítését a Sendmailbe.

Postfix

A Postfix 2.1 házirend démon felülettel rendelkezik. Ennek működése nagyon hasonló a Milteréhez: a Postfix csatlakozik a démonhoz, átadja a szükséges adatokat, majd a démon egy művelettel válaszol a Postfixnek. Ha a 2.0-s sorozat újabb, fejlesztői kiadását futtatjuk, akkor ellenőrizzük, hogy 2.0.18-20040122-es vagy újabb változattal rendelkezünk-e.

A házirend démonok beállításai a ***main.cf*** és a ***master.cf*** fájlban szerepelnek. Kezelésükről a Postfix gondoskodik, indításukat és leállításukat szükség szerint elvégzi, így ezzel nem kell foglalkoznunk. A Postfix házirend démon Perlben készült, működéséhez szükség van a normál ***Mail::SPF::Query*** könyvtárra.

Exim

Az Exim 4 újdonsága az Access Control Lists (ACLs), ami egy sokoldalú, kisméretű programozási mininyelv levélszemét szűrésére és egyéb házirend jellegű döntések leírásához. Az SPF Exim alatti használatához szükséges ACL kód nagyjá-

ból 12 sort tesz ki. Telepíteni kell a ***Mail::SPF::Query*** könyvtárat és futtatni ennek SPF démonját, amely megadott foglalaton hallgatózik. Az SPF ACL csatlakozik az ***spfd***-hez és átadja neki az ügyfél IP-címét, a HELO kapcsolót és a MAIL FROM küldő címet. Ezután kap valamilyen SPF-eredményt, választ az SMTP-kiszolgálóra vonatkozóan, valamint egy Received-SPF fejlécsort. Az ***spfd***-t külön kell indítani.

Qmail

A Qmail nem rendelkezik olyan beépülő modul felülettel, mint a többi MTA. Létezik viszont olyan SPF-folt, amely az SPF-et közvetlenül a Qmailbe építi. Emellett sok Qmail-használó ***qpsmtpd*** segítségével szűri leveleit. Aki ezt a megoldást választotta, az beépülő modulként könnyen be tudja kapcsolni az SPF-et.
A ***C SPF*** könyvtár készítésében James Couzens játssza a főszerepet. A ***libspf***-hez Qmailhez és egyéb MTA-khoz készült folt egyaránt tartozik.

A beépülő modul kipróbálása

A beépülő modul telepítése és bekapcsolása után két ellenőrzést kell végrehajtani. Az első és legfontosabb annak ellenőrzése, hogy a tiszta levelek átjutnak-e. Ha nem, akkor lehetséges, hogy nem fut valamelyik szükséges program, ekkor végezzünk ismételt ellenőrzést. Ha továbbra is gondjaink vannak, állítsuk vissza a régi rendszert, és jelezzük a hibát az spf-help levelezési listán.
A második próba alkalmával a hamisított levelek elutasítását kell ellenőrizni. Ha kézzel lépünk be az SMTP-kiszolgálóra, akkor ***MAIL FROM:linuxjournal-test@altavista.com*** származással hozzunk létre egy levelet. Az ***altavista.com*** tartományt levelezésre nem használják, ezért ilyen küldőre minden esetben FAIL jelzést kell kapnunk. Többen kérdezték, hogy a próbaüzenetekben szerepeljen-e a test/teszt szó. Ezt kockázatos megtenni, mert ha megbízható ügyfelet vélnek felismerni, saját MTA-nk és SPF-ünk hamis levelet is átengedhet. Telneten keresztül tehát ne lépjünk be a helyi gépre, inkább használjuk gépünk valódi állomásnevét, és ha mód van rá, a kapcsolatot külső állomásról kezdeményezzük. Ha 550-es választ és az ➲ http:\\spf.pobox.com/why.html oldalra hivatkozó hibaüzenetet kapunk, a rendszer működik. Ha másodlagos MX-et használunk, akkor utasítsuk SPF-ügyfelünket, hogy ne tagadja meg ennek leveleit. Minderről részletesen a beépülő modul telepítési útmutatójából tájékozódhatunk.

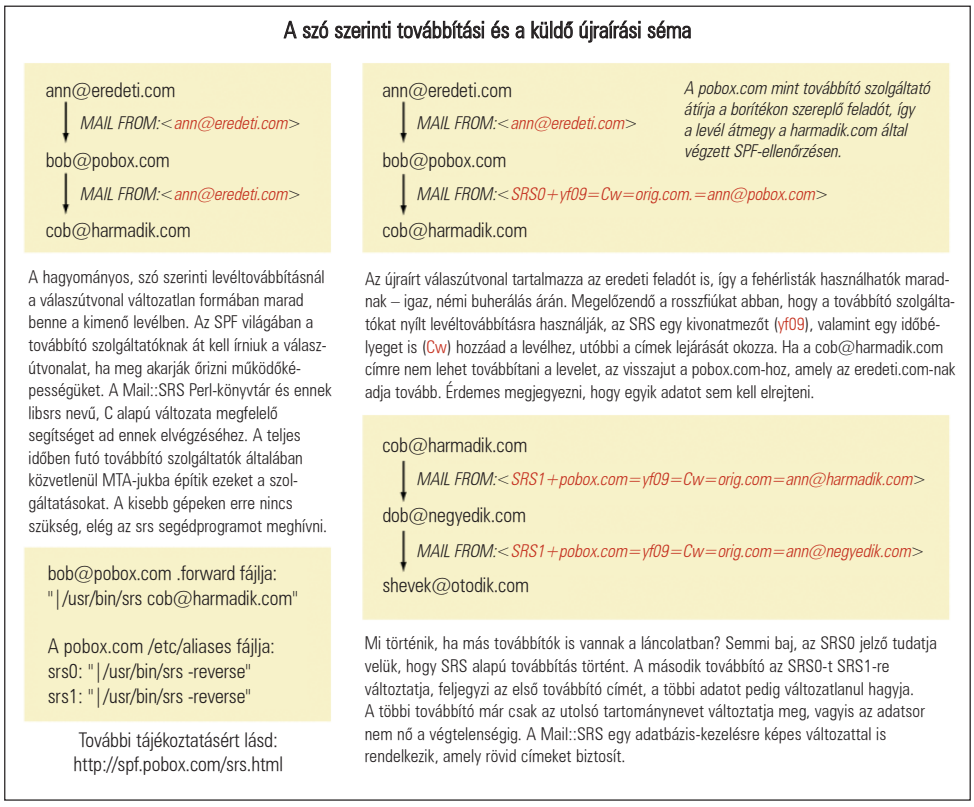
Received-SPF: a kódok jelentése

Mint azt látni fogjuk, leveleink immár egy Received-SPF fejlécet is tartalmaznak, amelyben különféle eredménykódokat találhatunk:

- `NONE`: A tartományhoz nem tettek közzé SPF-bejegyzéseket. Az MTA-nak a szokásos módon kell folytatnia munkáját.
- `PASS`: A levél nem hamis, de az sem biztos, hogy valódi. Ne feledjük, a levélszemetek küldői is közzétehetnek SPF-bejegyzéseket. A tartományt valamilyen fehérlista alapján ellenőrizni kell. Ha a küldő egy általunk megbízhatónak vélt fehérlistán szerepel, akkor a további ellenőrzéseket nyugodtan elhagyhatjuk.
- `FAIL`: A levél hamis, nyugodtan eldobhatjuk. Csekély valószínűséggel előfordulhat, hogy a levél tiszta feladótól érkezett, ám annak beállításait hibásan adták meg. Ha ez a helyzet, akkor a küldő fél hibaüzenetet kap, amelyben levélküldő ügynökének (Mail User Agent, MUA) SMTP AUTH használatára való beállítására szólítjuk fel. Az SPF tervezési elve szerint jobb egy szívélyes hibaüzenet küldése mellett hibát elkövetni, mint mély hallgatással egy levélszemetes ládába hajítani az üzeneteket.
- `SOFTFAIL`: Lehetséges, hogy az üzenet hamis, de a tartomány internetszolgáltatója már megkezdte a felhasználók rendszereinek átállítását SMTP AUTH használatára, tehát előfordulhat, hogy a levél tiszta. Az üzenetet célszerű fogadni, de érdemes további ellenőrzéseknek alávetni.
- `NEUTRAL`: A tartományban megkezdték az SPF bevezetését, alapértelmezett válaszuk `?all`. Szeretnék azt a látszatot kelteni, mintha a válasz NONE lenne, amíg a SOFTFAIL és a FAIL alapértelmezett válaszokat be nem vezetik. A nagyméretű, több millió ügyfelet kiszolgáló internetszolgáltatók lassan követik a dolgokat – ez van, nem az ő hibájuk.
- `ERROR`: Alkalmi hiba lépett fel a DNS-keresésnél. Normál esetben ilyenkor saját MTA-nknak 450-es kódú ideiglenes hibát kell jeleznie.
- `UNKNOWN`: Állandó jellegű hiba miatt az SPF-keresés félbemaradt. Lehetséges, hogy gépelési hiba van a bejegyzésben, esetleg egy másik tartományra mutat, amelyhez viszont nem tartozik SPF-bejegyzés.

Az SPF bevezetésének ára

Az elmúlt tíz évben az elektronikus levelezés egyre nagyobb szerephez jutott. Hogy pontosan mennyire függünk tőle, arról csak akkor kapunk képet, amikor egy-egy féreg elárasztja a hálózatot. Az elemzők ilyenkor rutinszerűen közlik, hány milliárd dolláros kárt okoznak a gazdaságnak a kéretlen levelek és a vírusok. Az SPF sikere is bizonyítja, az emberek nagyon várják már a változásokat.
Minden változásnak megvan azonban az ára. Ha volna valami fájdalommentes megoldás a levélszemét gondjára, már nyilván mindenki bevezette volna. A levélszemét elleni küzdelem régóta húzódik, mert a legnagyobb szakértők képtelenek megegyezni a szükséges ellenlépésekben – szerencsére a vita lezárulni látszik. Minden levélszemét elleni megoldásban közös és alapvető elem a küldő fél hitelesítése. Erre már számos modellt dolgoztak ki, ám ezek közül az SPF „megnevezett küldő” szemléletű megoldását a legkönnyebb megvalósítani. A jövő kétségkívül a titkosításé, de arra még várni kell. Az elsősegélyként alkalmazható SPF előnyei azonnal megmutatkoznak, és megvalósításával sem kell késlekednünk. De mi az SPF használatának ára? Minden megnevezett küldő alapú sémánál két dolog módosulásával kell számolni. Az első az, hogy az SPF ellehetetleníti a szó szerinti levéltovábbítást. (*1. ábra*) Vannak olyan szolgáltatások, amelyeket az emberek általában azért vesznek igénybe, mert változatlan e-mail címet biztosítanak. Ezek a UNIX ***.forward*** és a ***/etc/aliases*** fájloknál megismert módon továbbítják a leveleket. Amikor egy levél elhagyja a kiszolgálót, a borítékján szereplő válaszútvonal változatlan marad. Az SPF használatakor azonban a továbbított levelek hamisnak tűnnek. A megoldás az, hogy a továbbító szolgál-



1. ábra SPF használatakor az elektronikus levelek hagyományos továbbítása nem lehetséges

leveleket eldobják. Az újabb, haladó szellemiségű webhelyek, mint például az ***Orkut***, valami ilyet tesznek. Ha viszont a levelek fontosak, és elküldésükre a webhelyre szabályosan bejelentkezett felhasználók nevében kerül sor, valamint a webhely üzemeltetője korábban ellenőrizte a felhasználók e-mail címét, akkor a webhely SRS-t is alkalmazhat – vagyis, ha beágyazzák a felhasználó válaszcímét, akkor a visszapattanó leveleket is könnyedén el tudják juttatni rá. Jogos a kérdés: mi lesz az átállás ideje alatt? Nem lesznek leállások, amíg a továbbító szolgáltatók nagy nehezen megvalósítják az SRS támogatását? Mi lesz azokkal a szolgáltatókkal, akik túl lassan lépnek, vagy képtelenek alkalmazkodni? Van itt egy apró titok. Nagyjából sejteni lehet, kik lógnak ki a sorból. Van egy fehérlistát, amelyen a jó szándékú hamisítók szerepelnek. A listán szerepel például a ***pobox.com***, az ***acm.org***, az eBay és számos „elküldöm ezt a cikket e-mailben” jellegű szolgáltatást kínáló hírmagazin. Az eddig említett SPF-ügyfelek mindegyike képes kezelni a fehérlistát. A fehérlista formájában tehát minden ismert SPF-ügyfélnél megvan az utolsó esély a végleges elutasítás előtt. Ha az anyukánk küld nekünk egy levelet az AOL-os hozzáféréséről az ***acm.org***-os címünkre, akkor az SPF-ügyfelünk fogadni fogja a levelet, noha műszakilag hamis lesz. (Ha a levelet út közben olyan rendszer – például egy barátunk linuxos kiszolgálója – is továbbítja, amely nem szerepel a fehérlistán, akkor hozzá kell adnunk saját MTA-nk fehérlistájához ezt a gépet.) Amikor az ***acm.org*** megvalósítja az SRS támogatását, a kérdés megoldódik. Az SPF-et bírálók ellenérvként a levéltovábbítás ellehetetlenülését szokták felhozni. Az SPF közösség nem hagyta szó nélkül a kifogásokat, mindent megtett az átállás megkönnyítésének érdekében. Két megoldást is kidolgoztak, egy rövid és egy hosszú távút. Jó ha tudjuk, minden változás fájdalmas. Az SPF-re való áttérés is gondokkal jár, ám a betegségtől gyakran csak a kellemetlen injekció segítségével szabadulhatunk meg. Márpedig az elektronikus levelezés nagyon beteg. Vannak, akik szerint a levélszemét meg fogja ölni, de én ezt nem hiszem. Szerintem az SPF biztos kézzel fogja a gyógyulás útjára segíteni.

tatók újraírják a levelek válaszútvonalát. Ugyancsak így kell tenniük a leveleket a .forward és a ***/etc/aliases*** fájlok alapján továbbküldő szervezeteknek. Ez a megoldást SRS-nek (sender rewriting scheme, küldő újraírási séma) nevezzük. Az eredeti küldő címét egy újraírt, SPF-megfelelő válaszcímbe ágyazza be. Ha egy üzenet visszapattan, akkor hozzánk érkezik be, ilyenkor ki kell hámoznunk a címet, majd továbbítani a levelet az eredeti feladónak. A továbbító szolgáltatóknak akkor is meg kell tenniük mindezt, ha nem használnak SPF-et, az internetszolgáltatók ugyanis már végeznek SPF jellegű ellenőrzéseket. Az SPF csupán egy szabványos eljárást biztosít arra, amit a legtöbb helyen már most is megtesznek. Ahogy a felelősséggel vezetett szolgáltatók néhány éve megszüntették a nyílt levélküldési lehetőségeket, úgy a következő hónapok során az SPF-megfelelő továbbítást is be fogják vezetni. A ➲ http:\\www.pobox.com már alkalmazza az SRS-t, és hamarosan más továbbító szolgáltatók is követni fogják a példáját. A jó hír az, hogy az SPF-et fejlesztő közösség az MTA-khoz tartozó SRS kódot is elkészítette. Ezek a foltok ugyanonnan érhetők el, mint az SPF foltok. Az SRS elterjedése csupán idő kérdése. Az SPF azonban a webről küldött leveleket is megállíthatja. Az üdvözlőlapok küldésére használható oldalak és az „elküldöm ezt a cikket egy barátomnak” jellegű szolgáltatások általában nemcsak a From: fejlécben de, borítékon is a szolgáltatást igénybe vevő személy címét tüntetik fel. Az SPF az ilyen leveleket nem képes megkülönböztetni a hamisítottaktól.
A gondra kétféle megoldás létezik. Az első, hogy úgy döntenek, az ilyen levelek nem túl fontosak, válaszútvonalként a ***senki@példa.com***-ot adják meg, aztán a visszapattanó

Linux Journal 2003. november, 115. szám

Meng Weng Wong

A Yum használata az RPM-frissítésekhez

A legtöbb ismert biztonsági kockázat elkerülhető, ha eltávolítjuk a nem használt programokat és naprakészen tartjuk a rendszerünket. Ehhez egy új eszközt mutatok be, amelyet már három népszerű Linux rendszercsomag használ.

A rendszerünk sokszor a programhibák miatt válik sebezhetővé. Hogy az ebből eredő kockázatot csökkentsük, kiemelkedően fontos, hogy frissítsük a Linux rendszert minden alkalommal, ha egy új biztonsági javítócsomag válik elérhetővé. A kérdés súlyát jól mutatja, hogy annak idején hosszabb írást is szenteltem a témának, mind a Linux Journal oldalain („Staying Current without Going Insane", 2002 júliusi szám), mind a könyvem (Building Secure Servers With Linux, Biztonságos kiszolgálók építése a Linux segítségével) 3. fejezetében.
A Debian és SuSE rendszerek frissítése az írások megjelenése óta eltelt két év alatt sem változott jelentősen, még mindig az **apt-get** és a ***YaST*** a feladat végrehajtására leginkább használt eszköz. A Red Hat világa azonban egy új és figyelemre méltó eszközzel lett gazdagabb, amit ***Yum*** néven emlegetnek (a ***Yellow Dog Updater, Modified*** kifejezés után). Ebben a hónapban azt fogom elmondani, hogy miképpen szerezhetjük be ezt a kis programot, hogyan helyezhetjük üzembe, és mi módon használhatjuk fel a Red Hat, Fedora vagy Mandrake rendszerünk naprakészen tartásában.

A Yum legfontosabb vonásai

Ahogy a program neve is utal rá, a ***Yellow Dog Updater, Modified*** Yellow Dog Updater, vagyis Yup program leszármazottja, amely a Macintosh gépekre írt Yellow Dog Linux Rendszercsomag része. Noha a Yup kizárólag csak a Yellow Dog (Macintosh) rendszereken fut, a Yum működik Red Hat, Fedora, Mandrake és Yellow Dog Linuxon, amelyekben a Yup-ot váltotta fel. A Yum a Duke Egyetemen működő ***Linux@DUKE*** csapat projektje, mely csapat két legbefolyásosabb tagja ***Seth Vidal*** és ***Michael Stenner***.
Pár szóban összefoglalva, a Yum az RPM-et használó rendszerekben azt a funkciót tölti be, amit az **apt-get** a Debianban, vagyis egy egyszerű parancsot biztosít, amelyet programcsomagok automatikus telepítésére vagy frissítésére használhatunk, miután önműködően megtörtént az összes olyan egyéb csomag telepítése és frissítése is, amelyek szükségesek a kívánt program függőségeinek kielégítésére.
A Yum valójában két parancsból áll: a `yum` az ügyfél-oldali parancs, míg a `yum-arch` a kiszolgáló oldali utasítás, amellyel a Web- vagy FTP-kiszolgálóink Yum-tárrá való alakításához szükséges fejrész-állományokat hozhatjuk létre.
A `yum-arch` parancs ismertetése meghaladja ennek a cikknek a kereteit, de mindenképpen szükségünk lesz rá, ha nyilvános Yum-tárat szeretnénk létrehozni, saját Yum-tár kialakítását tervezzük a helyi rendszereink karbantartásához, vagy pedig egy nem hálózati Yum-tárat hozunk létre a merevlemezünkön. A `yum-arch` parancs használata egyszerű, és a ***yum-arch(8)*** man-oldal minden szükséges információval elátja a felhasználókat.
Az `apt-rpm` paranccsal szemben – amely az `apt-get` egy népszerű átültetése az RPM-alapú rendszercsomagok kezelésére – a Yum kifejezetten az RPM-formátumú csomagokhoz lett létrehozva. Mindehhez ***Michael Stenner*** még azt is hozzáteszi, hogy „a Yum tervezése során elsődleges szempont az egyszerűség és megbízhatóság volt, s egyúttal nagyobb a hangsúly a biztonságon és megbízhatóságon, mint az ügyfél oldali testreszabás esetében."

A Yum beszerzése

A Yum letöltését kínáló Weboldal (amelynek címét a Kapcsolódó címek részben találhatjuk meg) elmagyarázza, hogy az egyes Red Hat vagy Fedora változatok esetén a Yum melyik változatát érdemes letöltenünk.
Ha a Fedorát használjuk, A Yumot a Fedora Core 1 tartalmazza, a ***yum-2.0.4-2.noarch.rpm*** csomag pedig megtalálható a Fedora 1-es számú telepítőcédéjén. Amennyiben az általunk használt rendszer a Mandrake 9.2, a ***yum-2.0.1-1mdk.noarch.rpm*** csomagot a telepítőcsomag ***contrib/i586*** könyvtára tartalmazza.
A Yum teljes egészében Python nyelven íródott, ezért ahhoz, hogy a Yum RPM-csomagjait telepíteni tudjuk, szükségünk van a Fedora/Red Hat ***python, gettext, rpm-python*** és ***libxml2-python*** csomagjainak (vagy ezek Madrake-megfelelőinek) feltelepítésére. Jó eséllyel ezek a csomagok már korábban a rendszerünkre kerültek.

Hol találhatunk Yum-tárhelyeket?

Honnan tudja a Yum az RPM-csomagokat letölteni? Az erre alkalmas hely rendszerint a Világháló valamelyik távoli szöglete. Mivel a biztonsági kérdéseket igen fontosnak tartom, innentől kezdve a Yum-ot a biztonsági foltok letöltésének szemszögéből fogom vizsgálni. A cikk hátralévő részének fókuszában a hálózaton keresztül történő frissítések

1. lista A Fedora Core 1 /etc/yum.conf állománya

```
[main]
cachedir=/var/cache/yum
debuglevel=2
logfile=/var/log/yum.log
pkgpolicy=newest
distroverpkg=fedora-release
tolerant=1
exactarch=1

[base]
name=Fedora Core $releasever - $basearch - Base
baseurl=http://fedora.redhat.com/releases/fedora
-core-$releasever

[updates-released]
name=Fedora Core $releasever - $basearch -
Released Updates
baseurl=http://fedora.redhat.com/updates/release
d/fedora-core-$releasever

#[updates-testing]
#name=Fedora Core $releasever - $basearch -
Unreleased Updates
#baseurl=http://fedora.redhat.com/updates/testin
g/fedora-core-$releasever
```

2. lista A testre szabott [base] szakasz

```
[base]
name=Fedora Core $releasever - $basearch - Base
baseurl=http://mirror.eas.muohio.edu/fedora/
➥linux/core/$releasever/$basearch/os/baseurl=
➥http://fedora.redhat.com/releases/
➥fedora-core-$releasever
gpgcheck=1
failovermethod=priority
```

állnak, de a teljesség érdekében meg kell jegyeznem, hogy a Yum képes az RPM-csomagok helyi, vagy olyan, látszólag helyi fájlrendszerekből való olvasására is, mint amilyen az NFS.

Akár távoli, akár helyi kiszolgálóról van szó, az RPM-csomagoknak egy Yum-tárat kell alkotniuk. Ennek tartalmaznia kell egy ***headers*** nevű könyvtárat, amely magában foglalja a Yum számára az RPM-csomagok függőség-ellenőrzéséhez és megfeleltetéséhez szükséges információkat. Ez az oka annak, hogy nem választhatunk ki tetszésünk szerint egy régebbi Red Hat tükrözést vagy Mandrake CD-ROM-ot. Amennyiben a Fedora Core 1 vagy 1.90 változatot használjuk, lehetőségünk van a Yum-ot bármelyik Fedora tükrözésre ráirányítani. Mivel a Yum a Fedora hivatalosan is támogatott frissítő eljárása, a tükrözések már eleve Yum-tárhelyként is szolgálnak.

Hallottunk már a Fedora Legacy Project-ről? A Fedora erőfeszítéseinek ez a vonulata a korábbi Red Hat rendszercsomagokhoz kínál biztonsági frissítéseket, jelenleg a Red Hat 7.3, 8.0 és 9.0 változatokhoz. Ebből következik, hogy sok Fedora-tükrözés Red Hat frissítéseket is tartalmaz Yum-tárhely formátumban.

Ha kétségeink lennének, a különböző rendszercsomagok számára készült Yum-tárhelyek egy kicsi, de jól használható listáját találjuk a Kapcsolódó címek szakaszban. A listában szereplő hivatkozások mindegyike egy szövegblokkot is tartalmaz, amelyet közvetlenül az */etc/yum.conf* fájlunkba is átmásolhatunk – erről hamarosan részletesen is szó lesz. Ha más megoldást nem találunk, írjuk be a Google-be a rendszercsomagunk nevét és a Yum repository (Yum-tár) kifejezést, így szintén rábukkanhatunk a megfelelő helyekre.

A Yum beállításai

A Yum beállítása meglehetősen egyszerű, mindössze egyetlen fájl szerkesztéséről van szó, amelynek a neve – ahogy az megjósolható – */etc/yum.c.* Az *1. lista* azt az alapértelmezett */etc/yum.conf* fájlt mutatja be, amely a Fedora Core 1 Yum RPM csomagban juthat el a felhasználóhoz.

Amint látható, ez a fájl globális változók beállításának listájából áll, amelyet egy vagy több [server] blokk követ ([base] és [updates-released] az 1. listán). Ezek mindegyike különböző típusú RPM-csoportok beállításait határozza meg. Nem szándékozom az összes lehetséges globális illetve kiszolgáló-blokk beállítást ismertetni, erre használhatjuk a ***yum.conf(5) man***-oldalt is, de vizsgáljunk meg néhányat a legfontosabbak közül.

Az általános (global) szakaszban a `debuglevel` azt határozza meg, hogy a Yum kimenete mennyire legyen bőbeszédű. Ennek az értéke 0-tól (nincs semmilyen kimenet) a 10-es értékig (a legrészletesebb kimenet) terjedhet. Az 1. listán látható az alapértelmezett beállítás, amelynek értéke 2. Amennyire én tudom, ez a `debuglevel` értéke csak a szabványos kimenetet érinti, vagyis nincs hatással a Yum naplófájljára, amelynek a helyét a **`logfile`** értéke határozza meg. Most ezt az értéket mégis 4-re szeretném átállítani, amely értéket úgy kaptam meg, hogy egy kicsit eljátszadoztam a `yum` parancs `-d` paraméterével. Például a (`yum -d 4 yum`-parancsok) utasítással ugyanazt a hatást érhetjük el és ez a beállítás felülbírálja a `debuglevel` értékét.

Szintén az általános szakaszban találjuk a `pkgpolicy` változót, amely azt határozza meg, hogy egy adott csomag melyik változatát használja a program, amikor az több `[server]` blokkban is előfordul. A **`distroverpkg`** változó a helyi kiadás-fájl csomag (***release file package***) nevét adja meg A kiadás fájl, amely például */etc/fedora-release* vagy */etc/redhat-release* lehet, a Linux rendszercsomagunk nevét és változatszámát tartalmazza.

Minden egyes `[server]` blokk egy RPM-halmazt határoz meg. Jobban örülnék neki, ha ezeket inkább `[package-type]` blokkoknak neveznék, ugyanis nem a kiszolgáló, hanem az RPM-csoport az, ami ezeket megkülönbözteti. Egyetlen blokk is tartalmazhat több kiszolgálóra mutató URL-t. Az 1. listában a `[base]` blokk egyetlen URL-t tartalmaz, amely a fő Fedora tárhelyre, a ➲ http:\\www.fedora.redhat.com címre mutat. Azok a Fedora-tükrözések, amelyek ugyanezeket az RPM-gyűjteményeket tartalmazzák, további sorokban adhatóak

meg ugyanebben a szakaszban. A `[server]` blokk bármely sorában használható két változó: a `$releasever`, amely a használt Linux rendszercsomag verziószámát adja meg, és a `$basearch`, amely a rendszerünk processzorát sorolja a megfelelő processzorcsaládba. A processzorcsaládok itt a legáltalánosabb értelemben szerepelnek, ebben az összefüggésben például az Athlon processzorok az i386 család tagjai.
Lehet, hogy a Yum RPM-ünk által telepített */etc/yum.conf* fájl rendben működik, de az 1. listában látható alapértelmezett ➲ http://fedora.redhat.com... hivatkozásokat legalább egy tükröroldalra mutató hivatkozással kell kibővítenünk. Ezzel minimálisra csökkenthető a veszély, hogy egy kiszolgáló elérési problémái miatt a frissítési folyamatunk nem sikerül. Ne feledjük a böngészőprogramunkkal ellenőrizni, hogy a *yum.com* fájlhoz hozzáadott hivatkozások mindegyike egy olyan könyvtárra mutat, amely tartalmaz egy *headers* nevű könyvtárt is. Arra is ügyeljünk, hogy a megadott URL végére kitegyük a záró perjelet.
Az 1. listával kapcsolatos másik említésre méltó dolog, hogy hiányzik egy fontos `[server]` szakaszbeli beállítás, mégpedig a `gpgcheck`. A 2. *listán* láthatunk egy javított `[base]` blokkot, amely használja ezt a lehetőséget.
A `gpgcheck=1` beállítással azt érjük el, hogy a Yum minden letöltött RPM-csomagban ellenőrzi a GnuPG aláírást. Ahhoz, hogy ez működjön, szükségünk van a megfelelő GnuPG-kulcsok RPM-adatbázisunkba történő befoglalására. A Fedora Core 1 rendszereknél ezek a kulcsok a fedora-release csomag részeként már a rendszerünkre települtek. Ezeknek az RPM-adatbázisba történő másolásához a következő parancsot kell futtatnunk:

```
rpm --import /usr/share/doc/fedora-release-1/
➥RPM-GPG*
```

Az `rpm --import` parancs URL-t is elfogad paraméterként, így ha a Yum-forrásunk GnuPG-kulcsa a hálózaton keresztül hozzáférhető, az alábbi formát használhatjuk:

```
rpm --import
http://your.distro.homepage/GPGsignature
```

amelyben a `http://your.distro.homepage/GPGsignature` részt a megfelelő valós hivatkozással kell helyettesítenünk. Úgy tűnhet, hogy ezzel csak bonyolítjuk a dolgot, pedig érdemes használni ezt a beállítást. Az évek során számos Linux-terjesztéssel foglalkozó oldalról töltöttek le a gyanútlan felhasználók trójai programokkal vagy egyéb „finomsággal” fertőzött programcsomagokat. Ezeknek a kivédésére a legmegfelelőbb eszköz az RPM GnuPG aláírásokat támogató tulajdonságának a kihasználása.
A 2. listában eszközölt másik említésre méltó változtatás, a `failovermethod=proirity` változó megadása, amely arra utasítja a Yum-ot, hogy a listában felsorolt URL-eket sorrendben, a legfelsőtől lefelé haladva próbálja felhasználni. Az alapértelmezett viselkedés a `failovermethod=roundrobin`, amely beállítás hatására a Yum a felsorolt hivatkozások közül véletlenszerűen választ ki egyet. Én személy szerint a prioritásos működést részesítem előnyben, mert így lehetőségem van arra, hogy a közelebbi, gyorsabb tükrözéseket tegyem az elsődleges helyekké.

3. lista Frissítések ellenőrzése (a kimenet az olvashatóság érdekében enyhén rövidítve és formázva lett)

```
[root@iwazaru-fedora etc]# yum check-update
Gathering header information file(s) from
server(s)
Server: Fedora Core 1 - i386 - Base
Server: Fedora Core 1 - i386 - Released Updates
Finding updated packages
Downloading needed headers
getting /var/cache/yum/updates-released/headers/
➥coreutils-0-5.0-34.1.i386.hdr
coreutils-0-5.0-34.1.i386 100% |===============|
➥13 kB  00:01

Name                 Arch  Version   Repo
-------------------------------------------------
XFree86              i386  4.3.0-55  updates-
                                     released
XFree86-100dpi-fonts i386  4.3.0-55  updates-
                                     released
XFree86-75dpi-fonts  i386  4.3.0-55  updates-
                                     released
XFree86-Mesa-libGL   i386  4.3.0-55  updates-
                                     released
```

A Yum önműködő futtatása

Elérkeztünk a könnyű részhez, a yum parancs használatához. A Yum futtatásának két módja létezik, az egyik, hogy kézzel elindítjuk a parancssorból, a másik pedig, hogy az */etc/init.d/yum* indító parancsfájl segítségével önműködővé tesszük az indítást. Ha engedélyezett – ezt a `chkconfig --add yum` parancs futtatásával magunknak kell megtennünk –, akkor ez a parancsfájl egyszerűen ellenőrzi a futtatófájlt, amely a */var/lock/subsys/yum* fájl, s amelyet a *cron.daily* feladat, a *yum.cron* vizsgál meg. Ha a parancsfájl engedélyezett, vagyis ha a futtatófájl létezik, a cron-feladat először is futtatja a Yum parancsot a frissített Yumcsomag ellenőrzésének és telepítésének céljából, ezután pedig ellenőrzi az összes többi csomag esetleges frissítését, és szükség esetén telepíti is azokat. Mindeközben a Yum a háttérben önműködően felold minden fontos függőséget. Ha egy frissített csomag függ egy másik csomagtól, a Yum kikeresi és telepíti az a csomagot, még akkor is, ha korábban ez nem történt még meg.
A legtöbb felhasználó jó hasznát veheti ennek a hatékony parancsfájlnak. Ellenben ha a környezetünk megkívánja a minden részletre kiterjedő változás-ellenőrzést, és nem szeretnénk, hogy bármilyen új program önműködően a rendszerünkre kerüljön, kézzel kell futtatnunk a Yum-ot.

