Beköszöntő

Linuxvilág

■ Februári számunkban elsősorban a linuxos kiszolgálók és az asztali rendszerek üzemeltetőinek igyekeztünk kedvezni. A két tábor igényei, illetve érdeklődési köre természetesen erősen eltér, de a ***Linuxban*** éppen az a szép, hogy titkárnőtől a rendszergazdáig mindenki találhat benne számára érdekes dolgokat.
Az emberek többsége – állítólag – hátulról, a sport rovat irányából kezdi olvasni az újságot. Nálunk ugyan ilyen nincs, de remélhetőleg kiválóan helyettesíti a multimédiáról és szórakozásról szóló rész. Maradva tehát a nemes hagyományoknál, kezdjük mi is innen az ismerkedést.
Aki számítógép helyett (mellett?) TV-vel akarja magát mérgezni, annak nagy segítség lesz ***Medve Zoltán*** TV-kártyák beüzemeléséről szóló cikke. Vele, és ***Kovács Zsolttal*** újraindítottuk a körülbelül egy éve szünetelő játékrovatot is. A jövőben különösen ***Zsolt*** jeleskedik majd a nyílt forrású vérontás, száguldozás és egyéb, ehhez hasonló szabadidős tevékenységek bemutatása terén. E havi számunkban a régi kiadású játékok ***Linux*** alatt megvalósítható újraélesztéséről írt nekünk.
Gyors ugrás a ***Magazin*** rovathoz...
Sajnos nem emlékszem pontosan hol, de minden valószínűség szerint valamelyik ***Linux HOWTO***-ban olvastam azt a kezdő rendszergazdáknak szóló tanácsot, hogy ha elvesztenék a türelmüket az idegőrlő hackelés során, keressenek valamilyen nagy méretű, súlyos tárgyat (a szerző emlékezetem szerint egy ruhásszekrényt tartott a legmegfelelőbbnek) és rugdossák kitartóan, amíg el nem múlik a roham. Nos, kérem, a magyar fejlesztéseknek köszönhetően nekünk jobb ötletünk is van. Lehet a rugdosás helyett például gyurmázni. A gyengébb idegzetűek tehát fáradjanak el egy hobbiboltba, vásároljanak némi süthető gyurmát, és tartsák mindig kéznél a krimpelő, a csavarhúzó és a telepítőlemezek közvetlen közelében. Ja, és mindenképpen olvassák el ***Hargas Ildikó*** cikkét, amelyből megtudhatják, hogyan kell az említett eszközök különleges felhasználásával sütni való pingvint formázni.
Ennyit a sportról, most pedig lássuk a többit.
A fejlesztői sarok ***Radics Péter*** ADA programozási nyelvet bemutató cikkével indul. Az ADA megalkotói szerint azért készült, hogy – logikai szigorának köszönhetően – nehezebben lehessen a fejlesztés során hibázni. Talán éppen ezzel magyarázható, hogy a nyelv a slendrián fejlesztőknek soha nem nyerte el a tetszését. Ha valaki utána szeretne járni, hogy mi is valójában a helyzet, nyugodtan megteheti, hiszen ***Linux*** alatt is létezik hozzá fejlesztőeszköz.
A ***vi*** szerkesztő ***vim*** nevű továbbfejlesztett változatát számos programozó tartja a legjobb fejlesztőkörnyezetnek. Hogy miért, azt megtudhatjuk a ***Fejlesztői sarok*** második cikkéből.
Az alkalmazások közül ebben a hónapban a ***Firefox*** (***Horváth Ernő***), a ***Bochs*** emulátor (***Markó Imre***), az ***Enlightenment*** környezet (***Apagyi György***) és a ***Nero for Linux*** (***Kenderesi Norbert***) kerül terítékre.
Ami pedig az üzemeltetést illeti, ***Auth Gábor*** az ***RRDTool***-ról kezd egy sorozatot, ***Novák Áron*** a ***Grsecurity*** foltot mutatja be, míg ***Markó Imre*** az ***IPTraf*** nevű hálózatelemző szoftverről írt.
A kiszolgálók üzemeltetői természetesnek veszik, hogy rendszereik biztonságával is foglalkozniuk kell. Ugyanez az asztali rendszerek esetében már nem ennyire egyértelmű, pedig a biztonsági beállításoknak itt is óriási jelentősége lehet. Ezt a témát, vagyis a desktop rendszerek biztonsági kérdéseit boncolgatja egy most induló sorozatban ***Molnár Norbert***. És ha már a biztonságánál tartunk, néha nem a külső ellenség az, akitől tartanunk kell, hanem ***Murphy***, aki mindent elront, amit el lehet. Ilyenkor jön jól aztán a biztonsági mentés, amelynek elkészítéséről ***Illés Viktor*** írt nekünk.

Kellemes időtöltést, jó szórakozást kíván a Linuxvilág stábja!

Hírek

Desktop Multiplier próbamenet

A kanadai ***Userful Corporation*** bejelentette, hogy korlátozott ideig ingyenesen elérhetővé teszi ***Desktop Multiplier*** alkalmazásának két felhasználós változatát. A ***Desktop Multiplier*** segítségével egy ***Linuxot*** futtató számítógép – külön billentyűzet, monitor és egér biztosításával, illetve szükség szerint további videokártyák beépítésével – több, akár 10 felhasználót is képes kiszolgálni. A két felhasználó számára készült változat kipróbálásához különösebb gépbővítést sem kell végezni, egy két kimenettel rendelkező videokártya is elegendő, a külső eszközök csatlakoztatása után azonnal rendelkezésre áll a második munkaállomás.

➲ ***http://userful.com/products/free-2-user***

OX-demó

Az ***Open-Xchange*** fejlesztői újabb eszközt vetnek be, hogy az érintettek figyelmét ráirányítsák termékükre: a csoportmunka-alkalmazást „élő” ***CD*** lemezt is kínálnak. A lemezen található, ***Knoppix*** alapú rendszert elindítva telepítés és a meglévő környezet módosítása nélkül, kockázatmentesen próbálhatók ki az ***Exchange*** helyettesnek szánt összeállítás 5-ös változatának szolgáltatásai, mint a levelezés, a naptár-, névjegy és feladatkezelés, a projektkövetés, a dokumentumok megosztása és a tudásbázis. A beetetést szolgálja, hogy a bemutató lemezen az ***Outlook OXtender*** is megtalálható – az a kiegészítő, és kizárólag díj ellenében elérhető modul, amellyel a ***Microsoft Outlook*** használói is csatlakozni tudnak az ***Open-Xchange*** kiszolgálókhoz.

➲ ***www.open-xchange.com/live***

Munkaállomás-bőrönd

Az asztali számítógépek és a hordozható gépek közötti úton sokféle módon sikerült már letáborozni, a ***Dell*** azonban úgy vélte, tudnak ők még újat mutatni a világnak. Össze is állítottak egy hordozható munkaállomást, majd ráaggatták a ***XPS Mobile Concept PC*** nevet. A szerény 20”-os kijelzővel felszerelt, valójában a kijelző talpában elhelyezett kisméretű géphez érintőpaddal ellátott billentyűzet és nyolc hangszóró is tartozik, és az egész csomag táskaszerűen összefogható, összehajtható, majd monitor hátoldalán kialakított támasztéknál fogva kényelmesen magunkkal vihető. A belsőleg minden földi jóval, többek között kétmagos processzorral rendelkező gép a noteszgépeknek aligha támaszt konkurenciát, a sokat utazó grafikusok és tervezőmérnökök igényeinek azonban biztosan kiválóan megfelel.

Kívül a belül

Ki ne ismerné a bekarikázott ***Intel Inside*** jelmondatot? 1991 óta számtalan számítógépen és hirdetésben láthattuk viszont, most azonban elérkezett a nyugdíjazás ideje. Maga a logó megjelenését tekintve visszaidézi elődjét, az ***„odabent Intel van”*** jelmondat helyét azonban átveszi az „ugorj előre”. Érdekes kérdés, hogy az asztali processzorok terén az elmúlt időszakban meglehetősen halovány szereplő, ám a mobil technológiák piacán komoly sikereket elérő ***Intel*** miért éppen most határozta el magát a váltás mellett, az azonban tény, hogy az egyszerű személyi számítógépek szerepe – illetve az általuk elérhető haszon – egyre kisebb, a digitális szórakoztató eszközök, a mobilitás és a számítástechnikával összefonódó szórakoztatóelektronika térhódítása viszont egyre komolyabb. Az ***Intel***-külcsín megújulása is azt sugallja: itt valami történik; már csak azt nem tudni, hogy valóban elértünk valahova a fejlődés során, vagy csak látványos marketinghúzásra volt szüksége egy az útját kereső vállalatnak.

Nyílt titkok

Különleges játékszerrel kedveskedik a ***WD*** az egyedi számítógépek építése iránt rajongóknak. A ***WD Raptor X*** merevlemezek a kedvelt ***Raptor*** sorozat legújabb tagjai, kapacitásuk immár eléri a 150 GB-ot, csatolófelületük továbbra is ***SATA***, fordulatszámuk maradt 10000 1/s, azonban házuk egyik oldala átlátszó. Aki ebben leli örömét, az megfelelő, például plexiből készült ház beszerzése után esténként kellemes neonfénynél követheti a meghajtó író-olvasó fejeit hordozó karok csapkodását. A ***Raptor X*** ára már kevésbé kellemes, a cég weblapján az előrendelés keretében 349 dolláros, vagyis 70 ezer forintos áron kelleti magát.

➲ ***http://www.wdraptorx.com***

A társasjátékok feltámadása

Entertaible névvel – egyelőre kísérleti példányként létező – elektronikus játéktáblát mutatott be a ***Philips***. A készülék lényegében egy 30"-os ***LCD*** érintőképernyő, amit vízszintes elhelyezésre terveztek, és kiegészítő funkcióként képes a bábuk, kockák, játékelemek helyzetének felismerésére. A képernyőn meg lehet jeleníteni a hagyományos társasjátékok játékmezőit, és máris előállt egy a korábbi játékok hangulatát megőrző, azokhoz képest mégis sokkal több lehetőséget kínáló játék. Lehetőség adódik például arra, hogy a játékosok személyre szabott stratégiai tanácsokat kapjanak, megjelenítsék a játékszabályokat, elmentsék a játékállást, illetve kihasználják mindazokat a multimédiás és hálózati szolgáltatásokat, amiket egy ilyen eszköz nyújtani képes. Az ***Entertaible***-t eleinte bároknak, kaszinóknak kínálják majd, de később otthoni használatra szánt változatot is megjelentetnek belőle – vélhetően a késleltetésben a nagyméretű ***LCD*** kijelzők ára is szerepet játszik.

Csillagfényes estékre

Az amatőr csillagászokat célozza a ***Celestron*** legújabb érdekessége, a ***SkyScout*** készülék. A csillagkémlelő eszköz képes arra, hogy különféle érzékelők segítségével megállapítsa saját helyzetét, és az égbolt éppen megfigyelt részének legérdekesebb objektumaival kapcsolatos információkat adjon át szöveg és beszéd formájában. A ***SkyScout*** arra is alkalmas, hogy bármely csillagot azonosítsunk vele, elég a kiszemelt égitestre irányítani, adatbázisa alapján azonnal kikeresi a vonatkozó tudnivalókat. Az USB kapcsolaton keresztül frissíthető szoftver futtató készülék a fontosabb égi eseményeket is ismeri, így használója figyelmét rá tudja irányítani az érdekesebb eseményekre, például a meteorrajok érkezésére vagy az űrrakéták fellövésére.

➲ ***http://www.celestron.com/skyscout***

Sportnotebook

Az ***ASUSTeK*** és az ***Automobili Lamborghini*** együttműködése során egyesülnek az információtechnológia és az autóipar fejlesztései és öröksége – bármit is jelentsen ez, az ***Asus*** noteszgépei között hamarosan a ***Lamborghini Notebook Series*** tagjait is megtaláljuk majd. A sportautókról ismert felirattal és márkajelzéssel ellátott gépek belsejébe vélhetően mindenből a legnagyobb és leggyorsabb kerül majd, illetve a sárga vagy fekete színben kérhető gépek külseje, különleges, tükörhatású festése is az exkluzivitást fogja majd tükrözni. A ***Lamborghini*** sorozat tagjainak áráról egyelőre nem tudni semmit, ám az ***Acer*** nem túl rég kiadott ***Ferrari*** sorozatának sorsából kiindulva eleinte valóban a különlegességek közé sorolódnak, majd néhány hónap elteltével dicstelenül a leszállított árú maradékok közé kerülnek. A híres sportautók hiába számítanak érdekességnek akár 30 évesen is, a nevüket viselő számítógépeket ez sem mentheti meg a néhány hónap alatt történő elavulástól.

Sugárzó övezet

A ***SercoNet*** nevű, floridai cég a vezeték nélküli hálózatok fontos problémáját célozza meg fejlesztésével: a felhasználók igényeit ki nem elégítő lefedettséget. A vezeték nélküli hozzáférési pontokat vásárlóknak a statisztikák szerint körülbelül egynegyede elégedetlen a termékkel, és jelentős részük további antennákat, erősítőket kénytelen vásárolni – amelyeket viszont még nagyobb arányban hordanak vissza az üzletekbe. Ezen próbál segíteni ***WirePlus Broadband*** technológia, amely apró, a hozzáférési pont közelében és az épület más pontjain található telefonos aljzatokba csatlakoztatható eszközökben testesül majd meg. Az eszközök a telefonkábeleket rádiós jeltovábbításra fogják használni, így a vezeték nélküli hálózatok lefedettségi területét jelentős mértékben meg tudják majd növelni. A megoldás érdekessége, hogy a kiegészítő eszközök nem végeznek semmilyen magasabb szintű hálózati tevékenységet, szerepük kizárólag annyi, hogy a falakban húzódó kábelek mentén továbbítsák a rádiós jeleket, gyakorlatilag megtoldva ezzel a hozzáférési pont antennáját. A piacon ugyan kaphatók már olyan eszközök, amelyeket hasonló feladatra terveztek, ám ezek hálózati kapcsolatot létesítenek például az erősáramú hálózaton keresztül, így inkább jelismétlő szerepet játszanak. A cég elképzelései szerint az első ***WirePlus*** termékek a jövő év első negyedében jelenhetnek meg, nagyjából 50-60 dolláros áron.

➲ ***www.serconet.net***

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

Mi újság a rendszermag fejlesztése körül

Ahmad Reza Cheraghi elkezdett dolgozni egy olyan infrastruktúra kialakításán, amely lehetővé teszi, hogy a kernel ***.config*** fájlját a rendszer automatikusan, a hardvert átvizsgálva állítsa össze. Ez a projekt tulajdonképpen nem is egy teljes automatizálási megoldás kialakítását tűzte ki célul, inkább egy olyan keretrendszer létrehozását, amely lehetővé teszi és összerendezi a későbbi fejlesztéseket. ***Ahmad*** úgy gondolja, hogy a projektbe beszálló fejlesztők idővel egyre több hardverdetektáló modult írnak majd, amelyeket az ő keretrendszeréhez csatlakoztatva a ***make autoconfig*** parancs találatai egyre pontosabbak lesznek. A projekt meglehetősen vegyes érzelmeket váltott ki a fejlesztői közösségből. ***Hua Zhongot*** például egyenes felvillanyozta az a lehetőség, hogy a kernelfordítás mágikus művelete ismét egy lépéssel közelebb kerülhet a közönséges felhasználókhoz. ***Roman Zippelnek*** ugyanakkor komoly kétségei vannak azzal kapcsolatban, hogy egy efféle projekt valaha is sikeresen befejeződhet. Szerinte ez a kezdeményezés legjobb esetben is sokáig félkész állapotban hervadozik majd. ***David Teigland*** egy ideje már próbálkozik azzal, hogy fölvetesse a hivatalos kernelbe a ***GFS2*** klaszterfájlrendszert. Az ezzel kapcsolatos első probléma maga a név. A ***GFS*** fejlesztői egyrészről úgy gondolják, hogy művük megérett a kettes verziószámra, másrészről viszont nem hajlandóak elismerni, hogy ezzel gyakorlatilag egy új projektet hoztak létre, aminek ismét át kell esnie a szokásos vizsgálatokon és engedélyezési folyamaton, mielőtt bekerülne a rendszermagba. A ***GFS*** jelenleg része a kernelnek, vagyis egy közönséges frissítésnek ennél valóban sokkal könnyebb útja lenne a kernelfáig. Sokan azonban úgy gondolják, hogy itt ennél többről van szó. A ***GFS2*** nem teljesen kompatibilis az előző változattal. Különösen a lemezformátum változott jelentősen a két verzió között, méghozzá annyira, hogy ***GFS*** könyvtárszerkezeteket nem is lehet becsatolni ***GFS2*** alá. Ezen kívül más, kisebb eltérések is vannak a két változat között, így érhető, hogy ***Andrew Morton*** vonakodik zöld utat adni a ***GFS2***-nek, amíg a szerzők kellően meg nem győzik arról, hogy ezt kell tennie. Ami a jelenlegi helyzetet illeti, a meggyőzés még folyamatban van. Hosszú bizonytalanság után a ***FUSE (Filesystem in Userspace)*** végül bekerülhetett a kernelbe. Ami azt illeti, ***Linux Torvalds*** soha nem szerette ezt a projektet, mivel szerinte egy fájlrendszer lényegét lehetetlen igazán jól elszigetelni a kernel belsejétől. ***Szeredi Miklós*** a fő fejlesztő azonban végül válaszolt minden ellenvetésre, és hitelt érdemlően megmutatta, hogy a rendszerben benne maradt furcsa részletek jelenleg semmilyen módon nem kerülhetők el. Igazából eddig is csak ennyi hiányzott a jóváhagyáshoz. Mivel a ***FUSE*** régóta anélkül szerepelt ***Andrew Morton -mm*** fájában, hogy különösebb frissítést igényelt volna, vagy hogy bármely más fejlesztő bármilyen konstruktív javaslattal tudott volna előállni az említett problémákkal kapcsolatban, ***Andrewnak*** és ***Linusnak*** nem volt több ellenvetése. A ***FUSE*** először a ***2.6.14-rc1*** fában fog megjelenni, a ***2.6.14***-es stabil rendszermagnak pedig hivatalosan is része lesz.

Kétségek merültek fel azzal kapcsolatban, hogy ***Andrew Morton*** bírja-e kernellel kapcsolatos teendők miatt a rá nehezedő terhelést. ***Linus Torvalds*** nemrég a következőt mondta ezzel kapcsolatban: ***„Attól tartok hogy Andrew egyszer csak ott fog tartani, ahol én néhány évvel ezelőtt. Túlhajszolt lesz és feszült a több tonnányi kernelfolt miatt."*** Valóban igaz, hogy ***Andrew*** kezén napi szinten is rengeteg ilyen kódrészlet halad át, de szerinte az igazi gond nem a mennyiséggel van. A problémát azok a foltok jelentik, amelyek egyszerűen nem működnek, hiszen ezzel adott esetben órák telhetnek el. Egy-egy végleges ***-mm*** fa kiadása meglehetősen időigényes, mert legalább négy különböző architektúrán kell hibátlanul bebootolnia, és legalább héten gond nélkül fordulnia. Bármilyen hiba esetén a problémás foltot el kell távolítani, majd elölről kell kezdeni az egész folyamatot. A gyakorlatban ez az jelenti, hogy ha csak egyetlen ilyen hiba is bekövetkezik, az a kérdéses kernelváltozat hivatalos megjelenését a következő napra tolja át. A dolognak pedig itt még mindig nincs vége, hiszen mindeközben folyamatosan áramlanak befelé az újabb és újabb foltok. Mindezek ellenére ***Andrew*** egyelőre nem érzi túlterheltnek magát, sőt arra is van ideje, hogy figyelemmel kísérje a kernelhibák adatbázisát és nyaggassa a fejlesztőket, hogy válaszoljanak a munkájukkal kapcsolatos kérdésekre.

Greg Kroah-Hartman átadta az ***I2C*** alrendszer fejlesztését ***Jean Delvare***-nak. Ők ketten tulajdonképpen hónapok óta szoros együttműködésben dolgoztak ezen a projekten, így nyilván jó előre felkészültek erre a váltásra, amit ***Greg*** most hivatalosan is bejelentett. Ő maga amúgy továbbra is részt vesz a fejlesztésben, csak kisebb kapacitással, az ***I2C***-vel kapcsolatos foltokat pedig továbbra is rajta keresztül kell benyújtani ahhoz, hogy ***Andrew Morton*** és ***Linus Torvalds*** beillessze azokat az általuk fenntartott fákba.

Zack Brown

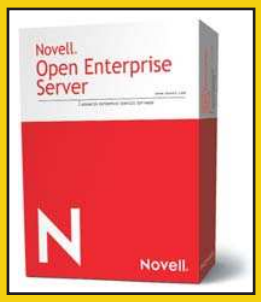
Linux Journal 2005. 141. szám

600 000 munkaállomásból álló rendszert épít a Novell

A Novell nyílt forráskódú, szabványalapú technológiai megoldásait választotta heterogén informatikai környezetének felügyeletére, biztonságossá tételére és integrálására az Egyesült Királyság Nemzeti Egészségügyi Szolgálata (National Health Service)

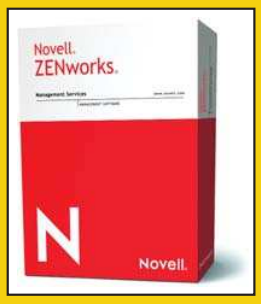
A *Novell* és az *Egyesült Királyság Egészségügyi Minisztériumának* vezető ügynöksége 39 millió dollár értékű szerződést kötött olyan átfogó biztonsági, felügyeleti és infrastruktúra szoftvermegoldások létrehozására, amelyek magasabb szintű egészségügyi szolgáltatásokat biztosítanak az *Egyesült Királyság* állampolgárai számára. A *Nemzeti Egészségügyi Szolgálattal (National Health Service – NHS)* kötött hároméves megállapodás a *Csatlakozás az Egészségért (Connecting for Health)* program keretében lehetővé teszi, hogy a teljes szervezeten belül, az *NHS* több mint 600 000 munkaállomásból álló infrastruktúráján alkalmazzák a *Novell* megoldásait. A várakozások szerint ez jelentős költségmegtakarítást eredményez az egészségügyi szervezet számára. A *Novell* stratégiai partnerként további támogatást nyújt az *NHS* számára *IT* programjának megvalósításában, amely a betegellátás és ahhoz kapcsolódó

Az NHS-nél üzemeltetett Novell megoldások alapelemei



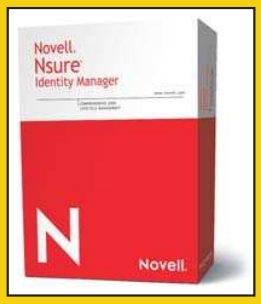
Novell Open Enterprise Server

A világ első vegyes forráskódú vállalati megoldása biztonságos és megbízható hálózati szolgáltatáscsomag, amely a *NetWare* és *SUSE Linux Enterprise Server* kombinálásával fejlett tárolási, fájlmegosztási, nyomtatási, felügyeleti, csoportmunka és alkalmazásszerver szolgáltatásokat nyújt. A *Novell* teljes támogatási rendszerét kínálja a nagyvállalati igények kielégítéséhez.