A Yum manuális indítása

Ha csak egy listát szeretnénk kapni az elérhető frissítésekről anélkül, hogy bármit is feltelepítenénk, használjuk a `yum check-update` parancsot (*3. lista*).

Egyetlen frissítés telepítéséhezés az összes olyan frissítés végrehajtásához, amire a függőségek miatt szükség van, a `yum update csomagnev` parancsot használhatjuk, pédául `yum update yum`.
Ezzel valójában csak magát a Yum csomagot frissítjük. Ha tényleg elérhető a Yum egy frissített csomagja, a rendszer a folytatásra és telepítésre buzdít. Amennyiben a Yum-ot egy parancsfájlból hívjuk, és azt szeretnénk, ha az összes feltett kérdésre önműködően `-y` legyen a válasz, használjuk a `-y` kapcsolót:

```
yum -y update yum
```

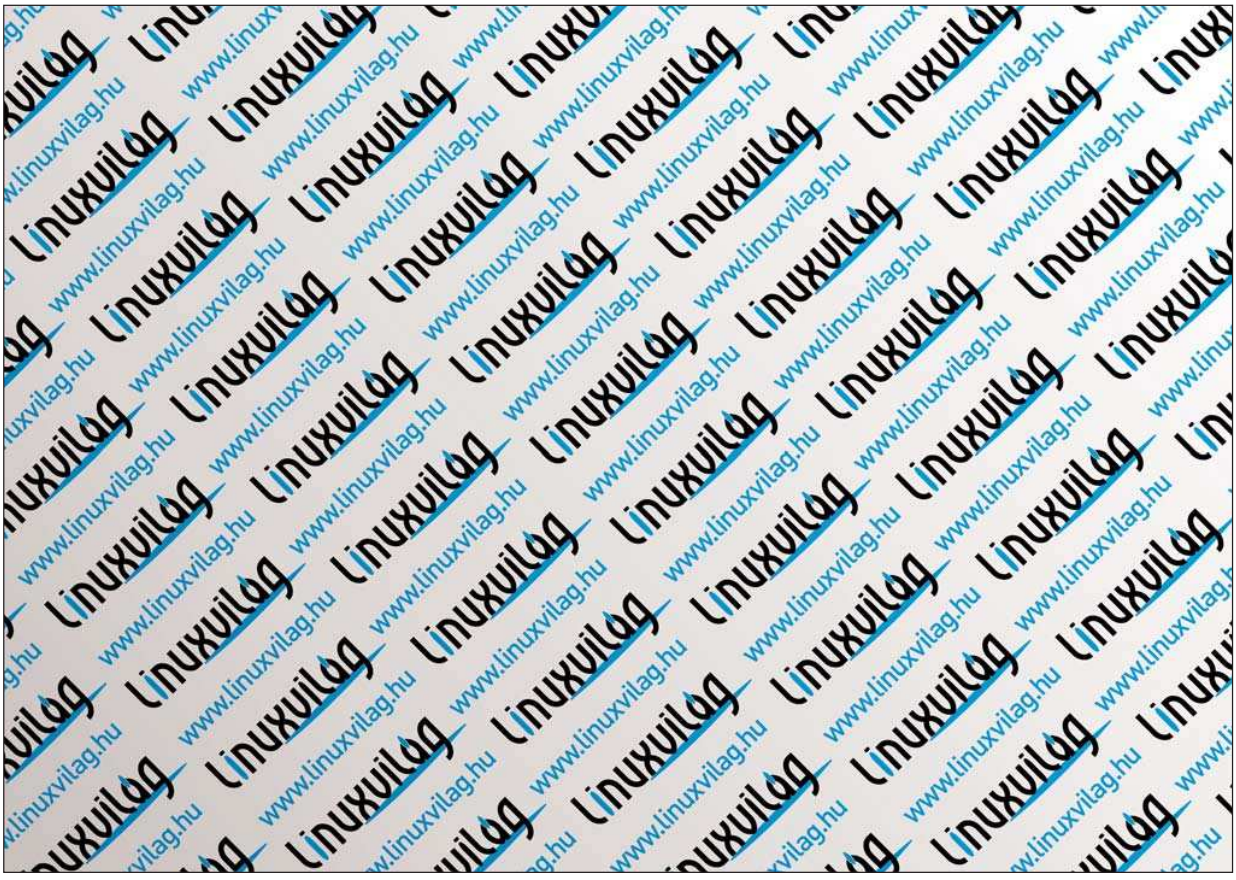
A `check-update` parancs, vagyis a Yum parancs megfelelő formájának futtatása, nem kötelező a frissítések telepítése előtt. Ha úgy tetszik, használhatjuk közvetlenül a `yum update` parancsot is – ez ugyanazokat az ellenőrzéseket végrehajtja, mint a `yum check-update`.
Az utolsó mintaként bemutatott parancsban egyetlen frissítendő csomagot, a yum-ot adtuk meg. A rendszerünkön lévő összes telepített csomag frissítésének kezdeményezéséhez mindössze el kell hagynunk az utolsó paramétert, a csomag-meghatározást: `yum update`.
Miután a Yum ellenőriz minden frissítési lehetőséget, és kiszámolja a függőségeket, bemutat egy listát azokról a frissítésekről, amelyeket le szeretne tölteni. Ha nem használtuk a `-y` lehetőséget, kérdést kapunk, hogy tényleg le szeretnénk-e tölteni és telepíteni az összes felsorolt csomagot.
A teljesség kedvéért íme egy jutalomtipp: új csomagokat is telepíthetünk a Yum segítségével – mint ahogy erre esetleg már rá is jöttünk. Bármelyik csomag esetében, amelyet tartalmaz az */etc/yum.conf* fájlban felsorolt források valamelyike, használhatjuk a `yum install csomagnev` parancsot a kérdéses csomag legfrissebb változatának telepítéséhez, kiegészítve azokkal, amelyektől függő helyzetben van. Például a *vsftpd* nevű FTP-kiszolgáló csomag telepítéséhez a következő parancsot futtatnánk: `yum install vsftpd`.

További információk

Amennyiben a Yum használatával kapcsolatban bármilyen problémánk adódna, a hálózaton keresztül bőséges leírás – köztük két kitűnő GYIK áll rendelkezésre. Erre vonatkozóan lásd a hálózaton keresztül elérhető források jegyzékét. Ha az on-line dokumentáció sem segít, rendelkezésünkre áll a Yum levelezőlistája is. Mielőtt azonban feladnánk egy kérdést, ne mulasszunk el a problémánkra vonatkozóan egy-két keresést feladni a Google keresőben. E cikk írása közben számos olyan kérdéssel találkoztam a Yum levelezőlistáján, amit a Google segítségével meg tudtam oldani.

Linux Journal 2004. június, 122. szám

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



RPM tranzakciók és visszavonás

Ha egy frissítés jól működik, néhány új képességhez, vagy jobb teljesítményhez jutunk. Ha azonban becsúszik egy hiba, az egész hétvégénk tönkremehet. Nézzük meg, hogyan térhetünk vissza a korábbi verzióhoz és tarthatjuk karban rendszerünket.

Hányszor fordult már elő, hogy valami csodásnak ígérkező új programot telepítettünk fel, csak azért, hogy szinte azonnal kiderüljön: valójában egyáltalán nem is akartuk felrakni? Sőt, a dolog még kellemetlenebb lehet, ha a feltelepítendő programhoz jó néhány egyéb csomagot kell frissítenünk további párat pedig az alapoktól felraknunk. Ha a dolgokat az eredeti állapotba szeretnénk visszaállítani, több forrásból kellene előkeresnünk a továbbfejlesztett csomagok korábbi verzióját, visszalépni ezekre a verziókra, valamint leszedni minden újonnan felrakott csomagot. Természetesen ha nem tartjuk rendszeresen nyilván milyen új csomagokat változtattunk meg és mi is volt a korábbi verzió, a dolgok még rosszabbra fordulhatnak. Hát nem lenne sokkal egyszerűbb, ha csak megnyomnánk egy gombot, vagy kiadnánk egy parancsot, és máris visszatérhetnénk a régi verzióhoz?
Néhány környezetben a frissítés előtti állapothoz való visszatérés nem csupán lehetőség, hanem egyenesen követelmény. A telekommunikációs cégek felszerelésének fejlesztésekor például a programozó és alkatrész szolgáltató cégeknek minden fejlesztést egy karbantartási ablaknak nevezett, korlátozott időszakban kell elvégezniük. Ugyanebben a karbantartási ablakban a továbbfejlesztés során bekövetkezett valamennyi változtatást vissza is kell tudniuk állítani. Ha nem sikerül a karbantartási ablakon belül befejezni a műveletet, annak súlyos pénzügyi következményei lehetnek.

Frissítés és helyreállítás, nehézkesen

Bármennyire is kívánatos lenne az RPM-ek automatikus visszaállítása egészen mostanáig nem volt erre lehetőségünk. Az igazság kedvéért hozzá kell tennünk, hogy az RPM támogatta a csomagkészletek visszafejlesztését (*downgrade*). Például, ha a *foo-1-1* RPM csomagról a *foo-1-2* verzióra fejlesztettük, az `rpm` parancsot az `--oldpackage` kapcsolóval futtatva visszaléphetnénk a korábbi verzióhoz; valahogy így:

```
# rpm -Uvh --oldpackage foo-1-1.i386.rpm
Preparing...                ################
[100%]
Upgrading...
   1:foo                    ################ [100%]
```

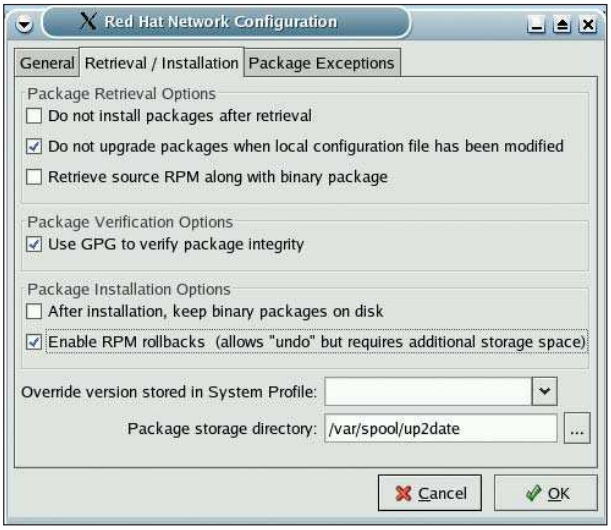
Amennyiben a *foo-1-2* fejlesztéséhez nem volt szükségünk semmilyen további RPM csomagra, az `--oldpackage` kapcsoló kiválóan működik. Mindössze a korábbi *foo-1-1* RPM csomagot kell megtalálnunk és kész is vagyunk. Azonnal megváltozik a helyzet azonban, ha a *foo-1-2* más csomagoktól is függ, amelyeket így szintén fel kellett tennünk. Ez esetben elő kell keresnünk ezeket az RPM-eket különféle helyekről – a telepítő lemezről, a terjesztésünk hibajegyzék oldaláról, egyéb RPM tárhelyről vagy a projektek weblapjairól.
Miután levadásztuk az összes szükséges RPM-et , valamennyit vissza kell fejlesztenünk majd letörölnünk a frissen telepítetteket. Ha ezt fordított sorrendben csinálnánk, letörölnénk a frissen feltelepített RPM-eket és csak aztán fejlesztenénk vissza a frissített RPM-eket, akkor az RPM hibaüzenetekkel örvendeztetne meg bennünket, miszerint ezekre a csomagokra a *foo-1-2* csomagnak szüksége van. Tömören: ez idáig több RPM csomag visszaállítása igencsak nehézkes volt, és tele hibalehetőségekkel.

Tranzakció alapú visszavonás és rendszermentés

2002 elején, ***Jeff Johnson***, az RPM aktuális gazdája, megpróbált ellenszert találni erre a visszavonási problémára és az RPM 4.0.3-as változatában bevezette a tranzakció alapú visszavonási képességet. Ez már magában hordozta az RPM csoportok automatikus visszavonásának lehetőségét. Akárcsak más új képességek, ez is kicsit elnagyoltra sikerült, és – az RPM levelezési (***rpm-list@redhat.com***) lista néhány levelétől eltekintve -, teljesen dokumentálatlan volt. Az elmúlt másfél évben a tranzakció alapú visszavonás folyamatosan fejlődött. A Red Hat 9-el érkező, jelenlegi 4.2-es RPM kiadásban például már teljesen használható ez a funkció.

A tranzakció alapú RPM visszavonás működési elve

A színfalak mögött az RPM minden feltelepített RPM csoportot egy-egy tranzakciónak tekint. Ez igaz, ha egyetlen RPM-et telepítünk fel (egy RPM tranzakciója) és akkor is, ha több RPM-et teszünk fel egyszerre. Minden ilyen tranz-



1. ábra RPM visszavonás engedélyezése az up2date programban

akció egyedi tranzakció azonosítót (TID) kap. Ahogy az RPM felkerül és frissül, bejegyzését az RPM adatbázisban megjelöljük a telepítésekor felhasznált TID azonosítóval. Ezáltal az RPM nyomon tudja követni, hogy mely RPM csomagokat mely tranzakciók során telepítettük és frissítettük. Ha az RPM visszavonja a tranzakciócsoportot, el kell tudnia érni a tranzakció végrehajtása idején a rendszeren lévő csomagokat. Ezt a problémát úgy oldja meg, hogy törlés előtt valamennyi csomagot újracsomagolja, majd eltárolja őket a ***repackage*** könyvtárban (alapértelmezés szerint, */var/spool/repackage*). Az újracsomagolt csomagok az RPM által kezelt és a törlés pillanatában a rendszeren található valamennyi állományt, a régi RPM-ek fejléceit illetve a régi RPM-el kapott parancsfájlocskákat (***scriptlets***) tartalmazzák. Felmerül a kérdés, ugyan mi köze mindennek a frissítésekhez? Ha ugyanis frissítünk egy RPM csomagot, attól még nem töröljük le – nem igaz? De bizony, letöröljük. Ugyanis az RPM frissítése két részből áll: az új csomagot feltelepítjük, a régit pedig letöröljük. Ennek megfelelően, valahányszor frissítünk egy csomagcsoportot, az RPM előbb újracsomagolja az összes frissítendő csomagot, ezután telepíti az újakat, végül pedig letörli a régieket. Amikor az RPM újracsomagolja a régi csomagokat, egyúttal meg is jelöli azokat a futó tranzakció TID azonosítójával. Ennek köszönhetően többé nem kell a netet, tárolókat, vagy mentéseket böngészni a frissített csomagokat keresendő. Mivel az újracsomagolt csomagok azokat az állományokat tárolják amelyek a frissítés idején a rendszerünkön voltak, a beállításfájlokat sem kell mentésekből visszaállítanunk. Mellékhatásként az újracsomagolt csomagban található állományok `md5` ellenőrzőösszegei valószínűleg hibásak lesznek, ugyanis az RPM ezeket nem számítja ki újra az újracsomagolt csomag létrehozása során. Szerencsére ez a tranzakciók visszaállításakor nem okoz gondot az RPM-nek, de ha közvetlenül szeretnénk használni csomagokat használnunk kell a `--nodigest` kapcsolót.

Az RPM-nek a ***repackage*** könyvtár feltöltése után a csomagok visszaállításához már csak a visszaállítás célpontjára (azaz a dátumra, amelyre vissza kell állni) van szüksége.

Az RPM a TID alapján el tudja dönteni milyen tranzakciók történtek rendszerünkön a visszaállítási időpont óta. Ez után az RPM veszi ezeket a tranzakciókat, sorba rendezi a legfrissebbtől a legkevésbé frissig, majd valamennyin végrehajtja a következő lépéseket:

- Kikeresi az összes ilyen TID jelet viselő újracsomagolt csomagot .
- Kikeresi az összes jelenleg feltelepített csomagot amelyet ilyen TID jelöl de nincsen megfelelő újracsomagolt csomagja.
- Felépíti a visszaállítási tranzakciót. Az újracsomagolt csomagok ebben mint telepítendők szerepelnek, a megfelelő újracsomagolt csomag nélküli csomagok pedig törlendő elemként.
- Lefuttatja a frissen elkészített visszaállító tranzakciót.

Ezeket a műveleteket ismételgetve a legfrissebbtől a céldátumhoz legközelebbi vagy azzal egyező dátumú csomagig, az RPM végiglépked a visszaállítási időpont óta bekövetkezett valamennyi tranzakción és eltünteti hatásukat.

Hogyan használhatjuk a tranzakció alapú RPM visszaállítást

Azt gondolhatnánk, hogy ez elég bonyolult folyamat, de a tranzakció alapú visszaállítás valójában rendkívül egyszerű. Példaként telepítsünk egyetlen RPM csomagot majd vonjuk vissza. A leglényegesebb dolog amire emlékeznünk kell, hogy valahányszor frissítünk vagy törlünk egy csomagot, soha ne felejtsük el előtte az RPM-nek kiadni a régi csomag újracsomagolását végző parancsot. Ehhez a `--repackage` kapcsolót kell használnunk:

```
# rpm -Uvh --repackage foo-1-2.noarch.rpm
Preparing...  ########################### [100%]
Repackaging...
   1:foo      ########################### [100%]
Upgrading...
   1:foo      ########################### [100%]
```

A kapcsoló hatására az RPM előbb újracsomagolja a régi csomagot, majd frissít az új változatra. Törléskor szintén meg kell adnunk a `--repackage` kapcsolót, valahogy így:

```
# rpm -e --repackage foo
```

Az RPM törléskor nem mutat semmilyen kimenetet, de ha törlés után belenézünk az újracsomagolási könyvtárba, megtalálhatjuk az újracsomagolt csomagokat.
Az RPM tranzakció visszavonására a `--rollback` kapcsolót használjuk, amelyet a visszavonás időpontja követ.
A visszavonás időpontja lehet valódi dátum vagy olyan kifejezés mint „egy órával ezelőtt” (a dátumértelmező ugyanolyan formátumot használ, mint a `cvs(1)` parancs `-D` kapcsolója). Így, ha már eltelt egy óra a *foo* telepítése óta, és úgy döntünk, még sincs rá szükségünk, a következő sort gépeljük be:

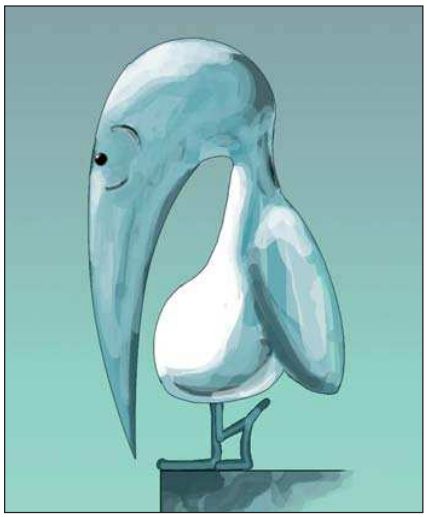
```
# rpm -Uvh --rollback ´2 hours ago´
Rollback packages (+1/-1) to
```

```
Thu Jul 31 23:26:52 2003 (0x3f29ddfc):
Preparing...  ########################### 100%]
   1:foo      ########################### [ 33%]
```

A `Rollback packages (+1/-1)` kimenet mutatja, hogy az RPM egy darab csomagot (az előző verziót) fog felvenni, és egy csomagot (a jelenleg telepített verziót) fog törölni.

Hogyan készítsünk visszavonás alapú rendszert?

A tranzakció alapú visszavonás csak annyira használható amennyire az újracsomagolt csomagjaink tárháza. A leggyorsabban úgy tudjuk elrontani a dolgot, ha valamit a `--repackage` kapcsoló nélkül frissítünk vagy törlünk. Márpedig saját tapasztalataim szerint rendkívül könnyű elfeledkezni erről a kapcsolóról. Ezért aztán ha tranzakció alapú visszavonást szeretnénk használni, érdemes úgy beállítani az RPM-et, hogy automatikusan újracsomagoljon minden törlést. Állítsuk be a `%_repackage_all_erasures` makrót a 1-re a */etc/rpm/macros* állományban. Ha a fájl nem létezik nyugodtan hozuk létre:



```
%_repackage_all_erasures 1
```

Alapértelmezés szerint az RPM nem von vissza frissen telepített csomagokat; azaz nem töröl le olyan csomagokat, amelyek a frissítésünk időpontjában még nem voltak a rendszeren. Valószínűleg nem szeretnénk, ha ez lenne az alapértelmezett működési forma, úgyhogy meg kell mondanunk az RPM-nek, hogy engedélyezze a törlést. Állítsuk a `%_unsafe_rollbacks` makrót olyan dátumra, amelyen túl már nem szeretnénk, ha az RPM visszavonáskor teljes törlést végezne. Erre a célra például nagyon jól megfelel az az időpont amikor a teljes rendszert feltelepítettük. A dátumot ***epoch*** után másodpercben kell megadni. Az ***epoch*** óta eltelt másodpercek számát a `date` parancs segítségével állíthatjuk elő:

```
date --date="2003/8/1" +%s
1059710400
```

(a dátumot átírtuk magyar sorrendre, helyes Local beállítású UNIXon ennek mennie kell. – a szerk.)

Ha például azt szeretnénk megadni az RPM-nek, hogy a 2003/8/1 előtti csomagokat ne távolítsa el teljesen (lásd a fenti példában használt dátumot), a következő sort kell a */etc/rpm/macros* fájlba írnunk:

```
%_unsafe_rollbacks 1059710400
```

Egyetlen dolog maradt még, amit esetleg érdemes átállítani, mégpedig az, hogy hol tárolja az RPM az újracsomagolt csomagokat. Tehetjük ezt például azért, hogy elegendő hellyel rendelkező partícióra helyezzük őket. Az újracsomagolási könyvtár megváltoztatásához a `%_repackage_dir` makrót kell az új csomaggyűjtő hely elérési útjára állítanunk:

```
%_repackage_dir /my_rp_repository
```

Készítettünk tehát egy rendszert, amely automatikusan újracsomagolja a törléseket (így se mi, se más nem felejti el), visszavonáskor törli az újonnan felrakott csomagokat (de nem törli az egész rendszerünket) és a nekünk megfelelő helyen tárolja az újracsomagolt állományokat.

RPM tranzakciók visszavonása up2date segítségével

Red Hat 9 alatt az RPM tranzakció alapú visszavonási módszerét használva az `up2date` program is támogatja a visszavonásokat. Amennyiben be akarjuk állítani a tranzakció alapú visszavonást mindössze az `up2date-config` programot kell lefuttatnunk, rákattintanunk a Retrieval/Installation fülre, majd kiválasztanunk az Enable RPM rollbacks jelölőnégyzetet (*1. ábra*). Magát az RPM-et az előző szakaszban leírtak szerint kell beállítanunk. Az RPM és az `up2date` program beállítása után rendszerünket az `up2date` segítségével frissítve, az RPM frissítés előtt elkészíti az RPM-ekhez tartozó újracsomagolt csomagokat.
Az ismert visszavonási időpontok listáját a következő parancs segítségével kapjuk meg:

```
up2date --list-rollbacks
```

Hatására valami ilyen listát kapunk:

```
# up2date --list-rollbacks
install time: Sun Jul 27 20:49:55 2003
tid:1059353395
                [-] goo-1.0-1.0:

install time: Tue Jul 29 20:44:25 2003
tid:1059525865
                [-] foo-1-2:
```

Ez a parancs akkor is jól jöhet, ha egyébként nem használjuk az `up2date` parancsot, ugyanis az `rpm` parancs nem képes ilyen adatok megjelenítésére.
A tranzakció visszavonását a `--undo` kapcsolóval végezzük, amely az utoljára telepített tranzakciót vonja vissza. Egyszerűen adjuk ki a következő parancsot:

```
# up2date --undo
```

Amennyiben több tranzakciót szeretnénk visszavonni, futtassuk többször a parancsot. A grafikus felületen keresztül nem tudunk visszavonást kérni.

Automatikus visszavonás folt

Az RPM általában a legkisebb fáradtság elvén dolgozik, azaz, ha egy vagy több RPM csomagot nem tud feltelepíteni, a tranzakcióban található egyéb csomagok ettől még felkerülnek. Bizonyos környezetekben ez a kívánatos viselkedés, máskor azonban sokkal jobb lenne ha automatikusan visszavonná a hibás tranzakciót. Minthogy én ilyen környezetben dolgozom (telekommunikáció), írtam egy automatikus visszavonás (***auto-rollback***) foltot. Ezzel a folttal az RPM-et úgy is be lehet állítani, hogy hibás tranzakció esetén automatikusan vonja vissza a hibás tranzakciót. Amennyiben az RPM a *%post* parancsfájlban okoz hibát, sajnos az RPM-et hátrahagyja; remélhetőleg ez hamarosan ki lesz javítva (van kedve valakinek foltozni?). Ha használni szeretnénk ezt a funkciót, a foltot (vagy a foltozott RPM-eket) a ➲ http: \\www.lee.k12.nc.us/~joden/misc/patches/ rpm címről tölthetjük le. Az automatikus visszavonás folttal rendelkező RPM rendszerünk telepítése után be kell állítanunk az RPM-et, hogy használja az automatikus visszavonás képességet. Ehhez az */etc/rpm/macros* állományt kell átszerkesztenünk, felvéve a következő makródefiníciót:

```
%_rollback_transaction_on_failure 1
```

Ezután ha majd legközelebb telepítünk illetve frissítünk egy RPM csoportot, és az egyik hibásan sikerül, akkor az RPM automatikusan visszavonja a hibás tranzakciót, kivéve persze amelyik a `%post` parancsfájl futtatása során okozott hibát.

Összefoglalás

A tranzakció alapú RPM visszavonás hatékony módszer az RPM frissítések előtti helyzet visszaállítására. Egy biztos alapot ad, melyre építve a rendszerfrissítő programok (***up2date***, ***yum*** és az ***apt-get***) automatikus visszavonási képességet nyújthatnak a felhasználónak. Ugyanakkor a tranzakció alapú visszavonást nem igényli mindenki. ***Jeff Johnson*** szavaival élve „a `--rollback` kapcsoló... tökéletes rendszeradminisztrációt igényel és inkább gépezet mint rendszabály.” A tranzakció alapú visszavonás mindent-vagy-semmit jellegű dolog. Figyelni kell rá, hogy minden törlés újracsomagolódjék, hiszen az RPM tranzakció visszavonási képessége csak annyira használható amennyire az újracsomagolt csomagkészlet. A rendszergazdának biztosítania kell némi külön tárhelyet az újracsomagolt csomagok számára. Végül, az tranzakció alapú RPM visszavonás folyamatos fejlesztés alatt áll. Amennyiben szeretnénk, hogy az RPM rendszerfrissítéseinket gyorsan vissza tudjuk vonni, valószínűleg pontosan ez az amit a doki ajánl.

Linux Journal 2004. május, 121. szám

James Olin Oden (joden@lee.k12.nc.us) programozó a Tekelec-nél. UNIX-típusú rendszereket kezelt és alattuk fejlesztett több mint egy évtizede. Ő a Tech Tracker (tt.lee.k12.nc.us) e Web alapú IT-követő rendszer alkotója is.

LSB alkalmazásokat készítése

Ne kössük Linux programunkat egyetlen terjesztéshez. Tegyük mindenhol futtathatóvá egy valamennyi nagyobb terjesztés által használt szabvány segítségével.

A Linux Szabványos Alap (***Linux Standard Base, LSB***) az alkalmazás és a futási környezet közötti felületet szabványosítja. Több terjesztés szerzett már futtatási környezetéhez ilyen bizonyítványt. Ebben a cikkben lépésről lépésre megismerkedhetünk az LSB csatolófelülettel.

Az LSB eredete

Az LSB-Projekt 1997-ben alakult, hogy megoldást találjon az egyre komolyabbá váló kompatibilitási problémákra. A különféle terjesztések eltérő élvonalbeli programverziókat használtak és azokat eltérő opciókkal fordították le. Következésképpen gyakran előfordult, hogy az egyik terjesztésen lefordított program nem futott a másik terjesztés alatt. Még ennél is nagyobb gondot jelentett, hogy időnként az alkalmazások akkor sem működtek, ha ugyanazon terjesztés másik verziójáról volt szó.
Az LSB célja eredetileg az volt, hogy egységes megvalósítási gyűjteményt készítsen, amely majd a GNU/Linux rendszer alapjául szolgálhat. A megvalósítási gyűjteményhez lejegyzett specifikációt készítettek. Sok terjesztés azonban nem fogadta kedvezően ezt az ötletet, ugyanis saját alapprogramjuk fejlesztésébe komoly erőforrásokat fektettek, amit versenyelőnyként értékeltek.
Az érdekelt felek aztán megváltoztatták az LSB Projekt alapvető céljait, hogy ezáltal az egész közösségre nézve elfogadható megoldást találjanak. Ezzel elsődleges szerepet kapott a megvalósítással szemben az írott specifikáció, így az LSB irányadó képesség/verzió párok helyett inkább viselkedési útmutatóvá vált. Az új megoldást három fő ág jellemzi: az írott specifikáció, amely a rendszer viselkedését adja meg; a formális tesztkészlet, amely a specifikációnak megfelelő megvalósítást ellenőrzi; és a minta-megvalósítás, amely a specifikációra mutat példát.

Az LSB szerkezete

Az LSB specifikáció jelenleg egy ***gLSB*** nevű általános részből és az ***archLSB*** gépfüggő részből áll. A ***gLSB*** tartalmazza a gépek (architektúrák) közös jellemzőit; nagyon igyekeztünk mindent a ***gLSB***-ben megadni. Az ***archLSB*** tartalmazza mindazon részleteket, amelyek processzortípusonként eltérőek, például a gépi utasításkészletet és a C könyvtár szimbólum verzióit.

Az LSB tartalma

Amennyire csak lehetséges, az LSB már létező szabványokra épül, ideértve a POSIX-ból kifejlődött Single UNIX Specifikációt (SUS), a ***System V Interface Definition*** (SVID) és a ***System V Application Binary Interface*** (ABI) szabványokat. Az LSB az ABI szabvány ELF definícióit és a SUS csatolófelület viselkedési irányelveit használja. A formális listában egyrészt azt határozza meg, hogy mely csatolófelületek mely könyvtáron keresztül érhetőek el, másrészt pedig a hozzájuk tartozó adatszerkezeteket és állandókat is. A jelenleg érvényben lévő könyvtárak listáját a ***„Linux Szabványos Alapkönyvtárak”*** széljegyzetben találjuk.
Az LSB ABI részén kívül az ajánlás felsorolja az alkalmazásokhoz tartozó parancsfájlokban felhasználható parancsokat is. Egyben azt is kimondja, hogy az alkalmazásnak tartania kell magát a ***Fájlrendszer Hierarchia Szabványhoz*** (FHS).
Az LSB egyik kiegészítője a csomagolási formátum. Az LSB kimondja, hogy a csomagformátumnak az RPM fájlformátum csoport tagjának kell lennie, ugyanakkor nem szabja meg, hogy a terjesztésnek kötelező RPM alapúnak lennie. Mindössze azt köti ki, hogy valamilyen módon helyesen fel kell tudni dolgoznia az RPM formátumú állományokat.
Az utolsó említésre méltó dolog a parancsértelmező neve. Az alkalmazás indításakor a parancsértelmező indul elsőként és felelős az a program többi részének valamint a megosztott könyvtáraknak a folyamat címterébe töltéséért. Hagyományosan a ***/lib/ld-linux.so.2 -t*** használják, de az LSB e helyett a ***/lib/ld-lsb.so.1*** nevet határozza meg az IA32 rendszereken. A ***/lib/ld-arch-lsb.so.1*** nevet általában más géptípusokhoz használják. Ezzel az operációs rendszernek lehetőséget adunk, hogy ha a folyamatfeldolgozás elején valami különleges dologra lenne szükség, az alkalmazásnak helyes futási környezetet biztosíthasson. A parancsértelmezőt a következő GCC paraméterrel tudjuk megváltoztatni:

```
-Wl,--dynamic-linker=/lib/ld-lsb.so.1
```

Az itt bemutatandó eszközök mindezt megteszik helyettünk.

Az LSB fordítási környezet

Az emberek rég felfedezték, hogy a kódváltoztatások sokkal egyszerűbbek és olcsóbbak ha a fejlesztési folyamat korábbi szakaszában kerülnek elő. Ezt szem előtt tartva az LSB Pro-

jekt létrehozott egy fordítási környezetet, amely segít az LSB szabványú alkalmazások készítésében. A fordítási környezetben találunk néhány tiszta fejlécállományt, ***stub*** könyvtárakat és fordító csomagolásokat (***wrappers***).
Az LSB e definíciók nagy részét adatbázisban tárolja. Az egyébként kézzel csak nagyon nehezen szerkeszthető specifikációs részleteken felül képesek vagyunk olyan tiszta fejléc fájlokat és ***stub*** könyvtárakat készíteni, amelyek kizárólag az LSB által meghatározott elemeket tartalmazzák. Az adatbázis segítségével biztosíthatjuk, hogy a változtatások és bővítések során az eszközök és a specifikációk folyamatosan szinkronban maradjanak. A telepítendő csomagok listáját a „Linux Szabványos Csomagok" széljegyzetben találjuk.
Az LSB szabványnak megfelelő alkalmazás készítésekor az első lépés, hogy a kódot az LSB fejlécekkel fordítsuk le. Ha a fordítás sikertelen, akkor valószínűleg valamilyen LSB-n kívüli elemet használunk. Ez persze nem feltétlenül végzetes, de mindenképpen érdemes rá különös figyelmet fordítani. Az LSB fejlécek az */opt/lsbdev-base/include* könyvtár alá települnek. Egy gyors teszt kedvéért adjuk meg a GCC-nek a `-I/opt/lsbdev-base/include` kapcsolót és figyeljük meg mi történik. Ezt és néhány további lépést a később ismertetésre kerülő fordító-csomagolás elvégzi helyettünk.
A kód lefordítása után a következő lépés és teszt a kód végleges alkalmazássá szerkesztése (***link***-elése). Általában ez a lépés valahogy így néz ki:

```
gcc -o app1 obj1.o obj2.o -lfoo
```

Az LSB stub könyvtárak a */opt/lsbdev-base/lib* könyvtárban találhatók, melyet a fordítónak a -L kapcsolóval tudunk megadni. Ezeket a ***stub*** könyvtárakat csak fordításhoz használjuk. Futásidőben általában a rendszer saját könyvtárait fogjuk felhasználni. Akárcsak az előbb, a később bemutatásra kerülő fordító csomagolás mindezt elvégzi helyettünk.
Az alkalmazás összeszerkesztése után az `ldd` parancs segítségével meg tudjuk nézni milyen megosztott könyvtárakat használtunk fel. Ezen a ponton az LSB-ben megadott (és a „Linux Szabványos Alap könyvtárak" széljegyzetben felsorolt) könyvtárakon kívül semmilyen más könyvtárnak nem szabad megjelennie. Ha mégis ilyesmi történik, további lépéseket kell tennünk és statikusan kell beszerkesszük őket. Általában a `-Wl,-Bstatic` és `-Wl,-Bdynamic` kapcsolókkal tudjuk megadni, ha az adott könyvtárat statikusan szeretnénk beszerkeszteni. Mostanra már biztosan ráérezttünk: a fordító csomagolás ezt is elvégzi helyettünk.
Példaképpen, nézzük meg hogy néz ki egy tipikus xpdf alkalmazás:

```
# ldd /usr/bin/xpdf
  libXpm.so.4 => /usr/X11R6/lib/libXpm.so.4
  libt1.so.1 => /usr/lib/libt1.so.1
  libfreetype.so.6 => /usr/lib/libfreetype.so.6
  libSM.so.6 => /usr/X11R6/lib/libSM.so.6
  libICE.so.6 => /usr/X11R6/lib/libICE.so.6
  libX11.so.6 => /usr/X11R6/lib/libX11.so.6
  libpaper.so.1 => /usr/lib/libpaper.so.1
  libstdc++-libc6.2-2.so.3 =>
          /usr/lib/libstdc++-libc6.2-2.so.3
  libm.so.6 => /lib/libm.so.6
  libc.so.6 => /lib/libc.so.6
  /lib/ld-linux.so.2 => /lib/ld-linux.so.2
```

És ilyen egy LSB-nek megfelelő xpdf:

```
# ldd /opt/lsb-xpdf/bin/xpdf
    libSM.so.6 => /usr/X11R6/lib/libSM.so.6
    libICE.so.6 => /usr/X11R6/lib/libICE.so.6
    libX11.so.6 => /usr/X11R6/lib/libX11.so.6
    libm.so.6 => /lib/libm.so.6
    libgcc_s.so.1 => /lib/libgcc_s.so.1
    libc.so.6 => /lib/libc.so.6
    /lib/ld-lsb.so.1 => /lib/ld-lsb.so.1
```

A nem-LSB könyvtárak meg sem jelennek az alkalmazás futásához szükséges könyvtárak közt, hiszen statikusan fordítottuk őket az alkalmazásba. Ez persze hátrányokkal is jár: az alkalmazás végrehajtható állománya nagyobb lesz, ugyanakkor kevésbé függ az operációs rendszertől.

Egyszerűsítsünk

Végül elérkeztünk az `lsbcc` és `lsbc++` fordító csomagolóhoz. A kettő valójában ugyanaz a program; egyszerűen csak más néven hívjuk meg őket, jelezve, hogy C vagy C++ módot szeretnénk. Az elképzelés szerint az `lsbcc`-t használjuk ha GCC-vel fordítunk és az `lsbc++` változatot, ha a *g++* fordítót használjuk.

Linux Szabványos Alapkönyvtárak

Az LSB 1.3 szerint a következő megosztott könyvtárak adottak az LSB-ben. Minden más könyvtárat statikusan kell az alkalmazásba szerkeszteni.

Alapkönyvtárak: ***libc***, ***libm***, ***libpthread***, ***libpam***, ***libutil***, ***libdl***, ***libcrypt***, ***libncurses*** és ***libz***.
Grafikus könyvtárak: ***libX11***, ***libXt***, ***libXext***, ***libSM***, ***libICE*** és ***libGL***.
Ahogy az LSB verziók fejlődnek, a könyvtárak listája is egyre bővül majd.

Linux Szabványos Alapcsomagok

Az LSB fejlesztői környezetet letölthetjük a Linux Standard Base oldaláról (lásd az on-line Források szakaszt); letöltéshez egyszerűen kövessük a hivatkozásokat. A következő csomagokat kell telepítenünk:

- ***lsbdev-base:*** ez tartalmazza a fejléceket és könyvtárakat.
- ***lsbdev-cc:*** tartalmazza a fordító-csomagoló eszközt.
- ***lsbdev-chroot:*** tartalmazz az alternatívaként használható, chroot alapú környezetet.
- ***lsbdev-c++:*** tartalmazza a statikus libstdc++ könyvtárat, melynek segítségével bizonyos C++ alkalmazásokat átvihetünk LSB 1.3 alá.

A csomagoló eszköz az összes megkapott kapcsolót értelmezi, majd kissé átrendezi őket. Ez után néhány további kapcsolót is beiktat, hogy a normál rendszerkönyvtárak előtt az LSB szerinti fejléceket és könyvtárakat érjük el. Az eszköz felismeri a nem-LSB könyvtárakat is, és kikényszeríti statikus fordításukat.
Minthogy az LSB-által nyújtott fejlécek és könyvtárak a keresési útvonal elejére kerültek, általában biztonságosan használhatunk nem LSB alapú könyvtárakat is. Azért persze nem árt, ha megnézzük, nem függenek-e valamitől, amit szándékosan hagytak ki a LSB fejlécek és könyvtárak közül, illetve vigyázni kell arra is, hogy statikusan szerkesszük be őket. Az `lsbcc` ilyen módon többnyire teljesen átlátszóan tud működni.

Az LSB fejlesztési környezet használata

Az LSB fejlesztési környezett csomagjainak telepítése után a példaalkalmazás átültetése (***porting***) a szokásos három lépéses műveletre egyszerűsödik, egy apró különbséggel:

```
CC=lsbcc ./configure
make
make install
```

(Mivel nem minden .configure figyeli a CC környezeti változót, érdemes elolvasni a help-jét, miképpen adhatjuk meg neki az lsbcc-t (pl: `./configure –cc=lsbcc`) – a ford.)

A ***configure*** parancsfájl az `lsbcc`-t használja a GCC helyett, és ezzel le is vezényli a az LSB környezet különféle tesztjeit, ráadásul beállítja a programhoz szükséges módosításokat és korlátozásokat. Általánosságban a teljes funkcionalitás közel azonos lesz azzal mintha GCC-t használtunk volna. Gyakorlatképpen próbáljuk meg lefuttatni a `configure` parancsfájlt mindkét módon és hasonlítsuk össze a kapott eredményt. Azzal, hogy a `configure` parancsot az `lsbcc` használatára utasítottuk, egy másik előnyhöz is jutunk: a létrehozott `make` fájlokban a CC változó automatikusan az `lsbcc` lesz, nem kell tehát figyelnünk, hogy minden egyes `make` futásakor megadtuk-e (`make CC=lsbcc`).
Az `lsbcc` parancs alapértelmezés szerint a GCC-t hívja meg a módosított paraméterekkel, de környezeti változóban megadhatunk neki más használandó fordítóprogramot is. Bármilyen fordítóval tud dolgozni, a feltétel, hogy parancssori kapcsolói sszeférjenek a GCC kapcsolóival.

Teszteszköz

Az alkalmazás elkészítése után az `lsbappchk` programmal próbálhatjuk ki, hogy valóban megfelel-e az LSB elvárásainak. A program ellenőrzi az alkalmazásban felhasznált megosztott könyvtárak listáját, továbbá megvizsgálja, valóban csak az LSB által jóváhagyott csatolófelületeket használjuk-e. Nézzünk egy példát a használatára:

```
# /opt/lsbappchk/bin/lsbappchk /bin/ls

/opt/lsbappchk/bin/lsbappchk for LSB Specification
➥1.3.3
Checking binary /bin/ls
Incorrect program interpreter: /lib/ld-linux.so.2
Header[ 1] PT_INTERP    Failed
Found wrong interpreter in .interp section:
➥/lib/ld-linux.so.2 instead of: /lib/ld-lsb.so.1
DT_NEEDED: librt.so.1 is used, but not part
➥of the LSB
Symbol clock_gettime used, but not part
➥of LSB
```

Az LSB nem követeli meg, hogy az operációs rendszer által nyújtott eszközök maguk is megfeleljenek az LSB követelményeinek. Ezért nem is igazán várjuk el, hogy a terjesztés saját `/bin/ls` programja átmenjen ezen a teszten. Egyszerűen csak kézenfekvő példa volt.
Az `lsbappchk` kimenetéből megtudhattuk, hogy az `/bin/ls` nem felel meg az LSB elvárásoknak. Az első probléma, hogy nem az LSB által megadott parancsértelmezővel, azaz a `/lib/ld-lsb.so.1`-el fordították le.
A következő gond, hogy az alkalmazás a ***librt.so.1*** könyvtárat igényli, amely nem része LSB által meghatározott könyvtárkészletnek. Végül pedig, hogy felhasználja a `clock_gettime()` függvényt, de nem statikusan szerkesztettük az alkalmazásba (a ***librt.so.1*** könyvtárban találnánk meg egyébként).
Az ilyen alkalmazások javítása általában úgy történik, hogy újrafordítjuk az `lsbcc` segítségével, amely helyesen állítaná be a parancsértelmezőt és a ***librt.a*** könytárat használná a ***librt.so*** helyett. Néha, a statikusan beszerkesztett könyvtár miatt újabb nem-LSB szimbólumok kerülnek az alkalmazásunkba, így aztán az egész folyamatot többször meg kell ismételnünk.
Néhány nagyobb alkalmazás vagy egymáshoz szorosan kapcsolódó alkalmazáscsoport esetében kívánatos lehet olyan megosztott könyvtárak létrehozása, amelyet csak ezek az alkalmazások használnak. Ez LSB alatt is engedélyezett feltéve, hogy a megosztott könyvtár az alkalmazás részeként települ valamint az alkalmazás saját adatterületén és nem a rendszer-programkönyvtár mappák valamelyikében található. Az `lsbappchk -L` kapcsolója segítségével a teszteszköznek megadhatjuk azon megosztott könyvtárak teljes elérési útját, melyek az LSB- helyesség szempontjából az alkalmazás részének tekintünk. Nézzünk meg például az LSB szerint fordított Apache Web kiszolgálót, amely három saját megosztott könyvtárat használ:

```
# /opt/lsbappchk/bin/lsbappchk
  ➥-L /opt/lsb-apache/lib/libaprutil.so.0
  ➥-L /opt/lsb-apache/lib/libexpat.so.0
  ➥-L /opt/lsb-apache/lib/libapr.so.0
  ➥/opt/lsb-apache/sbin/httpd
  /opt/lsbappchk/bin/lsbappchk for LSB
  ➥Specification 1.3.3
  Adding symbols for library /opt/
  ➥lsb-apache/lib/libaprutil.so.0
  Adding symbols for library /opt/
  ➥lsb-apache/lib/libexpat.so.0
  Adding symbols for library /opt/
  ➥lsb-apache/lib/libapr.so.0
  Checking binary /opt/lsb-apache/sbin/httpd
```

Csomagolás

Amint azt korábban említettem, az LSB kimondja, hogy a csomagnak az RPM fájlformátumot kell használnia. Ez nem jelenti azt, hogy az alkalmazásunk csomagját az RPM programmal kell elkészítenünk, bár általában valóban ez a legpraktikusabb megoldás, főleg akkor, ha már egyébként is használjuk. Másik lehetőség lehet például, hogy a csomagot Debian formátumban készítjük el, majd az `alien` segítségével alakítjuk RPM csomaggá. Akár más eszközöket is használhatunk az RPM formátumú csomag létrehozásához. Van ugyan egy kezdetleges `mkpkg` nevű programunk az RPM fájlformátum készítéséhez, de minden valószínűség szerint a legelszántabb hackereken kívül más számára csak valamilyen segédprogramon keresztül használható. Alkalmazáskészletünkben most elkészítjük az alkalmazást, majd egy ideiglenes ***root*** könyvtárba telepítve meghívjuk az RPM-t a csomag összecsomagolásához és telepítéséhez. Kicsit nehézkesnek tűnhet ugyan, de viszonylag probléma mentesen működik és valamivel egyöntetűbb eredményt biztosít a vadonban fellelhető különféle RPM változatoknál. Nézzük meg, hogy néz ki az `xpaint` alkalmazás példa definíciós állománya:

```
Summary: An X Window System paint program
Summary: XPaint
Name: lsb-xpaint
Version: 2.6.2
Release: 3
Vendor: Free Standards Group
License: MIT
Group: Appbat/graphics
Buildroot: /usr/src/appbat/pkgroot/lsb-xpaint
AutoReqProv: no
PreReq: lsb >= 1.3

%description
LSB szabványnak megfelelő xpaint verzió.
Az XPaint X Window System alapú, színes
képszerkesztő program.
Az Xpaint elsősorban a célból került az LSB
Alkalmazás készletbe, hogy rajta keresztül
bemutathassuk az X11 könyvtárak
használatát.


%pre

%install

%post

%preun

%postun

%clean

%files

%attr ( - bin bin ) /opt/lsb-xpaint
```

Ezen és az alkalmazáskészletben megtalálható alkalmazások fordításához és csomagolásához használt forráskódot az LSB projekt CVS fájában találjuk meg.

És ez tényleg működik?

Igen, tényleg működik, bár az igazat megvallva, időnként még mindig beleszaladunk néhány alkalmazásba amelyek nem mindig követik a tiszta, hordozható kód szabályait. A LSB ellenőrző készlet részeként létrehoztuk egy, az itt leírt eszközökkel lefordított alkalmazás- sorozatot. Eme alkalmazások közt megtaláljuk az Apache, Samba, Lynx, Python, xpdf és groff programokat. Megpróbáltunk olyan valódi alkalmazásokat összeválogatni amelyek a lehető legnagyobb területet lefedik az LSB csatolófelület készletéből.

Mi a helyzet a C++ alkalmazásokkal?

Az LSB 1.3 nem támogatja a C++ nyelvet, így a szükséges könyvtár statikus beszerkesztésének szabálya lép érvénybe. Ennek elkerülésére az LSB 2.0 változatába már beépítettük a C++ támogatást. Közreadjuk az ***lsbdev-c++*** csomagot, amely tartalmazza a `lsbcc` segítségével beállított és lefordított ***libstdc++*** könyvtárat. Ezzel és a GCC 3.2 változatával úgy tűnik jó eredményeket lehet elérni. Más fordítókat és különféle C és C++ könyvtárakat is kipróbáltunk, de az alkalmazás jellegétől függően különféle problémákba ütköztünk.

Célok és elképzelések

Az LSB fejlesztésénél általános szinten további könyvtárakat fogunk felvenni a specifikációba, feltéve, hogy közmegegyezés alakul ki szükségességük tekintetében és elértek egy adott stabilitási szintet. Ezáltal eltüntethetjük a terjesztés által kiadott és az LSB-nek megfelelő alkalmazások készítése közötti rést.

Az LSB fejlesztői környezet esetében tovább javítjuk majd az eszközök használhatóságát és átlátszóságát. A fejlesztői környezet karbantartása igen aktív, és az eszközöket használók visszajelzéseit nagyra értékeljük. Az LSB 2.0 változatban megjelenő C++ segítségével, a fejlesztési környezet lecserélheti a manapság használt ***lsbdev-c++*** csomagot a ***C++ stub*** könyvtárra, amely így az alap LSB fejlesztői csomagba vándorol.

Jelenleg jó néhány kapcsolót be kell állítanunk az ***rpmrc*** vagy az ***rpmmacros*** állományban ha az RPM-et LSB szabványú csomagok előállítására szeretnénk rábírni. Reméljük idővel elő tudunk majd állítani egy LSB módot az ***rpmbuild*** programhoz, amely mindezt automatikusan megoldaná.

Linux Journal 2004. május, 121. szám

Stuart R. Anderson elkövette azt a hibát, hogy meghallották amikor kijelentett, hogy „tudom hogyan kellene ezt megoldani", és így azóta az LSB Written Specification vezetője lett. Amikor nem az LSB-n dolgozik, Stuart szorgosan ügyködik Dél Carolina felvilágosításán a nyílt forrású témákban, egyesével térítve meg a cégeket. Az anderson@freestandards.org címen érhető el.

Slash

Az egyik legrégebbi és legtöbb szolgáltatást nyújtó webközösség-rendszer egyben az egyik legnehezebben telepíthető is. Reuven ezúttal mod_perl-lel és egyéb előfeltételekkel kapcsolatos tanácsokkal segít Slash webmesterré válni.

A webnapló, vagy más néven blog népszerűsége néhány éve megállíthatatlanul nő, és semmi jele annak, hogy kicsit is alábbhagyna. Noha sok webnapló-író továbbra is független szolgáltatások – például a LiveJournal vagy a Blogger – segítségével jegyzi le gondolatait, egyre könnyebbé válik saját kiszolgálón webnaplót működtetni. Az elmúlt néhány hónapban számos programcsomagot megnéztünk, amelyek biztosítják ezt a lehetőséget. Ezek közé tartozik a COREBlog, amely egy Zope termék, és a Blosxom, egy Perl-ben íródott CGI programcsoport.
A múlt hónapban, amikor a XOOPS-ot vizsgáltuk, egy kicsit másmilyen weblog-rendszert láttunk. A XOOPS, akárcsak az unokatestvérei, a PHPNuke és a PostNuke, a csoportok és tagok felügyelete mellett lehetővé teszi a felhasználóknak tartalom-kezelés és online közösség létrehozását is. A XOOPS segítségével egyszerűen lehet a rendszer minden felhasználójának saját, egyéni webnaplót biztosítani. Bármennyire is népszerű a XOOPS, a közösségi programok ősatyja vitathatatlanul a Slash, amely a közkedvelt ➲ http:\\www.slashdot.org és a ➲ http:\\www.use.perl.org webhelyek motorja. A Slash elsősorban hírek és hozzászólások közzétételére szolgál, de nagyon hatékony webnapló szolgáltatása is van, amely a rendszer minden felhasználója számára elérhető. Néhányan erre azt mondanák, hogy a Slashdot maga egy közösség által szerkesztett webnapló, és ez a nézet számomra is egész meggyőző. A legjobb az, hogy ez a webnapló szolgáltatás együttműködik az oldal többi elemével, így ha egy másik felhasználótól különösen éles elméjű vagy ízléstelen hozzászólást olvasunk, rögtön megnézhetjük a naplóját, hogy többet tudhassunk meg róla. Ebben a hónapban egy egyszerű Slash webhely telepítését és beállítását nézzük meg, amely beépített támogatást biztosít a felhasználóknak és a webnaplóknak (vagy ahogy a Slash-zsargon mondja: „journal”-nak). A következő hónapban közelebbről tanulmányozzuk a Slash webnapló lehetőségét, valamint azt, hogyan lehet beállítani és változtatni, hogy megfeleljen egyedi igényeinknek.

Háttér

A Slash terjesztésének és megvitatásának a fő helye a Slashcode. A cikk keletkezésekor, 2004. áprilisának elején, a legutolsó hozzászólás címe: „Barátságos Slash telepítés”, amelyben a szerző azt kérdezi, létezik-e a Slash telepítésének világos, egyszerű módja. Sajnos, úgy tűnik, a válasz nem – és ennek számos oka van:

- A Slash leírása és használati utasítása az emberek többségét a hivatalosan kiadott ***.tar.gz*** csomagokhoz vezeti, amelyek több, mint két éve elavultak.
- A legújabb CVS (***Concurrent Versions System***) változat szabadon hozzáférhető és naprakész, de nem kizárt, hogy megbízhatatlan, hacsak nem tudjuk, melyik címkézett változatot kell letölteni.
- A telepítés nagyrészt manuális művelet, számos kritikus ponton hibalehetőségekkel.
- A Slash leírása nem túl jó minőségű.
- Ha sikerül is a kód egy friss, működő példányát telepíteni, a teljes Slash telepítéshez szükség van a ***mod_perl***, a Template Toolkit, a MySQL telepítésére, egy csomó más Perl modulra és független eszközre, a saját, kissé eltérő és nem szabványos beállításaikkal.

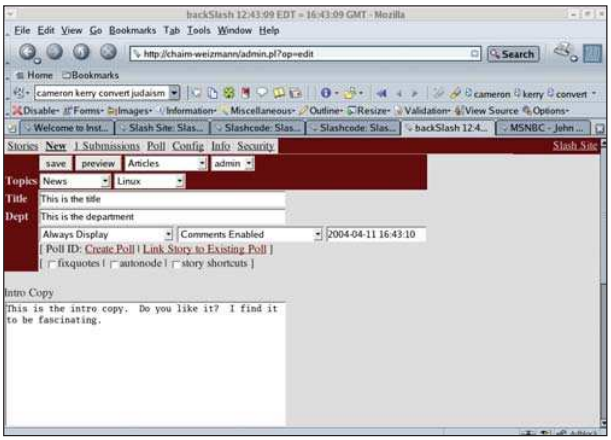
Ha járatosak vagyunk a ***mod_perl***-ben, a MySQL-ben, és az Apache-ban, akkor a Slash telepítése egy kissé kellemetlen, de elvégezhető folyamat. Ha kevésbé vagyunk gyakorlottak, az eredmény valószínűleg megéri a fáradságot, számítsunk rá, hogy menet közben nagyon sokat megtanulunk ezekről az alkalmazásokról. Lehet azonban, hogy egyszer csak azon kapjuk magunkat, hogy az IRC csatornán, vagy a webhelyen keresünk tanácsokat és ötleteket.
A Slash telepítésének első lépése az Apache, a ***mod_perl*** és a MySQL telepítése. A MySQL bármelyik újabb változata megfelelő, az igazi gond az Apache-csal és a MySQL-lel lehet, amelyeket elsőre talán bonyolult telepíteni. Szerencsére, az évek során a telepítés folyamatának ez a része alapvetően nem változott, ami azt jelenti, hogy követhetjük az InstallSlash oldalon található MySQL, Apache és ***mod_perl*** telepítési utasításokat. (Lásd az online források részt). Ha sem a Perl, sem az ***expat*** nincs telepítve a rendszeren, akkor ehhez is segítséget nyújtanak az InstallSlash utasításai. Ne feledjük, hogy RedHat és Fedora rendszereken nem csak az ***expat*** RPM-et, hanem az ***expat-devel*** RPM-et is telepíteni kell. Meg kell adnunk egy MySQL adatbázist is, amelyben a hely adatai tárolódnak, alaphelyzetben ez a ***slashdb*** nevet kapja. Az

Apache és a *mod_perl* fordításakor az `EVERYTHING=1` megadásának szükségessége jelentős problémaforrás. Ez működésbe hozza a *mod_perl* összes horgát (*hook*), ezzel lehetővé teszi a *mod_perl* számára, hogy felülbírálja az Apache minden alapértelmezésbeli működését, beleértve a hitelesítést, az engedélyezést, az URL átírást és a naplózást. Az `EVERYTHING=1` megadása nélkül a *mod_perl* csak tartalmat hozhat létre. Ha a rendszerünket már telepített *mod_perl*-lel szereztük, valószínűleg az `EVERYTHING=1` megadása nélkül fordították, ami azt jelenti, hogy magunknak kell újrafordítanunk.
A Slash használati utasítás a `PERL_MARK_WHERE=1` beállítását is javasolja a rendszergazdáknak fordításkor, habár a *mod_perl* kódja és leírása szerint ez az utasítás törli a meghatározatlan értékekre vonatkozó figyelmeztetések nagy részét a hibanaplóból. Én figyelmen kívül hagytam ezt a tanácsot, és a Slash-t a létező *mod_perl* telepítésemen használtam, de nem tapasztaltam semmilyen káros hatást.

Ha úgy döntünk, hogy biztonsági okokból, vagy egyéni ízlésünk miatt megváltoztatjuk az értékeket, semmiképp ne felejtsük el, milyen neveket használtunk.
A *Bundle::Slash* letöltésének legnehezebb része a Template Toolkit telepítése. A Template Toolkit egy népszerű és hatékony sablon-rendszer a *mod_perl* alatt, ami arra szolgál, hogy egy Slash webhely különböző oldalait következetes és praktikus módon jelenítse meg.

Címkézett változatok letöltése a CVS-ről

Ennél a pontnál magát a Slash kódot kell letöltenünk és telepítenünk. Az *InstallSlash* használati utasítás végigkísér minket a SourceForge-on található, legfrissebb csomagolt változat (2.2.6) letöltésének és telepítésének folyamatán. Viszont, mint fentebb említettem, 2.2.6 kibocsátása óta eltelt két évben rengeteg fejlesztés történt, és ezek a javítások csak a CVS változatban elérhetőek.



1. ábra Ide írjuk a szöveget...

2. ábra ...és így jelenik meg.

Végül, telepítsük a rendszerre a CPAN-ről (olyan kiszolgálók világméretű hálózata, amelyek szabadon felhasználható Perl modulokat és leírásokat tartalmaznak) a *Bundle::Slash* csomagot, az önműködő CPAN eszközök segítségével. A *Bundle::Slash* tulajdonképpen semmilyen kódot nem tartalmaz, csak felsorolja a Slash futtatásához szükséges modulokat, így nekünk nem kell megjegyezni (és begépelni) az összes telepítendő Slash-hez kapcsolódó modult. Rendszergazdaként bejelentkezve a következő sor begépelésével telepíthetjük a modulokat:

```
# perl -MCPAN -e ´install Bundle::Slash´
```

Ha még ezelőtt sosem használtuk a CPAN-t, meg kell adnunk néhány CPAN-nel kapcsolatos paramétert, köztük a legközelebbi CPAN archívumot, amelyről modulokat tölthetünk le. Ez lehet, hogy elsőre egy kicsit bonyolult lesz, bár elfogadhatjuk az alapértékeket is, valószínűleg minden következmény nélkül.
A CPAN modulok telepítésének egyik nehézsége a *DBIx::Password*, amely a telepítésekor a webhelyre vonatkozó információkat kér. Az InstallSlash utasításai megmondják, mit kell válaszként beírni a parancssorba.

A következő paranccsal letölthetjük a legfrissebb változatot a CVS-ről, és elmenthetjük egy slash nevű könyvtárban.

```
cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:
➥/cvsroot/slashcode co -r T_2_3_0_148 slash
```

Ugyanakkor a legfrissebb CVS változat SourceForge-ról való letöltésének megvannak a maga buktatói, mert ezzel vállaljuk a kockázatot, hogy olyan kódot telepítünk, amit mostanában töltöttek fel, de nem ellenőriztek. A legjobb megoldás a címkézett változat használata. Ahogy a tapasztalt CVS fejlesztők tudják, minden CVS állománynak van egy változatszáma (*revision number*), mint például 1.5, vagy 2.8 vagy 3.1.1.2. Ezek a változatszámok csak az egyes állományok változataira vonatkoznak, és semmi közük a teljes projekthez. Ezért annak ellenére, hogy program projekt 2.0-s változatként kerül a nyilvánosság elé, az egyes állományok majdnem biztos, hogy nem 2.0-s változatszámúak. Hogy egy közös nevet (vagy számot) lehessen rendelni a projekt összes állományának a pillanatnyi állapotához, címkéket, vagy más néven jelképes változatokat kell használni. Amennyiben van írási jogunk, megcímkézhetjük a jelenlegi *release-2-0* könyvtárat, a következőképpen:

```
cvs tag release-2-0
```

A címkék nem tartalmazhatnak bizonyos karaktereket, például vesszőt vagy pontot sem. Ezért kell kötőjelet használni pont helyett annak jelzésére, hogy a programunk 2.0-s kiadású. Ez felveti azt a kérdést, hogy melyik címkét tudjuk vagy szükséges letölteni. A Slash esetében a kód lassan halad a 2.3.0-s változat felé, és a fejlesztők egységesítették a címkéket, a következőképpen: T_2_3_0_XXX. A T azt jelöli, hogy tesztelt, és az XXX minden egyes címkével növekszik. A cikk írásáig a legfrissebb címke a T_2_3_0_148, melyet a következőképpen tölthetünk le:

```
cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:
➥/cvsroot/slashcode co -r T_2_3_0_148 slash
```

Ez a parancs létrehoz egy ***slash*** nevű könyvtárat, és minden Slash-hez kapcsolódó kódot, könyvtárat és leírást ebbe tesz. Az ***INSTALL*** állomány, amely azoknak az utasításoknak a naprakész változata, amelyeket arra terveztek, hogy az éppen letöltött CVS változattal működjenek, különösen hasznos. Követtem a fájlban leírt utasításokat, begépeltem a `make install` parancsot az összes Slash alkotóelem telepítéséhez. A telepítési folyamat során kapott üzenet megmutatja, milyen parancsot tegyünk az Apache beállító fájl végére. Ez biztosítja, hogy az Apache tartalmazza a megfelelő Perl modulokat az induláskor, ezzel lényegesen csökkentve a ***mod_perl*** memóriaigényét és futási idejét.
Az egyik jó dolog a Slash-ben, hogy könnyedén tud több webhelyet kezelni ugyanazzal a kóddal. Azaz, ha úgy döntünk, hogy külön online közösséget szeretnénk a Perl, a Python, a Tcl és a Ruby számára, akkor a Slash ezt képes kezelni. Minden közösségnek saját gépnévre van szüksége, de ezek lehetnek egymástól teljesen különbözőek. Más szóval, egy Slash weboldal telepítése a Slash program telepítésétől különálló folyamat. Amikor a programot már telepítettük, alaphelyzetben a ***/usr/local/slash*** alá, a következő parancs futtatásával hozhatjuk létre az új webhelyet:

```
/usr/local/slash/bin/install-slashsite -u USER
```

Itt a USER ugyanaz a virtuális felhasználó, amelyet még a ***DBIx::Password*** telepítésekor hoztunk létre, alaphelyzetben a ***virtslash*** nevet kapja. Az `install-slashsite` még néhány kérdést feltesz, többek közt a rendszergazda felhasználónevét és jelszavát, ezután az Apache `apachectl` programmal való leállítására és elindítására utasít. Ez a program általában a ***/usr/local/apache/sbin*** alá van telepítve. Az `apachectl` program `restart` utasítását nem szabad használni, különösen, amikor ***mod_perl***-lel dolgozunk. Le kell állítani az Apache-ot, várni néhány másodpercet, hogy a folyamatok teljesen ki tudjanak lépni, és azután újraindítani.
Amikor megpróbáltam futtatni az install-slashsite-ot, észrevettem, hogy legalább egy CPAN modul hiányzik (***LWP::Parallel::UserAgent***). Nem tartott sokáig telepíteni, de csalódott voltam, amikor kiderült, hogy sem a ***Bundle::Slash***, sem a make install nem észlelte vagy javította ezt.

Egy egyszerű Slash weboldal

A Slash webhely most már használatra kész. Az én gépemen, amelynek a saját otthoni hálózatomon ***chaim-weizmann a*** neve, meg tudtam nézni a webhelyemet úgy, hogy a ***http://chaim-weizmann*** címre ugrottam.
Egy újonnan telepített Slash weboldal természetesen nem valami izgalmas. Ha azzal a rendszergazda felhasználónévvel és jelszóval lépünk be, amit az `install-slashsite`-nek adtunk meg, számos lehetőséget és menüt kapunk, amely elég nagy irányítást biztosít a rendszer fölött. Sok menüt nehéz megtalálni, és igénybe vesz némi időt, mire a Slash rendszergazda rájön, hogy hol hajtódnak végre a különböző változtatások, a webhelyen, vagy a lemezen lévő sablonokban és programokban. A ***slashguide.pod***, amely a CVS változattal kapott docs könyvtárban található, jó bevezető a Slash webhelyek karbantartásába.
Ha már valamennyire ismerjük az eredeti Slashdot webhelyet, rögtön elkezdhetünk történeteket feltenni, és gyűjteni a hozzászólásokat. A webhely rendszergazdájaként bejelentkezve kattintsunk a főoldal tetején a ***New*** (Új) hivatkozásra. Adjunk meg egy témát, egy címet, egy kategóriát, egy bemutató változatot (a főoldalon látható majd) és egy hosszabb változatot (az önálló történet-oldalon látható majd). Még közvélemény-kutatást is csinálhatunk, vagy összekapcsolhatunk egy történetet egy meglévő közvélemény-kutatással. Mikor jóváhagytuk a történetet, az egész világ számára láthatóvá válik. Az oldal felhasználóinak tetszés szerint lehetővé tehetjük a hozzászólást, sőt, még moderátorok is lehetnek. A Slash történetek közösségi moderálása és meta-moderálása tényleg a legizgalmasabb ötlet, amit a Slash letett az asztalra.