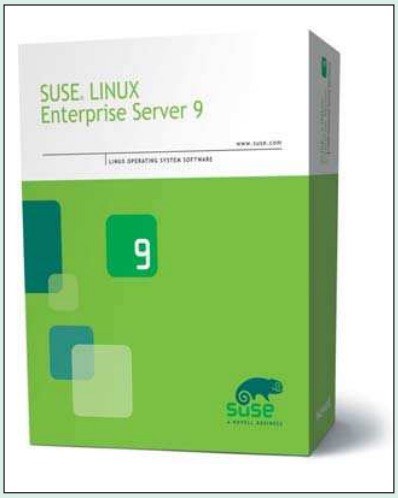
Novell ZENworks

A *Novell* integrált és teljes körű felügyeleti megoldása megbízható, automatizált és személyre szabott munkakörnyezetet biztosít az asztali, hordozható, szerver és kéziszámítógépek teljes életciklusán keresztül. *NetWare*, *Linux* és *Windows* platformon is megoldást nyújt, mind desktop, mind szerver oldalon. Leegyszerűsíti az informatikai erőforrások telepítését, kezelését és karbantartását, miközben csökkenti a költségeket, javítja a termelékenységet, és időt takarít meg.



Novell Nsure Identity Manager

A *Novell* metacímtár megoldása integrálja a különböző korokból származó informatikai rendszereket, szinkronban tartja az intézmény adatait és alkalmazásait. Az egységes, integrált adatok és alkalmazások a felhasználók számára személyre szabottan, biztonságosan, a szervezeten belülről és kívülről egyaránt, az év bármely napján elérhetők.



szolgáltatások, valamint az *NHS* szervezeti működésének javítását célozza.
Az *NHS* informatikai infrastruktúráját érintő főbb célkitűzések: a rendszer folyamatos magas szintű működésének biztosítása, a szoftvertermékek telepítésének és frissítésének egységesítése, valamint az *IT* rendszer hatékonyságának növelésével a dolgozók termelékenységének növelése is.
Az új informatikai infrastruktúra több mint 100 000 orvost, 380 000 ápolót és további 50 000 egészségügyi szakembert köt össze, biztosítva ezzel a több mint 50 millió angliai páciens zökkenőmentes ellátását.
Az *NHS* dolgozói számára a *Novell* szakmai képzést is biztosít.
Az adatok és alkalmazások szinkronban tartására, a hozzáférések és jogosultságok kezelésére az *NHS* teljes szervezete a *Novell Identity Managert*, a munkaállomások felügyeletére pedig a *Novell ZENworks* terméket fogja használni. Az *NHS* a szervezet egyes egységeinél a *Novell Open Enterprise Servert* is bevezeti, így lehetővé válik a megszokott, magas színvonalú *NetWare* szolgáltatások elérése *Linux* rendszereken is – a létező infrastruktúra lecserélése nélkül.

Kormányzati szinten is hódít a Novell SUSE Linux

Az elmúlt években fokozatosan egyre több kormányzati részleg vezette be a Linuxot, azonban ez az első eset Svájcban, hogy szövetségi szinten hivatalos közbeszerzési eljárás során döntöttek a Linux bevezetése mellett

A svájci szövetségi kormányzat is *Linuxra* vált.
A *Novellel* kötött kulcsfontosságú megállapodás keretében a *Novell* vállalati szintű *Linux* technológiáját vezetik be *Svájc* közszférájának *IT* infrastruktúrájába.
Hivatalos közbeszerzési eljárás során választották a *Novell* megoldásait, mivel a vállalat meggyőzően bizonyította, hogy a svájci közszféra elvárásainak legjobban megfelelő *Linux* megoldást kínálja. A megállapodás eredményeként több mint 3000, a svájci szövetségi kormány által működtetett szerver mostantól a díjnyertes *Novell SUSE Linux* platformon fut.
A *Novell SUSE Linux Enterprise Server* operációs rendszer nyerte el a *Legjobb Vállalati Szerverdisztribúció (Best Enterprise Server Distribution)* díjat, amelyet a *LinuxWorld Conference & Expo* rendezvényen adtak át *Frankfurtban*. A 200 tagú nemzetközi zsűri szavazatainak 53 százalékát begyűjtő *SUSE Linux Enterprise Server* maga mögé utasította a 37 százalékot kapott *Red Hat Enterprise Linux* és a 8 százalékot elért *Mandriva* rendszereket, így megkapta a legjobb vállalati *Linux* rendszernek járó *Linux New Media* díjat, amelyet idén hatodik alkalommal adtak át. A szakma elismerését jelentő díjat magasan kvalifikált zsűri adta át, amelynek tagjai között fejlesztők, szerzők, iparági szakértők és közigazgatási képviselők is voltak.

Linux New Media Díj 2005

Szoftver

Legjobb Vállalati Szerverdisztribúció

1.	Novell/Suse	53.3 %
2.	Red Hat	36.7 %
3.	Mandriva	8.3 %

A díjnyertes Novell SUSE Linux Enterprise Server

A *SUSE Linux Enterprise Server* egy biztonságos, megbízható platform nyílt forráskódú számítástechnika használatához a vállalaton belül.
A *SUSE Linux Enterprise Server* páratlan teljesítményt és méretezhetőséget kínál, átfogó, nyílt forráskódú funkcionalitást, valamint hardverplatformok és szoftvercsomagok széles körének támogatását. Ezenfelül nyílt alkalmazásprogramozási felületeket (*API*-kat) és más fejlesztőeszközöket is tartalmaz, amellyel leegyszerűsíthető a *Linux*-integráció és a rendszerek testreszabása.

Tux süthető gyurmából

Némi kreativitással megáldott rendszergazdák egyszerűen el tudják készíteni saját kis Linux kabalafigurájukat. A jószág aztán használható kulcstartóként, ceruzára húzható dísz gyanánt, vagy egyszerűen ülhet békésen az asztalon.

Vásároljunk valamilyen kézműves- vagy hobby-termékeket forgalmazó üzletben fekete, fehér és narancssárga süthető gyurmát. Általában 3-4 fajta gyurmát forgalmaznak, mindegyik jó, csak az a lényeg, hogy ujjal ne tudjuk könnyen benyomni. Ha túl puha az alapanyag, akkor nem lehet rendesen formázni, ha túl kemény, akkor pedig nehezebben tudunk vele dolgozni, és morzsálódhat is.

A gyurmák általában 130 fokon süthetőek, a sütési idő a mérettől függ. Az itt leírt figura körülbelül 10 perc alatt sül ki. Sütés után egy viszonylag kemény, a mindennapi használatnak ellenálló tárgyat kapunk.

Ha kulcstartót szeretnénk belőle, akkor szerezzünk be a fent említett üzletekben egy kisméretű képakasztót, és egy kulcstartókarikát is. Szükségünk lesz egy vékony gumikesztyűre is, hogy a figurán ne maradjanak ujjlenyomatok.

Minden lépést úgy kezdünk, hogy egy kis golyót készítünk, majd ebből alakítjuk ki a kívánt formát.

A testnek szánt golyó körülbelül 4 cm átmérőjű legyen. Ezután tojás alakúra hengergetjük, végül az ujjunk segítségével a tojásból egy kugliformát alakítunk ki.

Fehér gyurmából elkészítjük a kicsit tojás alakú hasát, majd rányomkodjuk a testére, a csőrét is megformázzuk, majd nyomjuk rá a fejére, úgy hogy a has véget is lefedje.

Ilyenkor alakítható ki a csőr végleges formája. Egy tű segítségével szúrjunk neki orrnyílást.

1. ábra A kiinduló golyó

2. ábra Tojás

3. ábra A végleges testforma

4. ábra A has arányosan a testhez

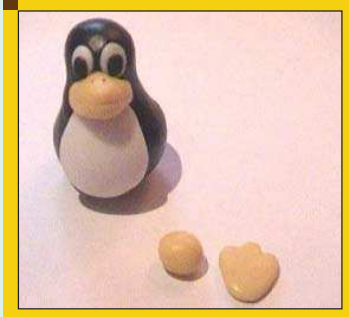
5. ábra Lekerekített háromszög a csőrnek

6. ábra Megtűzzük...

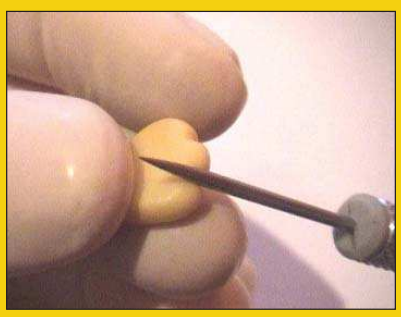
7. ábra Óvatosan a késsel!

8. ábra A szemet közvetlenül a csőr fölé tesszük

9. ábra A szembogarat le, és középre igazítsuk



10. ábra A láb és a test mérete legyen arányos!



11. ábra Kialakítjuk a tappancsát...

12. ábra Szárnyak...

13. ábra A jobb szárny felhelyezése

14. ábra Jobb láb a helyére

15. ábra Hölgyeim és uraim: Tux, sütés előtt, rémülten

16. ábra Sült Tux, kulcstartón

A csőrét egy élesebb eszközzel karcoljuk be: máris vigyorog. A szemének formázzunk kis fehér golyókat, majd lapítsuk ki ovális formává, és nyomjuk rá a fejére. Fekete gyurmából is készítsünk egy ovális formát, azt pedig nyomjuk a már a fején levő szem közepére.

Előkészítjük a talpait és a szárnyait: a talpnak egy háromszög alakú gyurmát készítünk, majd egy tű segítségével két helyen benyomjuk, a szárnyaknak pedig egy-egy elvékonyodó hengert formázunk.

Először a jobb szárnyvéget rányomjuk a pingvin hátára, majd a jobb talpat is rátesszük. A bal oldalon a talppal kezdjük, utána rögzítjük a szárnyát.

Ha már szembenéz velünk a figuránk, akkor az arckifejezésén tetszés szerint alakíthatunk.

Ha kulcstartót szeretnénk belőle, akkor a szúrjunk bele a fejébe sütés előtt egy vastagabb tűt. Ha megsült utána becsavarhatunk egy képakasztót a lyukba, és tehetünk rá kulcstartót.

Ha ceruzára szeretnénk ültetni, akkor óvatosan alulról egy ceruzával alakítsunk ki neki helyet.

Egy sima felületre helyezve tegyük a sütőbe, és a megadottak szerint süssük készre. Vigyázzunk, a gyurma csak akkor szilárdult meg igazából, ha már teljesen kihűlt!

Hargas Ildikó
(pomat@svajcisapka.net)

Kevés közöm van a Linuxhoz, sőt, nem is vagyok szakmabeli. Viszont szeretek mindenféle kézműves dologgal foglalkozni és ezzel másoknak örömet szerezni.

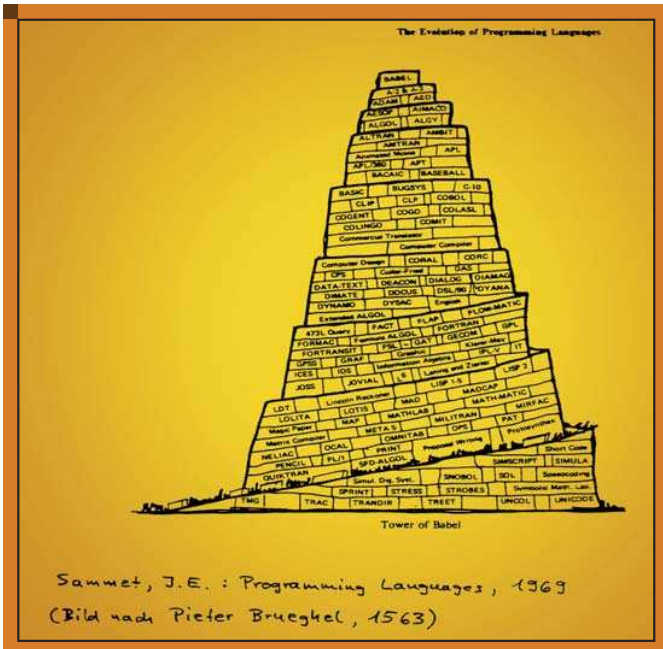
KAPCSOLÓDÓ CÍMEK

A fent említett hobbyüzleteket megtalálhatod a ➲ www.kreativ.lap.hu oldalon.

Biztonság, megbízhatóság, ADA

Biztosan felmerült már a kérdés sokakban, hogy azokon a helyeken, ahol életek függnek számítógépektől, ahol a hibák lehetőségét teljesen minimalizálni kell, milyen programozási eszközt, vagy eszközöket használnak. Ma már temérdek programnyelv létezik. Vannak ismertek és kevésbé ismertek. Véleményem szerint az utóbbiak közé tartozik az ADA nyelv, mely megpróbált békét teremteni a programozási nyelvek bábeli zűrzavarában, sokszínűségével, gazdagságával és elég szigorú szabályaival.

A nyelv története 1970-es évek elejére nyúlik vissza. Ekkor szervezte meg ugyanis a ***US DoD (United States Department of Defense)*** a ***HOLWG (High Order Language Working Group)*** nevű projektet ***William Whitaker*** vezetésével. Először nem is az volt a feladat, hogy teremtsenek egy új nyelvet. A projekt célja, az akkori programozási nyelvek vizsgálata volt. Sajnos a szigorú követelményeknek nem nagyon feleltek meg a korabeli nyelvek. A követelmények tömören a következők voltak: algoritmusok implementálásához jó és egyszerű jelölés rendszert biztosítson, eszközt adjon a programok bonyolultságának kezelésére, valamint lehetőséget adjon arra, hogy programozáskor el tudjunk vonatkoztatni a számítógéptől. 23 létező programozási nyelv jöhetett számításba: ***FORTRAN, COBOL, PL/I, HAL/S,***



1. ábra A bibliai párhuzam: Bábel tornyának története

2. ábra Ada Augusta Byron

TACPOL, CMS-2, CS-4, SPL/I, JOVIAL J3, JOVIAL J73, ALGOL 60, ALGOL 68, CORAL 66, Pascal, SUMULA 67, LIS, LTR, TRL/2, EUCLID, PDL2, PEARL, MORAL, EL/I

Sajnos a vizsgálat során a szakértők rájöttek: nincs olyan nyelv mely kívétel nélkül minden specifikált követelménynek megfelel. Arra az álláspontra jutottak, hogy ki kell alakítani egy új nyelvet, mely az eddigi nyelvek értékeiből merít és egymagában meg fog felelni a célnak. Tettek bizonyos megkötéseket. Például nem vehették számításba, az akkor már elavultnak számított nyelveket, mint a ***CORAL*** vagy a ***FORTRAN***. Voltak melyekből lehetett meríteni elemeket, például ***LIS***, ***RTL***. Alapnyelvnek választották a ***PASCAL***, ***ALGOL 68*** és ***PL/I*** nyelveket.

A kitűzött pályázatot végül is egy franciaországi labor nyerte el: ***CII Honeywell Bull*** és alapnyelvül a ***Pascalt*** választották.

A fejlesztés során a kiinduló projekt nevéből fakadóan a ***DoD-1*** nevet kapta, majd 1979 májusában keresztelték el, mint ***ADA***. A név ki mást takarhatna, mint a világ első programozóját ***Ada Augusta Byront***, ***Lord Byron*** lányát, aki ***Babbage*** asszisztense volt.

A fejlesztési szempontokat jobbára megbízhatóság és karbantarthatóság övezte. Ezért a nyelv erősen típusos lett és lehetőséget ad adatabsztrakciók kezelésére is. Karbantarthatóságot szolgálván több fordítási egységből áll. Ami pedig nagyon is lényeges, eszközöket szolgáltat a kivételesen hibás esetek kezelésére.

Az első 1983-as szabvány tehát a Pascalra épül, de ezen kívül hordozta többek között az ***EUCLID***, ***ALGOL 68***, ***SUE***, ***MODULA***, ***CLU*** nyelvek vonásait.

1995-re megnőttek az igények, megnőtt a hardverek kapacitása új technológiák terjedtek el, itt volt az ideje, hogy az ***ADA*** is felzárkózzon. Ebben az évben született meg az ***ADA95***, mely több új tulajdonságot hordozott: könnyebben illeszkedett más nyelveken íródott rendszerekhez, támogatta az objektum orientált programfejlesztést, a könyvtárszerkezetek létrehozását is hierarchikusabbá tették valamint párhuzamos programozást támogató elemekkel is bővült a repertoár.

Bár nehezen programozható, mégis valahol ebben rejlik az erőssége, nem hagyja, hogy a programozó butaságot kövessen el. Amire csak figyelmeztethet a fordító, arra figyelmeztet is, de nem csak a fordító lett ügyesen megalkotva, a nyelv szabályai eleve a logikus gondolatmenetet segítik elő.

Nagyon sok helyen használják, ahol a biztonság elsődleges szempont. ***ADA*** program irányítja többek között a ***Boeing 777***-esek erőforrás kezelő rendszerét valamint fék berendezéseit. Kigondolná hogy a ***GPS*** rendszerekhez tartozó műholdakban is helyet kapott pár ***ADA***-ban fejlesztett rendszer. A felhasználási területeket lehetne még sorolni: rakéták vezérlőrendszerei, párizsi, kairói, metró rendszer,a francia ***TGV*** irányítórendszere.

Az ***ADA*** felhasználási területeiről Többet megtudhatunk ha ellátogatunk a ➲ ***http://www.adahome.com/Ammo/Success*** címre.

Természetesen az ***ADA*** rendelkezik ***Linux*** alatt futó fordítóval is, melynek neve ***GNAT***. A fordító beszerezhető a ➲ ***http://www.gnat.com/*** címről, de a legtöbb ***Linux*** disztribúció alapból tartalmazza.

Most, hogy már tudunk valamit a nyelvről, ideje kipróbálni, hogy hogyan is néz ki a gyakorlatban. Mivel nem férne bele ebbe a szerény cikkbe, minden amit érdemes tudni, csak pár alap dolog kerül most a porondra, mely talán kedvet fog csinálni egyeseknek, hogy közelebbről is megismerjék e remek nyelvet.

Tehát egy ***ADA*** program egy vagy több ***programegységből*** áll, ezek többnyire külön is fordíthatóak, valamit egymásba ágyazhatók. Programegység lehet: ***alprogram***, ***csomag***, ***sablon***, ***taszk*** és ***védett egység***. Ne húzzuk tovább az időt! Lássuk végre egy ***„Hello World!”*** programot ***ADA*** nyelven (*1. lista*).

Már ebből a pár sorból is feltűnhet, hogy mennyire hasonlít a szintaxis a ***PASCAL*** nyelvre. Az ***ADA*** források általánosan ***adb*** kiterjesztésűek és lehetőleg megegyezik a fájlnév a programegység nevével, ebben az esetben ***hello.adb***. Az első sor megadja, hogy mely könyvtári egységet szeretnénk felhasználni. A második sor ebben az esetben a ***programegység*** (a programunk most egy fordítási egységet tartalmaz) ***specifikációja***, ami megadja, hogy mire és hogyan lehet felhasználni az általunk írt programegységet. A ***törzs*** pedig a 3-tól 5. sorig terjedő rész. Ez tartalmazza a programunk implementációs részét. A törzs is több részre bomlik: ***deklarációs rész***, ***utasítássorozat***, ***kivételkezelés*** (ez persze opcionális). Lássunk egy másik példát (*2. lista*).