Következtetés

A Slash-t nehezebb telepíteni, mint bármely programcsomagot, amelyről eddig szó volt. Ez részben azon múlik, hogy a szerzők milyen megoldást választottak, mivel a ***mod_perl***-t lényegesen nehezebb telepíteni, mint a XOOPS és a Zope által használt PHP-t. Ugyanakkor, a Slash több közösségi és webnapló lehetőséget kínál, mint más programcsomagok, közismerten nagy forgalmat képes kezelni, és igen magas szinten lehet karban tartani.
Ha nem félünk, hogy összepiszkoljuk a kezünket, és szeretnénk elérni azt a hatékonyságot, amit a Slashdot webhely biztosít, a Slash jó választás lehet a közösségek vagy egyének által vezetett webnaplók számára. Következő hónapban olyan személyes naplókat nézünk meg, amelyeket a Slash segítségével lehet létrehozni egy olyan társrendszerrel együtt, amelynek segítségével csoportosítani lehet a felhasználókat a rendszerben, olyan virtuális közösséget létrehozva, amely a valódi világot tükrözi.

Linux Journal 2004. július, 123. szám

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

Számítógéphálózatok (8. rész)

Elemi adatkapcsolati protokollok.

Sorozatunk jelen részében az adatkapcsolati protokollokkal foglalkozunk, azaz megnézzük, miként zajlik az adatkapcsolati rétegen „belüli” kommunikáció. Eddig sok érdekes dolgot hallhattunk az adatkapcsolati réteg feladatairól, most itt az ideje, hogy ezek gyakorlati megvalósításáról is szót ejtsünk. Adott tehát a probléma: A és B gép adatot cserélnének egymással. Mindezt egy olyan világban szeretnék megtenni, ahol a keretek eltünhetnek, megsérülhetnek, illetve a két gép sebessége között jelentős eltérés is előfordulhat. Hogy mindez ne jelenthessen problémát, szükség lesz az adatkapcsolati protokollokra.

Továbbra is az egyszerűből haladunk a bonyolultabb felé, azaz először egy olyan protokollal ismerkedünk meg, ahol csak az A gépnek van módja adatot küldeni a B felé, és élünk azzal a nevetséges feltételezéssel, hogy az A gép végtelen mennyiségű elküldendő adattal rendelkezik, illetve az adatok előállítása semennyi időbe se kerül. ***(A hálózati réteg mindig készenlétben áll).*** A későbbiekben megszabadulunk ezektől a megkötésektől, ezzel eljutva egy gyakorlatban is használható protokollhoz.

Keretek és csomagok

Ahhoz, hogy bemutathassuk az adatkapcsolati protokollok működését, fontos, hogy bizonyos dolgokban előre megállapodjunk. Most sem térünk el az általunk bemutatott hálózati modelltől, azaz a fizikai, az adatkapcsolati és a hálózati réteget továbbra is független, egymással párhuzamosan futó folyamatokként tételezzük. Minden réteg csak a „saját megfelelőjével” kommunikál, azaz az A gép hálózati rétege a B gép hálózati rétegével cserél információt. Emlékezzünk csak az első részben bemutatott filozófusok esetére: mindkét filozófus egymással kommunikál, de az üzenetváltást a titkárnők valósítják meg.

Ebben az esetben a hálózati réteg a filozófus, az adatkapcsolati réteg a titkárnő, a fizikai réteg pedig az email vagy fax, amin a titkárnők az üzeneteket továbbítják. A valóságban ezek a rétegek nem minden esetben vannak ennyire különválasztva. Az adatkapcsolati réteg lehet például a hardver része is. Ehhez a modellhez mi csak azért ragaszkodunk foggal-körömmel, mert így a különböző részfeladatok gyakorlati megvalósítását egymástól függetlenül is be tudjuk mutatni. Sőt mi több: egyik rétegnek sem kell tudnia arról, hogy a másik miként oldja meg a saját feladatát. Ugyanígy a filozófusokat sem foglalkoztatják az élet értelméhez képest olyan, teljesen jelentéktelen kérdések, hogy például a titkárnőjük email-en vagy faxon továbbítja az üzeneteiket.

Mostantól kezdve a hálózati réteg által egyszerre elküldött adatblokkot csomagnak nevezzük, az adatkapcsolati réteg „megfelelője” pedig a keret lesz. Ha a hálózati réteg adatot kíván küldeni, akkor ő azt először egy csomagba szervezi (hogy egy csomagnak milyen részei vannak, arról majd a hálózati réteg tárgyalásakor visszatérünk), ezt követően pedig átadja azt az adatkapcsolati rétegnek. Fontos, hogy míg a hálózati réteg számára a csomag egy jól meghatározott struktúrával bír, az adatkapcsolati rétegnek ez csak bitek sorozata.

Az adatkapcsolati réteg a csomaghoz hozzáragaszt némi plusz információt, ezeket a keret fejlécének fogjuk nevezni. A keret tehát két részből áll: a hálózati réteg számára is érdekes adatmezőből, és a fejlécből, amely további három mezőre oszlik.

Az első a ***kind***, amely egy logikai érték, és azt mondja meg, hogy van-e adat a keretben. Erre nem azért van szükség, hogy haszontalan üres keretekkel terhelhessük kedvünkre a hálózatot, hanem azért, hogy lehetőség legyen vezérlőkeretek küldésére is. Amikor ugyanis érkezik a keret, akkor annak adatmezejét az adatkapcsolati réteg mechanikusan továbbítja. Felmerülhet azonban az igény, hogy a két adatkapcsolati réteg kommunikálhasson egymással.

A keret fejlécéhez tartoznak még az ***ack*** és a ***seq*** mezők, ezekkel valósíthatjuk meg a keretek sorszámozását, illetve a nyugtázást.

Mivel az algoritmusokat a legegyszerűbben valamilyen programnyelvhez hasonló nyelven segítségével lehet ismertetni, mi egy C-hez nagyon hasonló, ám rendkívül magas szintű nyelvet használunk a protokollok bemutatására.

Korlátozás nélküli szimplex protokoll

Ez a lehető legegyszerűbb protokoll, amit csak el tudtunk képzelni. Mint már említettük, először azt a helyzetet vizsgáljuk, amikor csak az

A gép küld, akinek mindig van adata, és egy percet sem kell várni egy-egy keret előállítására. A vevő sincs híján a mesébe illő tulajdonságoknak: végtelen gyorsan képes feldolgozni a hálózaton hozzá érkező kereteket.

Ezt úgyis megfogalmazhatjuk, hogy a vevő végtelen puffer területtel rendelkezik, ahova bármennyi feldolgozásra váró keret befér.

Azon már meg se lepődjünk, hogy a kommunikcáiós csatorna annyira jó minőségű, hogy minden keret sértetlenül megérkezik.
A *2. listán* láthatjuk ennek a protokollnak egy megvalósítását. A `Kuldo()` függvény a küldő, a `Vevo()` pedig a fogadó gépen fut. Mindkét eljárás lelke egy végtelen ciklus. A küldő amilyen gyorsan csak tudja, fogadja az adatokat a hálózati rétegtől, és egyből be is rakja egy keret info mezejébe.
A keret egy összetett típus, ennek definiálását láthatjuk az *1. listán*. Ez a protokoll csak az info mezőt használja, hiszen forgalomszabályozásról, illetve nyugtázásról ő még semmit sem hallott.
A küldő miután összerakta a keretet (ami most még nem volt egy különösebben megterhelő feladat), egyből át is dobja a fizikai rétegnek, amely bitenként továbbítja azt a csatornán keresztül.
A vevő eljárás sem bonyolultabb ennél. Először is vár, hogy történjen valami. Jelen esetben a jövő varázsgömb nélkül is kiszámítható, ugyanis csak egy dolog történhet: egy sértetlen keret érkezik. Az „e" változó értéke tehát biztosan „`keret_erkezett`" lesz, így annak értékének ellenőrzése nélkül egyszerűen csak át kell venni a keretet a fizikai rétegtől, majd továbbítani annak info mezőjét a hálózati réteg felé.

1. lista A keret összetett típus

```
typedef struct
{
  bool kind;
  int seq, ack;
  csomag info;
} keret;
```

Szimplex „megáll és vár" protokoll

A számítógépeink továbbra is varázskábellel vannak összekötve, ahol nem sérülhetnek, illetve veszhetnek el a keretek. A vevő azonban már nem végtelen memóriájú, így előbb vagy utóbb a folyamatosan érkező keretektől megtelik a puffer, és kénytelen lesz a forrásállomáshoz szünetért könyörögni.
A megoldandó probléma tulajdonképpen nem más, mint hogy valamilyen módszerrel visszafogni a küldő állomást abban, hogy keretek sokaságát zúdítsa a vevőre. Általánosan úgy lehetne ezt megfogalmazni, hogy ha mondjuk a vevőnek T időbe telik, míg a fizikai rétegtől a hálózati rétegig eljuttatja az adatot (azaz T a `FizikaiRetegtol` és a `HalozatiRetegnek` utasítások végrehajtásának ideje), akkor a küldő T idő alatt átlagosan kevesebb mint egy keretet küldhessen.
Erre a legkézenfekvőbb megoldás az lehet, ha a forrásba „bedrótozunk" valamiféle késleltetést. A vevőnek azonban nem csak a keretek fogadásával kell foglalkoznia, hanem sok más egyéb feladata is lehet, például más gépekkel is kommunikálhat.
Így a T értéke folyamatosan változhat. Persze előfordulhat, hogy létezik a T-re egyfajta legrosszabb érték, amelynél a T értéke sohasem lehet nagyobb. Ha ezt választjuk a késleltetés időtartamának, akkor biztos, hogy a vevő minden esetben képes lesz feldolgozni az összes keretet, akármennyi más egyéb teendője is akad. Ezzel azonban az a gond, hogy rendkívül pazarló megoldás. Ha ugyanis a vevőnek éppen nincs semmi dolga, akkor sem fogja gyorsabban kapni a kereteket.
Praktikusabb megoldás tehát ha visszacsatolást létesítünk a vevő és a forrás között. Ez annyit tesz, hogy a vevőnek módjában áll visszaszólni a forrásnak.
A „megáll és vár" protokoll esetében például a forrás addig nem küldi el a soron következő keretet, amíg a vevő vissza nem küld egy „álkeretet", amellyel közli, hogy az előző keretet sikeresen feldolgozta, készen áll a következő fogadására *(3. lista)*. Az ábrán láthatjuk, hogy tulajdonképpen teljesen mindegy, hogy mi az álkeret, a forrásnak ugyanis csak a beérkezés a fontos. Rögtön ezt követően a forrás „felébred", és ismét elküld egy keretet. Ezután megint várakozik.
Másik fontos különbség az előző protokollal szemben, hogy itt már az adatok visszafelé is áramolnak. Ez azt jelenti, hogy az összekötő mágikus kábelnek kétirányúnak kell lennie. Az is igaz, hogy az adatok áramlásának iránya szigorúan változik, azaz hol az egyik, hol a másik küld.

Ha zaj van a csatornán...

Bonyolítsuk tovább a helyzetet és cseréljük le a zajmentes mágikus hálózati kábeleinket teljesen hétköznapiakra. Az előző protokoll nyilván használhatatlan, hiszen most már könnyedén előfordulhat, hogy egy-egy keret megsérül, esetleg el is veszik.
A nagy ötlet az, hogy módosítsuk az előző protokollt a következőképp: amikor a forrás elküld egy keretet, elindít egy időzítőt, és vár, hogy a vevő visszaküldjön egy nyugtát a keret sértetlen megérkezéséről.
Ha az időzítő lejártáig nem érkezik meg a nyugta, akkor a kérdéses keret valószínűleg elveszett, ezért a forrás azt ismét elküldi.
Ez jó megoldásnak tűnik, de végig számításba kell venünk azt az esetet is, amikor nem a keret, hanem a nyugta tűnik el. Ilyenkor a forrás nem értesül a keret megérkezéséről és ismét elküldi azt.
Ekkor a vevő ismét nyugtáz, és ha nem veszik el ismét a nyugta, akkor a forrás küldheti a következőt. Ha kicsit utánaszámolunk, láthatjuk, hogy a vevőhöz ugyanaz a keret kétszer is megérkezett. Rendkívül kellemetlen, ha például egy állomány átvitelekor annak bizonyos részei megduplázódnak.
Ezért a vevőnek valahogy fel kell tudnia ismerni azt, ha a beérkezett keret az előző ismétlése. A kereteket legegyszerűbben úgy különböztethetjük meg egymástól, ha sorszámmal látjuk el őket.
A sorszámot nyilván a keret fejlécében tárolnánk, és mivel nem jó, ha a fejléc túl nagy, ezért fontos a kérdés: a keretek sorszámait hány biten ábrázoljuk?
Könnyen belátható, hogy erre elegendő egyetlenegy darab bit is. Ha végig gondoljuk a protokoll működését, akkor megállapíthatjuk, hogy a nem egyértelmű keret csak az n. vagy az n+1. lehet.
Ha ugyanis az n. keret megsérül, akkor nem fog róla nyugta érkezni, ezért a forrás addig küldi, amíg végre hibátlanul át nem ér.

2. lista Korlátozás nélküli szimplex protokoll

```
Küldő:
void Kuldo()
{
  keret k;
  csomag cs;

  while(true) // végtelen ciklus
  {
    HalozatiRetegtol(&cs);
    // megkapjuk a hálózati rétegtől az
    // elküldendő csomagot
    k.info = cs;
    FizikaiRetegnek(&k);
  }
}

Vevő:
void Vevo()
{
  keret k;
  esemeny e;

  while(true)
  {
    EsemenyreVar(&e); // csak a keret_erkezett
                      // a lehetséges esemény
    FizikaiRetegtol(&k);
    HalozatiRetegnek(&k.info);
  }
}
```

3. lista Szimplex megall es var protokoll

```
void Kuldo()
{
  keret k;
  csomag cs;
  esemeny e;

  while(true)
  {
    HalozatiRetegtol(&cs);
    k.info = cs;
    FizikaiRetegnek(&k);
    EsemenyreVar(&e);
    // addig ne folytassuk, amig nincs ra
    // engedely
  }
}

void Vevo()
{
  keret k, s;
  esemeny e;

  while(true)
  {
    EsemenyreVar(&e); // csak a keret_erkezett
                      // a lehetséges esemény
    FizikaiRetegtol(&k);
    HalozatiRetegnek(&k.info);
    FizikaiRetegnek(&s); // alkeret kuldese
  }
}
```


Ha a nyugta veszik el, akkor ismét az n. keret kerül elküldésre. Ha a nyugta is rendben megérkezik, akkor kerül továbbításra csak az n+1. keret.
Folytatva a dolgot, ha az n+2. keretet elküldjük, akkor az az esemény magában hordozza azt is, hogy az m+1. keret nyugtája már sértetlenül megérkezett. Ez pedig azt jelenti, hogy az m. keret, és annak nyugtája is célba ért. Ezért az egy bites sorszám elegendő.
Azokat a protokollokat, ahol a forrás minden keret után a keret megérkezéséről pozitív visszajelzésre vár, ARQ-nak ***(Automatic Repeat reQuest – automatikus ismétléskérés)*** nevezik. Ezek is egyirányú ***(szimplex)*** protokollok, de már tudják kezelni az elveszett kereteket.
Mi történik azonban akkor, ha a forrás időzítője hamarabb jár le, mint ahogy a nyugta célbaérne. Ha már megtörtént a keret újraküldése, és az időzítő nullázása, akkor – ha a kérdéses nyugta mégis célba ér – a forrás azt fogja gondolni, hogy ez az imént elküldött keretre érkezett a nyugta. Ezért fontos, hogy a forrás valamiképpen tudomást szerezzen arról, hogy a beérkezett nyugta valójában melyik keret megérkezését erősíti meg. Persze ezt a problémát könnyen orvosolhatjuk azzal, ha maga a nyugta is tartalmazza a hozzá tartozó keret sorszámát. A *4. listán* láthatunk egy példát a zajos csatornán is alkalmazható szimplex megáll-és-vár protokollról. A forrás minden keret elküldése után elindít egy időzítőt. Ezután várakozik addig, amíg nem történik valami esemény. Ilyen esemény lehet például az, hogy a nyugta visszaérkezik, illetve visszaérkezik valami, de átviteli hibával, vagy esetleg nem érkezik semmi, és lejár az időzítő.
Az első esetben a forrás lekérheti a következő csomagot a hálózati rétegtől, invertálja a sorszámot (ha 0 volt, akkor 1 lehet, vagy pedig fordíva), és továbbadja a fizikai rétegnek. Ha azonban sérült nyugta, vagy éppen semmi sem érkezik, akkor ismét az előző csomagot küldi, és a sorszámot sem írja felül.
A vevő minden beérkezett keretet megvizsgál, hogy egész véletlenül nem-e duplikátum. Erre szolgál a „`vart_keret`" változó. Ha a beérkező keret sorszáma nem egyezik meg ennek a változónak az értékével, akkor biztos, hogy duplikátumról van szó.

Ráültetés (piggybacking)

Eddig olyan helyzeteket vizsgáltunk, amikor csak az egyik fél küld adatokat a másiknak. Igaz, a második és a harmadik protokoll esetében már a kommunikáció valójában kétirányú volt, tehát szükség volt egy olyan csatornára, ahol az adatok mind a két irányban áramolhatnak.

4. lista szimplex megall-es-var protokoll

```
void Kuldo()
{
  sorszam kovetkezo_keret; // a kovetkezo keret
                           // sorszama
  keret k;
  csomag cs;
  esemeny e;

  kovetkezo_keret = 0; // inicializaljuk
  HalozatiRetegtol(&cs);  // lekerjuk az elso
                          // elkuldendo csomagot
  while (true)
  {
    k.info = cs; // bemasoljuk a csomagot
                 // a keretbe
    k.seq = kovetkezo_keret; // beallitjuk a
                             // keret sorszamat
    FizikaiRetegnek(&k); // elkuldjuk a keretet
    IdozitoIndul(k.seq); // elinditjuk az
                         // idozitot
    EsemenyreVar(&e);
    // lehetseges esemenyek: keret erkezes,
    // idotullepes, ill. chesksum hiba
    if ( e == keret_erkezett ) // ha megerke-
                               // zett a nyugta
    {
      FizikaiRetegtol(&k); // megszerezzuk a
                           // nyugtat
      if ( k.ack == kovetkezo_keret )
      {
        HalozatiRetegrol(&cs);  // kovetkezo
                                // csomag ....
        inc(kovetkezo_keret); // invertaljuk a
                              // sorszamot
      }
    }
  }

void Vevo()
{
  sorszam vart_keret;
  keret k, nyugta;
  esemeny e;
  // mivel itt mar megserulhetnek a keretek,
  // ezert a cheksum_hiba is
  // lehetseges esemeny

  vart_keret = 0;
  while(true)
  {
    EsemenyreVar(&e);
    if ( e == keret_erkezett )
    {
      FizikaiRetegtol(&k); // megszerezzuk a
                           // keretet
      if ( k.seq == vart_keret ) // ha erre a
                                 //keretre vartunk
      {
        HalozatiRetegnek(&k.info); // atadjuk a
                             // halozati retegnek
        inc(vart_keret); //legkozelebb mar a
                         //masik sorszamra varunk
      }
      nyugta.ack = 1 - vart_keret; // melyik
                           // keretnek a nyugtaja
      FizikaiRetegnek(&nyugta); // elkuldjuk a
                                // nyugtat
    }
  }
}
```

Ilyen csatornát a legegyszerűbben úgy csinálhatunk, hogy két egyirányú csatornát teszünk egymás mellé, csak végpontjaikat „fordítva” kötjük be. Az egyik dróton tehát A-ból B-be, a másikon pedig B-ből A-ba áramolhatnak az adatok. Ez azonban meglehetősen pazarló megoldás, mivel egyrészt a sávszélesség nagyrésze kihasználatlan marad, másrészt a felhasználó két áramkörért fizet, de kapacitásban annyit kap, mintha egyet használna.

Az ideális megoldás nyilván az, hogy egy áramkört használunk az oda és visszamenő adatok továbbítására.

Erre láttunk példát az előző két protokoll bemutatásakor is. Ilyenkor az A-ból B-be, és a B-ből A-ba tartó adatok egymással összekeveredtek.

Amikor a B állomás is küld adatokat az A-nak, akkor mindkét félnek meg kell tudnia különböztetnie az adat- és a nyugta kereteket. Erre szolgál a keret fejlécében elhelyezett „kind” mező.

A hatékonyságot azonban még tovább is növelhetjük, például úgy, hogy a vevő nem küld minden keret beérkezése után azonnal nyugtát. Csökkenthetnénk a hálózat forgalmát úgy, ha a nyugtát egy kifelé menő adatkerethez „csatoljuk”.

Ha tehát beérkezik egy keret, akkor a vevő a nyugta küldésével addig vár, amíg a hálózati rétegtől nem kap elküldendő adatot. Ha ezt megkapja, akkor a nyugtát ráülteti a kifelé menő adatkeretre. Így a vevő két keret helyet csak egyet küld. Ezzel a megoldással sokat nyerhetünk, hiszen nem csak jobban használhatjuk ki a rendelkezésre álló sávszélességet, hanem a vevő oldalán kevesebb „keret érkezett” megszakítás váltódik ki. Ráadásul, hogy ez megvalósítható legyen, csak egy bitet kell módosítanunk a keret fejlécében.

Ez a módszer azonban csak akkor működik, amikor mindkét fél végtelen mennyiségű elküldendő adattal rendelkezik. Ha ugyanis a vevőhöz megérkezik egy keret, de annak hálózati rétege éppen semmit sem akar küldeni, akkor előfordulhat, hogy a forrás által elindított időzítő lejár.

Ez pedig a kérdéses keret ismétlését fogja eredményezni, tehát ha időközben érkezik is adat a vevő hálózati rétegétől, a nyugta már elvesztette jelentőségét.

Megoldás lehet a jövőbelátás, azaz megjósolni, hogy a hálózati réteg mikor fog adatot átadni, és a forrás időzítőjét ehhez hangolni. Erre azonban még a TV jós sem képes, ezért jobb megoldásnak tűnhet, ha a vevő inkább vár egy előre meghatározott ideig, majd ha addig nem kapott feladatot a hálózati rétegtől, elküldi a nyugtát egy önnálló keretként.

A következő részben innen folytatjuk. Izgalmakban továbbra sem lesz hiány: szó lesz a csúszóablakos protokollokról, végesállapotú automatákról, híres adatkapcsolati protokollokról (például a PPP-ről, amelyet az Internetben is széles körben alkalmaznak).

Garzó András (garzo@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.

Egy webkiszolgáló tündöklése és bukása

A cikk első feléből megtudhatod hogyan lehet öt perc alatt HTTP kiszolgálót írni. A másik felében rájössz, miért nem érdemes.

Szoftverfrissítések, hírek a nagyvilágból, tv-műsor, no és az ebédem. Ezeket nap mint nap az internetről szerzem be. Egészen pontosan meglátogatok a böngészőmmel egy olyan oldalt, ahol a web nyújtotta korlátlan multimédia lehetőségeket kihasználva választom ki a kényes ízlésemnek megfelelő programokat, híreket, filmeket és feltételeket. A háttérben a böngészőm egy távoli web kiszolgálóval beszélget a HTTP protokollt használva de ez engem hidegen hagy. Engem csak a pizzám érdekel. Viszont lehet, hogy írsz egy programot, ami egy-két adminisztrációs munkát tesz önműködővé, és jól jönne hozzá egy webes felület. A vezetőség talán azt se tudja, mi az az SSH, de szeretne új felhasználókat hozzáadni a rendszerhez. Vagy csak azt akarod tudni, hogy a csillogó-villogó külső alatt milyen fogaskerekek működtetik a webet. Mindkét esetben jó kiindulási pont lehet belekóstolni egy ilyen kiszolgáló lelki világába.

Az egyszerűség kedvéért egy olyan nyelvet fogunk használni, ami felépítésének köszönhetően gyors alkalmazásfejlesztést tesz lehetővé. Perlben fogjuk elkészíteni első szerverünket, így a fő irányelvek nem vesznek el a részletek tengerében. Ha még nem használtad ezt a szkriptnyelvet, itt az ideje, hogy ellátogass a ➲ http://www.perl.org/ címre. Ajánlom továbbá a perl kézikönyv oldalait, melyek Debian rendszer alatt a ***perl-doc*** csomagban találhatóak.

Mindössze egy Perl modulra fogunk támaszkodni a fejlesztés során, ez az *IO::Socket*. Ez része az általános Perl könyvtárnak (***Standard Perl Library***), így minden Perl terjesztés tartalmazza. A munkához tehát mindössze egy megfelelően telepített Perl változatra, illetve kedvenc szövegszerkesztődre lesz szükség. Utóbbiak közül grafikus felület alatt erősen tudom ajánlani a jedit-et (➲ http://www.jedit.org/). Számtalan kényelmi funkciója mellett hihetetlen nagy számú nyelvet támogat színkiemeléssel is, ezért egy próbát mindenkinek megér.

Akkor vágjunk neki! Ha már olvastad A Perl és a hálózat (Linuxvilág, 2002. március) című cikkemet, az első néhány sor nem fog meglepetést okozni.

Az első sorban megadjuk a szkript parancsértelmezőjének elérési útvonalát, és kezdeti paramétereit. Ennek segítségével a szkriptnek futtatási jogot adva, közvetlenül indítható parancssorból. A `-w` hatására minden figyelmeztető üzenet megjelenik az szabványos hibacsatornán (***standard error***).

Ezután két modult is használatba veszünk. Az első, a `strict` minden szabványos Perl program alapköve. Ennek hatására hibaüzeneteket kapunk, ha az egységesítés útjára lépett nyelvben nem szép szerkezetekkel dolgozunk. Az `IO::Socket` modul pedig a hálózatkezelést egy kellemesen használható, objektum-orientált felület mögé rejti.

```
my $docroot = "./web";

die "Usage: " . $0 . " {port}\n" unless (@ARGV > 0);
```

A változók deklarációja kötelező, ha strict üzemmódban vagyunk. Erre szolgál a `my` kulcsszó. A `$docroot` egy globális változó lesz, mely annak a főkönyvtárnak az elérési útját tartalmazza, amelyben az összes HTML állományt tárolni fogjuk. Ezzel ugyanazt szeretnénk elérni, mint az Apache a DocumentRoot direktívával. A következő sor ellenőrzi, hogy van-e átadott parancssori paraméter. Így várjuk ugyanis azt a kapuszámot, ahol a kiszolgáló figyelni fog és várja a beérkező kéréseket. A program használat alakját írja ki a képernyőre és megszakítja futását, amíg az átadott paraméterek száma nem nagyobb nullánál. Ehhez csak annyit kell tudni, hogy Perl alatt egy tömb (jelen esetben `@ARGV`) skalárként értelmezve a tömb elemeinek számát adja vissza.

```
my $server = new IO::Socket::INET (
        LocalPort => $ARGV[0],
        Proto => "tcp",
        Listen => SOMAXCONN,
        ReuseAddr => 1
);

die "Error creating socket.\n" unless defined
➥$server;
```

Ezután újabb két sor következik, amit csupán a könnyebb olvashatóság érdekében tagoltam több sorba. Előbb egy új változót hozunk létre, majd ellenőrizzük, hogy sikerrel jártunk-e. Első lépésben a new kulcsszóval az `IO::Socket::INET` osztályból hozunk létre egy példányt, melynek segítségével majd felépítünk egy TCP/IP kapcsolatot. A konstruktornak egy olyan asszociatív tömböt kell átadni, mely kulcs-érték párjaival leírja a felépítendő foglalat

(*socket*) típusát. A legfontosabb a parancssorban átadott kapuszám. A várakozási sort a lehető legnagyobbra állítjuk, illetve beállítjuk, hogy minden kapufoglalás előtt próbálja meg felszabadítani azt, ha szükséges (lásd: `IO::Socket::INET(3 perl)`).

```
$SIG{INT} = sub {
        print "Stopping WEB server.\n";
        close $server;
        exit;
};
```

Következő lépésben a megszakítás szignálhoz (***interrupt signal***) hozzárendelünk egy névtelen eljárást, amely a kiszolgáló biztonságos leállításáért felel. Ez azért szükséges, mert az elindított webszerver majd csak a Ctrl+C billentyűkombináció hatására fejezi be működését, azonban ekkor is fontos a biztonságos leállítás. A %SIG szintén egy asszociatív tömb, melyet a szignálok rövid neveivel lehet indexelni, és minden szignálhoz egy függvényt rendel. Jelen esetben Ctrl+C esetén kiírja a képernyőre az elköszönő üzenetet, lezárja a program elején nyitott foglalatot, majd kiszáll.

```
print "Starting WEB server (port " . $ARGV[0]
➥. ").\n\n";

for (my $client; $client = $server -> accept ();
➥close $client) {
...
}
```

Ez a ciklus lesz a kiszolgáló lelke. A '.' operátor hatására összefűzött karakterfüzér megjelenítése után valójában egy végtelen ciklus indul el. Ebből csak szignálok használatával lehet kilépni. A ciklus kezdetekor deklarálunk egy `$client` nevű változót, mely a csatlakozó ügyfél foglalatát fogja tárolni. Minden iteráció elején meghívjuk a `$server` objektum `accept ()` elemfüggvényét. Ez addig nem adja vissza a vezérlést, amíg egy új ügyfél nem érkezik. Ekkor felépíti a kapcsolatot, és visszaadja a kliens foglalatát. Az iteráció végén, amikor az ügyfelet kiszolgáltuk, ez utóbbit természetesen le kell zárnunk.

```
print localtime () . " Connect from "
➥. $client -> peerhost () . "\n";
my $line = <$client>;
print "  " . $line;
...
print localtime () . " Closing connection.\n\n";
```

A ciklus minden lefutásának elején kiszolgálóoldalon kiíratjuk az aktuális dátum mellett az ügyfél csatlakozásának tényét, illetve IP-címét. Ezt az adatot a `$client` objektum `peerhost ()` elemfüggvényével határozzuk meg. Ezután fogadjuk az ügyfél kérését, ami azt jelenti, hogy egy sort olvasunk be a kliens foglalatról a gyémánt (***diamond***) operátor segítségével. Ezt a `$line` változóban tároljuk, és szemléletesség kedvéért azonnal ki is íratjuk a kiszolgáló oldalán. Természetesen az iteráció végén a kapcsolat lezárásának ténye is megjelenik a képernyőn.

```
if ($line =~ /^GET (.*) HTTP.*\r$/) {
   my $file = $1;
   $file .= "index.html" if ($file =~
   ➥/\/$/);
   print $client "HTTP/1.0 200 OK\r\n";
   print $client "Server: BigWig WEB
   ➥server\r\n";
   print $client "Connection: close\r\n";
   print $client "Content-Type:
   ➥text/html\r\n\r\n";
   ...
}
```

Csak az ügyfél legegyszerűbb GET kéréseit dolgozzuk fel. A `$line` változót, amely a kliens által küldött sort tartalmazza, próbáljuk meg az =~ operátorral egy reguláris kifejezésre illeszteni. A '/' jelek között szereplő kifejezés a `GET /eleresi_ut/file HTTP/x.x` alakra illeszkedik, ahol `x.x` a protokoll változatszáma. A feltételes szerkezet magjába akkor lép be, ha a minta illeszkedett, sőt mi több a `$1` változó a zárójelezett `.*` miatt tartalmazni fogja a kért állományt. Ezt egy `$file` változóba helyezzük át az átláthatóság és a módosíthatóság végett, és a végére illesztünk egy „`index.html`" szöveget, ha '/' jelre végződik. Továbbá kiíratjuk a szabályos HTTP fejlécet az ügyfél oldalára.

```
$file = $docroot . $file;
if (-f $file and -r $file) {
  print "  Serving " . $file . "\n";
  open (INPUT, $file);
  while (<INPUT>) { print $client $_
  ➥. "\r\n"; }
  close (INPUT);
} else {
  print "  404 Not found\n";
  print $client "<HTML><HEAD><TITLE>
  ➥\r\n";
  print $client "404 Not found\r\n";
  print $client
  ➥"</TITLE></HEAD><BODY>\r\n";
  print $client "<FONT size=+2>Sorry,
  ➥the page you\r\n";
  print $client "have requested cannot
  ➥be found.\r\n";
  print $client "</FONT></BODY></HTML>
  ➥\r\n";
}
```

Végezetül az állománynév elé tesszük a főkönyvtár elérési útját, ellenőrizzük a file meglétét, és kiíratjuk az ügyfélnek. A `-f` és `-r` különleges függvények (lásd `perlfunc(1)`). Nemnulla visszatérési értékkel jelzik, ha a file közönséges, illetve ha olvasható. Ez esetben megtesszük a szokásos bejegyzést kiszolgálóoldalon, megnyitjuk az állományt, soronként kiíratjuk, majd lezárjuk. Ellenkező esetben pedig egy olyan HTML oldalt nyomtatunk a kliensnek, ami arról tájékoztatja, hogy a keresett oldal nem található. Ennek megfelelően már egy kis HTML tudással tetszőleges dinamikus tartalom készíthető.

```
print "  Creating dinamic content.\n";
print $client "<HTML><HEAD><TITLE>\r\n";
print $client "Dinamic page\r\n";
print $client "</TITLE></HEAD><BODY>\r\n";
print $client "<TABLE border=1>\r\n";
for (my $i = 1; $i <= 10; $i++) {
   print $client "<TR>";
   for (my $j = 1; $j <= 10; $j++) {
      print $client "<TD>" . $i * $j
      ➥. "</TD>";
   }
   print $client "</TR>\r\n";
}
print $client "</TABLE></BODY></HTML>
➥\r\n";
```

A fenti példa dinamikusan hoz létre egy 10x10-es szorzótáblát. Látható, hogy a HTML fejléc kiíratása után egy kettős for ciklussal készítjük el a táblázatot. A külső végigmegy a sorokon, és a belső kitölti azokat. Csak arra kell figyelni, hogy a nyomtatás ne a szabványos kimenetre (standard output) történjen, hanem az ügyfél foglalatára. Az eddigi példákból könnyen látható, hogy a foglalatok ugyanolyan egyszerűen írhatók-olvashatók, mint a közönséges fileleírók (***file handler***).
Utóbbi, dinamikus rész úgy van beillesztve, hogy még a kiszolgálás előtt történik egy esetszétválasztás. Ha a kérés *„**/dinamikus/index.html**”*-re vonatkozik, ez a rész fut le, egyéb esetben az állomány kiíratása. Tetszőleges böngészővel a ***http://localhost:port/eleresi_ut*** címre mutatva érhetjük el a helyi gépről a kiszolgálót, de ne felejtsük el, hogy a hálózat tetszőleges pontjáról megtalálható a megfelelő cím megadásával. A mellékelt képen jól látható, hogy kiszolgálónk támogatja az összes népszrű böngészőt.