1. lista Hellló világ! (hello.adb)

```
With Text_IO;
Procedure Hello is
Begin

Text_IO.Put_Line("Hello");
End Hello;
```

2. lista Adjunk össze két számot

```
With Integer_Text_IO;

Procedure Osszead is
       A,B:Integer;
Begin

Ada.Integer_Text_IO.Get(A);

Ada.Integer_Text_IO.Get(B);

Ada.Integer_Text_IO.Pu(A+B);
End Osszead;
```

3. lista „for" ciklus az ADA-ban

```
With Integer_Text_IO;
Procedure Forciklus is
Begin
        For i in 1..10 loop

Ada.Integer_Text_IO.Put(i);
        End loop;
End Forciklus;
```

4. lista „while" ciklus az ADA-ban

```
...
I:Integer;
Begin
        While I<=10 loop

Ada.Integer_Text_IO.Put(I);
                I := I + 1;
        End loop;
...
```

5. Lista „exit when" konstrukció az ADA-ban

```
...
        loop

Ada.Integer_Text_IO.Get(N);
                exit when N=1;
        end loop;
...
```

Esetünkben a deklarációs rész az *„A,B:Integer;"* sor. Ebben a részben szerepelhetnek, változók, konstansok, típusok és más programegységek is, hiszen említettük, hogy a programegységek egymásba ágyazhatók.
A változókhoz lehet kezdőértéket is rendelni: `A:Integer := 3;`. Néhány alapvető típus az *ADA*-ban: `Integer`, `Natural`, `Positive`, `Float`, `Boolean`, `Character`, `String`. Ezek beépített típusok.
A következőkben áttekintjük az alapvető programozási konstrukciókat. Például igen érdekesen alakult az *ADA*-ban a ciklusok helyzete.
Az egyszerű *„for"* ciklusok ügyesen úgy lettek kialakítva, hogy a ciklusváltozó a ciklusmagon belül nem változtatható meg. Gondoljunk csak bele hányszor estünk már abba a hibába, hogy egy cikluson belül megváltoztattuk a ciklusváltozó értékét és persze elszállt a programunk.
Az *ADA* ezt egyszerűen küszöböli ki (*3. lista*).
Amint észrevettük az *„i"* ciklusváltozót külön nem is deklaráltuk. A változó hatóköre csak cikluson belülre terjed ki. Értékére tehát csak hivatkozni lehet, átírni azt nem tudjuk.
Ha például visszafelé szeretnénk az [1,10] intervallumon *„száguldani"*, akkor az *„in"* szócska után csak be kell szúrnunk a *„reverse"* szót.
Lássunk példát *„while"* ciklusra is: *4. lista*.
Az eredményünk az előző példával megegyező. Egyszerűen írhatunk végtelen ciklusokat is, oly módon, hogy elhagyjuk a ***while <feltétel>*** részt, azaz csak `loop` és ***end loop*** közé építjük a ciklusmagot. Mivel az *ADA* támogatja a párhuzamos rendszerek fejlesztését, ezért került bele ilyen egyszerű módon a végtelenített ciklus. Az ilyen ciklusoknak a ***taszkok*** programozásában van nagy szerepe. Végtelenített ciklus kapcsán lehet megemlíteni az `exit when <feltétel>` konstrukciót. Ennek segítségével akár `while` vagy `for` ciklusból is ki lehet lépni (*5. lista*)
Az elágazások konstruálása is kicsivel szigorúbb, mint más nyelvekben. Minden *„if"* utasítás a következőképpen épül fel:

```
if <feltétel> then
utasítások
end if;
```

Az *„end if;"* mindig kötelező! Feltételek kiértékelésében optimalizálásra ad lehetőséget a következő megoldás: *„and then"*.
Ezt a konstrukciót kihasználva erőszakoljuk a rendszert feltétel lusta kiértékelésére. Például ha feltételünk a következő: *„A AND B"* akkor optimálisabb lehet *„A AND THEN B"* formában. Az előző esetben A és *B* feltétel értéke mindig kiértékelődik, a másodikban ha A hamis, akkor tudjuk, mivel ÉS műveletről lévén szó, hogy az egész feltétel hamis, tehát a programunk *B* értékét már figyelembe sem veszi.
Lássuk most hogyan is alkothatunk alprogramokat. Az alprogramokat *ADA*-ban két csoportra lehet bontani: ***eljárások***, ***függvények***. A ***függvények*** valamely értéket adnak vissza eredményül, az ***eljárások*** a program változóit változtathatják meg, tehát a program állapotterében *„turkálhatnak"*. Itt is újítást vezet be az *ADA*. Az alprogramok paraméterezhetőek ez aránylag természetesen hangzik, viszont a paraméterek specifikálásakor megadhatjuk, hogy azok ki illetve bemeneti paraméterek az alprogram törzse szempontjából. Hogy világosan lássuk miről is van szó, tanulmányozzuk a *6. listában* látható példát.
Amint látjuk az `in` illetve `out` szócskák segítségével adhatjuk meg egy alprogram számára, hogy paramétere milyen módon érhető el a törzsben. Függvények esetén mindig `in` módon, azaz csak olvasható módban kapjuk a paramétereket.
A `Fuggyveny`-ben például hibásnak számítana a `B := 3;` utasítás. Nem így az `Eljarasban` hiszen ott megadtuk, hogy a *„B"* írható és olvasható is egyben, az A viszont itt is csak olvasható.
A fenti példa jól szemlélteti az alprogramok két fajtájának szintaxisát is, valamint kiderül, hogy függvényekben a `return` kulcsszóval adhatunk vissza értéket.
Az *ADA* lehetőséget ad többször felhasználható fordítási egységek leprogramozására is. Ezek az egységek nagyszerű előnye, hogy típusokkal is paraméterezhetőek. Például írhatunk

6. lista

```
Function Fuggveny(A,B:
➥Integer) return Integer is
Begin
        Return A+B;
End Fuggveny;

Procedure Eljaras(A:in
➥Integer;B:in out Integer) is
Begin
        B := A + B;
End Eljaras;
```

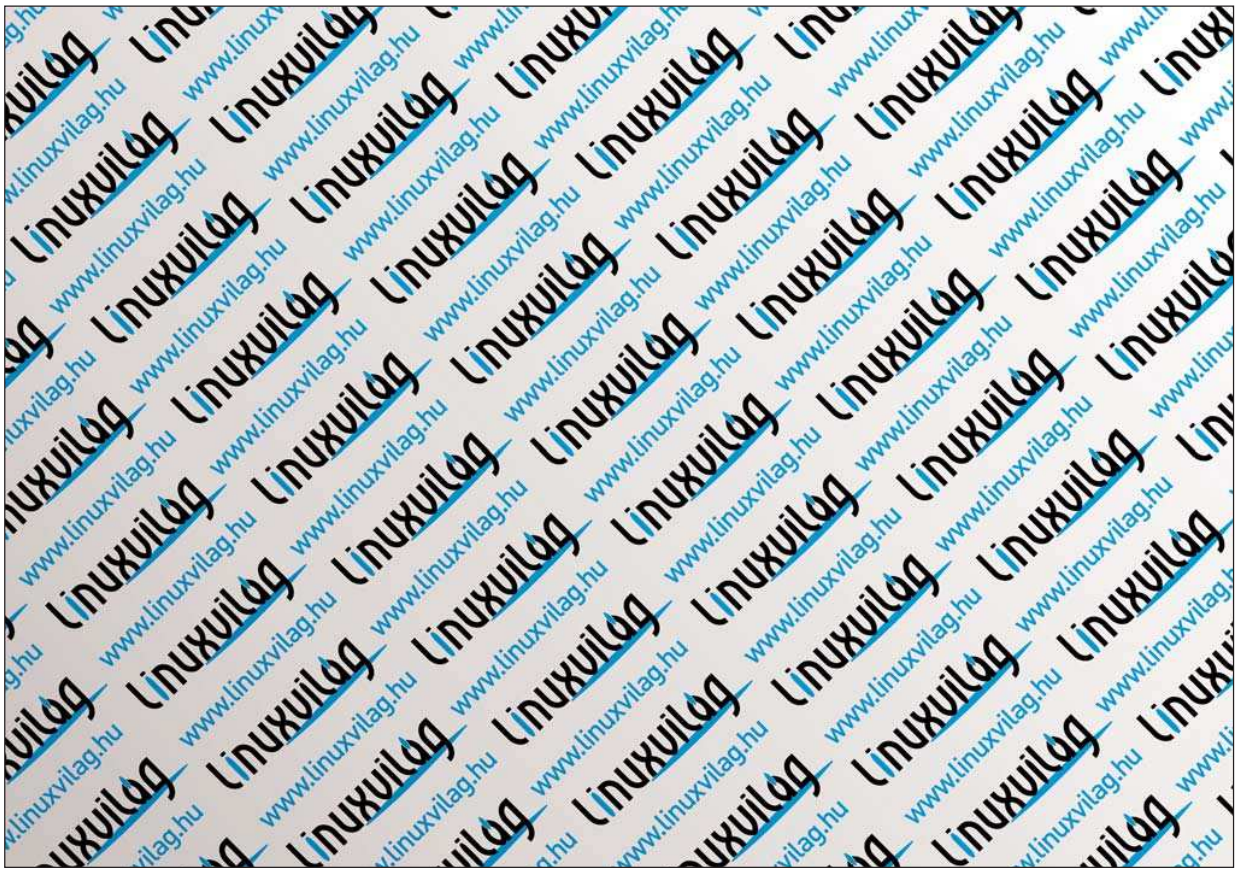
olyan csomagokat, melyek alkalmasak bármilyen típusra létrehozni egy vermet, vagy bináris fát. Ezzel a lehetőséggel a *C++*-ban is találkozhatunk. Viszont ami igen ritka és még a nagyszerű *C++* sem büszkélkedhet vele, az az, hogy az egyes csomagok nem csak típusokkal, hanem ***alprogramokkal is paraméterezhetőek!***
Az eddigiekből nem nagyon derülhetett ki, hogy miért is olyan kemény dolog *ADA*-ban programozni. A nehézségek, akkor kezdődnek mikor el kezdünk alkalmazni típusokat és azokból alkotott konstrukciókat.
A ***szintaxis*** talán nem is nehéz, hiszen aki kicsit is avatottabb a programozásban, biztos, hogy találkozott a *Pascal* nyelvvel. Ami új lehet szintaxis terén, például hogy tömbök esetén nem *[,]* jelek segítségével hivatkozunk az adott indexű elemre, hanem sima zárójelekkel. Tehát a

```
V:array of Integer(1..10);
```

10 hosszúságú, egészeket tartalmazó vektorunk 4. eleme: `V(4)`. Alapvetően azonban tényleg a *Pascal* szintaxisra épít a nyelv. Tehát ha a szintaxis nem is okoz, akkor a *„filozófia"* (nem teljesen a szemantika) okozhat fejtörést.
Az *ADA* nyelv igen nagy utat tett meg a mai napig. Fejlesztése természetesen nem ért véget. Az utolsó (*ADA95*) kiadás óta nem jelentkezett sok újdonság, de aki ellátogat a ➲ *http://www.gnat.com* címre, ott elég sűrűn láthatja az *ADA 2005* megnevezést. Valami várható a közeljövőben.
A nyelv sokoldalú, színes és nagyon hasznos dolgokat ad a programozó kezébe. Ne gondoljuk, hogy olyan könnyen megtanulhatjuk autodidakta módon, oktatóanyagokból a nyelvet, mint más esetekben!
A nyelv biztonságra törekszik, ezért sok szempontból megköti a programozó kezét. Szigorú típusossága elegendő ahhoz, hogy látszólag lehetetlenné tegye a könnyű programozhatóságot. Nagyon jó típuselméleti ismeretek szükségesek a könnyebb elsajátításhoz.
Manapság igen kevés programozó mondhatja, hogy tapasztalt *ADA* programozó. Viszont kitartással, türelemmel és sok sok idővel igen értékes tudásra tehetünk szert, ha magunkévá tesszük az *ADA* filozófiáját. Láthatjuk, hogy a szigor a precizitást és biztonságot vonja maga után és ez azt eredményezi, hogy a nyelv szélsőséges helyzetekben, hosszútávon üzemelő rendszerek fejlesztésénél, a levegőben repülőkön, a földön csillagászati kutató állomásokon, a föld alatt metrórendszerek irányításában is megállja a helyét. Mivel ember alkotta, ezért rendelkezik hibával, de a többi nyelvhez képest, az *ADA* kialakulását hatalmas nemzetközi erőfeszítések, tervezések előzték meg. A programozás terén ezért is említhetjük a precizitás, megbízhatóság szavak társaságában az *ADA* nyelvet.

Radics Péter
(peter.radics@gmail.com)

Az ELTE-n tanulok programtervező matematikus szakon. Hobbim a kosárlabda, autóvezetés, webdesign, programozás. Főleg webes alkalmazások fejlesztése érdekel. 4 éve megrögzött Linux felhasználó vagyok.



A Vim használata C programozóknak

Ahhoz, hogy kényelmesen tudjunk fejleszteni, nincs feltétlenül szükségünk egy integrált fejlesztőkörnyezetre. Az automatikus kiegészítéstől a ctags és a make meghívásáig a Vim-nek számos olyan szolgáltatása van, ami jelentősen megkönnyítheti egy C-ben fejlesztő programozó életét.

A *Vim* egy olyan rendkívül hatékony szövegszerkesztő, amely ugyan *Bill Joy* immár 30 éves vi szerkesztőjén alapul, de számos új és kellemes szolgáltatása is van. Ugyanakkor azok a szolgáltatások amelyek oly vonzóvá teszik a *Vim*-et mindazok számára, akik már ismerik, gyakorta elriasztják a kezdőket. Ennek a cikknek alapvetően az a célja, hogy segítsen a vállalkozó szelleműeknek a program megismerésében, különös tekintettel a *C* programozással kapcsolatos szolgáltatásokra.

A make parancs, és a fordít-tesztelszerkeszt ciklus

A programozók rendszerint úgy dolgoznak, hogy lefordítják a félkész anyagot, kipróbálják, majd addig szerkesztgetik újra és újra, amíg a kipróbálás során képes nem lesz azt a műveletsort végrehajtani, amire tervezték. Nyilván minden olyan eljárás, ami csökkenti az ezzel a ciklussal kapcsolatos műveletek számát, megkönnyíti a programozó életét. A *Vim* éppen ezt teszi, hiszen magából a szerkesztőből hívhatjuk meg a `make` parancsot, illetve futtathatjuk a kész kódot. Ha az aktuális könyvtárba van *Makefile*, elegendő a szerkesztőben kiadni a `:make` parancsot, és máris fordul a program.

Persze könyvtárat is válthatunk, ha épp erre van szükség, mégpedig a `:cd` parancs segítségével. Ha pedig netán elfelejtettük volna, hol is vagyunk éppen, adjuk ki a `:pwd` parancsot, és máris kiderül. Aki *FreeBSD*-t használ, és a `make` helyett a `gmake` programot szeretné használni, annak sincs túl sok extra teendője. Elegendő kiadnia a `:set makeprg=gmake` parancsot. Tegyük fel mármost, hogy paramétereket is szeretnénk átadni a `make`-nek. Mondjuk a fordításhoz e *gcc 2.96*-os változatát szeretnénk használni, nem az alapértelmezettet. Ezt így érhetjük el:

```
:set makeprg=gmake\ \CC=gcc296
```

Most jön a munka dandárja. Meg kell keresnünk az esetleges hibákat, meg kell találnunk a forrásban a nekik megfelelő kódsort, és kijavítani őket. Ha sorszámoztatni szeretnénk a kódot, adjuk ki a `:se nu` parancsot. A sorszámozást kikapcsolni a `:se nonu` paranccsal lehet.

Amikor elkezdjük fordítani a programot, de az nem sikerül, a *Vim* automatikusan az első problémás sorra ugrik. Ha a következő hibához akarunk ugrani, használjuk a `:cn` parancsot. Hasonlóan a `:cfirst` visszaugrik az első hibás sorhoz, míg a `:clast` az utolsóhoz. Ha kijavítottuk az összes hibát, ismét megpróbálkozhatunk a fordítással. A hibalista ismételt megtekintéséhez a `:clist` parancsot kell használnunk. Aki írt már programot, tudhatja, hogy mindez mennyire kényelmes. Ha a hiba javítása közben bele akarunk olvasni egy másik forrásfájlba, mondjuk a *foo.c*-be, adjuk ki az `:e foo.c` parancsot.

A *Vim*-ben számos olyan gyorsítási lehetőség van, amelyekkel elkerülhetjük a sok gépelést. Az egyik közülük az `:e #` parancs, ami az előző fájlra ugrik vissza. Ha látni szeretnénk az össze pillanatnyilag megnyitott fájl listáját, adjuk ki az `:ls` vagy a `:buffers` parancsot.

Ha úgy gondoljuk, hogy már zavaróan sok fájlt tartunk nyitva, és egyeseket be szeretnénk zárni, először szintén az `:ls` parancsot kell használnunk, ami valami ilyesmit jelenít meg:

```
2 #    "newcachain.c"
➥line 5
3 %a   "cachain.c"
➥line 1
```

Ha mondjuk a *newcachain.c* fájl be akarjuk zárni, akkor ezt a `:bd 2` vagy a `:bd newcachain.c` paranccsal tehetjük meg.

Ha olvasunk egy hosszabb kódot, gyakorta előfordul, hogy egy vagy több függvényt szeretnénk gyorsan átugrani, mert azokban nincs semmi pillanatnyilag érdekes. Ezt parancsmódban a `]]` billentyűkombinációval tehetjük meg. Ha visszafelé haladva akarjuk ugyanezt megtenni, használjuk a `[[` parancsot.

Használhatunk könyvjelzőket is bizonyos helyek bejelölésére. Címkeként használhatunk bármilyen kisbetűt. Ha mondjuk meg akarjuk jelölni a program 256-ik sorát, és ez a pontot „b”-nek szeretnénk hívni, álljunk a kurzorral a megfelelő sorra, majd adjuk ki parancsmódban a `mb` parancsot. Ügyeljünk rá, hogy a *Vim* soha nem igazolja vissza a parancsok végrehajtását, csak végrehajtja őket. Ha a könyvjelzők között ugrálva az előzőre szeretnénk visszaállni,

használjuk a `''` (két egyszeres idézőjel) parancsot. A `'a` parancs az a-val jelzett pontra ugrik, és így tovább.
Néha, de különösen ***Makefile***-ok szerkesztése közben jó ha látjuk, hogy egy üres hely szóközökből vagy tabulátorokból áll. Ilyenkor használjuk a `:se list` parancsot, mire a tabulátorok kék `^I` jelekként fognak megjelenni. Van amúgy egy másik módszer is: a `/\t` sárgával jeleníti meg a tabulátorokat.
A programozók gyakran végeznek az egész forrásban kereséseket, illetve globális szöveghelyettesítést. A ***Vim*** ehhez is kiváló támogatást nyújt.
A kereséshez egyszerűen adjuk ki a `/` parancsot, és a program máris a keresett szóra ugrik. Aki jobban szereti az ***emacs*** által használt inkrementális keresést, az adja ki a `:se incsearch` parancsot a keresés megkezdése előtt. Ezt a szolgáltatást később a `:se nois` paranccsal lehet kikapcsolni.
A keresés és csere szintén egyike a ***Vim*** legnagyszerűbb szolgáltatásainak. Ezt a műveletet végrehajthatjuk csupán a kód egy részén, amit a `v` paranccsal jelöltünk ki, megadhatunk neki kezdő és záró sorszámot, sőt kijelölhetünk a szövegben egy tetszőleges téglalap alakú területet is a ***Ctrl-V*** paranccsal.
Ha például az a feladat, hogy a 24. és 56. sorok között az összes `foo`-t cseréljük ki `bar`-ra, a következőt kell tennünk. Először is jelöljük ki a kérdéses területet a `:24,56` paranccsal. (A kijelölt területhez a két határoló sor is hozzátartozik!) Magát a cserét a `s/foo/bar` paranccsal végezhetjük el.
Ügyeljünk azonban rá, hogy a parancs ebben a formájában minden sorban csak a keresett szöveg egy előfordulását cseréli ki. A globális, minden előfordulást érintő cseréhez a `s/foo/bar/g` parancsot kell használnunk. Ha a cserét csak a keresett szövegrész bizonyos előfordulásainál akarjuk elvégezni, alkalmazhatjuk a `s/foo/bar/gc` parancsot is, amely minden egyes csere előtt rákérdez, hogy végrehajthatja-e a műveletet.
Előfordulhat, hogy a keresett karakterlánc része más kulcsszavaknak is. Tegyük fel például, hogy egy `in` nevű változó nevét akarjuk lecserélni valami másra. Ilyenkor nyilván nem szeretnénk, ha a csere a minta nevű változó „megfelelő” részét is érintené. A megoldás ilyenkor az, ha a keresést egész szóra korlátozzuk a következő módon : `/\<in\>/`.
A leggyakoribb az, amikor egy dolog valamennyi előfordulását ki szeretnénk cserélni egy adott fájlon belül. Ezt az `:1,$s/foo/bar/g` vagy a `:%s/foo/bar/g` parancsok valamelyikével tehetjük meg. Ha ugyanezt a műveletet valamennyi megnyitott fájlon akarjuk elvégezni, a `:bufdo %s/foo/bar/g` formát kell használnunk.
A keresés egy másik módja, ha a kurzorral ráállunk a kulcsszó egy előfordulására, majd parancsmódban beütjük a `*` parancsot. Erre a ***Vim*** a kérdéses szó minden előfordulását kiemeli. Ha egy keresést az adott kurzorpozíciótól visszafelé haladva akarunk elvégezni, használjuk a `?` parancsot a `/` helyett.
Ha egy keresés lefutott, a ***Vim*** megjegyzi annak eredményét. Ha tehát újra ugyanarra a szóra akarunk rákeresni, elég beüti a `/` vagy a `?` parancsot, a keresendő szövegre már nincs szükség.
A keresés mellékhatásaként a találatok kiemelve maradnak a képernyőn, ami szerkesztés közben elég idegesítő tud lenni, különösen, ha nincs is már rá szükség. A kiemelések eltüntetéséhez adjuk ki a `:se nohlsearch` vagy a `:nohl` parancsot.
Ha a kettőspont után egy ***Vim*** parancsot gépelünk, bármikor használhatjuk a ***Tab*** billentyűt, mire a program kiegészíti azt. Ha túl kevés az információ a kiegészítéshez, a ***Tab***-ot többször is megnyomhatjuk, mire újabb és újabb javaslatokat kapunk egészen addig, amíg meg nem találjuk az igazit.

A Vim és az Exuberant ctags

Az ***Exuberant ctags*** (lásd az elektronikus forrásokat) egy olyan külső program, amely a ***Vim*** számára a forráskódban való tájékozódást segítő tageket (jelzőket) képes előállítani.
Ha programunk teljes forráskódja egyetlen könyvtárban található, egyszerűen lépjünk be ide, és adjuk ki a következő parancsot:

```
$ ctags .
```

Ez létrehoz egy ***tags*** nevű fájlt, amelyben nevének megfelelően ezek a bizonyos tag-ek találhatók. Ezt az információt a ***Vim*** is értelmezi, és segítségével villámgyorsan képes odaugrani a különböző függvények, felsorolt típusok meghatározásához, az előfeldolgozónak átadott konstansok és makrók definíciójához (`#define`) és számos más a ***C*** nyelvre jellemző szerkezethez.
Ha a forráskód több könyvtárban szétszórva található, a ***ctags*** ezek helyét is bele kell foglalja az általa előállított információcsomagba. Ilyenkor lépjünk be a forráskód legfelsőbb szintű könyvtárába, és adjuk ki a

```
$ ctags -R .
```

parancsot, majd ellenőrizzük, hogy valóban létrejött-e a ***tags*** fájl. Ez utóbbit egyébként magával a ***Vim***-mel is megnyithatjuk.
Nos, most hogy már tag-jeink is vannak, lássuk, miként használhatjuk őket a forráskódban való mozgás során. A ***ctags*** segítségével való tájékozódás egyike a legnagyszerűbb dolgoknak, amik egy programozó rendelkezésére állnak.
Aki megtanulja használni ezt a szolgáltatást, olyan könnyen és gyorsan tud navigálni, hogy utóbb el se tudja képzelni, hogy tudott eddig meglenni nélküle.
Ha tehát létrehoztuk a ***tags*** fájlt, nyissunk meg próbaképpen egy tetszőleges forráskódot. Az egyetlen eltérés az eddigiektől az, hogy ha a teljes forrás több könyvtárban található, a kérdéses fájl pedig nem a gyökérkönyvtárban van, akkor is innen indulva kell megnyitnunk. Tegyük fel például, hogy forráskódunk könyvtárszerkezete a következőképpen néz ki:

```
common
  |
  ----> gui --> wxpython
  |       |
  |       ------>Tk
  |
  ----> backend -->
networking
  include
  user
```

Ha a ***common/backend/networking*** könyvtárban található ***tcp.c*** fájlt szeretnénk szerkeszteni, akkor a ***Vim*** segítségével most a következőképpen kell megnyitnunk:

```
$ vim common/backend/
➥networking/tcp.c
```

Régen, a *ctags* nélkül a következőt tettük volna:

```
$ cd common/backend/networking
$ vim tcp.c
```

Mivel a *tags* fájl a hierarchiában a *common* könyvtár fölött helyezkedik el, a *Vim* automatikusan tudni fogja ennek a helyét.
Van ugyanakkor egy másik módszer is. Megtehetjük, hogy a fent említett második (hagyományos) módszerrel nyitjuk meg a kérdéses fájl, majd már a *Vim*-ben kiadjuk a következő parancsot:

```
:se tags=../../../tags
```

Szerintem amúgy az első módszer könnyebben használható. Ha bármilyen módon megnyitottuk a fájlunkat, a függvények definíciói között a *Ctrl-]* billentyűkombinációval mozoghatunk. Ha oda akarunk ugrani bármilyen nyelvi szerkezet (függvény, konstans, makró vagy bármi más) definíciójára, vigyük fölé a kurzort, majd nyomjuk meg a *Ctrl-]* billentyűket. Röviden tehát egy függvény definíciójának megkereséséhez előbb meg kell találnunk egy pontot, ahol meghívjuk azt. Az már teljesen lényegtelen, hogy a függvény definíciója az adott fájlban, vagy egy egészen más könyvtárban és fájlban van. Tegyük fel például, hogy a *tcp.c* fájlban meghívunk egy `drawscreen()` nevű függvényt.
A tagek segítségével a *Vim* még akkor is megtalálja a függvény meghatározását, ha az azt tartalmazó fájl történetesen a *common/gui*-ban van.
Ha a keresés után vissza akarunk ugrani oda, ahonnan jöttünk, nyomjuk meg a *Ctrl-T* billentyűkombinációt. Újabb kereséshez nyomjuk meg újra a *Ctrl-]* billentyűket, ami után persze megint a *Ctrl-T*-vel térhetünk majd vissza.
Ha a keresett dolog nevének csak egy részét ismerjük, akkor a következőképpen is eljárhatunk:

```
:ta /function
```

Ha több ilyen rész is van a forráskódban, ez a parancs az első elforduláshoz ugrik. A következőre a `:tn` paranccsal válthatunk.
Ha egy függvénynek több definíciója is van, és választani szeretnénk közülük, megnyomhatjuk a *G Ctrl-]* billentyűket, vagy használhatjuk a `:tselect <tagname>` parancsot. Ezzel a módszerrel anélkül mozoghatunk a forrásban az egyes függvények között hogy tudnunk kellene, melyik pontosan hol található.