Ha ez mind ilyen szépen működik, akkor mi a baj?

A kiszolgáló első és legszembetűnőbb hibája, hogy Perlben íródott. A Perl egy szkriptnyelv, vagyis nem előrefordított. Nagy terhelés mellett nem nehéz megfektetni egy szervert, amely olyan nyelven íródott, ami futásidőben értelmeződik. Valódi, nagy erőforrásigényű alkalmazásokat érdemes inkább C/C++-ban készíteni. Ez a példa viszont egy állatorvosi ló. Azért íródott Perlben, hogy a legfontosabb részek jól felnagyíthatók legyenek, és ne zavarjanak minket az apró részletek. Ha az irányelveket elkaptad, nézz utána, hogyan írhatnád meg ugyanezt pl. C-ben.
Sajnos a szerverünk a dinamikus tartalom készítése miatt HTML kódot tartalmaz! Ez egy súlyosan hibás elgondolás. Valamivel tűrhetőbb megoldás lett volna, ha az oldal készítését egy külső szkriptre bízzuk, a szerverünknek így azt csak meghívnia kellene. Ezt az eljárást hívják CGI-nek (***Common Gateway Interface***), amikor egy külső program készíti a teljes választ az ügyfél GET kérésére. Ez azonban szintén egy túlhaladott elképzelés, ma már sokkal szívesebben használnak oldalba ágyazott szkripteket (ilyen a PHP).
A szerver nem tud egy időben egynél több ügyfelet kiszolgálni. Ha két kliens egyszerre csatlakozik, előbb a gyorsabbat kiszolgáljuk, ezalatt a másik várakozási sorba kerül (***listen queue***). Ezt nagyjából úgy lehet elképzelni, mint egy fogorvosi rendelőt, amelyben egyetlen orvos van. A várószoba méretét adhattuk meg a ***$server*** objektum létrehozásakor a Listen paraméterrel. Ahhoz, hogy több orvosunk legyen, gyermek folyamatokat kellett volna létrehozni minden egyes csatlakozó ügyfélnek, ám ez szintén elbonyolította volna a példát. Ha ennek a mikéntje érdekel, olvasd el a vonatkozó kézikönyv lapot: `perlfork(1)`.
Elég gyermekded módon bántunk a hibakezeléssel is. Ha az állomány nem létezik, a HTTP szabvány szerint nem szabad OK kódot adni az ügyfélnek. Mi a HTTP fejlécben kiadtunk egy HTTP 200 OK-t, majd nyomtattunk egy olyan HTML oldalt, ami leírja, hogy mennyire sajnáljuk, amiért nem tudjuk kiszolgálni. Már a fejlécben illett volna 404-es kóddal jelezni, hogy az oldal nem található. Ezeket a dolgokat nem szabad elnagyolni, mert a szabványtól való eltérés miatt semmi sem garantálja, hogy a böngésző tényleg megjeleníti a tartalmat.
Egy portál gyakran tartalmaz képet. Máskor hangot, videót, flash animációt, letöltésre váró állományt. A sokféle tartalom egységes kezelésére találták ki a MIME információt. Ez is a HTTP fejlécben utazik, és egy főcsoport/alcsoport formában szöveges leírással szolgál az adott állományról. Mi egyszerűen azt mondtuk, hogy minden text/html. Ennek az az eredménye, hogy ha egy képet kér egy kliens, azt is ezzel a MIME jelzéssel adjuk át, ezért nem a kép fog megjelenni a böngészőben, hanem a bináris tartalom.
Végül, de nem utolsó sorban alapvető biztonsági hibától szenved a szerver. A főkönyvtáron kívüli tartalom is elérhető, mindössze olyan GET kérést kell fogalmazni, amely kellő számú „..”-t tartalmaz.Mivel ez nem valódi chroot, ezért elég a böngésző címsorába azt írni, hogy `http://localhost:port/../web.pl`, és máris olvasható a webszerver forrása. Ez azonban nem az egyetlen olyan információ, amely bizalmas lehet, gondoljunk csak /etc/passwd állományra, amely bárki számára olvasható lesz, amint a támadó eltalálta a megfelelő számú “..”-ot.

Ha ez ennyire rossz, akkor mire volt jó?

Okos ember mások hibáiból tanul – szokták mondani. Ha ez egy informatikusra igaz, az hatalmas szerencse mindenkinek. Olvass utána azoknak a részeknek, amelyekről most hallottál először. Utána próbáld meg átírni a szervert úgy, hogy eggyel kevesebb hiba legyen benne – legyen az elvi, kényelmi, vagy biztonsági.
A fő az, hogy meglásd a saját kódodban is az ilyen hibákat és arra törekedj, hogy kijavítsd őket, ahelyett, hogy elfednéd.

Írj bátran, ha kérdésed van.
Sok sikert és örömet a kísérletezgetéshez!

Fülöp Balázs (admin@guardware.com)
Imádja a Túró Rudit, a Debian Linuxot és a teheneket. Kedvenc írója Slawomir Mrożek. Leginkább a számítógépes hálózatok biztonsága érdekli. A BME VIK műszaki informatikus szak hallgatója.

A Sudo-ról dióhéjban

...avagy hogyan indíthatjuk programjainkat álnéven?

Bizonyára minden rendszergazdában felvetődött már a probléma, hogy hogyan lehetne azt megoldani, hogy bizonyos kitüntetett felhasználók elindíthassanak bizonyos védett programokat, amelyekhez különleges jogosultságok szükségesek, s mindezt anélkül, hogy az adott felhasználónak meglenne ez a különleges joga. Ez a leggyakrabban akkor fordul elő, amikor bizonyos folyamatokhoz – például adatbázis mentéséhez – root jogosultságra van szükség, tehát csak a rendszergazda végezheti. A rendszergazdának azonban a legritkább esetben feladata az ilyen folyamatok indítása, tehát el kell látni a megfelelő felhasználót az adott jogosultsággal. Ez azonban veszélyes, mert ezzel a felhasználó akár szándékosan, akár figyelmetlenségből kárt okozhat. Ősidők óta létezik a problémára egy megoldás: a `setuid/setgid` használata, amelyet az egyes fájlokra határozhatunk meg Posix környezetben. Ha pl. a `setuid` bit be van állítva egy állományra, akkor azt bárki futtatja, a program azon felhasználó jogaival fog futni, akié az állomány. Tipikus példája ennek a `passwd` parancs, amely azáltal tudja írni a jelszófájlt, hogy a parancs tulajdonosa a root felhasználó, s be van billenve a setuid bit. Nem nehéz azonban rájönni, hogy ezt alkalmazva mondjuk az adatbázis-mentés példára azt kapjuk, hogy minden (legalábbis nagyon sok) felhasználó tudja futtatni az adott parancsot, nem csak az, akit a rendszergazda szeretne.
A probléma tehát az, hogy nem lehet elég finom felbontásban engedélyezni bizonyos parancsok futtatását ezzel a módszerrel.

A megoldás: Sudo

Ennek a kis programocskának a segítségével lehetőségünk nyílik arra, hogy az előre meghatározott szabályok szerint bizonyos felhasználók bizonyos programokat root vagy más felhasználó nevében futtathassanak. Az, hogy ki, mit és hogyan futtathat, a program beállítási fájljában kell rögzítenie a gép rendszergazdájának, és ezzel máris megoldódott a problémánk: akár felhasználónként és parancsonként egyesével is meghatározható, hogy ki és mit indíthat. A dolog a hétköznapi használatban úgy működik, hogy a parancs helyett a `sudo <parancs>` utasítást adjuk ki az értelmezőnek. A Sudo ekkor ellenőrzi a felhasználó és a parancs összerendelését, s a megfelelő feltételek mellett elindítja a programot úgy, mintha azt valóban az adott parancs futtatására jogosult személy tette volna, s a futás végén visszatér az elindított parancs visszatérési értékével.

A programról

A Sudo alapértelmezetten része szinte minden ma használatos terjesztésnek, így tehát csak annyi a dolgunk, hogy a rendszer csomagkezelőjével megkeressük, és hozzáadjuk a telepített összetevők sorához. Ha valamilyen oknál fogva ez nem sikerülne, még mindig letölthetjük a legfrisseb változatot az alkalmazás honlapjáról (➲ http://www.courtesan.com/sudo/) – igaz, csak forrás formájában. A binárisok csak nagy kereskedelmi Unix változatokhoz érhetők el.
A telepítésről tehát ennyit. A Sudo beállításai az általam használt Debian, Red Hat ill. SuSE terjesztésekben kivétel nélkül a */etc/sudoers* fájlban találhatók. Nekünk szinte mindig ezzel a fájllal kell majd dolgozni ahhoz, hogy testreszabjuk a programunkat.

A Sudo használata

A már fentebb említett `sudo <parancs>` stílusú használat esetén alapértelmezetten meg kell adnunk a saját felhasználói jelszavunkat. A helyes jelszó megadása esetén egy 5 perces érvényességű jegyet kapunk, ami azt jelenti, hogy a parancs következő 5 percben történő ismételt futtatása esetén nem kell a jelszót újra beírni. Ezzel a megoldással szerették volna a program írói megakadályozni azt, hogy egy otthagyott parancsértelmezőbe illetéktelenek bármit beirkálhassanak. A jegy érvényességének ideje természetesen átállítható, mint ahogyan a jelszó megadásának kötelező volta is, akár minden parancsra egyesével – mindezt a sudoers fájlból. A program nevéből is adódóan (SUperuser DO!) alapértelmezetten root-ként próbálja meg futtatni a kívánt parancsokat, de ettől a megfelelő kapcsolók használatával eltérhetünk. A lényeg az, hogy ha nem rendszergazdaként használjuk a Sudo-t, akkor minden kiadott parancs mögött kell, hogy legyen valami a */etc/sudoers* fájlban, különben a program nem fog lefutni.
A Sudo egyébként igen aktívan naplóz. Naplózza a próbálkozásokat és a sikertelen futtatási kísérleteket, de igény szerint rögzíti a sikeres próbálkozásokat is. Ha valaki olyan próbálkozik, akinek nincs joga a sudoers szerint az adott parancshoz, még levelet is küld – alapértelmezetten a root felhasználónak. A programnak egyébként rengeteg átállítható alapértelmezett beállítása van.
Hogy megtudjuk mennyi, adjuk ki a `sudo -L` parancsot.
S ha már a parancsoknál tartunk, nézzünk meg egy-két gyakrabban használt darabot.

`sudo -l`: Ezzel a paranccsal kilistázhatjuk, hogy nekünk, mint felhasználóknak milyen parancsok futtatásához van jogunk a sudo segítségével.
`sudo -v`: Felülírja a Sudo-tól legutóbb kapott jegyet, amely ugye a futtatás során jön létre, majd avul el, stb. Ha szükséges, bekéri a jelszót. Ezzel gyakorlatilag azt jelezhetjük a sudo számára, hogy még a gép előtt vagyunk, ezáltal a jegyünk újabb n percig érvényes – ugyanez történik egyébként akkor is, ha futtatjuk az adott parancsot.
`sudo -k`: Érvényteleníti az jegyet, amit a legutóbbi futtatás során kaptunk, tehát más még véletlenül sem indíthatja el az adott parancsot, ha esetleg kijelentkezünk, vagy otthagyjuk a terminált.
`sudo -b <parancs>`: Ennek hatására a kért parancs a háttérben fog futni. Ez akkor lehet hasznos, ha valami hosszan futó folyamatot indítunk, de közben szeretnénk visszakapni a promptot.
`sudo -u <felhasználó> <parancs>`: Ez esetben nem rootként próbálja meg futtatni a parancsot, hanem a megadott felhasználó nevén.
Ezeken túlmenően még számos kapcsolót használhatunk, héjprogramot jelölhetünk ki, másik sajátkönyvtárat adhatunk meg, és még sorolhatnám. A kedves olvasó a program telepítése után a sudo kézikönyvben (`man sudo`) olvashat róla.

A Sudo testreszabása

Mint már említettem, a Sudo testreszabása a */etc/**sudoers*** fájlon keresztül történhet. A programnak része egy olyan parancs is, amellyel ezt a fájlt lehet szerkeszteni, ez pedig a `visudo`. Gyakorlatilag a rendszer alapértelmezett szövegszerkesztőjét hívja meg, ám láthatatlanul azért csinál mást is a háttérben: Lezárja a ***sudoers*** fájlt a többi folyamatok elöl, így azok nem férhetnek hozzá, tehát nem fordulhat elő az, hogy egyszerre ketten egymásnak ellentmondásos értékekkel töltik fel az állományt. Ezen túl a szerkesztés után a parancs ellenőrzi a fájl tartalmát, és csak akkor menti el, ha nincs benne szintaktikai hiba. Ha van, akkor újra szerkeszthetjük a fájlt, vagy eldobhatjuk a változtatásokat, kivételes esetben megkérhetjük rá, hogy a szintaktiai hiba ellenére is mentse el a változtatásokat.
Maga a ***sudoers*** egyébként az EBNF (Extended Backus-Naur Form) nyelvet használja, ezen a nyelven kell tehát mondatokat a ***sudoers*** fájlba beírni, s ezt tudja értelmezni a program. Nekünk szerencsére nem kell megijednünk ettől a leírónyelvtől, ugyanis egyrészt nagyon egyszerű, másrészt a mindennapi használat során a példamondatok átalakítása bőven elég ahhoz, hogy elkészítsük a számunkra szükséges kifejezéseket. Ha szükségesnek érezzük, a nyelv gyorstalpalója megtalálható a ***sudoers*** kézikönyvében (`man sudoers`), sok más egyéb hasznos tudnivaló mellett.
A fájl kétféle adatot tartalmaz. Egyiküket alias-nak nevezi a leírás – ezek valójában globális változók. A globális változók egy részével felhasználói ill. parancscsoportokat hozhatunk létre, másik részével beállíthatjuk a már fentebb említett alapértelmezett értékeket: mi legyen az alapértelmezett héjprogram, mi legyen az alapértelmezett sajátkönyvtár, és még sorolhatnám...
A másik csoportja az adatoknak a felhasználókra vonatkozó előírások. Ezek mondják meg, hogy egy adott parancsot mely felhasználó indíthat el és milyen körülmények között.
A ***sudoers*** fájlban használt leírónyelv lehetővé teszi, hogy a kialakított felhasználói csoportokkal, munkaállomás-csoportokkal, parancs-csoportokkal nagy számú felhasználótábort is könnyedén kezelhessünk, mi több a munkaállomás-csoportok alkalmazásával az is megoldható, hogy ugyanazt a ***sudoers*** fájlt használjuk egy hálózat több gépén is. Ez a gyakorlatban nagyjából a következőképpen működik: megadjuk, hogy mely munkaállomáson futtatható csak az adott parancs, ezáltal ha másik gépre másoljuk, ott az azonos nevű felhasználó nem fogja tudni indítani az adott parancsot. Ezt általánosítsuk most gondolatban mind a parancsok terében, mind a fizikai elhelyezkedés terében, s máris megkapjuk egy központosított rendszer körvonalait, amelyben egy helyről szabályozható igen finom felbontásban az egész géppark felhasználótábora.
Végezetül nézzünk néhány egyszerű példát, amire nekünk, egyszerű felhasználóknak is szüksége lehet. Mindenekelőtt vizsgáljuk meg, hogyan is néz ki egy felhasználóra vonatkozó előírás:

```
<felhasználó> <mely gépeken futtatható> =
➥[(<milyen néven futtatható>)]
<parancs1>,<parancs2>,....
```

Mint látjuk, tényleg nem valami bonyolult dolog. Amit érdemes még tudni: létezik egy `ALL` kulcsszó, amely annyit tesz a mondatban is, hogy 'minden esetben'. Említettem még, hogy kikapcsolható a jelszó bekérése a Sudo használata esetén. Ez az első parancs elé biggyesztett `NOPASSWD`: kulcsszóval érhetjük el. Ha azt szeretnénk, hogy csak bizonyos parancsok esetében ne kelljen megadni jelszót, akkor a 'bizonyos parancsok' felsorolása után a többi parancs elé tegyük a `PASSWD`: kulcsszót. Ezzel körülbelül el is mondtam mindent, amire otthoni, vagy egyéb kisebb környezetben szükségünk lehet, lássuk a példákat:
`bela ALL = (ALL) ALL`: Ez a sor a sudoers-ben bela-nak gyakorlatilag root jogot ad anélkül, hogy Béla ismerné a root-jelszót. Minden parancsot futtathat mindenki nevében, minden gépen.
`bela ALL = (ALL) NOPASSWD: ALL`: Ez a sor akkor jó, ha otthon a munkaállomást felhasználóként használjuk, de gyakran szükséges root jog az egyes parancsokhoz. Ilyenkor kényelmetlen volna mindig su parancs után begépelni a jelszót, stb, ezért beírhatjuk a fenti sort a */etc/**sudoers***-be, s kényelmesen gépezhetünk egész nap.
`bela konyveles = /usr/bin/db_backup`: Béla a 'konyveles' nevű gépen futtathatja az adatbázis-mentést elvégző scriptet root-ként, de ezen kívül semmi mást, és meg kell adnia a parancs kiadása után a jelszavát. Ezek alapján már bárki könnyedén behelyettesítheti a kívánt felhasználó-parancs összerendeléseket, s hozzáolvasva a felhasználói leírásokból (man) már egész kis rendszer kialakítására képes.

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

GRUB rendszertöltő – az alapoktól kezdve (1. rész)

A LILO letaszíttatott trónjáról, lássuk az új uralkodó képességeit.

Sokan megfeledkeznek róla, de a rendszertöltő épp olyan fontos a működés szempontjából, mint maga a rendszermag. A rendszertöltő ugyanis a legelső felhasználói program, amely a gép indítását követően elindul. Feladata az operációs rendszer alapvető összetevőinek betöltése, majd a vezérlés átadása a már ébredező operációs rendszernek. Azt gondolhatnánk, hogy ez nem nagy feladat, és valóban: egy rendszermag betöltése nem a világ. Gondoljunk bele azonban abba, hogy az esetek többségében több eltérő felépítésű operációs rendszerünk van, melyeket felváltva szeretnénk használni – s mindeközben jó lenne, ha nem kellene egynél több rendszertöltőt alkalmazni. Ezen kívül minden haladó felhasználó vágya, hogy ne kelljen elpusztulni a rendszertöltő beállítása közben, egyszerű legyen, és ha kell mindenféle extra trükköt meg lehessen vele csinálni, vészhelyzetben fel lehessen éleszteni a gépet, és ehhez ne kelljen fejből tudnia, hogy mi merre található a gépen. A fentiek egy nagy rugalmasságot biztosító, mindenféle környezetben működő, sokféle operációs rendszert indítani képes, „felhasználóbarát" rendszertöltőt igényelnek. Mindezt egyben megtestesíti a GRUB (***GRand Unified Bootloader*** – Nagyszerű, Egységesített Rendszertöltő), amely ugyan még csak a 0.94-es változatánál tart, máris átvette az uralmat a régi, meglehetősen merev LILO-tól.

A GRUB tulajdonságai

Alapvető különbség a LILO-val szemben, hogy a GRUB nem kifejezetten Linux betöltéséhez íródott, hanem ***„Multiboot Standard"***-nak megfelelő operációs rendszerek indításához, mint például a GNU/Hurd. Ezek mellett természetesen be tudja tölteni a Linux és mindenféle BSD rendszereket is – és ezzel a sornak még mindig nincs vége. Hasonlóan elődjéhez a LILO-hoz, ez is használja a láncolt betöltés (***chainload***) módszerét, amellyel képes betölteni mindenféle kereskedelmi operációs rendszerek magját. Alapvető előnye minden eddigi rendszertöltőhöz képest, hogy a GRUB közvetlenül ismer jó néhány fájlrendszert, ahonnan fájlszinten képes elérni a betöltendő modulokat, rendszermagot. Ennek az előnye, hogy nem igényli a betöltendő fájlok fizikai helyének ismeretét, így nem kell folyton újratelepíteni, ha új rendszermagot fordítunk. Ha a régi nem indulna el, akkor egy egyszerű parancssor segítésével tetszőleges, a merevlemezen megtalálható rendszermagot a gép indításakor közvetlenül betölthetjükk. Ha pedig már itt tartunk, tegyünk említést a GRUB méltán népszerű héjprogramjáról. Ez a rendszerindításkor egy menüvel párhuzamosan a felhasználó rendelkezésére áll. Az elérhető néhány tíz parancs segítségével szinte minden problémát orvosolni tudunk – még a GRUB azonnali, más eszközre való telepítése is megoldható.

Említettem a menüt: a LILO-hoz hasonló szöveges menü áll rendelkezésünkre, amely az előre beállított paramétereknek megfelelően tartalmazza a betölthető operációs rendszerek nevét és indításának módját.

Mi az a láncolt betöltés?

A láncolt betöltés, más néven *chainloading* technika azt jelenti, hogy a rendszerbetöltő képes átadni a vezérlést más, már előre telepített rendszertöltőknek, épp úgy, mintha BIOS hívás kezdeményezte volna futtatásukat, ezáltal képes elindítani olyan operációs rendszereket is, amelyek rendszertöltési módszerét nem ismeri. Tipikusan ilyen például a Windows összes változata. Alapértelmezetten a betöltő ilyenkor az adott lemezrész (*partition*) betöltőrészében (*bootsector*) kap helyet, ide ugrat a GRUB is, ezáltal képesek vagyunk nemcsak Windows-t, de például OS/2-t is indítani a GRUB használatával.

A Multiboot szabvány ("Multiboot Standard")

Mindenegyes operációs rendszer saját rendszerbetöltőt, s hozzá saját betöltési technikát alakít ki, melynek következtében a területen rettenetes a káosz. Ezt a problémát próbálja megoldani a *Multiboot* szabvány a népszerű, szabad operációs rendszerek körében. A szabvány olyan egységes felületet határoz meg a rendszertöltő és az operációs rendszer között, amelynek használatával minden, a szabványnak megfelelő rendszertöltő képes elindítani minden, a szabványnak megfelelő operációs rendszert. A szabvány nem írja elő, hogy hogyan kell működnie a rendszertöltőnek, csupán azt, hogy milyen felület igénybevételével történhet az operációs rendszer betöltése.

A GRUB felépítése és működése

A GRUB három, egymástól élesen elhatárolható fokozatból (***stage***) áll: az első rész egy 512 bájt méretű programkód, amely a merevlemez fő rendszerindító területére (***Master Boot Record*** – MBR) íródik. Ez semmi mást nem csinál, mint betölti a következő fokozatot, amely hivatalosan a másfeledik sorszámot viseli. Ez még mindig egy viszonylag kis méretű rész, amely azonban már képes értelmezni egy adott fájlrendszert, és ezáltal el tudja érni a merevlemez adott lemezrészén található második fokozatot. Ez az utolsó fokozat tartalmazza tulajdonképpen magát a GRUB-ot. Itt van kódolva az összes fájlrendszer ismerete, a tényleges rendszerbetöltő, valamint a már emlegetett héjprogram, amellyel képesek vagyunk parancsokat kiadni a rendszerbetöltés kényes folyamatát szabályozva ezzel. Azért kell ilyen apró szakaszokra bontani a rendszertöltőt, mert az MBR-ben, illetve közvetlenül ezt követően nagyon kevés hely van, és nem férne el egy ekkora tudású program. Az MBR mérete pontosan 512 bájt, ide kerül hát az a program, amelyik tudja, hogy hol található az ő folytatása. A másfeledik fokozat még mindig korlátozott hellyel bír, ide sem fér el a teljes GRUB, de az már igen, hogy képes legyen megkeresni fájlszinten az adott lemezrészen, így nem függ az utolsó fokozat fizikai helyétől a rendszertöltő terület tartalma. A harmadik fokozat pedig már akármekkora lehet – figyelembe véve természetesen a rendelkezésre álló memória adta korlátokat.

A GRUB által támogatott fájlrendszerek

DOS FAT16/12, FAT32, Minix fs, ext2fs, ext3fs, ReiserFS, JFS, XFS, BSF FFS, VSTa fs, hálózatról történő rendszertöltés TFTP protokoll segítségével

A gép bekapcsolása után...

Először a gép BIOS-a betölti az MBR-ben található 512 bájtos kódot, amely jelen esetben a GRUB első fokozata. A fokozat hivatkozik az őt követő második fokozatra. Ez a rész nem mindig töltődik be, de ha meghívásra kerül, akkor ez lesz az a fejezet a történetben, amely ismeri azt és csak azt a típusú fájlrendszert, amilyenre a GRUB többi része telepítve van. Nem nehéz kitalálni, hogy minden ismert fájlrendszerhez létezik egy fokozat, s telepítéskor a megfelelő kerül a helyére. Az utolsó fokozat pedig maga a rendszertöltő, benne a legfontosabb résszel: a vezérlést átadó programkóddal. A telepítés során, ha nem használjuk a másfeledik fokozatot, akkor a második fokozat fizikai elhelyezkedése az első fokozatban rögzítődik, ellenkező esetben a másfeledik rész fizikai címe kerül az MBR-be, s a másfeledik részben csak az utolsó fokozat elérési útja található meg.

A GRUB újításai a menü és a héjprogram

A GRUB-ban található héjprogram egyrészről lehetővé teszi, hogy nekünk ne kelljen részletesen ismernünk a gépünkön található rendszereket pusztán azért, hogy hiba esetén is el tudjuk indítani a gépünket; másrészről azt is megengedi, hogy végrehajtsunk bizonyos műveleteket anélkül, hogy a rendszerünk elindult volna. Ez igencsak meglepő annak ismeretében, hogy eddig a rendszertöltés folyamán gyakorlatilag nem volt beleszólásunk az események alakulásába.

Az előző rendszerleállítást megelőzően kellett a betöltési jellemzőket precízen megadni, ellenkező esetben sanszos volt, hogy a gépünk nem indul többé. A GRUB a héjprogramja segítségével nem csak a betöltési paraméterek részletes átállítását teszi lehetővé, de megenged sok-sok rendszerszintű hívást is, mintha csak egy konkrét operációs rendszert használnánk. Ez a GRUB felhasználói oldalának alacsony szintje.

Minden ebben a parancssorban kell, hogy „lefusson". Kényelmetlen lenne azonban minden rendszerindítás során kézzel megadni, hogy melyik lemezrészről szeretnék betölteni, s milyen jellemzőkkel, ezért létezik erre a célra egy menü. Tudom, azt írtam, ez egy újítás, pedig ilyen már a LILO-nál is létezett. Van azonban egy elég nagy különbség: a GRUB esetében a menüpontok nincsenek „beledrótozva" a rendszertöltő területen található kódba. A GRUB ugyanis fájlszinten éri el a merevlemezen található menüállományt. Ez tartalmazza az általunk előre megadott indítható operációs rendszereket és még más beállításokat is. Az egyes menüpontok valójában parancsgyűjtemények, gyakorlatilag ide írhatjuk be előre azokat a parancsokat, amelyeket a rendszerindítás során kellene mindig kiadnunk, tehát a menüpont kiválasztása során a szükséges parancsok automatikusan „begépelődnek", aminek hatására megtörténik a rendszer betöltése – ennyi a menü titka. Ha van menü definiálva, akkor alapértelmezetten az jelenik meg induláskor, ha pedig nincs, akkor a GRUB „parancssor". Előző esetben is lehetőségünk van váltani a menüs és parancssoros nézet között, így válik lehetővé, hogy akár minden indítás során hegeszteni tudjuk a masinánkat.

Rendszertöltés a hálózatról?

Igen, a GRUB ezt is tudja. A parancssor lehetővé teszi, hogy akár kézzel, akár DHCP-n keresztül beállítsuk a gépünkben található hálózati eszközt, megadjuk a boot-kiszolgálót, ahol a rendszermag található, ezt követően pedig TFTP protokollon keresztül letölthessük az adott rendszermag-állományt, s el is indítsuk azt a gépünkön. Ez a megoldás akkor lehet hasznos, ha központilag szeretnénk tárolni, cserélgetni a rendszermagot, hogy ne kelljen az adott géphez ülni azért, hogy változtassunk a beállításokon. Akkor a legjobb megoldás, ha sok ugyanolyan gépünk van, és mindegyik ugyanazt a rendszermagot használja – ekkor elég a kiszolgálón kicserélni a rendszermag-állományt, s azonnal minden gép használni tudja.

Hogyan tovább?

Most , hogy ismerjük a GRUB képességeit, fel tudjuk mérni, hogy mire is van belőle szükségünk. Cikkünk második részében megnézzük, hogyan tudjuk telepíteni, használatba venni és testre szabni; megismerkedünk a menüszerkesztéssel, a héjprogram használatával, s az egyéb hasznos beállítási lehetőségekkel.

Komáromi Zoltán

Linuxos kiszolgálót mindenkinek! (9. rész)

A SuSE Linux, mint kiszolgáló, kisvállalati és otthoni környezetben.

Cikksorozatom eddigi részeiben a Kedves Olvasó megismerkedhetett sok hasznos szolgáltatással, amellyel a felhasználóinkat elkényeztettük. Most nézzünk egy újabb hasznos szolgáltatást, amellyel a felhasználókat fizikai helyüktől függetlenül hálózatba tudjuk fogni. Mostani cikkemben a virtuális magánhálózatokkal (VPN – ***Virtual Private Network***) foglalkozunk.

Egy kis elmélet

A VPN egy olyan szolgáltatás, amellyel bármilyen meglévő hálózati infrastruktúra felett saját hálózati környezetet tudunk kialakítani. Ennek segítségével megtehetjük, hogy meglévő hálózatunkat daraboljuk egymástól látszólag független hálózati egységekre, de azt is, hogy a felhasználóink a rendszerben számukra elérhető szolgáltatásokat az Interneten keresztül is biztonságos módon vehessék igénybe – mintha csak a helyükön ülnének. Ebben az esetben számukra teljesen átlátszó módon bekapcsoljuk őket a hálózatba, így a gépük olyannak fog tűnni, mintha az tényleg a lokális hálózatban lenne. A megoldás előnye, hogy sokkal biztonságosabb, mintha a tartalmakat az interneten tennénk hozzáférhetővé. Így ugyanis csak az férhet hozzá, aki a VPN elérésére jogosult, ráadásul az a megoldás kétirányú elérést tesz lehetővé. Tehát nem csak a VPN-en keresztül a hálózatba lépő felhasználó használhat erőforrásokat, hanem a hálózat tagjai is használhatják az Interneten keresztül VPN-nel csatlakozó felhasználók erőforrásait és szolgáltatásait.

És a gyakorlat

Ejtsünk pár szót a VPN működésének megvalósításáról. Amikor VPN kiszolgálót létesítünk, nem árt ha tisztában vagyunk azzal miként is valósul meg külső gépek bekapcsolása a lokális hálózatba. Mivel a VPN-be csatolt külső gépek potenciális kockázatot jelenthetnek, a tűzfalunk hangolását és a szolgáltatások elérésének korlátozását a VPN működéséhez kell igazítanunk.

A következőkben tárgyalt megoldások mindegyike úgy működik, hogy a VPN kliensek által elérhető publikus kiszolgálóra telepítjük a VPN démont, amelynek feladata, hogy a használt kiszolgálótól függően adott kapun várakozzon és figyelje a kapcsolódó klienseket.

Amint egy kliens megszólítja a szervert, az valamilyen azonosítási folyamat után becsatolja a klienst a hálózatba, mégpedig úgy, hogy mind a kliens, mind a kiszolgáló oldalán megjelenik egy virtuális hálózati csatlakozó, amelyen keresztül a kiszolgáló és a kliens bonyolítani tudja a kommunikációt. Az új csatlakozó megjelenése a rendszerben legkevesebb annyit jelent, hogy a gépek közötti elérések biztosításához módosítanunk kell a rendszerek útvonal választási tábláit (***route tables***), hiszen a VPN kiszolgáló a LAN és a VPN kliens között egy átjárót (***gateway***) fog jelenteni. Ideális esetben ezen az átjárón üzemeltetünk egy tűzfalat is, amely a VPN felől érkező klienseket az Internet felől érkező klienseknél megbízhatóbbnak kezeli, de semmiképpen nem biztosít minden olyan jogot, amit egy olyan gép kaphat, amely fizikailag a lokális hálózatban foglal helyet. Összefoglalva, a VPN kiszolgáló- és kliensoldalán egy-egy új, virtuális hálózati csatoló jelenik meg, amelyek a fizikai hálózati csatoló egy adott kapuját használva valósítják meg az adatátvitelt. Ezt az egy kapun történő kommunikációt nevezik alagutazásnak (***tunneling***). Az azonban, hogy a két VPN fél közötti adatfolyamot egy csatornába kényszerítettük még nem biztosíték arra, hogy ez a kommunikációs csatorna megbízható. Ugyan nem kötelező, de nagyon ajánlott valamilyen titkosítás használata, ugyanis ezzel nagyban megnehezíthetjük az esetleges támadó dolgát. A titkosításra ismertek szimmetrikus és nyílt kulcsú titkosítási megoldások is.

VPN megoldások

Ha úgy döntünk, hogy VPN-re van szükségünk és ezt egy Linux kiszolgáló segítségével szeretnénk megvalósítani, akkor minden bizonnyal leülünk a gép elé és elkezdünk keresgélni az Interneten. Ha ezt tesszük, akkor biztosak lehetünk abban, hogy találunk is jó néhány megoldást a problémára. Viszont ekkor felmerül a következő kérdés: Melyiket válasszuk? Nos, ez már nem ilyen egyszerű. Tulajdonképpen minden megoldásnak vannak előnyei is és hátrányai is, így – mielőtt valamelyik megvalósítás üzembe helyezése mellett döntünk – nem árt átgondolni, hogy mire is van szükségünk tulajdonképpen.

A VPN-t használhatjuk arra, hogy különböző operációs rendszerrel megáldott külső munkaállomásokat kapcsoljunk egy hálózatba, úgy, hogy azok csatlakozása a használt operációs rendszertől független legyen, és lekezelje azt a problémát, hogy a kliens helye, IP címe nagy valószínűséggel nem állandó – például egy notebook esetében. Előfordulhat ugyanakkor az is, hogy nem külső munkaállomásokat szeretnénk becsatolni, hanem több, egymástól független lokális hálózatot szeretnénk egy nagy virtuális

hálózattá összekötni, vagy éppen egy nagy lokális hálózatot szeretnénk több kis virtuális hálózatra osztani. Utóbbi esetben nagy valószínűséggel olyan kiszolgálókkal dolgozhatunk, amelyeknek az IP címe állandó és – optimális esetben – a gépek operációs rendszere is azonos: esetünkben ilyen például valamelyik Linux változat.

Noha első ránézésre nincs nagy különbség a két említett probléma között, azt azért jó tudni, hogy a két összeállításra más-más megoldás az optimális.

Nézzük meg a kínálatot, milyen megoldások állnak rendelkezésünkre. Használhatjuk VPN kialakításához a Poptop csomagot, az OpenVPN csomagot, vagy a Freeswan IPSec implementációját. Természetesen találhatunk egyéb megoldásokat is, ezek az általam használt és bemutatni kívánt rendszerek.

A Poptop egy Linuxos PPTP kiszolgáló. A PPTP (Point to Point Tunneling Protocol) egy Microsoft fejlesztésű VPN megoldás, amellyel meglehetősen egyszerűen csatolhatunk Windowsos és Linuxos PPTP klienseket a virtuális magánhálózatunkba. Amennyiben könnyen, gyorsan konfigurálható és egyszerűen használható VPN megoldást keresünk, akkor ez nagyon jó megoldás lehet. Különösen jól használható olyan környezetben, ahol a VPN kliensek valamilyen Windows operációs rendszert futtató gépek.

A Poptop rendszer egy PPTP protokollt társít a Linux PPP kiszolgálójához, így a rendszer a telefonos betárcsázáshoz teljesen hasonló módon konfigurálandó.

A Poptop szerverről bővebben a ➲ http:\\www.poptop.org címen olvashat az érdeklődő.

Az OpenVPN egy könnyen használható, SSL támogatást biztosító VPN démon, amelyhez létezik Linux és Windows oldali szoftver is, így ez is használható heterogén rendszerekben. Posix rendszereken való használatnál az OpenVPN csomag telepítése után létre kell hoznunk a megfelelő virtuális hálózati csatlakozókat, amelyeken keresztül a kommunikáció folyik majd. Ezután a VPN csatorna két oldalán el kell indítani egy parancssoros démont, amelyet a megfelelő paraméterekkel ellátva felépül a hálózatokat összekötő csatorna. Windowsos kliens esetében a projekt weboldaláról letölthető telepítőt kell lefuttatni, amely utána a rendszerben virtuális csatlakozókat fog létrehozni, amely csatlakozókon keresztül tudjuk a VPN erőforrásait elérni.

Az OpenVPN tartalmaz SSL támogatást, ami annyit jelent, hogy a kommunikációs csatorna forgalmát mindkét oldalon nyílt kulcsú titkosítási módszerrel titkosítjuk, így a csatorna forgalmának lehallgatása gyakorlatilag lehetetlenné válik. A módszer hátránya, hogy a kódolás erős gépet, vagy cél-eszközt igényel. Az OpenVPN projekt megtalálható az ➲ http:\\www.openvpn.sourceforge.net weboldalon.

Az IPSec (***Internet Protocol SECurity***) egy erős biztonsági megoldásokkal ellátott protokoll, amelyet az IETF (***Internet Engineering Task Force***) fejlesztett ki és amely megoldás az IPv6-nak már alapszolgáltatása lesz. IPv4-es rendszerünket is felkészíthetjük az IPSec támogatására, ha a megfelelő rendszermag módosításokat elvégezzük, valamint telepítjük a felhasználói csomagokat.

A fentiekből is látszik, hogy az IPSec több mint egy VPN megoldás: ez egy alacsony szinten megvalósított biztonsági megoldás. Az IPSec protokollnak több implementációja létezik, ebből az egyik az általam említett Freeswan/IPSec.

Emellett léteznek egyéb fejlesztések is, így például a Windows operációs rendszerek egy a Microsoft által készített IPSec megoldást tartalmaznak. Mivel az IPSec hamarosan, az IPv6 bevezetésével és széles körű elterjedésével a mindennapok részévé fog válni, ezért érdemes megismerkedni vele, érdemes tudni miként működik.

A Freeswan/IPSec projektről az érdeklődők egy kiterjed és nagyon alapos dokumentációt találnak a ➲ http:\\www.freeswan.org címen.

Használat a gyakorlatban

Gyakorlati útmutatónk alkalmával a Poptop és az OpenVPN projektek kínálta megoldásokat tanulmányozzuk át részletesen. A Suse 9.0-s verziója már tartalmazza mindegyik szolgáltatást, ellentétben a 8.2 kiadással, amely csak az OpenVPN csomagot tartalmazza. Ettől még nem kell megijedni, a projektek weboldaláról minden esetben letölthetőek a legfrissebb telepítő készletek, így azok akármelyik Linux kiadáshoz telepíthetőek.

Suse 9.0 esetén a telepítés meglehetősen egyszerű. A YaST csomagtelepítőjében a megszokott módon megkeressük a kívánt csomagokat és telepítjük. Innentől kezdve a telepített rendszer konfigurálható, használható.

Poptop kiszolgáló beállítása

Először nézzük a Poptop PPTP kiszolgáló beállítását. Ehhez telepíteni kell a Poptop csomagot, valamint a PPP csomagot, és a rendszermaghoz PPP támogatást kell biztosítani. Amennyiben a PPP modul le van fordítva, úgy az */etc/**modules.conf*** állományban be kell jegyezni, amennyiben nincs a rendszermaghoz PPP támogatás, úgy a támogatás bekapcsolásával újat kell fordítanunk. (A Suse által telepített rendszermaghoz van PPP modul fordítva, így annak fordításával nem kell bajlódnunk. Amennyiben ADSL, vagy modemes kapcsolatot használunk, úgy a PPP meghajtó minden bizonnyal be van töltve, ugyanis az szükséges az előbb említett kapcsolatok kezeléséhez.)

Miután telepítettük a PPTP kiszolgálót néhány konfigurációs állományban módosításokat kell elvégeznünk. Ezen módosításokat most a dokumentáció alapján nézzük.

Amennyiben valamelyik állomány a mi rendszerünkben más helyen található, úgy ott az elérési utakat értelemszerűen módosítani kell. Az első módosítandó állomány a már említett */etc/**modules.conf***. Itt a következő modulok betöltését kell bejegyezni:

```
alias char-major-108 ppp_generic
alias tty-ldisc-3 ppp_async
alias tty-ldisc-14 ppp_synctty
alias ppp-compress-18 ppp_mppe
alias ppp-compress-21 bsd_comp
alias ppp-compress-24 ppp_deflate
alias ppp-compress-26 ppp_deflate
alias net-pf-47 ip_gre
```

A PPTP démon alapbeállításait az */etc/**pptpd.conf*** állományban találjuk meg. Ez az állomány nem tartalmaz túl sok beállítást, mindössze azt mondja meg, hogy a PPP beállítási állomány pontosan hol található, valamint egy IP-t ad a kiszolgáló VPN interfészének és egy IP tartományt ad a csatlakozó klienseknek.

Egy példa az */etc/pptpd.conf* állományra:

```
option /etc/ppp/options.pptpd
#debug
localip 192.168.0.1
remoteip 192.168.0.200-249
```

A fenti konfiguráció azt mondja, hogy a PPTP démon PPP állománya az */etc/ppp* könyvtárban az *options.pptpd* állományban van, a kiszolgáló a 192.168.0.1-es címet kapja és a kliensek a 192.168.0.200 és a fölötte lévő 50 címből gazdálkodhatnak. A debug kapcsoló bekapcsolásával elérhetjük, hogy a démon teleírja a naplót a futási állapot üzeneteivel, így adva egy nagyszerű eszközt az esetleges hibák elhárításához. A *pptpd.conf* állományban megnevezett PPP konfigurációs állományt az alábbi tartalommal lássuk el:

```
lock
#debug
name pptpd
nobsdcomp
proxyarp
ms-wins <hálózati-wins-kiszolgáló-ip-címe>
ms-dns <hálózati-dns-kiszolgáló-ip-címe>
```

Ezzel a beállítással egy titkosítás nélküli VPN csatornát létesítünk, ami ugyan működő megoldás, de nem egy biztonságos összeállítás. A Poptop PPTP kiszolgáló ellátható SSL titkosítással is, amennyiben a kiszolgálóhoz tartozó MPPE foltot is telepítjük. A foltozás letölthető a Poptop projekt weboldaláról is. Amennyiben telepítettük az MPPE és ChapMS kiegészítést, úgy az alábbi sorokat hozzáadva az *options.pptpd* állományhoz elérhetővé tehetjük a Windows kliensek által is titkosítást:

```
-chap
-chapms
+chapms-v2
mppe-40
mppe-128
mppe-stateless
```

A fenti szolgáltatások elérhetőségét, szükségességét az alábbi sorok értelemszerű beszúrásával állíthatjuk:

```
#refuse-pap
#refuse-chap
#refuse-mschap
#require-mschap-v2
#require-mppe
```

Ha végeztünk az *options.pptpd* állománnyal, akkor az */etc/ppp/chap-secrets* állományban megadhatjuk, hogy melyik felhasználó milyen jelszóval milyen gépről érje el a szolgáltatást. Ehhez mindössze egy sort kell felhasználónként beszúrni, ami a következőképpen néz ki:

```
felhasználó_neve pptpd "jelszó" <IP tartomány>
```

Pédául:

```
viktor pptpd "passwd" 192.168.0.100
```

Ezzel a viktor nevű felhasználó a passwd jelszóval a 192.168.0.100-as című gépről tud bejelentkezni. Amennyiben az IP cím helyére *-ot teszünk, akkor az az IP korlátozás teljes feloldását jelenti. Ha ezzel végeztünk, akkor nincs is más dolgunk, mint újraindítani a pptp kiszolgálót és már csatlakozhatunk is a géphez. Amennyiben a PPTP kiszolgáló tűzfallal védett, ne felejtsük el engedélyezni a 1723-as TPC és UDP portot. A Poptop kiszolgáló teljes dokumentációja és további érdekes és hasznos információk megtalálhatóak a projekt honlapján.

Az OpenVPN projekt

Az OpenVPN kiszolgáló telepítéséhez a YaST csomagtelepítőjével az *openvpn* csomagot kell telepítenünk. Ezután az */usr/sbin* könyvtárban megjelenik az *openvpn* nevű futtatható állomány. OpenVPN esetén alapesetben ezen az állományon kívül nem is lesz szükségünk semmilyen más konfiguráció elkészítésére, parancssorból paraméterezve tudjuk futtatni a kiszolgálót. Ez előtt azonban kell még tennünk egy-két előkészületi lépést. Amennyiben 2.4.7-esnél frissebb rendszermagot használunk – ami egyébként elég valószínű –, akkor telepítenünk kell a TUN/TAP meghajtót. Ha ez megvan, akkor létre kell hoznunk a */dev/net/tun* nevű eszközt az alábbi paranccsal `mknod /dev/net/tun c 10 200`, majd töltsük be a meghajtót a `modprobe tun` parancs futtatásával. Ha rpm csomagból telepítjük a programot, akkor az eszköz létrehozására nincs szükség, mert ezt a telepítő elvégzi. Amennyiben a kiszolgálón tűzfal fut, úgy ne feledkezzünk meg a tűzfalon a tun eszköz megfelelő beállításáról sem! Ha a fentiekkel mind elkészültünk, akkor jöhet amire vártunk, indíthatjuk a kapcsolatot. Ez, mint már említettük, parancssorból történik. Íme egy séma egy egyszerű, titkosítás nélküli VPN csatorna létrehozására:

```
openvpn --remote <ellenoldali-gép-hostneve> --dev
➥tun1 --ifconfig <helyi-tun-csatoló-IP-címe>
➥<távoli-tun-csatoló-IP-címe>
```

Nézzünk egy példát. Van két helyi hálózatunk, ahol az egyik hálózat átjárójának címe gw1.hu, míg a másik hálózat átjárójának címe gw2.hu, valamint a gw1.hu gépen 10.1.1.100 címet adunk a tun1 hálózati csatolónak, míg gw2.hu gépen 10.1.1.200-at. Ezután a gw1 gépen az alábbi parancsot kell futtatni:

```
openvpn --remote gw2.hu --dev tun1 --ifconfig
➥10.1.1.100 10.1.1.200
```

A másik, tehát gw2.hu gépen pedig a következő paranccsal építhető ki a kapcsolat:

```
openvpn --remote gw1.hu --dev tun1 --ifconfig
➥10.1.1.200 10.1.1.100
```

Mint látható a kapcsolat felépítése rendkívül egyszerű. Természetesen lehetőségünk van arra, hogy titkosított csatornát használjunk a kommunikáció során. Az OpenVPN támogat mind szimmetrikus, mind TLS alapú titkosítási módszereket, így mindenki választhat igényének megfelelően. Azok, akinek az érdeklődét sikerült felkelteni a VPN iránt, bőséges olvasnivalót találnak a témában a projektek honlapjain. Ezzel a cikkel a Suse Linux-ról szóló cikksorozatom végére értünk. Remélem sikerült kedvet csinálnom, nemcsak a Suse, de akármelyik Linux kiadás megismeréséhez.

Illés Viktor

A GIMP eszközkészlete

Sorozatunk aktuális részében megismerkedhetünk a GIMP bőséges eszköztárának további tagjaival és azok különféle beállítási lehetőségeivel. Ezek után két egyszerű példán keresztül bemutatom, hogy ezek az egyszerű eszközök, a megfelelő sorrendben alkalmazva milyen hatékony munkát tesznek lehetővé.

Az 1-es képen láthatjuk a program alapértelmezett eszközkészletét. Most csak azért került ide a kép, hogy olvasás közben ne kelljen ide-oda tekintgetve az olvasottak felét a monitorról, másik felét a papírról megérteni.

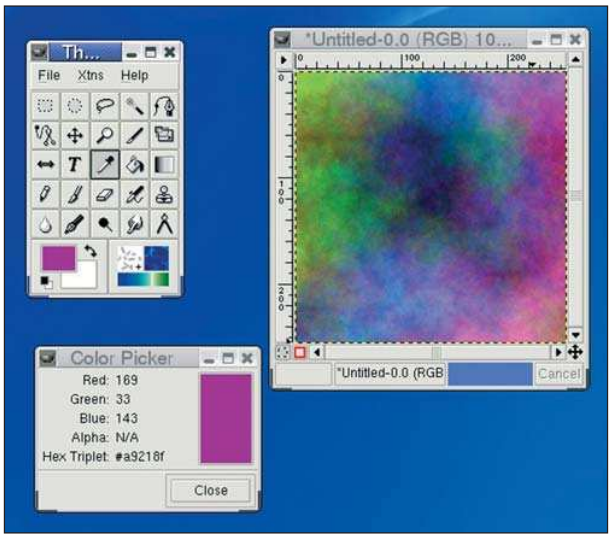
Minden eszköz esetében az eszközre jellemző tulajdonságokat elérhetjük és beállíthatjuk, ha az adott gombra duplán kattintunk a bal egérgombbal.

A kiválasztásra szolgáló eszközökkel már megismerkedtünk az első részben, így most folytassuk a sort a ***nagyítóval***. Már erről is esett némi szó, de érdemes megjegyezni, hogy a nagyítás nem csak pozitív előjelű lehet. A CTRL billentyű lenyomása mellett a képre kattintva, a nagyítás helyett kicsinyítést hajthatunk végre a GIMPben. Az eszköz beállításai között találhatunk egy bekapcsolható mezőt, amivel meghatározhatjuk, hogy a nagyítás során a program átméretezheti-e az ablakot.

A következő eszköz az ikonja alapján leginkább szikéhez hasonlítható. Nem meglepő a hasonlóság, ugyanis valóban egy vágásra használható eszközről van szó. A képből tetszőleges, téglalap alakú részeket vághatunk ki a segítségével.

Nagyon nagy hasznát vehetjük ennek a funkciónak akkor, amikor például kicsit ferdén olvastunk be egy képet a lapolvasóval, ferdén tartottuk a digitális fényképezőgépünket vagy csak egyszerűen a lényeget szeretnénk kiemelni egy képből. Ilyenkor jelöljük ki a vágóeszközzel a kívánt részt. Ekkor megjelenik egy párbeszédablak, ahol pontosan beállíthatjuk a kezdőpontokat és a méretet, majd a ***Crop*** gomb hatására a program kivágja a szükséges darabot. A méretet és a kezdőpontot a képen megjelenő kerettel is igényeinknek megfelelően változtathatjuk.

Fontos lehet megismerkedni a tükrözésre használható eszközzel, amit a kétfelé mutató nyíl ikon jelez. Alapállapotban a képet a függőleges tengelye körül fordítja el 180 fokkal, ha azonban lenyomjuk a CTRL gombot az egérgombbal együtt, akkor a program a vízszintes tengelyt veszi alapul. Már egy példába segítségével is láthatjuk, hogy mire lehet használni ezt az egyszerű megoldást. Aki olvasta a Blender sorozatot, annak rögtön eszébe juthat, hogy annak idején tengelyesen tükrözhető tárgyakat készítettünk, és azoknak csak egyik felét kellett modellezni. A GIMP-ben sincs ez másként, gondoljunk csak egy szívre, egy vázára, virágra vagy akár egy ablak megrajzolására.



1. kép A GIMP eszközkészlete

Szövegeljünk egy kicsit!

Felmerülhet a kérdés, hogy hogyan lehetne a GIMP segítségével megtervezni a cégfeliratunkat. Hogyan lehet magyarázó szöveget készíteni az oktatási anyagunkban szereplő képekhez? Egyáltalán lehet szöveget használ a GIMP-ben? Nos, a választ a 'T' betűt formázó gomb használatával kaphatjuk meg. Ha ezt a gombot választjuk, akkor az egérmutató átalakul a szövegszerkesztőkben használatos 'I' formájú mutatóvá és a képen kijelölhetjük a szöveg helyét. A szöveghelyzet megadása után egy betűtípust kell választanunk, ezt követően pedig a párbeszédablak alsó részén lévő mezőbe beírni a kívánt szöveget, ami rögtön a kiválasztott típussal jelenik meg, tehát tetszőlegesen válogathatunk, hogy megtaláljuk az igényeinknek megfelelő betűtípust. Ezután az OK gomb hatására a program elhelyezi a megadott helyen a szöveget és az kijelölt állapotban marad.

Ekkor még van arra lehetőségünk, hogy pontosítsuk a helyzetét, vagy új réteget hozzunk létre és arra helyezzük el.

Ha túl sok a szemét...

Milyen lehetőségeink vannak, ha egy képen bizonyos javításokat szeretnénk elvégezni? Ez több dologtól is függ. Ha például a javítandó területen hasonló mintázatok vannak (például szemét volt a park füvén és arra nincs szükség a képen), akkor használhatjuk a környező képrészleteket, és azokkal fedhetjük le a kívánt területet. Van azonban a képterületek másolásának másik módja is, amelyet majd a későbbiek során ismertetünk.

No de mi történjen akkor, ha nem hasonlók ezek a részletek, és kénytelenek lennénk képpontonként változtatni a színeket. Erre adhat megoldást a ***pipetta*** eszköz. Amikor ezt kiválasztjuk, és a képen egy pontra kattintunk, a GIMP megjelenít egy ablakot, amelyben többféleképpen is olvashatjuk a választott pont színét. Az ***1-es képen*** látható ez a párbeszédablak is. Fontos megjegyezni, hogy a pipetta használatakor az aktuális rajzolószín is megváltozik. Másik előnye ennek az eszköznek, hogy amikor weblapot készítünk, az átlátszó képek helyett használhatunk háttérrel rendelkezőeket, és a GIMP tizenhatos számrendszerben (ahogyan a HTML oldalakban szükséges) is kiírja a választott színt. Ennek alapján már könnyen beállíthatjuk a HTML oldal háttérszínét. A szükséges formában a ***Hex Triplet*** sorban látható a szín meghatározása.

Különös figyelmet érdemel a GIMP területkitöltő eszköze. Az ikonja leginkább egy festékes vödörhöz hasonlít. Alapvető működése szerint az éppen aktuális rajzolószínnel kitölti azt a területet, ahol lenyomjuk a bal egérgombot.

Fessük is ki...

Talán még nem szóltam arról, hogy hogyan lehet a GIMP-ben kiválasztani a használni kívánt színeket. Az eszköztáron látható két színes lapocska, aminek egy-egy sarka egy nyíllal van összekötve. Nos, mindkét lapocska az éppen használatban lévő színekre vonatkozik. Az előrébb lévő az előtérszín vagy rajzolószín, míg mögötte a háttérszín látszik. Az őket összekötő nyíl arra szolgál, hogy ezt a két színt felcseréljük.

Ha valami miatt meg szeretnénk változtatni valamelyik színt, akkor dupla kattintásra előbukkan egy párbeszédablak. Itt többféleképpen is megadhatjuk az új színt, használhatunk RGB értékeket, HSV értékeket és többfajta palettából az egérrel használatával is választhatunk. Ezek a beállítások a háttérszínére éppúgy alkalmazhatók, mint a rajzolószínre. Tehát a területkitöltés alkalmazható a megszokott módon, viszont a CTRL gomb lenyomása mellett a program a háttérszínt használja a színezéshez. További beállítási lehetőségekhez juthatunk ha kettőt kattintunk az eszköz gombján. Megadható az érzékenység, vagyis az, hogy az egérmutató alatt lévő képpont színéhez képest milyen mértékben vegye figyelembe a program a hasonló színeket. Például ha valamilyen árnyalást használtunk, akkor az alapértelmezett kitöltés szerint a halványabb részeket nem tudjuk befesteni a festővödör használatával. Ilyenkor az érzékenységet magasabbra állítjuk, majd megismételjük a műveletet és a GIMP ekkor már az elhalványuló körvonalakat is kitölti a megfelelő színnel.

2. kép A készülő logo

Természetesen nem csak egyszerű színekkel festhetünk, hanem alkalmazható az éppen aktuális kitöltő minta is. Ennek kiválasztása úgy történhet, hogy az eszköztárban, a színpaletta melletti mintázatra kettőt kattintunk, majd a megjelenő párbeszédablakból, kiválasztunk egy tetszőleges mintázatot.

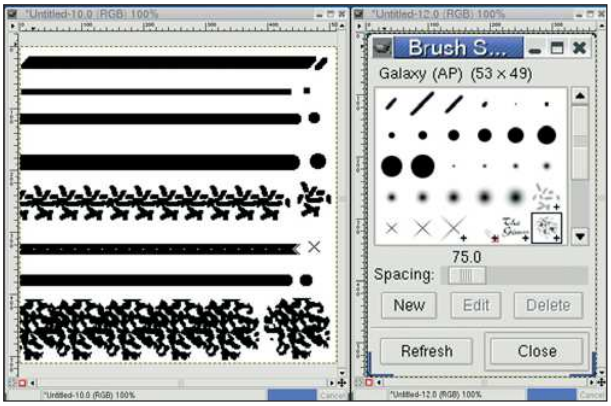
A következő eszközünk az átmenetek festésére szolgál és könnyen felismerhető az ikonja alapján. A GIMP-ben tetszőleges színátmenetet létrehozhatunk vagy a meglévőket is átalakíthatjuk. Kattintsunk duplán az eszköztárban az aktuális színeket mutató paletta mellett a színátmenetre. Szintén egy párbeszédablakból választhatjuk ki a beépített átmenetek egyikét, majd az ***Edit*** gomb segítségével igényeinknek megfelelően szerkeszthetjük.

Ha megelégszünk az egyszerűbb átmentekkel is, akkor a színátmenetfestő eszköz beállításainál választhatunk előtérből-háttérbe vagy háttérből-előtérbe tartó átmenetet. Itt adhatjuk meg azt is, hogy milyen formát kövessen a színátmenetes festés. A GIMP-ben többféle típust találunk, például egyszerű vonalmenti átmentet (***Linear***), gömbszerű átmenetet (***Spherical***), kúpszerű átmenetet (***Conical***), a kijelölés formájához igazodó átmenetet (***Shapeburst***) és spirál alakú átmenetet (***Spiral***). Mindezeket a beállítások ablakában a ***Gradient*** felirat melletti legördülő listából tudjuk kiválasztani.

Gyakorlat teszi...

Készítsünk most egy egyszerű feliratot, aminek a körvonalát kiemeljük más színnel. A felirat maga színátmenettel legyen kitöltve, és ne feledkezzünk meg a betűk belső részének kiemeléséről sem.

Készítsük el a feliratot a szöveges eszközzel és helyezzük el a kívánt helyre. Ezután a CTRL-L billentyűkkel jelenítsük meg a rétegek kezelésére szolgáló ablakot és az első gombra kattintva hozzunk létre új réteget. A létrehozott rétegen szín szerinti kiválasztással (***Kép menüje->Selection->Select by color...***) jelöljük ki a szöveget ismét. Most következhet az átmenetes kitöltés. Ezután növeljük meg a kijelölt területet a kívánt mértékben (1-2 pixel) a ***Kép menüje->Selection->***



3. kép Különféle keresztmetszetek

Grow menüpontok segítségével, majd hozzunk létre még egy új réteget. Ezt területet fessük ki olyan színnel, amivel a körvonal kiemelését szeretnénk viszontlátni. Az eddigi lépések alapján látható, hogy az átmenettel kitöltött felirat eltűnt, hiszen a körvonal kiemelésére szolgáló réteg eltakarja. A rétegek párbeszédablakában nyomjuk meg a lefelé mutató nyíllal jelzett gombot, így máris elértük a kívánt eredményt. Láthatjuk első átmenettel rendelkező, kiemelt körvonalakkal megjelenő feliratunkat. A 2-es képen a fenti lépések követhetők nyomon.

Fontos, hogy jól látható legyen. Pl. Két hasábra vagy hasonló. Ha nem oldható meg, akkor nem kell a kép és az előző bekezdés utolsó mondata sem.

Szabadkézi rajzoláshoz is többféle eszköz áll rendelkezésünkre. Az egyik a ceruza, ami kevesebb beállítási lehetőséggel főként egyszerű vonalas ábrák készítésére használható. Mindkét eszköz esetében érvényes, hogy ha a SHIFT billentyűt lenyomva tartjuk, akkor az utolsó pozícióból indulva egyenes vonalat húzhatunk. Utolsó pozíciónak az a képpont számít, ahol a vonalrajzolás előtt utoljára lenyomtuk a bal egérgombot. A rajzeszköz keresztmetszetét, vagyis a rajzolt alakzat formáját szintén az eszköztár alsó részén állíthatjuk be. A mintázatok gombja mellett található egy újabb gomb, amit alkalmazva (meglepő módon) egy párbeszédablakhoz jutunk. Ez látható a ***3-as képen*** (néhány mintával, melyek a ceruza eszközzel készültek), és innen tudjuk kiválasztani megfelelő formát.

A másik, szabadkézi rajzhoz használható eszköz az ecset. Szintén különféle keresztmetszetű darabokat használhatunk, azonban egy lényeges különbséget felfedezhetünk az ecset és a ceruza között. Az ecset alkalmazásakor az olyan formák, amiknek a széle elhalványul, valóban halványabb képet eredményeznek. Ez talán annak tudható be, hogy az ecsetek nem egységesen érintkeznek a vászonnal vagy papírral.

Köztudott az is, hogy az ecsetből hamarabb fogy ki a festék, mint ahogy egy ceruza elfogyna. A GIMP-ben ennek a festékfogyásnak is megvan a megfelelője. Az ecset beállításai között megadható a ***Fade*** érték, ami ezt a fogyást kívánja szimulálni. A másik érdekesség, hogy színátmenetes ecsetvonásokat is rajzolhatunk. A ***Gradient*** feliratnál beállítható, hogy milyen hosszú legyen egy-egy átmenet és a típus megadása után már rajzolhatunk is a szép ecsetvonásokkal.

Négyféle beépített típust találunk. A felső kettőt talán nem kell magyarázni. A harmadik, a fűrészfog-szerű ismétlés (***Loop sawtooth***), tulajdonképpen azt jelenti, hogy az átmenet végén a következő szín a színátmenet elején található lesz. A háromszög (***Loop triangle***) formájú átmenet esetén az utolsó szín elérésekor visszafelé jönnek a színek. Az elnevezéseket könnyebben megérthetjük, ha elképzeljük a háromszöget és a fűrészfogat. Ekkor a vízszintes tengely az idő-tengely, a függőleges pedig a színátmenet aktuális színének helyzete a teljes átmenetben.

Talán feltűnt a kedves olvasók egy részének, hogy ez utóbbi két eszköznél (és majd továbbiaknál is) a beállítások között találhatunk egy bizonyos ***Pressure Sensitivity*** feliratot.

A GIMP képes kezelni a rajzolótáblák nyomásérzékenységét, és ezt a kiegészítő információt különféle módon használhatjuk fel. Változtathatjuk vele a színek átlátszóságát (***Opacity***), az eszköz rajzolatának erősségét (***Hardness, Rate***), a méretét (***Size***) vagy a színátmenet éppen alkalmazott színét (***Color***). Természetesen ennek a lehetőségnek a használatához, olyan táblát kell használni, amit az XWindow rendszer is felismer és bemeneti eszközként kezelni képes. Ilyenek például a Wacom Intuos és a Graphire táblák, amelyek a Wacom IV és a Wacom V protokollt használják. Jelenleg néhány Linux változatban található támogatás az USB kapun csatlakozó eszközökhöz is.

Nem lehetne teljes értékű pixelgrafikus rajzolóprogram a GIMP, ha nem találnánk benne lehetőséget a változtatások visszavonására. Az egyik ilyen megoldás a szokásos visszavonás művelet, melyet a CTRL-Z billentyűvel érhetünk el.

A visszavonható lépések számát beállíthatjuk a már tárgyalt általános beállítások párbeszédablakban. Emlékeztetőül, ez az érték elérhető az ***Eszköztár ablaka->File menü->Preferences menüpont->Environment->Levels of undo*** beállításán keresztül.