A Vim és a Cscope

A *cscope* egy másik hatékony navigációs eszköz. Íme a program által felkínált menü:

```
Find this C symbol:
Find this global definition:
Find functions called by this
➥function:
Find functions calling this
➥function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this
➥file:
```

A programmal a *Vim* is képes együttműködni, így könnyedén ötvözhetjük az egyik hatékonyságát a másik kényelmével. A kapcsolat megteremtéséhez az egyetlen dolog, amit tennünk kell a

```
:cs add cscope.out.
```

parancs kiadása.
Akárcsak az előbb tárgyalt *ctags*, a *cscope* is egy indexfájlt hoz létre *cscope.out* néven. Ezt a következő paranccsal állíthatjuk elő:

```
$ cscope -Rbq
```

A szabályok is ugyanazok, mint a *ctags* esetében, vagyis ha a forráskód több könyvtárban szétszórtan helyezkedik el, akkor a fenti parancsot a könyvtárszerkezet gyökerében kell kiadnunk.
A történet folytatása is hasonló.
Ha egy a könyvtárszerkezet belsejében található fájlt nyitunk meg, akkor is a gyökérből kell indulnunk, majd a *Vim*-ben ki kell adni a `:cs add` *cscope.out* parancsot a kapcsolat felépítéséhez. A létező *cscope* kapcsolatokat a `:cs show` paranccsal listázhatjuk.
Hogy mi mindenre kereshetünk a *Vim*-en belülről, azt a `:cs <CR>` paranccsal nézhetjük meg. Ha például bele akarunk nézni egy forrásba, vagy egy fejlécállományba, egyszerűen gépeljük be a `:cs f f stdio.h` vagy a `:cs f f foo.c` parancsot.
Ha a *foo.c*-ben meghívott függvényekre akarunk rákeresi, használjuk a `:cs f d foo.c` parancsot. Ez megjeleníti az összes innen hívott függvény nevét. Ha fordított keresést szeretnénk végezni, vagyis azoknak a függvényeknek a nevére vagyunk kíváncsiak, amelyek a *foo.c*-re hivatkoznak, a `:cs f c foo.c` utasítást használjuk. Kereshetünk egrep minta alapján is a következőképpen: :cs f e varName és így tovább. A lehetőségeket a magában kiadott :cs parancs segítségével listázhatjuk.
Ha sikeresen beüzemeltük a *ctags* és a *cscope* rendszert is, kombinálhatjuk is őket, például így: `:cstag /foo`. Ez a parancs megkeresi az összes olyan függvényt, felsorolt típust vagy bármi egyebet, ami tartalmazza a `foo` karakterláncot.

A Vim és a szintaktikai kiemelés

Ha olyan szolgáltatása a *Vim*-nek, ami fölébe helyezi bármely más szerkesztőnek vagy integrált fejlesztőkörnyezetnek (*IDE*), nos akkor ez a szintaktikai kiemelés. A *Vim* színező képessége egyszerűen élvezetté teszi a vele való munkát. A szöveg szintaktika szerinti kiszínezése természetesen nem csak az életünket teszi színesebbé, hanem jelentsen megkönnyíti az elgépelések és egyéb hibák megtalálását akár a fordítás előtt. A színezés változása révén például azonnal kiszúrhatjuk a szinte leggyakoribb programozási hibát, a zárójelek hiányát. Azonnal látni lehet, ha nem zártunk be egy idézőjelet, vagy ha egy megjegyzés végéről hiányzik a `*/` jel. Szintén rögtön kibukik, ha netán megjegyzéseket akarunk egymásba ágyazni. Egy *C* programozó számára mindez nagyon hasznos, sőt néha életmentő lehet.
A *Vim* szintaktika szerinti színezőképességének bekapcsolásához általában semmit nem kell tennünk. Ha azonban valamiért úgy települne fel a program, hogy ez a szolgáltatása nincs alapértelmezésként bekapcsolva, akkor a `:sy on` paranccsal gondoskodhatunk a helyes működésről.

Akárcsak bármely más parancsot, természetesen ezt is felvehetjük a *~/.vimrc* fájlunkba.
Ha a szöveg még mindig nem színes, akkor a terminál beállításaival van valami gond. Ilyenkor azért még próbáljuk ki a `:se filetype` és a `:sy eanble` parancsokat.
Tegyük fel, hogy sikerült elővarázsolnunk a színeket, de valamelyik közülük nincs ínyünkre. A kék például elég rosszul látszik sötét háttér előtt, márpedig a *C* forráskódok megjegyzései alapértelmezésként ezzel a színnel jelennek meg. Ilyenkor több lehetőségünk is van. Használhatjuk például a `:se background=dark` parancsot, amivel a program tudtára adjuk, hogy milyen színű a háttér. De az is megoldás, ha kikapcsoljuk a megjegyzések színezését a `:highlight clear comment` paranccsal.
Aztán ha ez még mindig nem elég, akkor meg is változtathatjuk az alapértelmezett színeket. Ehhez először adjuk ki a `:syntax` parancsot, ami megjeleníti az összes szintaktikai elemet. Ezután adjuk meg a megváltoztatni kívánt csoport nevét. Ha például azt akarjuk, hogy a karakterláncok ragyogó fehérrel jelenjenek meg, ami fekete háttér előtt nyilván kiválóan olvasható, egyszerűen gépeljük be a következő parancsot:

```
:highlight String ctermfg=white
```

Aki *gvim*-et használ, annak viszont a következőképpen kell eljárnia:

```
:highlight String guifg=white
```

Természetesen bármely más szintaktikai csoport színét megváltoztathatjuk. A leggyakoribb elemek a következők: `Statement`, `Label`, `Conditional`, `Repeat`, `Todo` és `Special`. A színen kívül megváltoztathatjuk a kiemelés módját is. Beállíthatjuk például, hogy egy elemtípus legyen alahúzva, vagy legyen félkövér. Ha például a `NOTE`, `FIXME`, `TODO` vagy `XXX` elemeket alahúzva szeretnénk megjeleníteni, a következőt kell tennünk:

```
:highlight Todo cterm=underline
```

Így érhetjük el, hogy a színezéssel egyidejűleg az adott elem félkövér is legyen:

```
:highlight Repeat
➥ctermfg=yellow cterm=bold
```

Ezekkel a parancsokkal létrehozhatunk akár egy teljes kiemelési szabályrendszert, amit természetesen megint a *~/.vimrc* fájlban tárolhatunk.
Így valahányszor elindítjuk a *Vim*-et, a beállításokat végző parancsok rögtön az elején automatikusan lefutnak, és munka közben élvezhetjük kedvenc színeinket.

Változónevek automatikus kiegészítése

A *Vim* arra is képes, hogy automatikusan kiegészítse a változók neveit. Gépelés közben a ***Ctrl-N*** vagy a ***Ctrl-P*** segítségével válthatunk beszúrás üzemmódba. Fontos megjegyezni, hogy a kiegészítés csak ebben az üzemmódban működik. Minden más, eddig említett parancsot parancs üzemmódban kell használnunk. Ha több lehetséges kiegészítés is van, akkor a ***Ctrl-N*** többszöri megnyomásával léptethetjük a lehetőségeket.
Ennek a szolgáltatásnak az a legfőbb haszna, hogy segítségével könnyebb elkerülni az elgépeléseket, hiszen a szöveg jelentős részét nem mi visszük be kézzel, hanem a program „emlékezetből". A névkiegészítés akkor működik a leghatékonyabban, ha engedélyeztük a *Vim*-nek a *ctags* fájl használatát.

A forráskód formázása

A *Vim* elég jól érti a *C* nyelvet ahhoz, hogy automatikusan kezelni tudja a beljebbezést. Alapértelmezésként ehhez tabulátorokat használ, ami egyes programozókból ellenérzéseket válthat ki. Ha a teljes forráskódból el száműzni szeretnénk a tabulátorokat, adjuk ki a következő parancsot:

```
:set expandtab
:retab
```

Ez valamennyi tabulátort szóközökké alakít úgy, hogy a formázás változatlan maradjon. Miközben gépeljük a kódot, a *Vim* automatikusan formázza azt. Ez nyilván megkönnyíti az egymáshoz tartozó zárójelek felderítését. Ehhez amúgy használhatjuk a % parancsot is parancs üzemmódban. Álljunk rá a megfelelő zárójelre – legyen az kerek, szögletes vagy kapcsos – majd nyomjuk meg a % billentyűt. Erre a *Vim* odaugrik a megfelelő nyitó vagy záró zárójelre. Ugyanez a módszer működik a megjegyzésekre, valamint az `#if`, `#ifdef` és `#endif` szerkezetekre is.
Ha befejeztük a kód begépelését, és egyetlen paranccsal akarjuk megformázni, váltsunk parancsmódba, és gépeljük be a következőt: `gg=G`. Ezután szükség szerint átalakíthatjuk a tabulátorokat szóközökké a fentebb említett módszerrel. A megjegyzések formázását a gq parancs végzi. Ha csak egy kódrészletet akarunk beljebb tolni, jelöljük ki, és használjuk az = parancsot.
Végül ha valakit netán kifejezetten bosszantana a *Vim* automatikus formázása, ki is kapcsolhatja az a `:se nocindent` paranccsal. Érdemes ugyanakkor megemlíteni, hogy a formázásra más lehetőségek is vannak, amelyekről a `:help indent.txt` paranccsal olvashatunk.

A súgó

A *Vim* igen terjedelmes dokumentációval rendelkezik, amit a `:help` paranccsal böngészhetünk. Ha egy konkrét téma leírására akarunk ugrani, álljunk a megfelelő türkiz színű szövegrészre, majd üssük le a ***Ctrl-]*** billentyűkombinációt. A *Vim* súgója egyébként ugyanazt a navigációs mechanizmust használja, mint amit a tag-eknél láttunk.

Linux Journal 2005. 140 szám

Girish Venkatachalam
Szeret nyílt forrású operációs rendszerekkel, például OpenBSD-vel, FreeBSD-vel no és Debian GNU/Linuxszal játszadozni. Amikor nem a hackeléssel van elfoglalva, szeret biciklizni is. A girish1729@gmail.com érhető el.

KAPCSOLÓDÓ CÍMEK

A cikkhez tartozó elektronikus források a következő helyen találhatók:
➲ www.linuxjournal.com/article/8455

Szelíd Tűzróka – avagy mit tud a Firefox

Több mint 90 millió letöltés. Mi lehet az oka a Firefox sikerének? Vajon csak egy terjedő hóbortról van szó, vagy a Firefox tényleg ennyire jó, és megéri Tűzrókát költöztetni a számítógépünkre.

Megjelenése óta töretlen a *Firefox* népszerűsége, és hónapról hónapra egyre többen használják böngészésre a *Mozilla Alapítvány* legújabb üdvösköjét. Gondolom az olvasók közül is már szinte mindenki találkozott a programmal, abban is biztos vagyok, hogy szép számmal akadnak olyanok is akik már csak ezzel a programmal böngészik a világhálót.

A *Firefox* egyik legjobb tulajdonsága, hogy egyszerű a felülete, a felhasználó az első pillanatban átlátja a mi található az ablakban. Vessük csak össze mondjuk az *Opera* böngészővel, ahol nekem percekig szokott tartani, hogy az ablakban látható felesleges oldal sávokat, gombokat eltüntessem a helyükről. Annak ellenére, hogy a *Firefox* felülete végtelenül leegyszerűsített, és maga a program is kicsi rengeteg lehetőség lakozik a szoftverben. A maximális mértékben lehetőségünk van a böngészőt saját igényeink szerint alakítani. Azon felül, hogy a böngésző megjelenését tudjuk változtatni, lehetőségünk van biztonságosabbá, és sokkal gyorsabbá tenni a böngészést, sőt akár az időjárás előrejelzés is állandóan a szemünk előtt lehet.

A *Szerkesztés* menü *Beállítások* menüpontjában tudjuk elvégezni a szükséges beállításokat. Ezeket most nem részletezném, hiszen a kezelőfelület végtelenül egyszerű, és logikusan felépített. Innen tudjuk a böngésző alapvető tulajdonságait beállítani.

1. ábra Új téma választása

A róka új ruhája

Először kezdjük a testreszabást a kinézet megváltoztatásával. Az *Eszközök* menü, *Témák* pontjában érjük el az aktuálisan telepített témákat. Alapbeállításként itt csak egy téma szerepel, de az ablak alján lévő *További témák letöltése* linkre kattintva a *Mozilla Update* honlapján találjuk magunkat. Itt rögtön megjelennek a legkedveltebb témák, de a bal oldalon található menüből kategóriánként válogathatunk a témák közül. Egy téma letöltése előtt mindenképpen ellenőrizzük, hogy az általunk használt verzióval kompatibilis-e. Saját *Firefox* verziónk számát a *Súgó* menü A *Mozilla Firefox névjegye* menüpontban tekinthetjük meg.

Az oldalról sok témát letölthetünk, majd azt amelyik a legjobban tetszik nekünk, válasszuk ki, és kattintsunk a *Téma használata* gombra. Az új téma érvénybelépéséhez újra kell indítanunk a böngészőt. A letöltött témákat természetesen el is távolíthatjuk, ha mégsem tetszenének, vagy már nincs rájuk szükségünk.

Harc a reklámok áradata ellen

Egy megfelelő megjelenés kiválasztása nagyon imponáló, de mint tudjuk a csomagolás nem minden. Ha már elegünk van abból, hogy egy-egy hírportál felületének szinte a fele ugráló, villogó képekkel van tele, akkor a *Firefox* a legjobb választásnak tűnik.

A *Firefox* lehetőséget ad különböző hasznos kiterjesztések letöltésére.

A reklámok áradata elleni harc két legjobb fegyvere az *Adblock*, és a *Flashblock*. Tapasztalataim szerint a kettőt együtt használva már nagyon jó hatásfokkal tudjuk kiszűrni a zavaró reklámokat.

A reklámokkal tulajdonképpen több baj is van. Az egyik az, hogy engem például zavar, hogy ott villog a szemem sarkában, továbbá elveszi az értékes helyet egyéb tartalomtól, így gyakrabban lapozhatok az oldalon.

A következő probléma ma már nem annyira égető, de reklámokkal együtt az egész oldal lassabban töltődik le, és jelenik is meg. Továbbá felesleges processzor kapacitást emészt fel a négy vagy öt villogó *Flash* objektum, olyannyira, hogy akár magunk is könnyen meggyőződhetünk róla egy rendszerfigyelő társaságában.

A felesleges processzor használat egy hordozható gép esetén pedig csökkenti a rendelkezésre állási időt.

Azoknak viszont akik még mindig modemmel, vagy *GPRS*-en keresztül csatlakoznak a világhálóra viszont mindenképpen kritikus szempont, a reklámok, és mindenféle *Flash* animációk blokkolása.

Az *Eszközök*, *Kiterjesztések* menüpontjában látjuk, hogy eddig milyen kiterjesztéseket telepítettünk fel.

A kiterjesztések telepítése során mindig csak megbízható forrásból származó programokat töltsünk le és telepítsünk.

2. ábra Egy ismert hírportál felülete (a reklámok helye pirossal megjelölve)

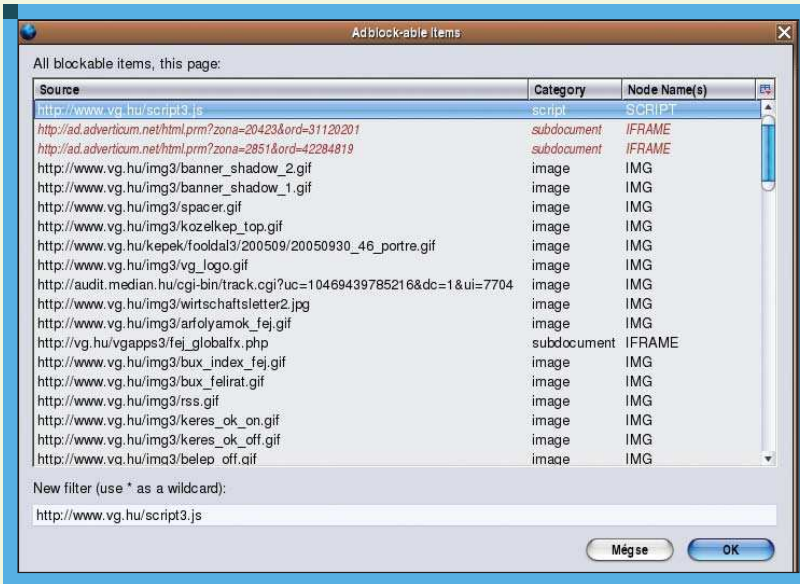
Amennyiben nem a Mozilla honlapjáról töltünk le kiterjesztést, akkor előbb engedélyeznünk kell az adott helyről való kiterjesztések letöltését.

Flashblock

Gyorsan le is szögezem, hogy nem minden *Flash* objektum reklám. Hiszen alapvetően a *Flash* szebb web oldalak létrehozására szolgál. Erre jó a *Flashblock* kiterjesztés, amely lehetővé teszi számunkra, hogy eldöntsük, hogy az adott *Flash* objektumot letöltse és megjelenítse-e a *Firefox*. A *Flashblock* a *Flash* animációk helyett egy „F” betűt jelenít meg, mely ha fölé visszük a kurzort, akkor megváltozik. és a magnókon, és cd lejátszókon használatos „lejátszás” gomb jelenik meg helyette. Erre kattintva van lehetőségünk az adott *Flash* engedélyezésére, és lejátszására. A kiterjesztés lehetőséget biztosít arra, hogy megadjunk olyan webhelyeket is, melyekről mindig minden *Flash* animációt elfogadunk.

Adblock

Az *Adblock* az előzőnél komplikáltabb eszköz a reklámok elleni harc terén. Ennek használatával nem csak *Flash* objektumokat, hanem képeket, szkripteket, sőt még kereteket is lehetőségünk van letiltani bizonyos oldalakon. Használata nem igényel különösebb szaktudást, mert a státuszbáron (a böngésző legalsó eleme) található *Adblock* gombra kattintva felkínálja nekünk a az adott oldalon blokkolható összes objektumot. Ebből a listából könnyen és gyorsan kiválaszthatjuk, azokat az elemeket, amikor blokkolni szeretnénk. Szerencsére mivel a web oldalakat logikusan építik fel, ezért a következőkhöz hasonló elemeket kell keresni: *ds.honlap.hu/...* ;www.valami.hu/ad?... *adserver.valami.com/...*

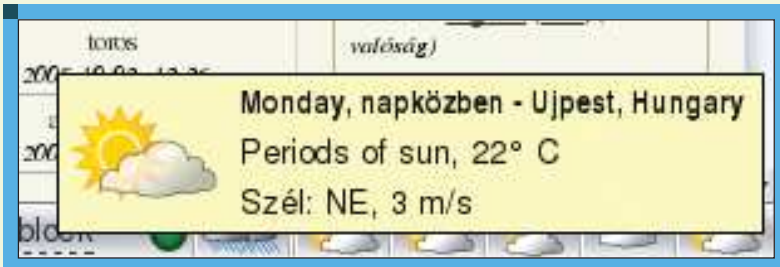


3. ábra Az Adblock segítségével blokkolható elemek listája

Szerencsére a kiegészítés támogatja a * használatát, mint az informatikában megszokott ismeretlen hosszúságú karaktersorozat, így nem kell minden egyes új hirdetést külön letiltani, egyszerűen csak nézzük meg az elérési út azon elemeit, melyek azonosak, és egy kellő hosszúságú karaktersorozatot cseréljünk le csillaggal, majd az OK gombra kattintva mentsük el a változást.

Például: www.valami.hu/ad* vagy ads.kiszolgalo.hu/* vagy banners.kiszolgalo.hu/*

Természetesen később vissza is



4. ábra A Forecastfox a közeljövő várható helyi időjárását mutatja

5. ábra A kis radar ikonja „mögött" Európa radarképe bújik meg

vonhatjuk a beállításokat, illetve időlegesen kikapcsolhatjuk az ***Adblock***-ot, ha éppen reklámokat nézni támad kedvünk.

Napszemüveg, vagy esernyő?

Aki azt hiszi, hogy a ***Firefox*** kiterjesztései csak a böngészéssel, és a weboldalak használatával kapcsolatosak, az téved. Itt van például a ***Forecastfox*** nevű kiterjesztés, mely nem más, mint egy időjárás előrejelző. Letöltés után természetesen most is újra kell indítani a ***Firefoxot***, majd meg kell adni a helyet, ahol tartózkodunk, természetesen olyan formátumban amit a rendszer megért. Ehhez használható egy kereső is, ahová ékezetek nélkül beírva a tartózkodási helyünket rögtön megkapjuk a hozzá tartozó kódsorozatot.