A másik lehetőség, a ***radír*** használata. Szintén könnyen felismerhető az eszköztárban, ikonja hasonlít egy szokásos radírhoz. Amikor használjuk, azt fogjuk látni, hogy az éppen aktuális rajzoló formával radírozhatunk, és a kiradírozott területek háttérszínnel jelennek meg. Beállításai között találhatunk egy ***Hard edge*** nevű lehetőséget, amellyel a radírnak a formától függetlenül éles szélei lesznek. Például, ha elmosott szélekkel rendelkező formát használunk, de ezt a lehetőséget beállítjuk, akkor a halványodó körvonalakat a program figyelmen kívül hagyja. Így egészen pontos műveleteket végezhetünk, például egy beolvasott vonalas rajzon vékonyíthatjuk a vonalakat vagy teljesen kitörölhetjük a felesleges részeket.

Az ilyen lapolvasóval beolvasott vonalas rajzok esetén előfordulhat, hogy a ceruzával készült rajzot egy kicsit elmaszatoltuk rajzolás közben. Ez a papíron nem olyan feltűnő, nem is zavaró, mégis akadályozhatja a további számítógépes feldolgozást. A lapolvasó ugyanis az ilyen alig látható maszatokat is képes megkülönböztetni a fehér papírtól, és annak rendje és módja szerint be is digitalizálja őket. A kép használhatóvá tétele a számítógép feladata lesz, erre lássunk most egy egyszerű megoldást.

Először próbáljuk a kontraszt növelésével eltüntetni a maszatokat. Ezzel a megoldással már jó eredményt érhetünk el, de előfordul, hogy még mindig maradnak felesleges foltok. A következő lépés legyen a szín szerinti kiválasztás.

4. kép Igazított és nem igazított mintázat

Válaszuk ki alacsony tűréssel a nem kívánt színeket és a CTRL-K billentyűkkel töröljük ki azokat. A tűrés változtatásával próbáljunk minél több felesleges színt kiválasztani, de vigyázzunk, hogy a fontos részeket ne töröljük ki. Most már lehet, hogy olyan vonalakból is eltűnt néhány képpont, amik szándékunk szerint folyamatos vonalak lettek volna. A folyamatos vonal és a körbezárt területek rendkívül fontosak a kifestés szempontjából.

A vonalak folytonosságának visszaállítására használhatjuk újra a szín szerinti kijelölést, most azonban azokat a színeket válasszuk ki, amelyeket látni szeretnénk a vonalak színeként. Növeljük meg a kijelölést, majd a kijelölt területeket fessük ki egy színnel. Így visszaállíthatjuk a vonalak folyamatosságát.

A vastag vonalak vékonyítására megoldás lehet, ha újra kijelöljük a kívánt vonalszínt a szín szerinti kiválasztással és ezután csökkentjük a kiválasztott terület méretét a ***Kép menü->Select->Shrink...*** menüpontok választásával.

Ezzel skeresen kiválasztottuk a megfelelő területeket. Invertáljuk a kijelölést a CTRL-I billentyűkkel, és fessük be újra – de most már a háttérszínnel – a kiválasztott területet.

Ilyen és ehhez hasonló lépésekkel alakíthatjuk át a beolvasott vonalas rajzot további feldolgozási (pl. színezés, árnyalás, különféle hatások hozzáadása) céljainknak megfelelően. Ha maradt felesleges rész, azokat kis munkával kiradírozhatjuk.

Ha már így belejöttünk a képfeldolgozásba, gondoljuk át, hogyan készíthetnénk a beolvasott vonalas rajzból, árnyalt képet. Bizonyára lesznek a rajzon olyan területek, amiket egyszínűre festhetünk. Ezekkel nem lehet gond. A színátmenetek alkalmazása is megkönnyíti a munkát, a jól megválasztott átmenet-típusokkal és irányokkal látványos eredményeket érhetünk el.

Előfordulhat, hogy bonyolultabb felület árnyalását szeretnénk elvégezni a programmal. Például, ha egy átmenetre szeretnénk mintákat festeni, készíthetjük azt egy másik rétegen, és az átlátszóság szabályozásával meghatározhatjuk a mintázat hatásának mértékét. A másik megoldás lehet, ha a festékszóró (***airbrush***) eszközt használjuk, finom átmeneteket hozhatunk létre egy nagyobb területen is.

Az eszköznek két érdekes beállítása van. Az egyik a nyomás, a másik pedig a ***Rate*** nevet viseli, amit leginkább úgy képzelhetünk el, mintha egy valódi festékszórón változtatnánk a szórófej méretét.

A változtatás hatására a festék gyorsabban áramlik, hamarabb látható a színezés eredménye. A valódi megoldáshoz hasonlóan ha egy területen többször haladunk át az eszközzel, akkor több festék kerül a vászonra, tehát a szín is sötétebb lesz.

A GIMP lehetőséget ad képrészletek egyszerű másolására is, mégpedig a ***nyomda*** eszköz segítségével. Szintén nagyon könnyel felismerhető az ikon alapján, tehát bizonyára mindenki könnyedén megtalálja az eszköztárban. Kétféleképpen használható, mindkét esetben a megjelenő lenyomat formája a beállított rajzeszköz keresztmetszet formájával egyezik meg. Az alapbeállítás szerint a kép egy részletét másolhatjuk más helyekre. Az ALT billentyű lenyomva tartása mellett egérkattintással tudjuk kiválasztani a forrásterületet, majd a billentyű felengedése után másolhatjuk a területet. Itt figyelni kell arra, hogy amikor lenyomjuk ismét a bal egérgombot, akkor a forrásterületet is elmozdul mégpedig az éppen érvényes helyet fogja követni. A forrás- és a célterület koordinátái közötti távolság állandó, egészen addig érvényes, amíg újabb forrásterületet nem adunk meg a programnak.

A másik használati mód, amikor az érvényben lévő mintával a valóságnak megfelelően nyomtassunk kis képeket. Az eszköz beállításait tartalmazó párbeszédablakban válasszuk a ***Pattern source*** lehetőséget és máris nyomtathatjuk a mintázatokat a vászonra. Természetesen a mintázatot meg is változtathatjuk munka közben.

További beállítási lehetőségek az igazításra vonatkoznak. Amennyiben az ***Aligned*** pontot választjuk, akkor a többszöri rajzolás során (két művelet közben felengedjük az egérgombot), a mintázat folytatólagosan kerül a képre, észre sem vesszük, hogy több részletben festettük ki a területet.

A ***Non Aligned*** „igazítás” kiválasztásakor a mintázat bal felső sarka az egérgomb lenyomáskor érvényes helyre kerül a képen. A különbséget jobban észrevehetjük, ha mindezeket ki is próbáljuk a gyakorlatban vagy megnézzük a négyes képen.

Most már rendelkezünk annyi tudással, hogy szebbnélszebb képeket, logókat vagy akár prospektusokat készítsünk. A megfelelő háttér kiválasztása után már csak a kisebb képeket kell elhelyeznünk a képen, a feliratokat elkészítenünk és akár saját honlapunkat, akár más nyomtatásra szánt cégbemutatónkat saját erőből örömmel megtervezhetjük és elkészíthetjük.

Mindenkinek kellemes alkotást és tanulást kívánok, a nagy nyári melegben nem mindig egészséges a napon sülni, és GIMP ismeretek mélyítésének is legalább annyi hasznát látjuk, mint a D-vitaminnak.

Fábián Zoltán

A Magyar Linux: UHU

Gyorsítsunk!

A Linux rugalmasságának rengeteg előnye van. Azt mindenki tudja, hogy kisebb-nagyobb műtéttel szinte bármilyen eszköz beleilleszthető a rendszerbe, de ez csak a legismertebb tulajdonsága. A rendszer hangolásáról viszonylag kevesebb szó esik, holott remekül felgyorsítható, ami eléggé komoly lépés a rendszer testreszabásában. Ebben a kis leírásban olyan rendszergyorsító tényezőkre próbálunk kitérni, amely a kezdők számára is könnyen végrehajtható. Nem igényelnek nagy szakértelmet, és nem kell nagyon bonyolult műveleteket végrehajtani hozzájuk. Persze az óvatosság nem árt. Hisz nem egyszer rendesen belemászunk majd a rendszer lelkivilágába. Azt se felejtsük el, hogy minden itt leírt tényező csak kis pluszt szolgáltat egymagában, viszont ha egyszerre többet alkalmazunk már szemmel látható a gyorsulás. Ez némileg módosítani fogja az UHU-Linux alapértelmezett beállításait figyelembe vett hardver igény adatokat. Értelemszerűen csodákra most sem leszünk képesek (többszörös növekedést senki ne várjon) viszont saját „sufni" méréseim szerint 50%-al biztosan felgyorsítható a rendszer.

A rendszer betöltődése

Az alapbeállításokkal rendelkező UHU-Linux pár olyan dolgot is betölt, amelyre az egyéni felhasználónak semmi szüksége. Mindössze azért van beállítva, mert valakinek biztosan jól jönnek. Példának okáért a ***firewire*** csatoló moduljának betöltése a felhasználók komoly hányadának felesleges. Ott van aztán az ISA támogatás, ami új gépeknél szintén felesleges lehet. Ha jobban megnézzük az UHU-Linux ***boot*** képernyőjét, akkor láthatjuk, hogy megkeresi a gépben lévő eszközöket, és működésre bírja őket.
És eközben megpróbál megtalálni olyan eszközöket is, amellyel nem rendelkezünk. Ezeket természetesen nem fogja megtalálni, a moduljaikat sem fogja betölteni, viszont fölöslegesen szöszmötöl a keresésével. Sok időt spórolhatunk meg azzal, ha ezeket nem keresi. Az UHU-Linux ezt a műveletet is a szkriptekben leírt módon végzi el. Ezen szkriptek az ***/etc/rc.boot*** könyvtárban találhatóak. Itt több szkript található, sorrendbe rendezve. A fájlok közül némelyikkel nem érdemes foglalkozni, hisz olyan alapvető hívásokat tartalmaz, amelyek nélkülözhetetlenek. Ezeknek a módosítása súlyos rendszerhibát idézhet elő. Ilyen fájl a ***„10-boot,15-boot,18-clock,20-urandom"***. Ezek a fájlok ráadásul rövidek, elhanyagolható növekedés érhető el a kiiktatásukkal, egyszóval nem éri meg a kockázatot a módosítása.
Nem így a ***12-detect***. A boot folyamat során a képernyőn lefutó utasítások 80%-át itt találjuk leírva. Ennek a fájlnak a módosítása viszont szemmel látható módon rövidíti meg a rendszer betöltődését. Szerkesszük hát meg.
Jelentkezzünk be a `su`-val konzolon, majd indítsuk el a ***Midnight Commander***-t, ami (mint már annyiszor tapasztalhattuk) egy kiváló, gyors, és kényelmes eszköz az ilyen műveletek elvégzésére.
Lépjünk be vele az ***/etc/rc.boot*** könyvtárba, és menjünk rá a ***12-detect*** fájlra. Ezután üssünk egy sima F4-et, hogy megnyissuk szerkesztésre. Ha vetünk egy pillantást a fájl tartalmára, rögtön látjuk, hogy egy konkrét szkriptről van szó. Ennek fényében tudjuk, hogy nem kell törölni a sorokat, hanem elégséges, ha a nem kívánt sor elé rakunk egy `#` -jelet. Ezzel ***komment*** jelzést adtunk egy sornak, tehát közöltük a rendszerrel, hogy ezt ne vegye figyelembe, ez mindössze egy olyan sor, amit magunknak írtunk. Ezután a jövőben az adott sort nem fogja figyelembe venni, tehát „hatályon kívül" helyeztük azt.
Általános recept nincsen arra nézve, hogy milyen opciókat hagyjon ki a rendszer a betöltődéskor, azonban pár szempont mindenképpen megemlítendő. Figyeljünk arra, hogy olyan utasítást még véletlenül se vegyünk ki, amely általános, és minden gépnél szükséges. Ilyen a „PCI eszközök keresése". Erre feltétlenül szükségünk van, még akkor is ha integrált alaplappal rendelkezünk, és egyetlen PCI-os kártya sincsen a gépben. Ugyanis az alaplapra integrált egységek is egy PCI sinen csatlakoznak egymáshoz.
Ott van viszont az `ISDN:*) - -;;` közötti rész. Ez nyugodtan kivehető, ha nincs ISDN hálózatunk, és ISDN kártyánk. Amennyiben szeretnénk használni, az USB modulokra is szükség van. Ha nincsen semmilyen ISA csatolójú eszközünk, (figyelem, itt is vegyük figyelembe az alaplap egységeit! Erről az alaplap kézikönyve tud minket alaposan tájékoztatni.) akkor az `ISA eszközök keresése...` részt is nyugodtan hatástalaníthatjuk.
Érdekes még a `További modulok betöltése...` rész, mert elképzelhető, hogy innen valamire van, valamire pedig nincs szükség. Rögtön a `probemod ide-scsi` sorral kezdődik. Erre mindenképpen szükségünk van. Ez ugyanis a SCSi emulációt hívja meg, amelyre a CD-ROM, (nem annyira) vagy a CD-RW eszközöknél van (nagyon!) szükség. Amennyiben ezt kivesszük, gondjaink lehetnek a CD eszközzel, és/vagy használhatatlanná válik a CD-RW eszközünk. Tehát megtartása melegen ajánlott. Nem úgy a `probemod serial`, és a `probemod lp`

soroké. Az első a soros port, a második a párhuzamos (nyomtató) port. Mindenki döntse el, hogy használja-e ezeket az eszközöket, merthogy korántsem biztos az USB-s nyomtatók, és PS2 egerek korában. (Ha külső modemet használunk, akkor az valószínűleg a soros portra csatlakozik, tehát a `probemod serial` megtartása szükséges lehet.) Az `IEEE 1394...` kezdetű rész viszont csak azoknak szükséges, akik FireWire csatolóval rendelkeznek, és használni is szeretnék azt. Ekkor hagyjuk bekapcsolva, ha nem használjuk, nyugodtan hatástalanítsuk.

Az ACPI sorokra szükség lehet. Ezek a fejlett energiagazdálkodást hivatottak kiszolgálni. Amennyiben szeretnénk ennek előnyeit élvezni hagyjuk meg, ha nem, kapcsoljuk ki. Az `apm`-et viszont érdemes meghagyni, hisz kényelmes az általa nyújtott szolgáltatás, de nem visz el sok erőforrást. Érdemes meghagyni a `Serial-ATA vezérlők keresése ...` sort is, hisz az ATA merevlemezek támogatását használhatjuk általa. Amennyiben készen vagyunk, F2-vel mentsük a fájlt, és indítsuk újra a gépet. A jó beállítások esetén minden általunk kívánt eszköz használható marad, de mégis szemmel láthatóan gyorsabban éled fel a számítógép.

Szolgáltatások

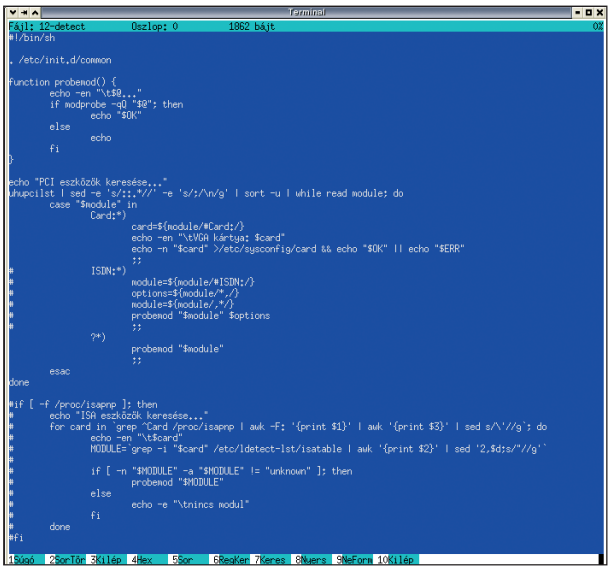
Számtalan alapértelmezett szolgáltatás indul el, amikor telepítjük az UHU-Linuxot. Ezek közül is van olyan, amire olykor nincs szükség, feleslegesen foglalja az erőforrásokat. Ezen szolgáltatások kezeléséhez az UHU vezérlőpult nyújt segítséget. Indítsuk el, és menjünk rá a „szolgáltatások" pontra. Itt láthatjuk listában, hogy milyen elérhető szolgáltatásokat futtathatunk, milyen futási szint tartozik hozzá, láthatjuk, hogy éppen fut-e, és beállítható, hogy automatikusan induljon-e, vagy sem. Ezekre olykor szükség van, olykor nem. Rövidke leírás is tartozik hozzá, hogy melyik milyen funkciót végez, tehát könnyen dönthetünk. Most csak a kicsit kétségesebb részekre térünk ki.

– `ADSL:` Erre akkor van szükségünk, ha ADSL kapcsolattal rendelkezünk, és az UHU tárcsázóban beállítottunk egy érvényes, és működő ADSL elérést. Ekkor ezt a pontot aktiválva a rendszer betöltődésekor azonnal indul az ADSL kapcsolat is, így ezzel már nem kell törődnünk. Alapértelmezetten ki van kapcsolva.
– `CUPSD:` Mindenféle nyomtatási feladatért felelős. Ő a nyomtató szerver démon, amire nyomtatóval rendelkező asztali gépeknél szükség van, (hálózati nyomtatóknál is) egy olyan notebook esetében mint ami nekem is van már nem. Ilyenkor nyugodtan kikapcsolható.
– `FAM`: fájl állapotváltozás figyelő. Szükséges, hagyjuk bekapcsolva. Ez tartja nyílván a fájlok állapotát, követi a változtatásukat.
– `GDM/KDM:` ez a gnome, vagy KDE bejelentkezés kezelő. Valamelyikre szükség van, hogy be tudjunk jelentkezni grafikus felületen, de csak az egyikre. Ha szerkesztjük figyeljünk arra, hogy a futási szintje legyen „5". Ugyanis a grafikus szerver ezen a futásai szinten üzemel.
– `nfs,smb:` Szerverek. Ezekre csak akkor van szükség, ha hálózati meghajtót is akarunk csatlakoztatni, kezelni (nfs) illetve szeretnénk windows-os gépekhez csatlakozni (smb). Ha nem, nyugodtan kikapcsolható.
– `ntp:` Ez egy érdekes szolgáltatás. Ez a kis démon az interneten keresztül hangolja össze a rendszer órát, az internetes órával. Ha az esetek döntő többségében nem lesz a gép az interneten felesleges a futtatása, hisz csak hálózat esetén használható. Én kikapcsoltam, mivel a notebook-om 90%-ban nincs hálózaton.
– `Postfix:` Amennyiben szerverként használjuk a gépünket úgy, hogy a leveleinket a *felhasználó@sajátdomain.hu* címen akarjuk kiküldeni, szükséges. Ha az internet szolgáltatónk pop/smtp szervereit használjuk, akkor nem szükséges. Én kikapcsoltam.
– `remotefs:` más, hálózati fájlrendszerek közötti tallózást tesz lehetővé. Ha semmilyen hálózati meghajtóval nem akarunk foglalkozni, (novell,NIS,nfs,smbfs) akkor nem szükséges.

Grafikus felület

Minden eddigi erőfeszítésünk hiábavaló lehet, ha egy, az erőforrásokat nagy kanállal evő ablakkezelőt használunk. A Linux ebből a szempontból is több alternatívát kínál, így a két fő szempont a gyorsaság, és a kényelem igen jól közelíthet egymáshoz. Tekintsük át őket röviden, a legtöbb erőforrást kívánó ablakkezelőtől a legkisebbekig.

– KDE: A ***K Desktop Environment*** a leginkább elterjedt, és legkényelmesebb ablakkezelő. Felülete rendkívül tetszetős, határozottan szép. Szolgáltatásai igen sokrétűek, jól konfigurálható, és egy teljesen komplett rendszert nyújt. A ***Kinfocenter*** remekül tájékoztat a hardverekről, a ***Koffice*** egy kellemes kis irodai programcsomag, és hosszasan lehetne sorolni a programokat , amiket tartalmaz. A teljes KDE felület több mint 250 MB, tehát igen nagy. Ez az az ablakkezelő, amelyet a leginkább előnyben részesítenek a kezdő felhasználók, a windows-os felületet kedvelők. A KDE 3.2 már eléggé gyorsan éled fel, és igen kellemes. Nagy hátránya, hogy a memória méretével arányos a sebessége. Ugyanis a KDE rendkívüli módon eszi a memóriát, így 128 MB rendszermemória alatt szemmel láthatóan lassú. Amennyiben a KDE felületen szeretnénk futtatni az ***Evolution***, ***OpenOffice***, vagy ***Mozilla*** alkalmazásokat, (esetleg mindet együtt) akkor a 256 MB-nyi memória igencsak elkél. A processzor órajele is legyen legalább 500 MHz, ha szeretnénk, hogy jól fusson. Sok ablak megnyitása, illetve a fent említett programok egyidejű futtatásakor még így is észrevehető, hogy bizony elfoglal nagyon sok erőforrást. Ez ter-



mészetesen a több Ghz órajelű processzorok korában nem akkora probléma, pláne, hogy manapság az 512 MB fizikai memória sem szokatlan. Régebbi gépeken határozottan kényelmetlen lehet, és nem ajánlott.

– Gnome: Igen fejlett ablakkezelő. Az UHU a 2.4.2-es verziót tartalmazza, de az internetről telepíthető a 2.6-os gnome is. Ez a legfrissebb. Jóval kevesebb erőforrással is beéri mint a KDE, és nagyon kényelmes használata. A 128 MB rendszermemória itt is javasolt, de ennyi elég is neki. Nem eszi nagykanállal ez erőforrásokat. Ez a felület a legalkalmasabb arra, hogy kényelmesen dolgozhassunk, és kíméljük az rendszer energiáját. Az UHU-Linux alapértelmezett felülete is ez.
– IceWM: Régi, mondhatni „klasszikus” felülete a Linuxnak. Nagyon kevés erőforrást igényel, és a Windows95-öt idézi a felülete. Gyors, még elég kényelmes. Néhány Linux terjesztés alapértelmezett felülete, (pl: ***Vector linux***). 64-128 MB alatti memóriával rendelkező gépek ideális felülete.
– XFCE: A fejlesztése jelenleg a 4.2 verziónál tart. Nagyon gyors, és az alkalmazások is nagyon gyorsan indulnak rajta. Kicsit szokatlan lehet a kezdő felhasználók számára a kezelése, de gyorsan elsajátítható. Sajnos ez a felület nem támogatja az ikonok elhelyezését a képernyőn, helyette egy igen átgondolt menürendszerrel rendelkezik, ahová gyerekjáték új bejegyzéseket felvinni. „súlycsoportban” az IceWm-el egyezik meg, és szintén tanácsolható a kis gépek felületének.
– BlackBox: Ez egy rendkívül kicsi, és villámgyors ablakkezelő. A menü a jobb egérbombbal csalogatható elő, betöltődése pedig még régi gépeken is 2-3 másodperc. Nagyon kevés erőforrást igényel. Az ikonokat ez sem támogatja. Hátránya, hogy a testreszabása némi szakértelmet igényel, mivel ehhez a beállító szkriptekben kell elmélyülni. Egyértelműen a haladó felhasználók felülete, akik már hozzászoktak az ilyen beállítási módhoz, és igyekeznek mind jobban kihasználni a Linux finomhangolhatóságát. 64 MB alatti gépekhez kiváló.
– Enlightenment: Egy nem túl elterjedt, és ismert ablakkezelő, de nagyon jó, mondhatnám kiváló. Nem sokkal tart tovább az indulása mint a ***blackbox***-nak, (3-4 másodperc) teljesen animált, kényelmes, és a funkcióinak a 90%-át beállíthatjuk grafikus menüből is. Ehhez is sok téma tölthető le az internetről, az egyszerűtől az egészen futurisztikusig. Ezt az ablakkezelőt kezdőknek is tudom ajánlani. Szintén alkalmas 64 MB alatti gépekre.
– WindowMaker: Ez egy szintén ismert, és régebb óta használatos ablakkezelő. Régi motorosok jól ismerik. Gyors, kevés erőforrást eszik, és nem támogatja az ikonos elrendezést. Pontosabban támogat valami olyasmit mint az ikonok, de korántsem abban a formában, amire az ember először gondolna. A WindowMaker a képernyő oldalán, és alján helyezi el ezeket, és nem a képernyő középső felületén. Így könnyen elérhető, mindössze kicsit szokatlan lehet. Szintén jó alternatíva a 64 MB alatti gépeknél.

Egy jól megválasztott ablakkezelő sokat javíthat a rendszerünk sebességén, terhelhetőségén. Általános recept erre sincsen, főleg,hogy az ablakkezelő megválasztása is inkább vallási kérdés. Persze kis gépen a KDE választása nem szerencsés választás, de általánosságban elmondható, hogy az erőforrás beosztás szempontján túl mindenki megtalálhatja azt az alternatívát, ami neki a legjobban tetszik, vagy a leghasznosabb.

A Rendszermag kezdőknek

A Linux lelke a rendszermag. Pontosabban (és ez a helyes megfogalmazás) a Linux maga a rendszermag. Minden más ami körülötte van, (programok, héj, stb...) csak egy nyílt forrású program, amely a Linux (a kernel) köré van helyezve, különböző feladatokkal. Amikor tehát a magot fordítjuk forráskódból, gyakorlatilag a Linuxot fordítjuk le.
Az UHU-Linux magja egy „általános” kernel, úgy elkészítve, hogy telepítés után a lehető legtöbb gépen használható, és működőképes legyen. Tömören, és lényegretörően megfogalmazva, a telepített kernelnek nem sok köze van a gépünkhöz. Sok olyan dolgot is tartalmaz, amelyre nekünk semmi szükségünk nincsen. A jól beállított, és az adott géphez fordított kernel nagyon stabil, és gyors működést tesz lehetővé.
A rendszermag fordítása messze meghaladná eme leírás kereteit, ezért nem is vesszük át alaposabban. Mindössze pár gondolatra térünk ki, amely segítség lehet a kivitelezésben, a kezdő felhasználók számára is.
Az UHU alapértelmezett magja a 2.4.24-7-es verziószámot viseli. Ahhoz, hogy tudjuk fordítani, szükséges a forráskód. Ehhez kétféle módon juthatunk. Az UHU-Linux második CD lemezéről, vagy az internetről. Mindenképpen javasolt az internetes telepítése. Ugyanis a CD lemezen található forráskód az UHU-Linux 1.1.1 kiadási időpontjában uralkodó állapotokat tükrözi, így minél távolabb van ez az időpont a jelentől, annál több hibajavításnak van híján. Az „internetes változat” azonban folyamatosan frissítve van, így tartalmazza a lehető legfrissebb javításokat, foltokat is. Ennek a forráskódnak a telepítése az `apt-get` el történik.

```
Apt-get install kernel-source
```

Ne lepődjünk meg, ha az `apt-get` először a ***kernel-headers*** csomagot telepíti. Erre feltétlenül szükség van.

Amikor az `apt-get` készen van, mi is munkához láthatunk. Kétféle módon is beállíthatjuk a kernel konfigurációt. Először lépjünk be a kernel forrásának a mappájába konzolról, természetesen rendszergazdaként.

```
cd /usr/src/linux
```

Ezután kétféle felületről konfigurálhatunk; A `make xconfig` paranccsal egy teljesen kulturált grafikus felületű beállítóprogramot kapunk. A `make menuconfig` viszont egy DOS-os időkből ismert grafikus képernyőt ad, ahol a kurzor mozgató billentyűkkel lehet manőverezni. Ekkor munkához láthatunk, és megszerkeszthetjük a saját, a gépünknek leginkább megfelelő konfigurációt. Az így elkészített, és elmentett konfigurációt fogja használni a fordítóprogram, amikor lefordítja a forráskódot. Ehhez a beállítások végeztével adjuk ki a `make bzImage` utasítást.
A beállítások során sokszor háromféle lehetőségünk van. A kívánt funkciót belefordíthatjuk a rendszermagba, ekkor a ***bzImage*** fogja tartalmazni. Választhatjuk azt is, hogy modulként legyen jelen (ekkor ***valami.o***-ként fordul le, de nem tartalmazza a ***bzImage)***, vagy el is távolíthatjuk a funkciót. Ebben az esetben a fordító a jövőben nem fog foglalkozni a forrásával.
Az UHU alapértelmezett rendszermagjának a fordításakor funkcionális hátránya nem lesz, ha nem fordítunk modult, hisz az alap kernel moduljai már léteznek a ***/lib/modules*** alatt. Ha új modult nem akarnuk fordítani, akkor a `make modules` és a `make modules_install` parancsok elhagyhatóak, és elégséges csak a ***bzImage*** fordítására koncentrálni.
Érdemes kitérni egy fontos tényre. Az eddigi gyakorlattal ellentétben, az alternatív ***kernel-smp*** az 1.1.1-ben már nem létezik. Tehát a több processzoros támogatás már nem külön kernelt igényel, hanem mindössze csak be kell kapcsolni a konfigurációban. Ez alapértelmezetten be van kapcsolva, és érdemes is úgy hagyni, még akkor is, ha tudjuk; csak szimpla processzorhoz fogjuk használni. Ugyanis tapasztalatom szerint ha ezt kikapcsoljuk, fordítási hiba lép fel.
Az új, immár a mi gépünkhöz fordított rendszermag a ***/usr/src/linux/arch/i386/boot/bzImage*** néven található meg. Ezt kell bemásolni a ***/boot*** mappába, ahol a ***grub*** a rendszer betöltésekor keresni fogja.
Amennyien nem vagyunk gyakorlottak a rendszermag fordításában érdemes átnevezni mondjuk `bzImage1`-re, és ennek külön bejegyzést írni a ***/boot/grub/menu.1st***-ben. Így ugyanis egy rossz beállításokkal fordított rendszermag elindítása után még nem lesz használhatatlan a gép, és hiba esetén még mindig használható az alapértelmezett.
Ha azonban szeretnénk a legújabb rendszermagot használni amit elkészítettek, akkor a 2.6-ost fordítjuk. A legfrissebb verzió a cikk írásának időpontjában a 2.6.7-es. Ennek a forráskódját a ➲ http://www.kernel.org weboldalról tölthetjük le. Ha letöltöttük a megfelelő forráskódot, akkor kicsomagolás után szintén lépjünk be rendszergazdaként a forráskód gyökérkönyvtárába, és ott adjuk ki a `make xconfig`, vagy `make menuconfig` parancsot.
Figyelem! A 2.6.x-es kernel teljesen új keletű fejlesztés. Számos, a 2.4.x-ben nem található újdonságot tartalmaz. Mindenképpen javasolt az idevágó fórumok olvasgatása, a tapasztaltabb felhasználókkal való konzultáció, vagy az adott újdonsághoz tartozó súgó elolvasása a beállítóprogramban.
Egy jól fordított 2.6.x-es rendszermag szintén nagyon sokat gyorsíthat a rendszerünkön, mivel új ütemezőjével, memóriakezelésével, és kernel szintű eszközgyorsítási (pl: egér) képességével meglepő plusz sebesség erhető el.
További fontos megjegyzendő, hogy a 2.6.x-es, és a 2.4.x-es kernel nem jól dolgozik össze. Tehát a 2.6.x-es kernel fordításakor feltétlenül fordítanunk kell a modulokat is, a `make modules` parancssal, és telepítenünk a ***/lib/modules*** alá a `make modules_install` utasítással.
Ha készen vagyunk, és az új kernel hibátlanul üzemel, és elégedettek is vagyunk vele, nyugodtan törölhetjük az új bejegyzést a ***grub***-ból, hogy az új rendszermag legyen az alapértelmezett.

Programok

A programok kérdéskörét nagyon sokszor elfelejtik a rendszerek hangolásánál. Egy adott feladatra létezhet nagyobb, és kisebb erőforrás igényű program, holott mindkettő tudja azt, amire szükségünk van. Általában egy adott program használatában sokkal inkább a megszokás dominál, mint a program tudása. Érdemes lehet átnézni a menüket, hogy egy adott funkcióhoz milyen programok léteznek, és mikor, melyik alkalmas a kívánt feladat végrehajtására. A kisebb, de nagyon gyors program használata jobb választás lehet, mint egy nagy, lassú, mely tudásának csak kis részére van szükségünk. Ez szintén teljesen egyén-, rendszer-, sőt feladatfüggő.
A Mozilla nagy és lassú. A feladatát (böngészés) jobban elvégezhetjük a FireFox,vagy az Opera segítségével. Az Operát nagyon melegen tudom ajánlani. Kicsi, és nagyon gyors, ráadásul bámulatosan nagy tudású böngésző, amelynek jelenleg a 7.51 verziója érhető el. Aki ismeri nem kell magyaráznom miért kiváló, aki nem, annak csak javasolni tudom a kipróbálását. Az Abiword nagyon szimpatikus kis szövegszerkesztő, ráadásul legalább kétszer olyan gyors, mint az OpenOffice.org. Egyszerű, sima dokumentumok szerkesztéséhez szerintem jobb választás.
Az Evolution nagyon jó program, ám aki csak levelezni akar, annak nagy, és felesleges. Ajánlható a Kmail, vagy a Sylpheed. Csevegéshez lomha lehet a Mozilla IRC-je, de még az Operáé is. Ekkor használhatjuk az Xchat, Ksirc, BitchX programok valamelyikét. Mindegyik nagyon jó, és kellően kevés erőforrást eszik, miközben a feladatát tökéletesen ellátja. A sort hosszasan lehetne folytatni. Érdemes egyszer átnézni a menüket, hogy mégis milyen programokkal rendelkezünk, az adott feladathoz. Egy a jól megválogatott program készlet gyors, hatékony munkát tesz lehetővé úgy, hogy a megfelelő tudásukban se legyen hiány.
A fent leírt módokon, és egy jól beállított rendszer, jól fordított 2.6.x-es rendszermaggal akár 60%-al is gyorsabban töltődhet be, és futhat. Rendkívül sok erőforrás takarítható meg, vagy használható ki jobban, hogy a rendszerünk valóban úgy működjön, ahogyan az a legoptimálisabb. Természetesen gyorsabb működés nem létezik annál, minthogy az adott rendszer/programok az adott géphez legyenek fordítva (a'la gentoo, ami bámulatosan gyors) de anélkül is sokkal jobb működés alakítható ki, ha nem az alapban telepített rendszert használjuk, hanem gépünkhöz, igényeinkhez igazítjuk.