A kódsorozat például így néz ki: '***EUR;HU;-;BUDAPEST;***' vagy '***EUR;HU;-;UJPEST;***'.

Ahogyan láttam elég sok magyar település megtalálható, így biztosan sikerül helyi időjárás előrejelzést varázsolni a ***Firefoxba***. Lehetőségünk van megadni, hogy a kapott adatokat a böngésző milyen mértékegység rendszerben jelenítse meg.

A szükséges adatokat a www.AccuWeather.com oldal szolgáltatja számunkra, és a várható és jelenlegi időjárás kis ikonok formájában jelenik meg. Amennyiben fölé visszük a kurzort, akkor sokkal részletesebb leírást kapunk az aznap várható időről, pontos hőmérséklettel, és várható szélsebességgel.
Ha a kurzort a kis radar ikon fölé visszük, akkor láthatóvá válik Európa radarképe. Ha rákattintunk a kis ikonra, akkor a ***AccuWeather*** oldala töltődik be, és ott rögtön nagyban is megtekinthetjük a radarképet.
Nekem különösen tetszett, hogy több napra előre lehet megjeleníteni a várható időjárást, illetve többféle profilt van lehetőségünk létrehozni. Így ha mondjuk utazgatunk ***Budapest*** és ***Bécs*** között, akkor csak egy kattintással máris a másik város időjárási adatait látjuk.
Az oldalon még rengeteg remek kiterjesztés található meg. Sajnos hely hiányában nem tudtam az összeset bemutatni, de megpróbáltam az általam egyik leghasznosabbnak találtakat kiemelni. Ugyanakkor megemlíteném még, hogy találhatóak a letöltések kezelését megkönnyítő kiterjesztések is. Vannak olyanok, amelyek a ***Könyvjelzők*** hatékonyabb, és rendszerezettebb kezelését valósítják meg.

Továbbá még kiemelném a ***Gmail Notifier*** nevű kiterjesztést, mely azoknak nyújt segítséget, akiknek van ***Gmail***-en e-mail címük, hiszen periodikusan lekérdezi, hogy vannak-e új, még olvasatlan levelek, és ezek számát a státuszbáron meg is jeleníti.

Az óvatos róka

Az egyik legjobban terjedő csalási forma manapság a világhálón a '***phising***', ami tulajdonképpen adathalászatot jelent. Ehhez a módszerhez nem kell feltétlenül feltörni a szolgáltató gépét, hogy a csalók érzékeny információk birtokába jussanak, hanem különböző „ötletes" módon veszik rá az adathalászok a gyanútlan felhasználót arra, hogy a szenzitív információkat tulajdonképpen önkéntesen adják ki.
Vegyünk egy példát, az én bankom internetes portálja a ***www.enbankomonline.hu***, ha valaki beregisztrálja mondjuk a ***www.komonline.hu*** webcímet, ami ugye senkinek sem fog feltűnni, és azon belül létrehoz egy másodlagos domaint 'enban' címen, akkor ennek elérése a következő lesz ***www.enban.komonline.hu***.
A csaló tükrözi a ***www.enbankomonline.hu*** kezdőlapját a saját másodlagos domainjére. Majd az áldozat címére küld egy e-mailt, hogy azonnal lépjen be az online rendszerbe, és változtasson meg ott valamilyen beállítást, az e-mail természetesen tartalmazza a hivatkozást az oldalra, hogy egyszerűbb legyen a belépés.
Ugyan a címsorban a www.enban.komonline.hu jelenik meg, de az eltérés csak egyetlen pont, mivel az oldal ugyanúgy néz ki, a gyanútlan felhasználó megpróbál belépni. Természetesen sikertelenül, de közben a jó felhasználó név, jelszó párost már a csaló ismeri is.
Ez egy kicsit kisarkított példa, de vegyük észre, hogy jelen esetben egyetlen ponton múlt az, hogy egy arra jogosulatlan személy hozzáfér bankszámlánkhoz, és onnan átutalást tud kezdeményezni.
A hosszú bevezető után, akkor nézzük is meg, hogy hogyan is tudjuk a ***NetCraft*** segítségével felvenni a harcot az adathalászok ellen. A ***NetCraft Toolbar*** egyetlen hátránya, hogy széles, így kicsit sok helyet foglal el a képernyőn.

6. ábra A www.linuxvilag.hu a NetCraft szerint nem adathalász oldal

A ***http://toolbar.netcraft.com/install*** oldalról szerezzük be a ***NetCraft*** eszköztárat, itt meg is jegyzem, hogy az eszköztár nem csak ***Firefox*** alá érhető el, hanem létezik belőle Internet Explorer böngészőhöz készült verzió is.
Az első két menüben a ***NetCrafthoz***, illetve az adott honlaphoz tartozó tevékenységeket találjuk, úgy, mint jelenthetjük a ***NetCraft*** számára ha az adott honlap adathalász gyanús, illetve például ajánlhatjuk az eszköztárat egyik ismerősünk számára.
Az adott lapra vonatkozó értékes információkat, viszont a függőleges vonaltól jobbra találjuk meg. Itt szerepel rögtön a ***RiskRating***, ami a megjelenített piros szín nagyságától függően jelenti, hogy az adott oldalon mekkora az adathalászat gyanúja.
A gyanú nagyságát a ***NetCraft*** például az alapján határozza meg, hogy ***IP*** cím vagy kiszolgáló név, illetve port szám van a címsorban, esetleg az adott tartományban már voltak adathalász oldalak, de még az oldalt kiszolgáló internet szolgáltató előélete is fontos szempont.
Az eszköztáron megtaláljuk még, hogy az adott oldal mióta létezik, nyilván egy régebbi oldalban jobban megbízhatunk, találunk egy rangsort is az oldal látogatottságáról. Továbbá annak az országnak a zászlója, ahol a kiszolgáló található, és megtudhatjuk az internet szolgáltató kilétét is.

A fürge róka

Az következő beállítási lehetőségeket az alábbi weblapon olvastam: (***http://www.freerepublic.com/focus/f-news/1299854/posts***). Az eddigieken felül is még tovább tudjuk növelni a böngészés sebességét, viszont a következő beállításokkal jobb ha óvatosak vagyunk, mert akár a böngész meghibásodását is okozhatja.
Először is a ***Firefox*** címsorába gépeljük be az ***about:config*** szöveget, majd üssünk egy ***Entert***. Keressük meg a következő bejegyzéseket az oldalon, és dupla kattintás segítségével állítsuk át őket az alábbiak szerint:

```
network.http.pipelining -> true
network.http.proxy.pipelining
➥-> true
network.http.pipelining.
➥maxrequests -> 8
```

Az utóbbi érték alap beállításként 4, és azt határozza meg, hogy egy oldal letöltésénél egyszerre hány kérést indítson a böngésző. A maximális szám ha jól tudom, akkor 8.
Továbbá ezen az oldalon bárhova az egér jobb gombjával kattintva kapunk egy helyi menüt, ebből válasszuk ki az ***Új -> Egész*** menüpontot, és írjuk be névnek az ***nglayout.initialpaint.delay*** majd az értékét állítsuk ***0***-ra.
Így a böngésző a kiszolgálótól kapott információt rögtön megjeleníti az ablakban.
További gyorsulást érhetünk el ha a ***browser.turbo.enabled*** értékét ***true***-ra állítjuk, bár én már nem tapasztaltam sok különbséget.

A ravasz róka

A ***Firefox*** nem csak a gyors böngészést, hanem a gyors kezelhetőséget is támogatja. Amennyiben a ***Ctrl+Enter*** billentyűkombinációt használjuk amikor el szeretnénk érni egy kiszolgálót, akkor a Firefox automatikusan kiegészíti a 'www.' és a '.com' karaktersorozatokkal, a ***Shift+Enter*** kombinációval a '.net' végű oldalakat érhetjük el könnyebben, míg a ***Shift+Ctrl+Enter*** a '.org' végű webhelyek elérésekor használható.
Ahhoz, hogy új weboldalra nézzünk el sem kell hozzányúlni az egérhez, üssük le a ***Ctrl+L*** billentyűkombinációt, és a kurzor rögtön a címsorba kerül, a ***Ctrl+K*** esetében pedig már gépelhetjük is be a keresendő kifejezést a keresés mezőbe.
Sokszor hasznos lehet, hogy minél nagyobb felületen jelenlen meg az éppen böngészett oldal tartalma, használjuk az ***F11*** billentyűt arra, hogy a ***Firefox a*** teljes rendelkezésre álló képernyőt kitölthesse, a normál módba való visszatéréshez ismét nyomjuk le az ***F11***-et.
Egyre ritkábban ugyan, de előfordulnak még olyan web oldalak, ahol egyes objektumok a ***<blink> HTML*** tag segítségével villognak, ami szintén zavaró lehet. A már ismertetett `about:config` beállítások között tehát a `browser.blink_allowed` sorban változtassuk meg a ***true*** értéket ***false***-ra, és többé nincs villogó szöveg.
Bizonyára másokkal is előfordult már az, hogy túl nagy felbontásban böngészett az Interneten, és a karaktereket ugyan lehetett növelni, de az oldal szerkezete, az egyes keretek (***frame)*** mérete nem volt növelhető, így a szöveg szerkezete kissé szétesett.
Erre is van megoldás, mégpedig szintén az `about:config` beállítások között állítsuk át `layout.frames.force_resizability` értékét `true`-ra.
Ennek egyetlen hátránya, hogy a keretek között vastagabb, ugyanakkor áthelyezhető sávok jelennek meg, melyek nem túl esztétikusan néznek ki.

Horváth Ernő
ernohorvath@gmail.com
24 éves, műszaki informatikus. Három évvel ezelőtt ismerkedett meg komolyabban a Linux rendszerekkel és emellett érdeklődik még a robotika és a biztonságtechnika iránt is. Ha lenne szabadideje sokat kirándulna, biciklizne és filmeket nézne.

KAPCSOLÓDÓ CÍMEK

- http://www.spreadfirefox.com/
- http://www.getfirefox.com

A Bochs emulátor

A Bochs egy olyan speciális program, amely az indítása után egy virtuális gépet hoz létre, melyet ugyanúgy kezelhetünk, mint ha valós számítógép lenne. Ez azért jó, mert így a más operációs rendszerre írt programokat saját operációs rendszerük alatt futtathatjuk.

Mi több, még virtuális gépünk processzorának típusát is meghatározhatjuk. Ez lehet ***386***, ***486***, ***Pentium***, ***Pentium Pro***, vagy akár ***AMD64*** típusú is, természetesen mindazon speciális utasításkészlettel, amelyek ezekben a processzorcsaládokban megtalálhatók. A perifériák kezelése is nagyon jó, mivel a gazda operációs rendszerek szolgáltatásait használja.
Ez azt jelenti, hogy minden hardvereszközt emulál és a hozzá intézett feladatokat átadja az operációs rendszer megfelelő moduljának. Mivel minden programutasítást és minden hardverhozzáférést emulál, ezért a ***Bochs*** sebessége nem túl nagy. Kellő türelemre van szükség, hogy a feladatok végrehajtását kivárjuk. Emennyiben nagyobb teljesítményre van szükségünk, úgy más emulátor programot célszerű használnunk, mint amilyen például a ***VMware***.

Telepítés

A ***Bochs*** telepítése nem nagy ördöngösség. ***Linux*** gazda operációs rendszert használva el kell döntenünk, hogy előre lefordított binárisból, vagy forrásból szeretnénk-e telepíteni. Mindegyiknek van előnye és hátránya egyaránt. A bináris állománnyal sok dolgunk nincs, egyszerűen telepíteni kell, azonban meg kell elégednünk a fordítást végző által meghatározott funkciókkal.
A forrásból telepítés több munkával jár, de cserébe személyre szabhatjuk a ***Bochs***-ot. Nézzük először a bináris állomány telepítését.
Töltsük le a ➲ ***bochs.sourceforge.net*** oldalról a legfrissebb verziót, ami a cikk írásának időpontjában a 2.2.1. A fájl ***bochs-verziószám.rpm*** nevet viseli, a cikk írásakor a legfrissebb verzió a 2.2.1, vagyis a telepítő készlete ***bochs-2.2.1.rpm*** néven érhetjük el. A telepítéshez rendszergazdai jogosultság szükséges.

```
root# rpm -iv bochs-2.2.1-1.i586.rpm
```

Amennyiben nem kapunk hibaüzenetet, végeztünk is a telepítéssel. Ha valamilyen hibába ütköznénk, olvassuk el a hibaüzenetet, amire az interneten rákeresvevalószínűleg megtaláljuk a megoldást. Sokszor a nem megfelelő függőségek miatt hiúsul meg a telepítés, ilyenkor lehetőségünk van forrásból új bináris állományt készíteni. Ha a telepítés rendben lezajlott, a rendszerünkben megjelent néhány új parancs, a `bochs`, ami maga a virtuális gép, a `bochs-dlx`, ami egy ***DLX Linux*** demó a próbálgatáshoz, a `bximage`, amivel lehetőség van a lemezek állományait létrehozni és a `bxcommit` ami speciális lemezképfájlok elkészítésére használható.
Rögtön próbáljuk is ki a virtuális gépünket, amihez egy terminálban adjuk ki a `bochs-dlx` parancsot. Megjelenik néhány sor, amely az alapvető ellenőrzéseket hajt végre és közli, hogy létrehozza a felhasználó saját könyvtárában a ***.bochsrc*** fájlt, valamint egy 10 MB nagyságú lemez képfájlt.

```
------------------------------------------------
       DLX Linux Demo, for Bochs x86 Emulator
------------------------------------------------
Checking for bochs binary...ok
Checking for DLX linux directory...ok
Checking for /usr/bin/gzip...ok
Checking for /root/.bochsdlx directory...
------------------------------------------------
To run the DLX Linux demo, I need to create
➥a directory called
/home/user/.bochsdlx, and copy some
➥configuration files
and a 10 megabyte disk image into the directory.
------------------------------------------------
Is that okay? [y/n]
```

Miután a fenti sorokat elfogadtuk, elindul a ***DLX Linux*** demója, és be tudunk lépni root-ként (*1. ábra*).
Amennyiben forrásból szeretnénk telepíteni a ***Bochs***-ot, akkor első lépésként töltsük le a ***bochs-2.2.1.tar.gz*** állományt és csomagoljuk ki a

```
tar -xzvf bochs-2.2.1.tar.gz
```

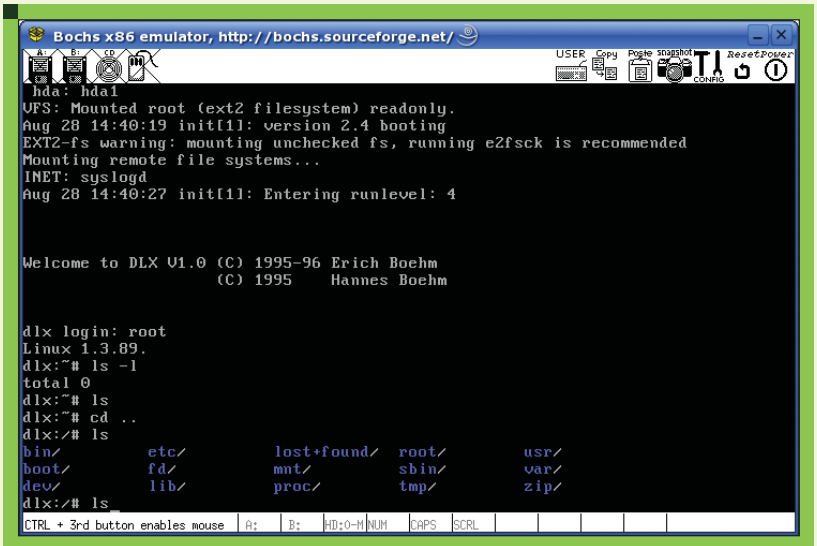
paranccsal. Ha belépünk a létrejött ***bochs-2.2.1*** könyvtárba, első lépésként adjuk ki a

```
./configure --help
```

parancsot és tanulmányozzuk át a hosszú listát, amiből megtudhatjuk, hogy milyen fordítási opciókat állíthatunk be. Miután kiválasztottuk a szükséges opciókat, elindíthatjuk a

```
./configure
```

parancsfájlt. Annak érdekében, hogy megkönnyítsék a felhasználók dolgát, elkészítettek néhány parancsfájlt, amelyek az egyes operációs rendszerekre vannak optimalizálva. Ezeket *.conf.operációs_rendszer* néven találjuk meg. A *Linuxhoz* a *.conf.linux* fájlt kell használni. Amennyiben más opciókat is szeretnénk használni, nyugodtan módosítsuk ezt a fájlt, a későbbiekben is felhasználhatjuk. A listában egy javaslom, hogy módosítsuk a processzor típust (`enabled-cpu-level`) 5-re (*Pentium* osztályú), mivel ez sokkal stabilabb, mint a *Pentium Pro* támogatás.



■ *1. ábra* Indul a DLX Linux


```
sh ./config.linux
```

Ennek hatására konfigurálásra kerül a *Bochs*, amit le kell fordítanunk a `make` paranccsal. Miután hiba nélkül ez is befejeződött, telepítsük fel a programot a `make install` paranccsal. Ne feledjük, hogy ehhez a lépéshez rendszergazdai jogokra lesz szükségünk!
Bármelyik módszert is használjuk, alapértelmezés szerint a *Bochs* a */usr/bin* könyvtárba települ, az osztott fájlokat a */usr/share/bochs* könyvtárban tárolja és a beépülő modulokat a */usr/lib/bochs/plugins* könyvtárba helyezi el.

A Bochs használata

Valószínűleg nem azért telepítettük fel a *Bochs*-ot, hogy egy egyszerű *DLX Linux* demót futtassunk rajta. A rendszer használatához szükségünk lesz magára a *Bochs*-ra, egy *BIOS* képfájlra, egy *VGA BIOS* képfájlra és valamilyen háttértároló képfájljára, vagy esetleg a fizikai eszközre, valamint egy, a beállításokat tartalmazó fájlra.
Ez utóbbi a *.bochsrc* nevet viseli és célszerűen a saját könyvtárunkban helyezhetjük el. A *BIOS* és a *VGA BIOS* a *Bochs* telepítőeszközzel érkezik, de az interneten is találhatunk mások által készített fájlokat.
A következőkben tekintsük át röviden a beállítófájl egyes sorainak jelentését.

```
romimage: file=$BXSHARE/BIOS-bochs-latest,
➥address=0xf0000
```

Ez megadja a *BIOS* képfájlnak a helyét, valamint azt, hogy ezt a *0000h* címre kell betölteni, ami a *RESET* állapotnak megfelelő indulási cím. A *$BXSHARE* változó a */usr/share/bochs* könyvtárra mutat.

```
vgaromimage: file=$BXSHARE/VGABIOS-lgpl-latest
```

A *VGA BIOS* fájl elérési útvonala és neve.

```
vga: extension=vbe
```

VGA kiegészítést kapcsolhatunk be vele, ami lehetővé teszi, hogy a telepített rendszerünket ne csak *640x480*-as felbontásban használhassuk, hanem az *SVGA* felbontások is elérhetők legyenek. Amennyiben ezt szeretnénk használni, vendég operációs rendszerhez be kell szereznünk egy *VBE* meghajtót.
Miután beállítottuk a *BIOS*-okat, meg kell adnunk, hogy mekkora memóriát használhat a virtuális gép. Ehhez a

```
megs: 128
```

sort kell a *.bochsrc*-be beszúrnunk, ami 128 megabájtban korlátozza a memóriahasználatot.
Természetesen a memória méretét a gépünk kiépítettsége alapján kell megállapítsuk. Ha túl kicsire vesszük, akkor nagyon lassú lesz az emulált rendszer, ha pedig túl nagyra, akkor lehet, hogy pazaroljuk a hasznos operatív tárat.
A paraméter maximális értéke `2048` lehet, de `1024` vagy efölötti értékekkel már meggyűlhet a *Bochs* baja, ezért óvatosan bánjunk vele.
Következhet a hajlékony- és merevlemezes meghajtók, valamint optikai tárolók meghatározása.

```
floppya: 1_44=/dev/fd0, status=inserted
floppyb: 1_44=b.img, status=inserted
```

Az A jelű meghajtó a `/dev/fd0` fizikai eszközhöz kapcsolódik, míg a *B* jelű a *b.img* fájlhoz. Ez utóbbi valamilyen hajlékony lemezről készített képfájl. Más lemezméretet is megadhatunk, de nem valószínű, hogy valakinek ennél kisebb méretű lemezekre lesz szüksége. A status kapcsolóval meghatározhatjuk, hogy a lemez behelyezett állapotban van, vagyis használatra kész. Ez a képfájloknál egyértelmű, de fizikai eszközöknél nem feltétlen.