Dancsok „strogg" Zoltán

BootSplash képernyő Uhu-Linuxon!

Hozzunk létre UHU linux alá is csinos háttérképeket a bootoláshoz!

A megfelelő anyag birtokában, minimális gyakorlattal ez mindössze 15 perc szöszmötölés után elkészül. Ennyit pedig messze megér a végeredmény. A probléma megoldásához több dologra is szükség van. Egy rendszermag forráskódra, egy rendszermag foltra, ami ezt a funkciót hozzáadja a fordítható rendszermaghoz, egy bootsplash-készítő segédprogramra, egy témára, és körülbelül 15-20 perc szabadidőre. Az UHU-Linux esetében a rendszermag forráskódja könnyen telepíthető az `apt-get install kernel-source` parancsal. Ez letölti, és telepíti a kernel-header csomagot is. Ha ez megvan, akkor a szerszámosláda többi elemére van már csak szükségünk. A ➲ http://www.bootsplash.org kifejezetten ezzel foglalkozik. Sőt, pár témát is találunk itt, és teljes leírásokat. Lássunk munkához! Az említett oldalról töltsük le a ***kernel stuff*** menüpont alól a megfelelő kernel foltot. Látható, hogy a 2.6.x, és a 2.4.22-höz vannak itt javítások. A verziószámok ne zavarjanak meg senkit, a 2.4.22-es tökéletesen működik az UHU 2.4.24-7-es kernelével. Ha letöltöttük a megfelelő ***bz2*** fájlt, akkor csomagoljuk is ki. Ha ezzel is megvagyunk, akkor semmi más teendőnk nincsen, mint feljavítani a telepített rendszermag forráskódját. Konzolban jelentkezzünk be rendszergazdaként a `su`-val. Lépjünk be a rendszermag forráskódjának a mappájába, a `cd /usr/src/linux` parancsal. Ha itt vagyunk, végezzük el a forrás foltozázását:

```
patch -p1 < /eleresiút/bootsplash-3.0.7-2.4.20-
➥vanilla.diff
```

Ha hiba nélkül lefutott a parancs, akkor a jövőben fordított kernel már képes lesz megjeleníteni a bootsplash képernyőt. Indítsuk el hát a rendszermag konfigurációs programját a `make menuconfig` utasítással. Keressük meg a ***Console drivers ->Frame-Buffer support*** részt, és engedélyezzük a ***VESA-VGA graphics console*** pontot. Ezután engedélyezzük a ***Use splash screen instead of boot logo*** pontot is. Ezzel meg is volnánk. Figyelem! A kernel ekkor még az alapbeállításokat tartalmazza. Érdemes átnézni, hogy még mire van szükségünk a fordítás elött. És mivel a kernel, és a modulok verziója egyezik, elhagyható a `make_modules`, és a `make modules_install` parancs. Elég csak a ***bzImage***-et fordítani. Ha készen vagyunk, akkor bátran fordítsuk le az új kernelt, a `make bzImage` parancsal. A fordítás után a kész ***bzImage***-t célszerű átnevezni a tesztek erejéig, mondjuk ***bzImage1***-re, és úgy bemásolni a ***/boot*** konyvtárban található eredeti ***bzImage*** mellé. Célszerű, ha lemásoljuk a ***/boot/grub/menu.1st*** fájlban az indító bejegyzést, és az új bejegyzésben hívjuk meg a ***bzImage1***-et. Ezáltal gyakorlatilag 2 üzemképes kernellel rendelkezünk, és hiba esetén nem válik használhatatlanná a gépünk. Ezután még két dologra van szükség. Egy témára, és egy programra, amely a kernel számára „ehető" formába hozza a képeket. Ehhez töltsük le a ***bootsplash-3.0.7.tar.bz2*** fájlt, és tömörítsük ki. A kapott forráskód könyvtárába belépeve adjuk ki a `make`, és a `make install` parancsokat. Ezzel lefordítjuk a programot, és telepítjük. Győződjünk meg róla, hogy telepdett a ***/usr/sbin*** könyvtárba. Ezután töltsünk le egy nekünk tetsző témát. A letöltött ***bz2*** fájl kitömörítés után azonnal haszálható. A kicsomagolt téma fájljai közül mindössze 3 dologra van szükség. A ***bootsplash-1024x768.jpg***, a ***silent-1024x768.jpg*** és a ***bootsplash-1024x768.cfg*** fájlokra. Most Midnight Commander-rel menjünk rá a ***.cfg*** fájlra, F4-el nyissuk meg, és a fájlban a két képnek adjuk meg a pontos elérési útját. (***/usr/share/splash/...jpg***) Amikor ez is megvan, akkor már csak létre kell hozni a megfelelő állományt, amit látni fogunk a boot folyamat során. Ehhez adjuk ki a következő parancsot.

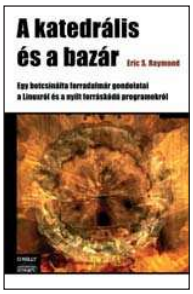
```
/sbin/splash -s -f /usr/share/splash/bootsplash-
➥1024x768.cfg > /boot/initrd.splash
```

Ha gond nélkül lefut, akkor létrejött egy a ***/boot*** mappában, az eredeti ***initrd*** mellett, egy ***initrd.splash*** fájl. Ez az, amire szükségünk van. Módosítsuk a ***/boot/grub/menu.1st*** fájlt ismét. Ezuttal azonban az új kernel (***bzImage1***) bejegyzését véglegesítjük. Utasítsuk, hogy a képernyőt kapcsolja a megfelelő felbontású VESA módba. Ezt a `vga=791` utasítással tehetjük meg. Ha a ***silent*** képet választjuk, akkor a teljes boot során egy képet mutat, míg ***verbose*** módban a kódsorok a kép elött futnak. Az ***initrd***-re utaló sort javítsuk át úgy, hogy az ***initrd.splash***-ra mutasson. Az egész tehát ilyen lesz:

```
   title UHU-Linux+splash
   kernel (hd0,1(/boot/bzImage1
root=/dev/ide/host0/bus0/lun0/part2 vga=791
➥splash=silent
   initrd (hd0,1)/boot/initrd.splash"
```

Ezután dőljünk hátra, és indítsuk újra a gépet. Ha mindent jól csináltunk, már gyönyörködhetünk is a képben.

Dancsok „strogg" Zoltán



Kiadja: Kiskapu
Felh. szint: kezdő
Ár: 2800,-
Cd melléklet: nincsen
244 old.

Eric S. Raymond: A katedrális és a bazár

A nyílt forráskód forradalma és története

Eric S. Raymond A katedrális és a bazár című könyve egy olyan mozgalmat, egy olyan szemléletet mutat be, amelyet a nagyközönség Magyarországon csak mostanában kezd megismerni. A nyílt forráskódú szoftverek, a nyílt forrású fejlesztés pár éve még úgy tűnt, mint néhány megszállott hobbija, semmi több. A nagy szoftveróriások ontották magukból a termékeket, szívták (f)el a nagytudású fejlesztőket, és mindenki őket tekintette az informatikai fejlődés mozgatóinak. Aztán a '90-es évek elején, a Helsinki Egyetem egy diákja egyedi ötlettel állt elő, amely – bár sokan tamáskodva fogadták – mára alapjaiban rengette meg a szoftveripart és változtatta meg az addig szentnek és sérthetetlennek hitt fejlesztési felfogást.
A Linux fejlesztési módja felrúgott minden addig használt szabályt, sémát. Nem jól szervezett fejlesztői csoportot hoztak létre, hanem közösséget szerveztek. A közösség tagjai látszólag ugyan szervezetlen csoportot alkottak, akik szervezetlen módon működnek, és ez sokakat eltántorított a csoport munkájának támogatásától. De a dolog valahogy mégis működni kezdett, és ahelyett, hogy sokak várakozásának megfelelően átláthatatlan káoszba süllyedt volna – virágozni kezdett. Egyre több és több szakember kapcsolódott be a Linux fejlesztésébe, újabb és újabb projektek indultak. És mindez éppen egybeesett az Internet robbanásszerű elterjedésének idejével, ami újabb lökést adott a Linuxnak. Mivel annak gyökerei az Internet ősének tekinthető ARPAnet hálózathoz csatlakozó Unixos rendszerekhez vezethetőek vissza, nem volt kétséges, hogy rövid idő alatt az Internetes kiszolgálók és munkaállomások uralkodó rendszerévé lép elő.
A könyv egy bennfentes tapasztalatain keresztül betekintést ad a nyílt forráskódú fejlesztés világába, bemutatja a fejlesztők csoportjait. Rámutat olyan ellenmondásokra, mint a nyílt forráskódú, szabadon használható szoftverek és a tulajdon kérdése, bemutatja a közösség mozgatórugóját, a fejlesztők egymással vívott presztízs-csatáját továbbá betekintést nyerhetünk egy olyan kreatív közösség életébe, amely a mai üzleties szemléletmód nélkül is sikerre tudta vinni ötletét. Annak is érdekes olvasmány a könyv, aki kicsit üzleti szempontból tekint a szoftverfejlesztésre. A könyv bemutat néhány téveszmét a nyílt forrású projektek finanszírozásával kapcsolatban, valamint mutat kilenc modellt ezen projektek finanszírozására.
A könyv ma még „csak érdekes olvasmány", de lehet, hogy pár éven belül kötelező irodalommá válik.

Kiadja: Kiskapu
Felh. szint: kezdő
Ár: 3980,-
Cd melléklet: 1 db
352 old.

Váltsunk Linuxra Marcel Gagné-val

Egy újabb útmutató a Linuxra váltóknak

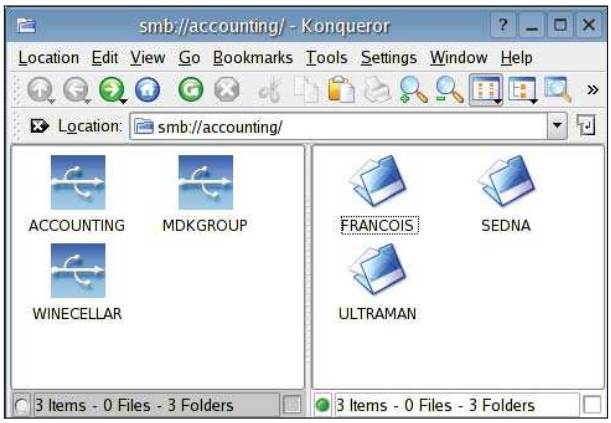
Marcel Gagné Válts Linuxra! című könyve alapos útmutató kezdő Linux felhasználóknak. A könyv az első gondolatok megszületésétől segíti a felhasználót a Linuxra való átállás során. A szerző a Linux bemutatását egészen az alapoktól kezdi. Első lépésben felteszi és meg is válaszolja a témával ismerkedőkben felmerülő kérdést: „Valóban ingyenes?"
A könyv első fejezetében megismerkedhetünk a GPL licenceléssel, a Linux által nyújtott biztonsággal, megbízhatósággal és teljesítménnyel. A könyv ugyanakkor nem elvakult, tisztában van az újdonsült Linuxos felhasználóknál jelentkező problémákkal is. „Van egy nyomtatóm, hangkártyám, lapolvasóm, régi jól megszokott programom. Miért nem működik?" A könyv foglalkozik a felhasználók által tapasztalható kisebb-nagyobb problémákkal is, így biztosítva azt, hogy senkit ne érjen váratlan csalódás a rendszer telepítése után. És ha már problémák felmerüléséről beszélünk, említsük meg, hogy a könyv nem megy el ezek mellett. Segíti az olvasót abban is, hogy ezekre a kisebb-nagyobb gondokra hol találhat gyógyírt.
A rendszer telepítése jól átgondolt ív mentén halad. Nem feltételezi egyik, vagy másik Linux kiadás meglétét, általánosságban mutatja be a csomagokat. Ez egyfelől előny, hiszen minden kiadás használója forgathatja a könyvet, másfelől feltételez némi talpraesettséget az olvasóról, de úgy gondolom ez a mai számítástechnikai világ velejárója.
Első lépésben telepítünk egy tetszőleges Linuxot, beállítjuk a rendszer indítását, felkészülünk a mindennapi munkához szükséges eszközök telepítésére. Második lépésben a szerző leírja, hogy miként telepítsük, konfiguráljuk és használjuk az egyik legnépszerűbb grafikus környezetet, a KDE-t.
A KDE bemutatása után következhet a KDE fontosabb programjainak ismertetése, így egy teljes fejezetet kapott a Konqueror. A szerző külön fejezetet szentel a számítógépünk különböző eszközeinek telepítésére, de az internetes kapcsolat beállítására és az elektronikus levelezésre is. Természetesen a különböző típusú felhasználói programokról sem feledkeztek meg, így a könyvből megtudhatjuk milyen irodai programot használjunk, hogyan állítsuk be és használjuk a lapolvasónkat, vagy mivel tudunk zenét hallgatni és videót nézni.
A könyv hasznos olvasmány mindenkinek, aki meg szeretne ismerkedni a Linux használatával a mindennapokban. Kezdőknek csak ajánlani tudom.

Illés Viktor

Keresztplatform? Helyes!

Linux alapú fájlkezelők könnyítik meg állományaink és nyomtatóink kezelését örökölt Microsoft Windows rendszereinken.

Igen, François elismerem. Ez valóban nagyon vicces. De amikor a múltkor említettem, hogy e havi témánk a kereszt-platform fejlesztés lesz, nem olyan platformokra gondoltam, amitől keresztbe állnak a szemeid, bár megértem, hogy néhány operációs rendszer kapcsán erre gondoltál. Amilyen elbűvölő kis képek ezek, azt hiszem a mai menühöz kiválasztott művek láttán lesznek akik felvonják majd a szemöldöküket, még ha közönségünk igen megértőnek mondható is. Miről is fecsegek itt, hiszen a mi kedves vendégeink már meg is érkeztek.
Üdvözlet, ***mes amis***, Chez Marcelnél, a különleges Linux csemegék otthonában. Helyezzék magukat kényelembe. François és jómagam éppen az e havi témáról, a rendszerfüggetlen fejlesztésről beszélgettünk, és kedvenc pincérem talán kicsit túl lelkes volt. Már-már úgy tűnhet, hogy ehhez egy fehér Zinfandelt kell hoznunk, de szerencsére ilyet éppen nem tartunk. François, a pincébe, ***immédiatement***! Hozzd fel 1992-es Napa Valley-i Cabernet Sauvignon-t. ***Vite***! Amint azt bizonyára sokan tudják, a Microsoft Windows továbbra is része az átlagos üzleti IT világnak. Sokunk számára létfontosságú, hogy meg tudjuk oldani a Windows és Linux közötti információcserét. Tegyük fel, valahogy sikerült meggyőznünk a vezetőséget, hogy Windows helyett Linuxot futtathassunk a munkaállomásunkon. Esetleg saját notebookunkat használjuk. Bármi legyen is az indok, meg kell birkóznunk a Windows munkacsoportokkal vagy tartományokkal, valamint a megosztott állományokkal és nyomtatókkal. Bár Judit a könyvelésen nem különösen kedveli Windows XP gépét, igen sok megosztott állomány található meg itt, olyan állományok, amelyeket a hálózati kapcsolatokon keresztül érhetünk el. Azt kérdezhetnénk magunktól, vajon mennyire nehéz a hálózati kapcsolatok nyújtotta előnyöket kihasználni. Nos, ez egy igen érdekes kérdés, tekintve, hogy milyen sok olyan fájl és nyomtatókiszolgáló található, amely valójában nem Windows-t, hanem a fájlszolgáltatásokat Samba rendszerrel kezelő Linuxot futtat. Ezek után nem is meglepő, hogy a Samba-val együttműködő ügyfélprogramok lassan kezdenek minden új Linux terjesztés részévé válni. Így aztán az `smbclient` programmal könnyedén felkapcsolódhatunk a hálózat Windows megosztásaira. Indulásként adjuk ki az `smbclient -L sedna` parancsot, amely a megosztások listáját adja vissza a következő alakban:



1. ábra Munkacsoport böngészés Konqueror-ral

```
Domain=[ACCOUNTING] OS=[Windows 5.1]
Server=[Windows 2000 LAN Manager]

        Sharename       Type       Comment
        ---------       ----       -------
        SEDNA_C         Disk
        IPC$             IPC        Remote IPC
        Reports         Disk
        Policies        Disk
```

Feltételezve, hogy van jogosultságunk a ***Reports*** mappa eléréséhez, a következőképpen csatlakozhatunk hozzá:

```
smbclient //sedna/reports -U winuser
```

A fenti példában Linux gépemről ***winuser*** néven jelentkezem be a Windows XP gépre. A rendszer ezután jelszót kér, majd a samba parancssorba jutunk amely a következőképpen néz ki:

```
Domain=[ACCOUNTING] OS=[Windows 5.1]
Server=[Windows 2000 LAN Manager]
smb: \>
```

Itt adjuk ki a `help` parancsot és az ***smbclient*** megmutatja nekünk, milyen parancsokat használhatunk a kapcsolat

ideje alatt. Sok közülük egyértelmű, ilyen a `dir`, `copy` és társai. Ugyan ez sem rossz megoldás, de vizuális értelemben nem valami csinos, és grafikus fájlkezelőnkben sem tudjuk használni, nem beszélve az OpenOffice.org alkalmazásairól. Akár KDE, akár GNOME rendszert használunk asztali környezetként, biztosak lehetünk abban, hogy rendelkezésünkre áll a hálózati kapcsolatok kiépítéséhez szükséges eszköztár. A legjobb az egészben, hogy mindezt rendkívül egyszerűen tehetjük meg. Először vessünk egy pillantást a Konqueror-ra. Nyissuk meg a Konquerort (akár a fájlkezelőt akár a böngészőt) és gépeljük be az `smb://` kifejezést a ***Location*** mezőbe. A hálózati megosztásaikat hirdető Samba kiszolgálók és Windows gépek már meg is jelennek a böngésző ablakában saját munkacsoport nevük alatt (például ACCOUNTING, SALESGRP). Az *1. ábrán* egy Konqueror alkalmazást láthatunk két-paneles nézetben; kattintsunk a menüsorban a Window elemre és és válasszuk a ***Split View, Left/Right*** pontot. A baloldali panelen találjuk a három aktív munkacsoportot tartalmazó alap hálózati böngésző nézetet. A jobb oldali panelen az ACCOUNTING munkacsoporton kattintottam egyet, hogy lássam milyen gépek tartoznak a csoporthoz. Ettől kezdve minden a szokásos fogd-és-vidd grafikus fájlböngésző módszerrel történik. Kattintással, be tudok lépni a ***cooking*** mappába, kikereshetem a megfelelő dokumentumot, amit meg szeretnék nyitni. Általában persze nem szeretnénk állandóan végigjárni az egész navigációs utat. Néhány kattintásnyival közelebb hozhatjuk megosztásainkat, ha az elérési utat betesszük könyvjelzők közé. Ha a GNOME felől közelítjük meg a dolgokat, ott van nekünk a Nautilus.
A módszer nagyon hasonló a Konqueror esetében látottakhoz. Indítsuk el a Nautilust és gépeljük a ***Location*** sorba az `smb://` kifejezést. A Nautilus megmutatja nekünk milyen aktív munkacsoportok vannak a hálózatunkon. A gép kiválasztásához kétszer kattintsunk az egyik munkacsoporton. A számítógépek listájából válasszuk ki a nekünk tetsző gépet és máris választhatunk a felajánlott erőforrások közül. Akárcsak az iménti Konqueror példánkban, megtakaríthatunk egy kis időt, ha a kiválasztott mappát a könyvjelzők közé mentjük. Mindkét javaslattal csak az a gond, hogy egyik sem teszi lehetővé a hálózati meghajtók állandó felcsatolását. A kiválasztott mappa eléréséhez egy kis parancssori munkát is kell végeznünk, ami nem túl nagy feladat, de nem is afféle kattintgatós megoldás, amit a Windows felhasználók szeretnének látni a rendszerünkön. Kérjük meg François-t, hogy töltse meg újra poharainkat mi pedig addig megnézzük, milyen megoldást találhatnánk erre a problémára. Ha kicsit rugalmasabb módszert keresünk a hálózati kapcsolataink kezelésére, csak egy pillantást kell vetnünk az Smb4K nevű szuper-klasszis SMB böngészőre, amely egyszerre hatékony és rugalmas. Ráadásul az Smb4K segítségével előre megnézhetjük a megosztásokat, sőt, rendszergazdai jogosultság nélkül is felcsatolhatjuk, induláskor automatikusan csatlakoztathatjuk és egyéb dolgokat művelhetünk velük. A cikk születésekor az Smb4K még csak a 0.3.2 kiadásnál tartott, de igen hasznos csomagnak találtam amely egyértelműen megéri a vizsgálódásra szánt időt. Az Smb4K telepítését forrásból a szokásos tömörítünk és fordítunk ötlépéses folyamattal végezzük:

```
tar -xzvf smb4k-0.3.2.tar.gz
cd smb4k-0.3.2
```

2. ábra Smb4K: lehet, hogy ez a tökéletes SMB böngésző?

```
./configure --prefix=/usr
make
su -c “make install”
```

A csomag telepítése után az `smb4k` paranccsal indíthatjuk. Az Smb4K indulása után azonnal hozzálát a hálózat feltérképezéséhez, kikeresve az aktív megosztásokat. A képességeket testre is szabhatjuk a menüsor ***Settings*** pontja alatti ***Configure Smb4K*** pont segítségével. Beállíthatjuk például, hogy szeretnénk-e, ha a megosztások automatikusan újracsatlakoznának. A grafikus felület könnyen tanulható és jól kezelhető. Az egész csomagot nagyon egyszerű használni. A képernyő bal oldali kezelőpaneljén találjuk a munkacsoportok, számítógépek és megosztások felsorolását. A megosztás felcsatolásához jobb gombbal kattintsunk a néven és válasszuk a `mount` parancsot. Ha előbb szeretnénk megtudni, mit is találunk az adott helyen, inkább a ***Preview*** pontra kattintsunk. A felcsatolt meghajtók a jobb felső ablakban meghajtó ikonként látszanak. Az egyik meghajtó ikonon duplán kattintva megnyílik a Konqueror. Ha parancssorból kiadjuk a `df` parancsot, láthatjuk, hogy a meghajtók használatra készen felcsatolódtak, mégpedig saját könyvtárunk alá az Smb4K útvonal előtaggal. A *2. ábrán* bemutatott példának megfelelő lista a következőképpen nézne ki:

```
Filesystem        Size Used Avail Use% Mounted on
//SEDNA/Reports   4.0G 3.0G  1.1G  75%
➥/home/marcel/smb4k/SEDNA/Reports
//FRANCOIS/wine    13G 8.8G  3.3G  73%
➥/home/marcel/smb4k/FRANCOIS/wine
```

Mostantól bármilyen alkalmazásom – legyen az KDE, GNOME, héjprogram-alapú vagy bármi más – könnyen elérheti a megosztásokat. A hálózati szomszédságunkba látogatni soha nem volt még ilyen egyszerű.

Következő találkozásunkig, mes amis, igyunk egymás egészségére. A vôtre santé! Bon appétit!

Linux Journal 2004. július, 123. szám

Marcel Gagné

VI. GNU/Linux Konferencia
Felhívás előadóknak!

A Linux-Felhasználók Magyarországi Egyesülete 2004 novemberében (szombati napon) rendezi meg a VI. GNU/Linux szakmai konferenciát. Az előadásokat délelőtt 10:30-tól este 17:30-ig tartjuk.

Előadók jelentkezését várjuk az alábbi témákban:

– Rendszeradminisztráció
– Biztonság
– Programozás Linuxon
– Multimédia, grafika
– Honosítás
– Hazai fejlesztések
– DTP Linuxon
– Linux az üzleti életben
– Linux felhasználói lehetőségek
– Linux az oktatásban/államigazgatásban
– Egyéb érdekes, Linux-hoz kapcsolódó témák

Az előadásvázlatok beérkezésének határideje: augusztus 10. 00:00. Ekkorra tervezzük a konferencia honlapjának a megnyitását is, ahol azonnal elindul az előregisztráció és a vázlatok közötti szavazás, ezért kérnénk a határidő betartását.

A vázlat javasolt mérete egy A4-es gépelt oldal (0.5-1kByte), amit az abstract@lme.linux.hu címre várunk.

Az előadások előzetes válogatásáról augusztus 14.-ig küldünk értesítést. A végleges cikkek leadásának határideje szeptember 30., a prezentációké pedig október 7. Nyomtatásban már megjelent, vagy a konferencia időpontjáig megjelenő írást nem lehet beküldeni.

Az előadás anyagainak formátumára vonatkozó követelmények a

➲ http://lme.linux.hu/rendezvenyek/konferencia/kiadvanyforma.html

oldalon találhatóak meg.

A konferencia kiadványába csak a határidőre beérkező anyagok kerülnek be, amelyek közül a közönségszavazatok alapján legjobbnak minősülők kerülnek előadásra.

Az előadókat, mint mindig, VIP-vendégként üdvözöljük, azaz ebédhez és egyéb szolgáltatásokhoz jutnak.

A legjobb előadás – közönségszavazatok alapján – jutalomban részesül a konferencia végén.

Az LME elnöksége

Deus Ex: A játék!

Igen, a cím magáért beszél. Ez A Játék. Maga a program elvileg semmiben sem különbözik egy sor másik, hasonló darabtól, amelyek globális összeesküvés-történetet, remek grafikát, és zenét vonultatnak fel. A Deus Ex azonban ezt olyan magas szinten, és olyan kidolgozottan teszi, amely a kiadásakor egyedülállóvá tette. FPS és RPG keveréke, játékmenetével pedig a stílus legjobb darabja lett.

A történet 2052-ben játszódik. A világ káoszba süllyedt. Rengeteg a szegény ember, a hajléktalan, a lét bizonytalan. Óriáscégek uralkodnak, amelyek árnyékszervezeteik révén amolyan sötét irányítóként állnak a kormányok mögött, és felett. Ebben a világban a bűnözés teljesen összefonódott a legális üzlettel, a törvényt egyénileg értelmezik. És teljes mértékben globális a terrorizmus, ahol terroristák jól szervezett csoportjai dolgoznak össze a világ minden pontján.(illuminati hongkongban, a siluette párizsban, de terrorcsoport működik new-yorkban is) Ekkor lépünk be a UNATCO (***United Nations Anti Terrorist Coalition***) kötelékébe, a bátyánk után, JC Denton néven. Kiképzésünk után azonnal a terroristák után küldenek minket, ahol több küldetést kell sikerrel megoldani. Természetesen ez nem egyszerű feladat. A pályák tele vannak csukott ajtókkal, biztonsági kamerákkal, érzékelőkkel, automata gépfegyverekkel, őrökkel. Az RPG jelleg miatt skill kell szinte mindenhez. Amelyek természetesen fejleszthetőek. Minden feladat, küldetés, vagy egy új titkos hely felderítése tapasztalati pontot ér, mely beváltható képességre, vagy meglévő képesség fejlesztésére. Mindentudó karakter természetesen nem létezik. Vehetünk/kaphatunk/találhatunk kiegészítőket a fegyvereinkhez, amellyel gyakorlatilag egyéni, a nekünk leginkább megfelelő fegyverarzenált tudjuk kialakítani. Egy jól összerakott fegyvertár rendkívül hatékony.

Azonban azt nem árt észben tartani, hogy a játék erős Thief beütéssel bír, tehát a hentelés, akármilyen fegyvertárral is tegyük, a biztos halált jelenti. Előnyben kell részesíteni a lopakodást és a gondolkodást. Főleg a gondolkodásra lesz nagy szükség, hisz a párbeszédek során a beszélgető partnereink olykor mint a vízfolyás úgy hazudnak, megpróbálnak tőrbe csalni, hamis információt adnak. A számítógépek feltörése, és a mailek elolvasása viszont nagy segítség a tisztább kép kialakításához. Még bonyolultabbá teszi a játékot, hogy egy-egy cselekedetünk kihat a játék teljes menetére. Amint kezd kirajzolódni a történet, és Denton ügynök elkezd a dolgok mélyére látni, elmosódik a határ a jó, és a rossz fiúk között. Kétségessé válik, hogy kit is szolgálunk, kik azok akiket terroristáknak bélyegzünk, és mi a célunk. Ehhez azonban szigorúan titkos létesítményekbe kell belógni, sok kódot feltörni, és jó pár kulcsfigurával beszélgetni, no meg persze elvállalni a küldetéseiket. A játék legfőbb előnye, hogy a játékos aktív részese a cselekménynek. Minden, (tényleg minden) rajta múlik, és szinte tapintható a felelősség, hogy mit is eredményezhet egy rossz döntés. A Deus Ex szinte új etalont teremtett azzal, hogy beutazhatjuk a világot. NewYork és környéke, Hong-kong, Párizs, arizonai sivatag, 51-es körzet a főbb helyszínek, és a környező vidékük. Alattuk/bennük titkos kutató létesítmények, katonai raktárak, stb...
És mi sem lehetne meggyőzőbb érv a hangulat mellett, hogy a játék végén




dönteni kell. Háromféle módon dönthetünk, teljesen szabadon: Világuralom és új rend a földön, vagy szövetségre lépünk azzal aki megszervezte a világuralmat, esetleg ellenszegülünk, és ennek elpusztításával felszabadítjuk az emberiséget. Meglepő a játékmenetben (és az elején teljesen szokatlan), hogy gyakorlatilag mindenkivel beszélgethetünk. Mindent megtehetünk. Kirabolhatunk egy boltot, újságot olvashatunk, ha látunk egy lámpát felkapcsolhatjuk, ihatunk a vízből ha vízcsapot látunk, lehúzhatjuk a WC-t, automatából vehetünk üdítőt, és csokit, ehetünk, szó szerint sörözhetünk egy cigarettával, akár még kábítószert is fogyaszthatunk. A pályákon egyetlen olyan tárgy sincsen, amivel ne tudnánk mit kezdeni. Valóban hihetetlen, hogy ilyen szabadság létezik egy játékban. És mindez Linuxon is mehet.

A telepítése!

A játéknak natív linuxos portja nincsen sajnos. Anno a Lokigames kezdte el, de pont a DeusEx volt béta állapotú, amikor a cég csődöt jelentett. Így a natív kapu sosem készült el. Viszont tökéletesen működik a winex-es telepítője, ami semmi más, mint egy deusex winex-es konfiguráció, összedrótozva a Loki féle telepítő szettel. Ennél fogva (talán mondanom sem kell már, hisz mindenki tudja ezt) a telepítéshez kapcsoljuk ki az `automount`-ot. (az */etc/sysconfig/automount* fájl törlése, aztán újraindítás. Az elindításához csak újra létre kell hozni ezt a fájlt, és ismét újraindítani a gépet). A más rendszer alá készült DeusEx telepítő CD-re lesz még szükségünk, és természetesen a winex-re.
(➲ http:\\www.transgaming.com)
A DeusEx telepítőt a ➲ http://liflg.sourceforge.net/ oldalról tudjuk lehúzni. Telepítése a teljesen megszokott loki-módon történik. Külön öröm, hogy ha telepítve van a winex 3.x, és ezt a telepítőt lefuttatjuk, gond nélkül telepedik a játék, sőt mindenféle bütykölés nélkül fut is.

Gépigény

A játék gépigénye kicsit magasabb az emuláció miatt. De tényleg nem vészes, sőt meglepően jól fut. Egy 800-as duron processzoron, egy GeForce 2 Titanium kártyával 1024x768-as felbontásban remekül megy a program. Akinek mégis gondja akadna vele, az természetesen csökkentheti a felbontás, vagy a textúra részletességet is. Jó játékot mindenkinek ezzel a kiváló programmal.

Dancsok „strogg" Zoltán
(strogg@mail.tvnet.hu)
Jelenleg technikai szerkesztőként dolgozik a BME-OMIKK-nál, ahol oktat is. Emellett egyetemi képzésben vesz részt, programozó matematikus szakon. Négy éve foglalkozik Linuxszal. Szabadidejében operációs rendszereket gyűjt és weblapot vezet.