A merevlemezes és az optikai meghajtók megadása előtt meg kell határoznunk, hogy a maximum négy lemezvezérlőből mennyit szeretnénk használni. Ehhez az alábbi sorokat szúrjuk be a .bochsrc fájlba:

```
ata0: enabled=1, ioaddr1=0x1f0, ioaddr2=0x3f0,
➥irq=14
ata1: enabled=1, ioaddr1=0x170, ioaddr2=0x370,
➥irq=15
ata2: enabled=1, ioaddr1=0x1e8, ioaddr2=0x3e0,
➥irq=11
ata3: enabled=1, ioaddr1=0x168, ioaddr2=0x360,
➥irq=9
```

Tiltani az egyes vezérlőket az `enabled = 0` értékkel lehet. Alaphelyzetben csak egy vezérlő aktív, a többi tiltott állapotú. Miután eldöntöttük, hogy mely vezérlőket szeretnénk használni, meg kell határozni, hogy milyen eszközök találhatók a vezérlőkön. Az ***ATA*** eszközökhöz ugyanazt a sort kell megfelelő paraméterekkel használnunk.

```
ata0-master: type=disk, path=/home/jimi/inst/
➥bochs-20050731/c.img, mode=flat cylinders=2031,
➥heads=16, spt=63
ata0-slave: type=cdrom, path=/home/jimi/inst/
➥bochs-20050731/win98.iso, status=inserted
```

Mint látható, a `type` határozza meg, hogy lemezről (`disk`) vagy ***CD-ROM***-ról (`cdrom`) van-e szó. Meg kell adnunk az elérési útvonalat, ami lehet egy képfájl, de lehet akár egy fizikai partíció vagy eszköz is. Ez utóbbival nem árt óvatosnak lenni, mert akár a fájlrendszer is megsérülhet egy esetleges ***Bochs*** összeomlás esetén. Javaslom, hogy a fizikai partíciókról hozzunk létre képfájlt, aminek pontos menetét a cikk későbbi részében természetesen ismertetni fogom.
A lemezeknél használható `mode` kapcsoló adja meg, hogy milyen típusú képfájlokkal dolgozunk.
A leggyakrabban a `flat` típusú fájlokat használjuk, ami egyszerű felépítést és fix méretet jelent, de használhatunk `concat` (több fájlból álló kép) és `growing` (igényeknek megfelelően növekvő méretű) típusokat is. Mint a fenti sorban látható, még a lemezegység geometriáját is megadhatjuk, amit a cilinder (`cylinders`), fej (`head`) és a szektorszámmal (`spt`) jellemeznek.
Miután beállítottuk a háttértárakat, meg kell határoznunk azt is, hogy mi legyen az indítási sorrend. Ehhez szúrjuk be a

```
boot: cdrom, disk, floppy
```

sort. A sorrend természetesen bármi lehet.
Amennyiben szeretnénk, ha a ***Bochs*** a műveleteit folyamatosan naplózza, akkor használhatjuk a

```
log: bochsout.txt
```

sort. ***Linux*** alatt a `/dev/tty` vagy akár a `/dev/null` eszközöket is használhatjuk.
A mai rendszerekben a legtöbb eszköz a ***PCI*** sínrendszert használja a kommunikációhoz, ezért ezt a ***Bochs*** is támogatja. A PCI eszközök használatához szükségünk van a

```
i440fxsupport: enabled=1, slot1=pcivga,
➥slot2=ne2k
```

sorra, ahol a `slot` kapcsolókkal megadhatjuk, hogy melyik csatolóra milyen eszköz kapcsolódik.
Összesen 5 csatlakozó áll rendelkezésünkre, amelyekbe `ne2k` (hálózati kártya), `pcivga` (***VGA*** kártya ***PCI*** csatolóval), `pcidev` (***PCI*** eszközazonosítás és erőforrás kiosztás, de még nagyon kezdeti stádiumban van) és `pcipnic` (hálózati kártya ***PCI*** sínnel) eszközöket „csatlakoztathatunk”.
Előfordulhat, hogy szükségünk van soros portokra, ehhez a

```
com1: enabled=1, mode=mouse
```

sort helyezzük el a ***.bochsrc***-ben. A `mode` kapcsoló értéke `mouse` (szabvány soros egér), `null` (üres be- és kimenet), fájlnév (például `dev=/dev/ttyp9`), valamint `raw` (tényleges hardverport, egyenlőre fejlesztési fázisban van) lehet.
Egy modernebb soros kommunikáció, az ***USB*** támogatás is fontos lehet. Ehhez a

```
usb1: enabled=1, ioaddr=0xFF80, port1=mouse,
➥port2=keypad
```

sort kell használnunk. Ez egy ***i440FX PCI*** lapkakészletet emulál, amelynek része az ***USB*** vezérlő. Mivel minden vezérlő két porttal rendelkezik, ezért a port kapcsolóval megadhatjuk, hogy melyikre milyen eszköz csatlakozik. A használatához szükség van a ***PCI*** támogatás bekapcsolására.
Persze a párhuzamos portot is hasonló módon kezelhetjük, amihez a

```
parport1: enabled=1, fájl="dev/lp0"
```

sort kell használnunk. Ezen portokat az `enabled=0` kapcsolóval lehet tiltani.
Hasonlóan az egér engedélyezéséhez szúrjuk be a

```
mouse: enabled=1, type=imps2
```

sort, míg a tiltáshoz ugyanezt, csak `0` értékkel. A `type` kapcsoló mondja meg az egér típusát, ami `serial` (soros), `serial wheel` (soros egér görgővel), `ps2` (***PS/2***), `imps2` (másik fajta ***PS/2***), vagy `usb` (***USB*** egér) lehet. A soros egér használatához szükség van egy egér típusú (`type=mouse`) soros, míg az ***USB***-s egérnél egy ***USB*** port beállítására.
Amennyiben hangot is szeretnénk a virtuális gépen futó alkalmazásokból kicsiholni, szükség lehet a ***Sound Blaster 16*** hangkártya támogatására. Ehhez az alábbi sort kell beszúrnunk:

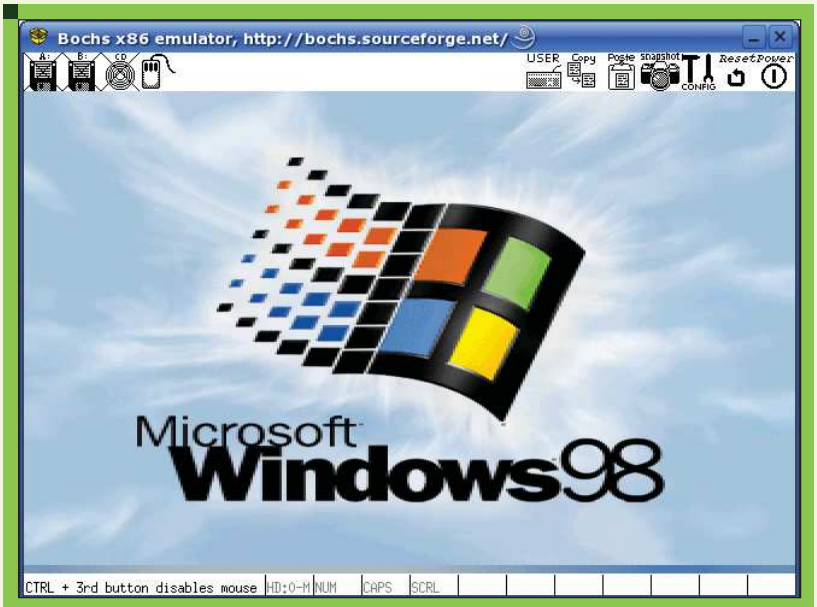
```
sb16: midimode=1, midi=/dev/midi00, wavemode=1,
➥wave=/dev/dsp, loglevel=2, log=sb16.log,
➥dmatimer=600000
```

A `midimode` kapcsoló megadja, hogy milyen módon használjuk az eszközt. Ha értéke `0`, akkor nincs adatkimenet, míg az `1` adatkimenetként használja az eszközt. A `wavemode` hasonló célokat használ, hasonló beállítási lehetőségekkel.
A log kapcsoló határozza meg, hogy a műveleteket a ***Bochs*** hova naplózza. Ehhez szorosan kapcsolódik a `loglevel`

kapcsoló, ami a naplózási eseményeket határozza meg. Értéke leggyakrabban 2 (súlyos hibákat naplóz), vagy 3 (összes hibát naplózza), de 0 és 5 között bármilyen egész szám lehet, a magasabb szám részletesebb naplózást jelent. A *DMA* ciklusok beállítását teszi lehetővé a `dmatimer` kapcsoló.
Bizonyos operációs rendszereknél szükség lehet az emulált gép sebességének a megadására. Ehhez használjuk az

```
ips: 15000000
```

sort, ahol a szám a másodpercenként végrehajtott utasítások számát adja meg. Értékét gyakorlatban tudjuk legjobban meghatározni, amit megkönnyíthet, ha a fordításnál használjuk a `--enable-show-ips` opciót.
Amennyiben szeretnénk hálózatot is használni a vendég operációs rendszerünkben, akkor definiálnunk kell legalább egy hálózati csatolót. Ehhez a



■ *2. ábra* Indul a Windows 98

```
ne2k: ioaddr=0x240, irq=9, mac=b0:c4:20:00:00:00,
➥ethmod=linux, ethdev=eth0
```

sort használhatjuk *Linux* alatt. Az `ioaddr` kapcsoló a perifériacímet, az `irq` a megszakítási vonal számát adja meg. A `mac` kapcsoló a kártya fizikai, adatkapcsolati rétegbeli címét határozza meg, az `ethmod` kapcsolóval állítjuk be az operációs rendszer típusát és végül az `ethdev` kapcsolóval pedig az eszköz nevét. *Windows* alatt ugyanez a sor az alábbiak szerint módosulna:

```
ne2k: ioaddr=0x240, irq=9, mac=b0:c4:20:00:00:01,
➥ethmod=win32, ethdev=KÁRTYANÉV
```

Alapértelmezésben a *Bochs AT* típusú, angol kiosztású billentyűzetet tesz elérhetővé. Ez sok esetben nem megfelelő ezért szükség lehet a ezeknek a megváltoztatására. A

```
keyboard_type: mf
```

megfelelő a legtöbb rendszer számára, de ha nem működne megfelelően, próbálkozhatunk a

```
keyboard_type: at
```

opcióval is. A billentyűzetkiosztás már egy kicsit bonyolultabb, mivel a fizikai gombkiosztáshoz egy más kódkészletet kell párosítanunk. Erre szolgál a

```
keyboard_mapping: enabled=1,
➥map=elérési_útvonal/billenytűzetkiosztás.map
```

sor. Mint látható, a kiosztást egy fájl tárolja, amelyet megpróbálunk begyűjteni az internetről (nekem sajnos nem sikerült megbízható fájlt találni), vagy létrehozunk mi magunk egyet. Ez utóbbi esetben jó szolgálatot tehet az *xkeycaps* nevű program, amelyet forrás formájában letölthetünk a ➲ *http://www.jwz.org/xkeycaps/* oldalról vagy ha segítségül hívjuk a *Google* keresőt, hamar ráakadhatunk a bináris változatra is.
Miután létrehoztuk a *.bochsrc* fájlt, már csak a lemezképet kell létrehoznunk, amit többféleképpen megtehetünk.

Merevlemez képfájl létrehozása

Ebben az esetben a *Bochs*-szal szállított bximage programra lesz szükségünk. A parancs kiadása után első kérdése a programnak, hogy hajlékony- vagy merevlemez fájlt kívánunk létrehozni (`Please type hd or fd. [hd]`), alapértelmezés értéket a szögletes zárójelek között láthatjuk.
A következő kérdés, hogy milyen típusú legyen a képfájl (`Please type flat, sparse or growing. [flat]`), itt maradhatunk a `flat` típusnál, vagy ha nem tudjuk a méretét a fájlnak megtippelni, akkor válasszuk a `growing` típust.
A harmadik kérdés a partíció méretére vonatkozik (`Enter the hard disk size in megabytes, between 1 and 32255`), ami 1 és 32255 MB közötti lehet. Miután ezzel is megvagyunk, kapunk egy összesítést arról, hogy milyen geometria paraméterekkel jön létre a fájl.
Már csak a fájl nevét kell megadnunk (`What should I name the image?`).
Ez egy nyers, formázatlan partíció, amit majd célszerűen az operációs rendszer alatt meg kell formáznunk.
Akkor is hasonló módon kell eljárnunk, ha egy meglévő operációs rendszer partícióját szeretnénk használni, mint lemez képfájl. Első lépésben elkészítünk egy, a fizikai partíciónak megfelelő (nem fontos, de jobb, ha ugyanakkora) lemez képfájlt a `bximage` programmal. Ezt követően a elindítjuk a másik operációs rendszert, kiürítjük a *TEMP* könyvtárakat, eltávolíttatjuk az ideiglenes állományokat és elvégeztetünk egy töredezettség-mentesítést is. Ez ezért fontos, mert így a fájlok folytonosan fognak elhelyezkedni a kép-

■ *3. ábra* Virtuális hardvereszközök

fájlban, ami sebességnövekedést jelent. Igyekezzünk azokat az állományokat is áthelyezni egy másik partícióra, melyek nem képezik az operációs rendszer szerves részét, mivel feleslegesen növelik a képfájl méretét. Miután ezzel is végeztünk, újra ***Linux*** rendszerre váltsunk át és csatoljuk fel a partíciót egy ideiglenes könyvtárba. Például ***Windows*** esetén az alábbi parancsokat adjuk ki:

```
mkdir /windows  #létrehozzuk a /windows
➥könyvtárat
mkdir /windows/c  #azon belül a c könyvtárat
mount -t vfat /dev/hda1 /windows/c/ #felcsatoljuk
➥a hda lemez első partícióját, a fájlrendszer
➥itt vfat (FAT32), de lehet más is
```

Az ***mtools*** csomag segítségével most használhatóvá kell tennünk a nyers képfájlt. Az ***mtools*** olyan parancsoknak a gyűjteménye, amelyeket megszokhattunk a ***DOS*** alatt és mindegyik „m” betűvel kezdődik.
Hozzunk létre a saját könyvtárunkban egy ***.mtoolsrc*** fájlt, amelyben helyezzük el az alábbi sort:

```
drive c: file="/elérési_útvonal/c.img"
➥partition=1
```

Következő lépés, hogy particionálnunk kell a képfájlt. Ehhez az `mpartition` programot kell használnunk, aminek általános alakja:

```
mpartition -I -s szektorszam -t cillinderszám -h
➥fejek_száma c:
mpartition -cpv -s szektorszam -t cillinderszám
➥-h fejek_száma c:
```

Az első parancs beállítja a partíciós táblát és törli a létező partíciókat, míg a második parancs létrehozza az egész képfájlt elfoglaló partíciót. A szektorszámot, a cilinderszámot és a fejek számát a `bximage` kimenetéből tudhattuk meg, amikor létrehoztuk a merevlemez képfájlt.
Következő lépés a létező partíció átmásolása a képfájlba. Ehhez az alábbiakra van szükség:

```
mcopy -s /windows/c/* c:
```

Ezzel meg is vagyunk a képfájl létrehozásával, már csak be kell állítanunk a ***.bochsrc*** fájlban, hogy erről történjen meg a rendszer indulása.

Képfájl létrehozása CD-ROM-ról

Sok esetben előfordul, hogy a telepítőkészlet ***CD***, vagy ***DVD*** lemezen érkezik. Célszerű ebből egy képfájlt készíteni, mert tapasztalataim szerint a ***Bochs*** ezt jobban szereti, nem lehet lemezolvasási hiba, bár helyet mindenképpen igényel.
A képfájl elkészítéséhez használjuk a

```
dd if=/dev/cdrom of=win.iso
```

parancsot, aminek eredményeként a dd egy meglehetősen hosszú munkába kezd, aminek a végeredménye egy ***win.iso*** nevű telepítő képfájl. Ezt már megadhatjuk a ***.bochsrc***-ben a cikkben ismertetett módon.
Az interneten kutakodva, akár csak a ➲ ***bochs.sourceforge.net*** oldalt tekintve jó néhány operációs rendszer képfájlját megtalálhatjuk, ami sok munkától tud megkímélni minket. Nyugodtan próbálgassuk ezeket, hiszen a fizikai rendszerünkben semmiféle változtatásra nincs szükség, így biztonságosan próbálhatók az operációs rendszerek.
A ***Bochs*** indítását a `bochs` paranccsal intézhetjük el. Ennek eredményeként megjelenik egy menü, amelyből az 5-ös menüpontnak kell aktívnak lennie. Amennyiben a 2-tes lenne az, a ***Bochs*** nem találja a ***.bochsrc*** fájlt. Ilyen esetben ellenőrizzük, hogy abban a könyvtárban adtuk-e ki a ***bochs*** parancsot, amelyikben a ***.bochsrc*** található.
Miután elindul a ***Bochs***, megjelenik egy ablak, amelyben elindul a virtuális gép és ha mindent jól állítottunk be, a lemez képfájlokból betöltésre kerül az operációs rendszer (*2. ábra*).
Az ablak tetején egy fejléc található, amely segítségével a ***Bochs*** eszközeit használhatjuk. Az ikonok magukért beszélnek, amely hardvereszköz át van húzva, az jelenleg nem engedélyezett a virtuális gépben (*3. ábra*).
A hardvereszközök munkavégzést az ablak alján látható állapotsor megfelelő része mutatja. Ha használunk egeret és az engedélyezett is a ***Bochs***-ban, akkor amikor az ablak fölé mozgatjuk az egérkurzort, automatikusan átadja a vezérlést a vendég operációs rendszernek. Ahhoz, hogy visszakaphassuk az egeret, használjuk a ***CTRL*** és az egér harmadik gombját egyszerre. Ez tiltja a virtuális gépben az egeret, engedélyezni ugyanezzel a billentyűkombinációval lehet.
A fentiek alapján remélem mindenki be tudja állítani a ***Bochs***-ot és tesztelni tudja a különböző operációs rendszereket. Nem szabad elfeledkeznünk róla, hogy zárt forráskódú, kereskedelmi termékeket csak akkor használhatjuk az emulátorban is, ha rendelkezünk érvényes licenccel. A ***Bochs*** alapvető célokra remekül használható, elsősorban azoknak ajánlható, akik nem engedhetik meg maguknak a drága, nagy teljesítményű kereskedelmi virtuális gépeket (mint amilyen a ***VMware***), de mégis szükségük van más operációs rendszer futtatására.

Markó Imre (linux@akribisbt.hu)
Hardvermérnök és mérnöktanár végzettsége van. Saját cégében Linux rendszerek tervezésével és üzemeltetésével foglalkozik. Ezen kívül egy főiskolán oktat, elsősorban hardveres tantárgyakat.

Grafikus felületek – Enlightenment

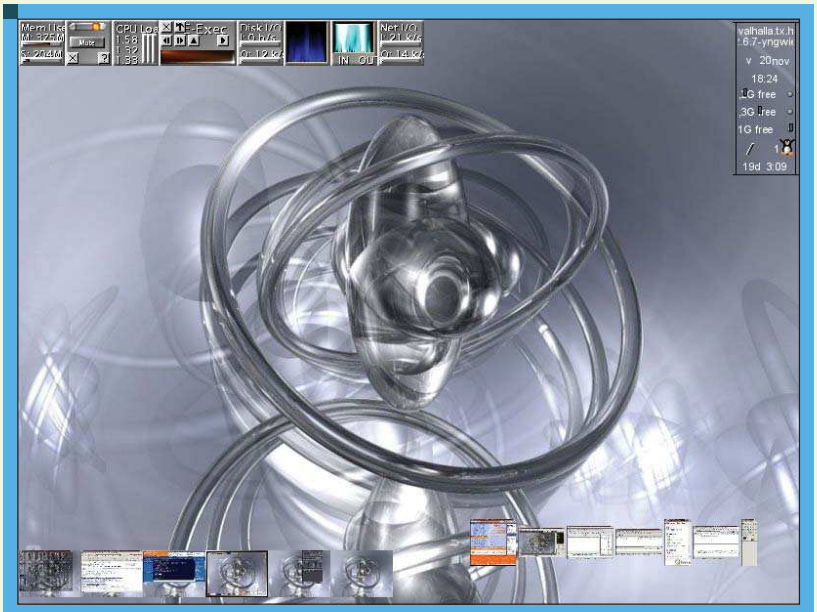
Enlightenment azaz felvilágosodás. Egy ablakkezelő, amely kicsi, gyors és gyönyörű. Kell ennél több? Sokaknak igen. Nekik készült ez a cikk.

Linux alá rengeteg grafikus felület létezik. Az egyik olyan apró, hogy szinte semmit sem tud, a másik hatalmas, mint egy külön operációs rendszer. Mindenkinek megvan a saját kedvence. Sokaknak fontos, hogy ***Linux*** alatt is ***Windows*** hatásokat mutasson az ablakozó. Legyenek ikonok, legyen tálca, az ablakműveletek is hasonlóan működjenek. Nekik tiszta szívvel lehet ajánlani a ***KDE***, ***Gnome*** párost. Mindkettő viszonylag jól használható, sok pluszt tartalmaz (levelezőprogram, szövegszerkesztő, stb), és nagyon hasonlít a ***Windowsra***.

A gyakorlottabb felhasználók zöme, illetve a kezdő felhasználók egy kisebb csoportja viszont kerüli e két monstrumot. Részben azért, mert már olyan szinten vannak, hogy kattintás helyett akár egy beállítófájl átírásával is tudnak szabályozni dolgokat, részben próbálnak minél több memóriát felszabadítani, ezáltal gyorsabbá tenni gépüket. Manapság ugyan már alapértelmezett a minimum ***256MB*** memória, sőt, akár ***512MB***-ot is mondhatnék, de azért jócskán akadnak olyanok is, akiknek gépe szerényebb teljesítménnyel bír.

A ***KDE*** és a ***Gnome*** töménytelen memóriát fogyaszt, ráadásul annyi modulból tevődik össze, hogy néha az egyes modulok kiakadnak egymástól. Még mielőtt valaki megkövezne e két grafikus felületóriás gyalázása miatt, szeretném leszögezni, hogy erről szó sincs. Mivel a felhasználók 90%-a e kettőt használja, ezért őket úgysem tudom meggyőzni arról, hogy használjanak mást.

Vagy mégis? A döntés szabad.



1. ábra Enlightenment a mindennapi használatban

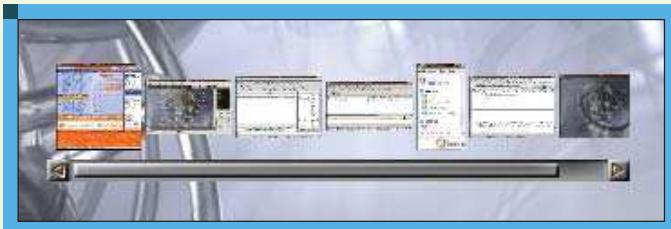
Sokáig jómagam is ***KDE***-t használtam. Egy kezdőnek tökéletes. Én azonban jó ideig lusta kezdő voltam, szerettem, hogy hasonlít a ***Windowsra***. Egy idő után viszont mindent meg lehet unni, mindenki elkezdi keresni az újat, a mást. Grafikus felületek hadát próbáltam ki, míg végül kilyukadtam az ***Enlightenment***nél.

Az Enlightenment

Nehéz dolgom lesz ezen a remekmű a bemutatásával. Kezdhetném azzal, hogy mennyire különbözik a ***Windows***-kinézetű és -stílusú ablakkezelőktől és folytathatnám azzal, hogy milyen jól támogatja az ***átlátszóságot*** (transparency).

Én mégis a megszokott bemutatási módot választom, mindenre kitérve.

Az ***Enlightenment*** (➲ ***http://enlightenment.org***) egy kis erőforrásigényű grafikus felület Linux alá. Nemhiába hangsúlyoztam ki a ***Linuxot***: a ***Linux***-felhasználók az ***Enlightenment*** célközönsége. Több ezren használják, és a fejlesztési szakaszok is itt élveznek prioritást. Ahogy már azonban egy nagyobb projektnél megszokhattuk, több platformot is támogat: ***Solaris/SPARC, AIX, *BSD, IRIX, GNU Darwin***. Ezen operációs rendszerekhez érhető el bináris csomag. Tesztelt rendszerek még a ***Tru64 UNIX***, a ***HP-UX***, és az ***OS/2***, de binárist ezekhez nem találunk. A forráskódot a ➲ ***http://enlightenment.org/Enlightenment/Get_Enlightenment/index.html*** oldalról tölthetjük le.



2. ábra Az Iconbox, vagyis az Enlightenment tálcája

Még itt az elején leszögezném, hogy ez a cikk az ***Enlightenment DR16***-ról szól.
Létezik egy új projekt ***DR17*** néven, ami merőben más attól, amiről írni fogok, ráadásul még eléggé kezdeti stádiumban van. Rengeteg új dolgot fog tartalmazni, például mozgó háttérkép megjelenítését is. Ennél többről most ne essék szó.

Telepítés és az első lépések

Az ***Enlightenment*** *(1. ábra)* telepítése roppant egyszerű. Mivel minden ***Linux*** disztribúció tartalmazza, ezért a CD-kről könnyen feltelepíthető.
A saját rendszeremnél maradva (***Debian GNU/Linux***):

```
apt-get install enlightenment
```

Ha az ***Enlightenment*** feltelepült, akkor a legtöbb disztribúció automatikusan berakja a választható grafikus felületek listájába. Amennyiben használunk ***KDM***-et, ***GDM***-et, illetve ***XDM***-et, akkor nincs más dolgunk, mint kiválasztani az ***Enlightenment***-et a listából. Akik nem használnak ***gui-loadert*** (grafikus felület betöltő), azok a

```
startx enlightenment
```

paranccsal indíthatják a felületet.
Merőben más, mint a megszokott! – gondolhatnánk. És igazunk is van. Az ***Enlightenment*** nem hasonlít semmire, amit eddig számítógépen láttunk. Egy grafikus felület, ami teljesen máshogy működik mint a többi! – gondolhatnánk újfent.
Igen, első ránézésre valóban úgy látszik, mintha tényleg forradalmi alkotás lenne (és valljuk be, valamilyen szinten az is), de amint egy kicsit közelebbről megvizsgáljuk, rájöhetünk, hogy ugyanazokat tudja, mint a többi ablakkezelő. Csak kicsit máshogy.

A legszembetűnőbb dolog, hogy nincsenek ***ikonok*** és nincs ***tálca***. Természetesen ez sem újdonság. Sok ikon nélküli, de tálcás grafikus felület létezik: ***Blackbox, Fluxbox, Openbox, Xfce***, stb.
Sőt, van csak ikonos, de tálca nélküli ablakkezelő is, a ***Windowmaker***.
Most mindenkiben felmerülhet a kérdés, hogy ha nincsenek ikonok és nincs tálca, akkor hogyan lehet ezt a felületet kényelmesen használni. Megnyugtatok mindenkit: remekül. Galád módon elárulom, hogy van tálca és vannak ikonok is, csak nem abban az értelemben kell rájuk gondolni, ahogyan egy „normális” ablakkezelőnél tettük.
Lássuk most az imént említett szokatlan részeket

Az Iconbox, avagy a tálca

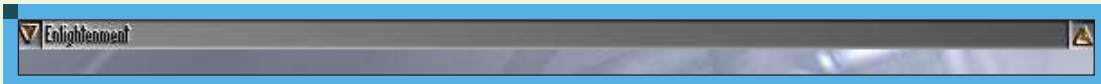
Amint az előbb kiderült, ez a fura felület is rendelkezik „tálcával”. A neve azért ***Iconbox***, mert az alkalmazások ikonjai kerülnek bele. Választhatunk azonban ***mini-snapshot*** nézetet is, ami a program aktuális állapotának kicsinyített képét rakja az ***Iconbox***ba.
Ha ezen ikonok fölé visszük az egérmutatót, akkor témától függően (az ***Enlightenment*** témákról majd a későbbiekben fogok szólni pár szót) látványosan megjeleníthető az alkalmazás neve.
Hogy érthetőbb legyen, megpróbálom elmagyarázni. Egy téglalap alakú területet képzeljünk el, mely a képernyő bármely részére mozgatható. Lehet vízszintes vagy függőleges is. Amennyiben egy futó programot, például egy böngésző-ablakot ***minimalizálunk***, az bekerül az ***Iconboxba***. Tehát úgymond a tálca szerepét látja el ez a kis alkalmazás.
Nagyon tetszetős, lehet átlátszóvá és keret nélkülivé is tenni, illetve hasznos a ***mini-snapshot*** funkciója miatt, ahogy a kép is mutatja *(2. ábra)*.
Elsőre kicsit szokatlan a „tálcapártiaknak”, de kis idő elteltével bárki beláthatja, hogy igazán briliáns ötleten alapszik.

A Pager, avagy a munkaterület-választó

A legtöbb linuxos grafikus felület rendelkezik a ***multiple desktops*** (több „asztal”) funkcióval. Azért nagyon hasznos dolog ez, mert nincs megkötve a kezünk, másképpen mondva nem vagyunk „tálcához kötve”. Ez annyit jelent, hogy nemfeltétlen kell minden programot „letenni” a tálcára ahhoz, hogy egy takarásban lévőt használni tudjunk. Én általában 6 darab virtuális asztalt használok, a következő programokkal: 1. konzol, 2. böngésző, 3. ***irc***, 4. ***xmms*** és ***msn***, 5. letöltőprogramok, 6. egyéb.
Nem feltétlen ebben a sorrendben, de nálam ez bevált. Természetesen ez nem zárja ki azt, hogy mondjuk 6 terminál legyen megnyitva, vagy 12 munkaterületet használjunk.
Az ***Enlightenment*** 32 darab virtuális asztalt enged használni egyidejűleg.

3. ábra A Pager, a munkaterület-váltó

A ***Pager*** *(3. ábra)* nagyon intelligens munkaterület-váltó. Sok dolgot megenged, amikre nem is gondolnánk elsőre.
Ha egy ***Pager***-beli program fölé visszük az egérkurzort, akkor automatikusan ráközelíthetünk, illetve



4. ábra A Dragbar, ahonnan minden elérhető egyszerre

a *Pager*en belül is „áthúzhatunk" programokat az egyik virtuális asztalról a másikra.
Nézzünk erre egy példát: az egyik asztalunkon fut egy ***xmms***, de eltakarja egy ***böngésző***. Ekkor nem muszáj azt tennünk, hogy átváltunk arra az asztalra, majd „lerakjuk" a böngészőt az ***Iconbox***ba. Egyszerűen a böngésző fölé visszük az egérkurzort a *Pager*ben, majd „megfogjuk" egy kattintással és lenyomott egérgomb segítségével áthúzzuk egy másik asztalra a *Pager*ben.
Ez egy remek funkció, én számtalanszor használom.
Természetesen a ***Pager*** is bárhova elhelyezhető az asztalunkon, ebben sincs megkötés. Lehet vízszintesen és függőlegesen használni. Keret nélkül természetesen olvad bele a háttérbe, akár az ***Iconbox***.

A Dragbar, avagy mit nekem Pager

Be kell valljam, én ***Pager***-párti vagyok, így ezt a funkciót nem használom. A ***Dragbar*** *(4. ábra)* azt a célt szolgálja, hogy minden futó alkalmazást egy helyről érjünk el.
Akinek bonyolult a ***Pager***, vagy zavarja, hogy az ***Iconbox***ban képek vannak nevek helyett, azoknak nagyon hasznos lehet.
A ***Dragbar*** egy ***Enlightenment*** témától függően tetszetős csík, melyet a képernyő valamelyik szélén, illetve alján vagy tetején lehet elhelyezni. Jobbklikkre megjelenik az összes futó program listája, melyből kiválaszthatjuk azt, amellyel épp foglalkozni szeretnénk.

5. ábra Felhasználói-menü

Az Enlightenment menürendszere

Összetett menürendszerrel találjuk szembe magunkat. Tulajdonképpen 3 fő részre oszthatjuk az ***Enlightenment*** menüt, melyeket a megfelelő egérgomb lenyomásával hívhatunk elő. Mindegyik egérgombhoz társul egy.
Lássuk akkor ezeket.

Bal egérgomb, avagy User Menu (felhasználói menü)

Kezdjük rögtön egy tippel. Ez a menü nem létezik rögtön első indítás után. Nekünk kell létrehozni. Kicsit előre fogok szaladni a menütémában, de nem tehetek róla, ezt muszáj meglépni ahhoz, hogy legyen felhasználói menünk.
Felhasználói menüt a következőképpen hozhatunk létre. Középsőklikk az asztal bármely pontjára, majd ***Maintenance*** menüpont, majd ***Regenerate Menus***. Ez a kis művelet szükséges, de nem elégséges feltétele a ***User Menu*** (bal egérgombos menü előcsalogatása) létrehozásának. Ezek után már működik a bal egérgombos kattintás, de üres menü fogad minket (jobb esetben néhány alapértelmezett program, például terminál-emulátor, vagy böngésző szerepelhet benne).
Az elégséges feltétel egy kicsit bonyolultabb: mint a név is mutatja (***User Menu***), nekünk kell kézzel feltölteni a menü adatait.
Ezt a következőképp tehetjük meg.
Egy tetszőleges rejtett fájl és könyvtár mutatására alkalmas fájlkezelővel lépjünk be a ***home*** könyvtárunkba, majd keressük meg a ***.enlightenment*** (rejtett) könyvtárt.
Aki nem riad el a parancssortól (és itt megjegyezném, hogy a ***Linux*** egy parancssoros rendszer, ezért nem árt minden dolgot parancssorból is végrehajtani, legalább gyakorlásképpen), az a

```
cd ~/.enlightenment
```

paranccsal is megteheti ugyanezt a lépést. Ebben a könyvtárban az Enlightenment konfigurációs fájljait találjuk. Keressük meg a ***user_apps.menu*** nevű fájlt, majd szerkeszthetjük. Amennyiben vannak már elemek a menünkben, úgy könnyen rájöhetünk egy új elem felvételének miként jére.
Természetesen fennállhat az is, hogy üres a fájl, ezért most nyújtok egy kis segítséget a kezdő lépésekhez:

```
"Felhasználói alkalmazások"
"aMSN" NULL exec "amsn"
"xchat" NULL exec "xchat"
"MPlayer" NULL exec "gmplayer"
```

Magyarázat: Felhasználói alkalmazások lesz a menübejegyzés neve, illetve 3 program fog benne szerepelni: ***amsn***, ***xchat***, ***mplayer***.
Az első idézőjeles rész a program nevét mutatja, amely majd megjelenik a menüben. A `NULL` itt azt jelenti, hogy ikon nélkül szeretném látni a menübejegyzést (természetesen az ***Enlightenment*** menüje támogatja az ikonokat is; ilyenkor meg kell adnunk az ikon elérési útját; ez fog kerülni a `NULL` helyére).
Az `exec` végrehajtást jelent, majd a sor végi idézőjeles rész a futtatandó alkalmazás elérési útja. A példában azért nincs teljes elérési út, mert a */**usr**/**bin***, illetve */**bin*** alatti alkalmazások ***Linux*** alatt direkt névhívással elérhetők. Ha mondjuk a */**opt**/**bin***-be installáljuk az ***amsn***-t, akkor természetesen `"MPlayer /opt/bin/amsn"`-t kell írnunk a sor végére.

6. ábra Minden menüpont egyben

Erre alapozva létrehozhatjuk a saját alkalmazásaink menüjét, majd ha kész, akkor a fent említett módon (középsőklikk, ***Maintenance/Regenerate Menus***) hozzuk érvénybe a változásokat.
A felhasználói menü része még az ***Enlightenment*** újraindítása (***Restart Enlightenment***), illetve a kijelentkezés (Log Out).

Középső egérgomb, avagy minden egyben

Ebben a menüben mindent megtalálunk. Az imént említett felhasználói menütől kezdve az asztalbeállításokon át a globális beállításokig. Lássuk akkor, hogy mikből is áll ez a rész.

User Menus – Az előbb tárgyaltuk.

Debian Menus (csak ***Debian GNU/Linux*** alatt) – Disztribúciófüggő menü. Az ***Enlightenment*** támogatja mind a ***KDE***, mind a ***Gnome*** menübejegyzéseket és importálja őket a saját menüjébe. Tehát például ***SuSE Linux*** alatt ***SuSE Menus*** lesz a menü neve.

Desktop – A legfontosabb ebben a menüben, hogy általa lehet kezelni a háttérképeket.

Enlightenment háttérkép-beállítás gyorstalpaló: belépünk a home könyvtárunk rejtett ***.enlightenment*** könyvtárába, majd létrehozunk egy új könyvtárat ***backgrounds*** néven, azaz

```
cd ~/.enlightenment
mkdir backgrounds
```

Bemásoljuk a ***.jpg*** kiterjesztésű képfájlokat a ***backgrounds*** könyvtárba, majd újrageneráljuk a menüket a szokásos módon. Ezután a középső egérgombos menüben a ***Desktop*** menüpont alatt megjelenik egy ***Backgrounds*** menüpont, mely megjeleníti az előbbi képek kicsinyített mását. Nincs más dolgunk, mint kiválasztani a nekünk legmegfelelőbbet és rákattintani. Máris megjelenik az új háttérkép.

Settings – Később térek ki rá.

Themes – Eljött az idő egy teljes renoválásra! Az ***Enlightenmenthez*** rengeteg téma található és kivétel nélkül mindegyik impozáns. Letöltünk egy nekünk tetsző témát egy ***Enlightenment*** témákkal foglalkozó oldalról (például ➲ http://e.themes.org) és kitömörítve bemásoljuk a ***~/.enlightenment/themes*** könyvtárba (ha ez a könyvtár nem létezik, akkor hozzuk létre). Újrageneráljuk a menüket, majd a helyi menüben a ***Themes*** pont alatt megjelenő menüből kiválasztjuk a nekünk megfelelőt.

Maintenance – Kiüríthetjük a ***Pager***, a ***háttérválasztó***, a ***konfigfájlok*** gyorsítótárát, akár külön, akár egyszerre. A másik fontos menüpont itt a sokat emlegetett menügenerálás.

A középső egérgombra előtűnő helyi menü tartalmaz még súgót (***Help***), az ***Enlightenmentről*** és az aktuális témáról információkat (***About Enlightenment***, ***About this Theme***), illetve újraindítást és kijelentkezést.

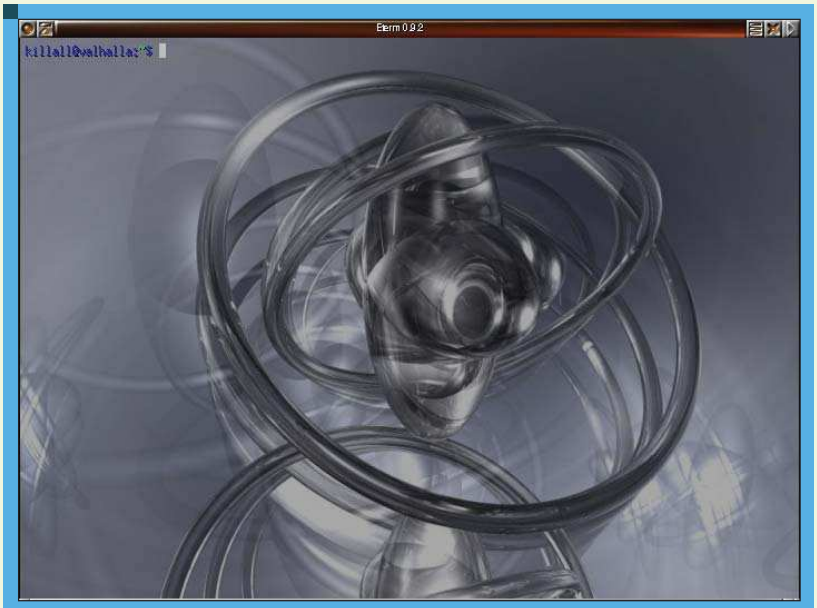
Jobb egérgomb, avagy Settings

Az egyéb beállításokról, vagyis a globális ***Settings*** menü tartalmáról az *1. Táblázat* ad áttekintést.

Az Eterm, avagy az Enlightenment Terminál

Minden valamirevaló grafikus felület rendelkezik saját fejlesztésű terminálemulátorral. Elég csak a ***Konsole***-ra, vagy ***Gnome-terminalra*** gondolni.

7. ábra Settings-menü



8. ábra A tetszetős Eterm

1. Táblázat *Beállítások*

Focus Settings	Ablakfókuszálás beállítása
Move & Resize Settings	Ablakmozgás, illetve ablakméretek állítására szolgál.
Pager Settings	A Pager beállításai.
Window Placement Settings	Ablakok elhelyezkedése.
Multiple Desktops Settings	A már említett „több desktop egyszerre” beállításai.
Virtual Desktop Settings	Virtuális asztalok számának beállítása.
Autoraise Settings	Ablakok automatikus felnyílásának beállítása (ez elég idegesítő tud lenni, érdemes nem bekapcsolni).
Tooltip Settings	Az Enlightenment majdnem minden részéhez segítséget nyújt tooltip formában (annyit jelent, hogy ha bármilyen rész fölé visszük az egeret, akkor kis buborékokban segítséggel lát el bennünket). Kezdőknek hasznos lehet, folyamatos használat során viszont idegesítő.
Audio Settings	Ablakmozgásokhoz lehet rendelni hangeffekteket. Aki szereti ezt, annak hasznos lehet. Engem speciel zavar.
Group Settings	Egyben és külön-külön is lehet állítani az ablakok kinézetét. Ez a menüpont az egybeni változtatásra szolgál.
Remember Settings	Ha valamit átállítunk, azt az Enlightenment megjegyzi. Ebben a menüpontban lehet megtekinteni és módosítani az emlékeztető listát (Remember List).
Special FX Settings	Dragbar beállítások és egyéb speciális effektek. Például be lehet állítani a kötött oldalszéleket, ami azt jelenti, hogy nem tudunk programokat áthúzni az egyik asztalról a másikra. Hasznos menüpont.
Desktop Backgrond Settings	Minden egyes virtuális asztalunknak más és más hátteret állíthatunk be.
Misc. Settings	Ablakok fejlécének használata és elhagyása.

Az ***Enlightenmenté*** az ***Eterm*** ***(8. ábra)***. Funkciókban gazdag: támogatja a színes karaktereket, az átlátszóságot, sőt, szabadon választott háttérképet is rakhatunk bele. Nagyon elegáns, kiváló benne dolgozni. Aki óvakodik a parancssortól, az ***Etermmel*** meg fogja szeretni.

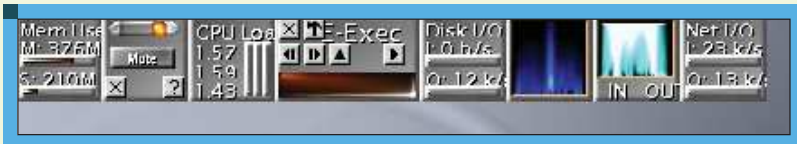
Bővebb információhoz adjuk ki a

```
man Eterm
```

parancsot (vigyázzunk: nagy E-vel írandó!), melyből megtudhatunk minden opciót és kapcsolót.

Az Enlightenment Epplet-ek, avagy figyeljük rendszerünket igényesen

Az ***Enlightenment*** kifejlesztett egy rendszerfigyelő alkalmazáscsaládot amelyek összefoglaló neve az Applets (kisalkalmazások) ***Enlightenmentesítéséből*** keletkezett. Ezek az ***Epplet***-ek *(9. ábra)*.



9. ábra Enlightenment Epplets, az igényes rendszermonitorozás

Rengeteg ***Epplet*** létezik, hasznos, illetve mókás célokra. Hogy csak néhányat említsek: van internetforgalom-figyelő, merevlemez-figyelő, hangerő-szabályzó, processzorfigyelő, mini-paranccsor, stb. Telepítésük hasonlóan egyszerű, mint az ***Enlightenmenté*** (***Debian GNU/Linux*** alatt):

```
apt-get install epplets
```

A többi disztribúciónál keressük meg a ***CD***-n, és a szokásos módon telepítsük. Érdemes őket kipróbálni, hasznosak, és jól mutatnak az asztalon. Amennyiben feltelepültek, a ***User Menu/Enlightenment Epplets*** menüpont alatt lesznek megtalálhatóak.

Remélem sikerült felhívnom a figyelmet erre a felületre. Elsőre talán furcsa a használata, de szerintem aki szereti kipróbálni az új és nem megszokott dolgokat, az nagyon meg fogja szeretni.

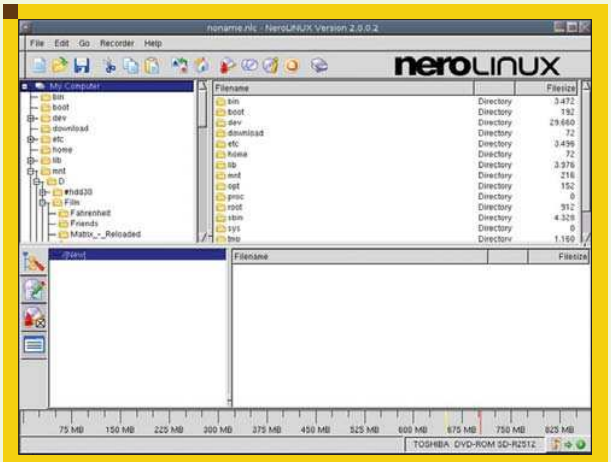
Apagyi György, (killall)
(killall@linuxforum.hu)

24 éves, jelenleg az ELTE programozó matematikus szakán másodéves hallgató. Hobbija a zene (gitározás), az olvasás (Stephen King) és a számítástechnika (Linux, Unix, VMS).

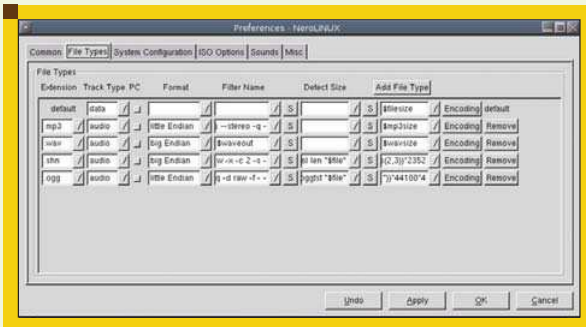
A császár új írója – NeroLINUX

Nem tudom ki hogy van ezzel, az én kedvenc CD-író programom a Nero. Kezdetben ugyan (amikor még gyerekcipőben jártak ezek a dolgok) használtam más szoftvereket is, de aztán ennél ragadtam le. Az ok elég egyszerű: ez volt a legjobb. Megbízható volt, nem szorított felesleges korlátok közé, mindemellett a kezelőfelülete barátságos, nem is beszélve a varázslóról, amivel még a legtapasztalatlanabb felhasználó is könnyen tudott mindenféle dolgokat CD-re írni.

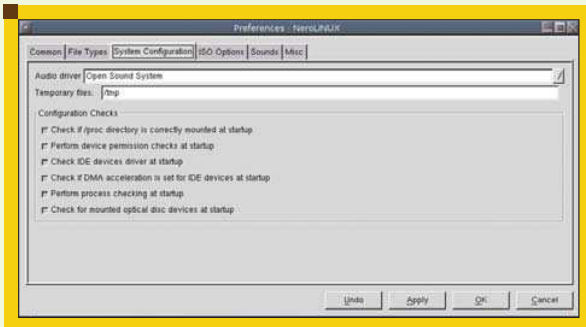
Ez a múlt persze komoly kihívás egy új verziónak, hiszen a tapasztalt felhasználó minimum ugyanezt fogja elvárni a *NeroLINUX*-tól is. Az már első pillanatra világossá vált számomra, hogy nem igazán kezdőknek készítették a programot, hiszen nincs futtatható telepítője. Maradtak a szabványos linuxos telepítőmechanizmusoknál, sőt itt sem vitték túlzásba a kényelmet.
A weboldalon elérhető egy *rpm* és egy *deb* csomag, valamint a *Gentoo portage*-ról is lehet telepíteni. Mindenesetre kétségbeesni nem kell, ha nem ért az ember annyira a *Linuxhoz*.
Az oldalon fellelhető egy *QuickStart Guide*, amiben minden – még a kernelkonfiguráció is – le van írva.
Első „érdekes” élményem a demó verzióhoz fűződik. A drága ugyanis közölte velem, hogy már lejárt a próbaidőszak, úgyhogy mindenképpen kell neki sorozatszám. Kicsit meglepődtem, tekintve, hogy nem sokkal korábban sikerült feltelepítenem először. Aztán elővettem egy OEM szériaszámot, de köszönte szépen, de nem kérte. Így végül kénytelen voltam kölcsönkérni egyet a próba idejére.
Túljutva a hercehurcán azonnal belevetettem magam a beállítások rengetegébe. A windowsos verzióban is ezzel szoktam kezdeni, így rögtön feltűnt, hogy a túlírási lehetőséget kihagyták a csomagból. Olyan nagy szükség ugyan nincs rá, de néha adódhat úgy, hogy pár megabájttal vagy másodperccel többet akarunk írni egy lemezre. Na mindegy. Valahogy csak meglesznünk nélküle...
Kisvártatva az is feltűnt, hogy külső programokat is használ. Nem mintha titkolná ezt a gyártó, csak én nem a *QuickStart Guide* elolvasásával kezdtem, amiben szerepel ez az információ. Ezek után viszont már elolvastam a Nero oldalon található információkat és a dokumentációt is, mivel felvetődött bennem a gyanú, hogy esetleg csak egy felületet kaptunk egy már meglévő linuxos *API*-hoz. Szerencsére nem így van, nem vagyunk szemfényvesztés áldozatai.
A *NeroLINUX* a *Nero API*-t használja a különböző médiák égetéséhez, ami, be kell vallani, nagyon jól végzi a dolgát.
A külső programokat a zenék lejátszására vagy az íráshoz való dekódolásra használja, így simán lehet például *mp3*-ból vagy *ogg*-ból zenei lemezeket írni. Ezen programok használata bizonyos szinten testreszabható, szóval a néminemű mazochizmussal megáldott felhasználóknak is tud kellemes perceket szerezni a szoftver. Ráadásul ez még nem minden.
A *Nero* lehetőséget ad arra is, hogy új fájltípusokat ismertessünk meg vele, így ha nem vagyunk elégedettek a meglévőkkel, vagy valamilyen egészen egzotikus kodekkel kezelt fájlból akarunk zenéket írni, arra is van lehetőség.
Tovább böngészve a beállítási lehetőségeket, kiderül, hogy a gyengébb géppel rendelkezőknek is kedveztek. Ugyanis ha valakinek nincs elég erős vasa ahhoz, hogy röptében (on-the-fly) írjon *mp3*-ból zenei *CD*-t – vagy a vas elég erős, és ehhez mérten eléggé terhelt is –, akkor annak lehetősége



1. ábra Nero indítás után



■ *2. ábra* Fájl típusok beállítása avagy mazochizmus Nero módra

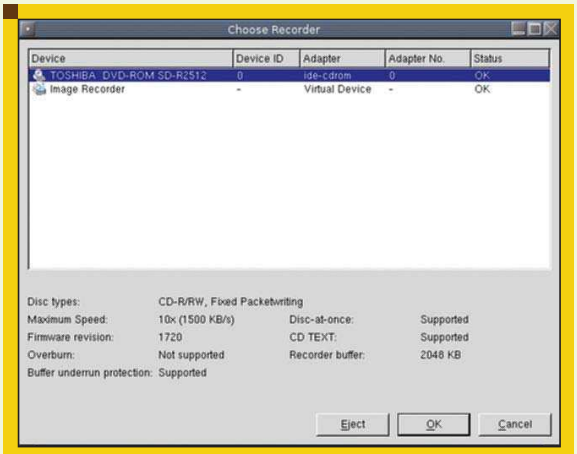


■ *3. ábra* Fontosabb beállítások

van az úgynevezett precache bekapcsolására. Erre egyébként nem csak lassú gépeknél lehet szükség, hanem például ha régi *CD*-t akarunk másolni, amit nem tud elég gyorsan olvasni az olvasó. Tegyük azért hozzá, hogy a *Nero* használja puffer kiürülés elleni technológiákat, így annyira nem létkérdés ez az opció, de mindenképpen jól használható.

Van a precache-nek egy másik változata is. Előfordulhat, hogy nem lehet jól megállapítani egy *mp3* hosszát, mert hibás. Ilyenkor, ha jobb gombbal húzzuk át a zenét, akkor kicsomagolja az egészet *wav*-be és így már meg tudja állapítani a szám hosszát.

Apropó zenék. Miközben állítgatjuk össze a kiírandó zeneszámokat, simán belehallgathatunk bármelyikbe. Itt viszont véget is ér a dolog, mert se azt nem látjuk, épp hol tart a szám, se belepörgetni nem tudunk. Viszont ahhoz, hogy tudjon lejátszani, be kell állítanunk az megfelelő meghajtót (audio driver). Itt az a két



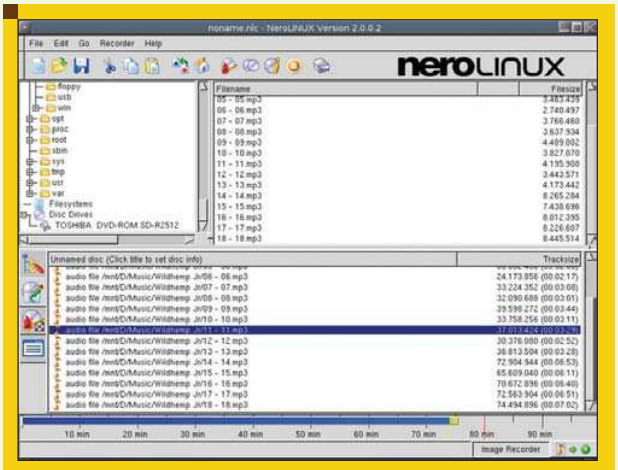
■ *4. ábra* Író kiválasztása a régi jól bevált módon

darab lehetőség szintén nem túl bőséges, de legalább a legfontosabbak megvannak: *OSS* és *eSound*. Véleményem szerint az *ALSA*-t nem kellett volna kihagyni, de hátha a következő verzióban.

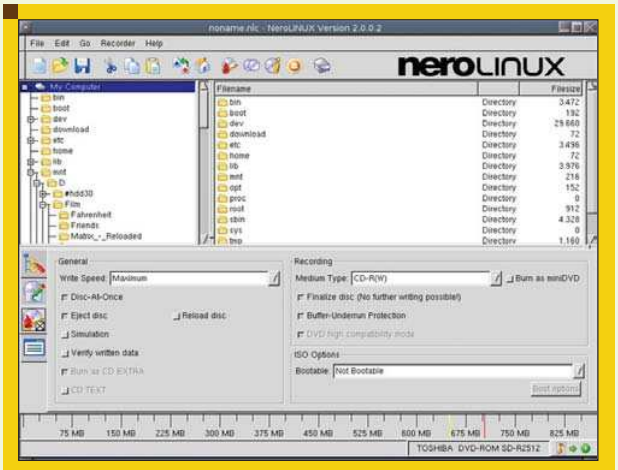
Ezzel át is rágtuk magunkat a fontosabb beállításokon, úgyhogy nézzük tovább, mire képes még a program. Aki próbált már régebben *CD*-t vagy esetleg *DVD*-t írni *Linux* alatt, az tisztában van a téma körüli kernel kavalkáddal. Lőn ugyanis, hogy a 2.4-es kernelben lévő *ide-cdrom* driver nem vette a fáradtságot, hogy írási képességekkel is rendelkezzen, így kerülő megoldást kellett találni. Ez pedig úgy nézett ki, hogy beletettek a kernelbe egy *SCSI* emulációs modult, ami a meglévő *IDE* írót *SCSI* íróként láttatta a rendszerrel. Ennek aztán meg is lett az eredménye. Lehetett írni a *CD*-ket… maximum 4x sebességgel. Aztán jött a „hős" 2.6-os kernel teljesen újraírt *ide-cdrom* meghajtóval plusz *DMA* támogatással, és mindenki örülhet, lehet szélnél is sebesebben írni.

Természetesen a *NeroLINUX* mindkét kernelt jól kezeli, sőt a dokumentációban – mint azt már említettem – végigvezetnek minket a különböző beállítások miként jén is. A program egyébként automatikusan érzékeli a környezetet – ez részben testre is szabható –, és kiabál, ha úgy érzi, valami nem stimmel. Nekem is szólt, hogy be kellene kapcsolni a *DMA*-t, különben búcsút mondhatok a sebességnek. Nagyon jól el lett találva az író meghajtó kiválasztása. Egyrészt teljesen olyan, mint Windows alatt, ami rögtön jól jön, ha valaki nem nagy bűvész, másrészt ott is nagyon jól el lett találva. Valamint szintén megtalálható az *Image Recorder*, amivel *Nero* képfájlokat (image) készíthetünk mind *CD*/*DVD*-ről, mind általunk összeállított fájlokból, zenékből stb. Külön tetszett, hogy bármikor cserélgettem a meghajtóban lévő lemezeket, mindig egyből frissítette a lemezinformációkat. Egy szó mint száz, nagyon jól meg van csinálva ez a rész.

Na, ha mindezeken a dolgokon túljutottunk, és úgy érezzük, hogy kész van a kicsike a munkára, akkor vessünk egy pillantást a felületre, amivel előkészítjük az írást. Azt kell mondjam, hogy sikerül *Ahead*-éknál egy első ránézésre számomra kicsit szokatlan, de annál jobban használható, mégsem varázslós *GUI*-t összehozni. Mint az a képeken is látszik, a kezelő felület alapból két részre van osztva. Az egyiken a fájlrendszerünket látjuk, a másikon pedig a készítendő



■ *5. ábra* Minden együtt: könnyen kezelhető felület, jobb alsó sarokban zene lejátszásra szolgáló gombbal



■ *6. ábra* Írási beállítások

CD/DVD fájlrendszerét. Az előbbi nem túl érdekes, mindenki látott már hasonlót.

Utóbbi annál inkább. Négy fülre van osztva. Ebből az elsőt használva írhatunk adatokat a *CD/DVD*-re. Itt könyvtárakat hozhatunk létre, fájlokat dobálhatunk bele, stb. A második fül segítségével zenéket írhatunk. Ide is simán beledobálhatjuk az írni kívánt számokat, aztán hadd menjen. Természetesen lehetőség van sorrendváltoztatásra, törlésre, stb. A két fül egyszerre is használható, így tudunk kevert *CD*-ket készíteni. Némi kényelmetlenséget okoz, hogy nem használható a *DEL* gomb törlésre, ehelyett mindig jobb egérkattintás hatására lenyíló helyi menüben kellett a *remove*-ot nyomkodnom.

A harmadik fül az írási beállításokat tartalmazza teljesen átlátható módon a negyedik fülön pedig az írási naplót láthatjuk.

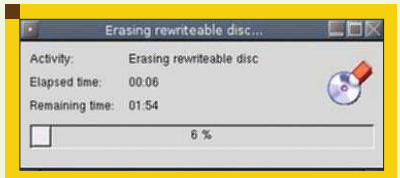
Ha mindent beállítottunk, akkor jöhet az írás. Mint már említettem, az író mag rendkívül jól végzi a dolgát, teljesen megbízható, és nem kell vérre menő harcot vívnunk ahhoz, hogy írás közben más programokat használhassunk. Számomra külön érdekes volt, hogy megnyitási/lezárási fázisban sem akad le a rendszer, bár gondolom ez inkább az operációs rendszer, mint a *Nero* tulajdonsága.

Azonban sajnos nem csak jót tudok írni a szoftverről. Azt ugyan nagyon jól csinálja, hogy ha nem üres, de újraírható lemezt tettünk be akkor felajánlja a törlést és ha kérjük, törli is a lemezt. Nagyon durva hibának tartom viszont, hogy ha írás közben *Cancelt* nyomunk, akkor egy az egyben lezárja a *CD/DVD*-t és kész. Semmi kérdés, hogy tényleg meg akarjuk-e szakítani vagy ilyesmi. Hát enyhén meglepődtem ezt látva.

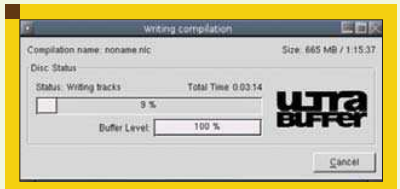
A másik dolog, amivel nem teljesen voltam kibékülve zenei *CD* esetében, hogy nem lehet állítani a számok közti szünetet. Szerencsére nem hagy a számok között szünetet, ami a legjobb lehetőség ebben az esetben, de úgy érzem ez elég súlyos hiányosság. Szintén elfeledkezhetünk a zenei állományok manipulálásáról, mivel ezt a funkciót is kihagyták belőle.

Adat *CD* esetén sikerült valamibe belenyúlnom, ugyanis 5-ből 5-ször „elszállt" minden szó nélkül, amikor az egyik könyvtáramat – nem éppen kevés fájlal – próbáltam meg CD-re archiválni. A hiba okára nem jöttem rá, és más dolgokat simán rá tudtam írni, úgyhogy nem foglalkoztam a dologgal tovább, de nagyon jól mutatja az eset, hogy még nem teljesen kiforrott a szoftver.

Fontos lehet még az is, hogy amíg a *Nero* megy, addig abszolút nem tudjuk használni a *CD/DVD* meghajtót, vagyis a benne lévő adathordozó tartalmát nem tudjuk megnézni. Ez azért van, mert *Linux* alatt nincs lehetőség a meghajtó teljes zárolására, és a program így próbálja meg megakadályozni, hogy valaki a lehető legrosszabb pillanatban belenyúljon a dolgokba. Ennek ellenére a *QuickStart Guide*-ban említést tesznek pár programról – *KDE* és *Gnome* kapcsán – amelyek tönkretehetik az éppen készülő lemezt, így érdemes használat előtt mindenképpen végignyálazni ezt a dokumentációt.



■ *7. ábra* Intelligens lemezkezelés…



■ *8. ábra* …és írás, „unintelligens" Cancel gombbal…

Összefoglalva jó kis program ez, és biztos sokaknak megér 19.99 dollárt, de azért még van hova fejlődnie, különösen, hogy vannak ingyenes versenytársak is hasonló képességekkel (*x-cd-roast*). Azonban ha hasonló módon és igényességgel végzik a fejlesztést, mint a *Windows* esetében, akkor egy újabb platformon lehet a legjobb a *Nero*.

Kenderesi Norbert (wildhemp@gmail.com) Programozó vagyok, már 12 éves korom óta ezzel foglalkozom. A Linuxszal valamikor 98 környékén ismerkedtem meg. Kedvencem a Debian volt, amíg néhány hónapja ki nem próbáltam a Gentoo-t.

Az asztali rendszerek biztonságáról (1. rész)

A számítógépek biztonsága napjaink egyik legfontosabb kérdésévé vált. UNIX alapú rendszereken leginkább a szerverek biztonsága a fő téma, mivel elterjedt vélekedés, hogy biztonságosabbak a Windows rendszereknél.

Ez általában igaz is, de az egyéni felhasználó végül könnyen óvatlanná válhat, az ismeretek hiánya vagy a nem kellő odafigyelés pedig akár helyrehozhatatlan károkat is okozhat. A desktop gép szinte állandóan ostrom alatt van: otthon gyermekünk, munkahelyünkön a munkatársak vagy éppen főnökünk tehet valami nem kellemes dolgot gépünkkel.

Nem beszélve a nyilvános helyeken lévőkről: könyvtárban, internet kávézóban, e-ponton működő desktop rendszerekről: ezek biztonsága még inkább fontos kérdés. A fenyegetés ezen kívül jöhet a belső hálózatról és az internetről egyaránt. De *Murphy* óta azt is tudjuk, hogy ami elromolhat, az el is romlik, tehát végső soron a számítógép önmaga ellenségévé is válhat. Ideje tehát megerősíteni saját vagy cégünk gépeit!

Mit is védünk?

Sokszor elégnek érezzük, ha munkahelyi rendszergazdánk felkészültnek mutatkozik és mindenféle trükkös módszerekkel körülbástyázza a szervert. Otthon pedig ha jelszóval kell bejelentkeznünk, máris „védve van a gép”. De mit is veszíthetünk?
Ezt érdemes átgondolnunk. Ha csupán internetezünk, akkor nem sokat, hiszen legfeljebb időveszteség ér bennünket az újratelepítés miatt. Sokan úgy gondolják, a nyílt forráskód világa, és a linuxos munkaállomások védve vannak a hálózatról érkező támadásokkal szemben.
Pedig ez nincs így. Valószínűleg fontos dolgokat is tárolunk gépünkön, amit nem szeretnénk más kezébe adni vagy éppen örökre elveszíteni. Nagy gonddal csiszolt beállításaink is képezhetnek értéket, amikkel kényelmesre és kellemesre faragtuk kedvenc operációs rendszerünket. Ha email klienst is használunk, akkor általában teljes levelezésünk is gépünkön tárolódik, amit nem szeretnénk mások orrára kötni, sőt az is bizalmas információ tárgyát képezi, ami a böngésző gyorsítótárában vagy az előzményekben megőrződik.
Nyilvános helyen lévő gépen általában erős korlátozásokat is be kell vezetni, hogy a felhasználók ne tehessenek meg bármit, sőt inkább csak egy-két, általunk kívánatosnak tartott dolgot (pl. böngészőhasználat, adatbázis elérése). Védeni kell a gépet az erőszakos újraindítások vagy feltörési kísérletek ellen is.
Meg kell akadályozni azt is, hogy nem megfelelő tartalmat töltsenek le, installáljanak fel, például egy trójait vagy egy rootkitet.
A felhasználói alkalmazások esetenként elég tág teret kínálnak a nem megfelelő használathoz, korlátozásaink tehát ide is kiterjedhetnek.
Az operációs rendszer szintjén is sok tennivalónk akad, mert ezek gyakorta csak kevés védelemmel településnek, kivéve az eleve biztonságosra tervezett rendszereket, mint amilyen az *OpenBSD*. A hálózati védelmünk gyakorta rendszergazdánk kezében van, otthon azonban – vagy ha mi vagyunk a hálózati rendszergazda – ennek megoldása is ránk hárul.
Védeni érdemes még fájlrendszerünket is, ha úgy véljük, egy esetleges támadó hozzáférhet adathordozóinkhoz, így azokat esetleg könnyedén lemásolhatná vagy módosíthatná. Az indítási folyamatba való esetleges beavatkozás is szóba jöhet. Ha letiltunk bizonyos eszközöket a *BIOS*-ban, úgy ennek védelme sem felesleges. Védelmünk legkülső eleme a hardver és annak elhelyezése, ami már fizikai védelem, ezért ezzel itt nem fogunk foglalkozni.
A biztonsági intézkedések tárgyalását abban a sorrendben tárgyalom, ahogy a gépünk is elindul, tehát a bekapcsolási folyamattól kezdve az operációs rendszeren keresztül a felhasználói alkalmazásokig és az általuk kezelt adatokig.

A BIOS védelme

Ez tulajdonképpen azt szolgálja, hogy illetéktelenek ne módosíthassák a *BIOS* beállításokat (*supervisor password*), illetve csak a jelszó ismeretében töltse be az operációs rendszer (*user password*). Mivel az egyes *BIOS* típusokhoz az interneten megkereshető általános jelszó is, amit a gyártó beleéget a chipbe, vagy pedig a gép szétszedése után a jelszó

törölhető, így ez elég gyenge védelem, viszont arra mindenképpen jó, hogy megakadályozza a kezdők károkozását, a profik számára pedig azt, hogy gyorsan behatoljanak a gépbe.
A *BIOS*-ban letilthatjuk a nem kívánatos külső perifériák használatát – floppy, *CD*/*DVD* meghajtó, sőt *USB* – , amiket arra használhatnak, hogy feltörjék a gépet. Természetesen ha rendszeresen használjuk ezeket a perifériákat, akkor ez nem túl kényelmes megoldás, hiszen minden egyes használat előtt vissza kell állítani ezeket a beállításokat, majd újra beállítani. Megoldás lehet még ezen eszközökről való bootolási lehetőségek tiltása is.
Persze a *BIOS* említett viszonylag egyszerű feltörhetősége miatt a nyilvános és jobban védett helyeken célszerű megoldás a fizikai kontaktust is megszüntetni, tehát kihúzni ezen eszközök adatkábelét, tápkábelét és csak akkor összeállítani, amikor esetleg installálunk valamit. Ez már elég komoly akadály lehet a próbálkozóknak, még a „varázsjelszóval" felszerelkezve is, a gép házának megbontásával remélhetőleg már kevesen élnek.

A bootmanager biztonsága

Egy *UNIX*-szerű rendszer használata esetén célszerű *GRUB*-ot vagy *LILO*-t telepíteni a boot szektorba, amelyek menürendszerrel láthatók el, így képesek többféle operációs rendszert, eltérő verziókat is elindítani. Itt kétféle védelemmel is élhetünk: megakadályozhatjuk, hogy illetéktelenek bármilyen operációs rendszert elindítsanak vagy pedig hogy nem biztonságos módon (például bejelentkezési shell nélküli) rendszert indíthassanak el. Az elsőnél jelszóval kell védenünk a boot managert, a másodiknál pedig nem szabad engednünk az automatikus bejelentkezést az operációs rendszer elindítása után.
Ez utóbbi történhet úgy is, hogy a bootmanager menüjéből eltávolítjuk a *„single"* módú indítási lehetőségeket vagy úgy is, hogy bár ezt engedjük, a *„single"* módban beállítjuk, hogy kötelezően be kelljen jelentkezni.
GRUB jelszó létrehozására kétféle lehetőségünk van: az egyiknél szimpla szöveges módban beírjuk a jelszót a konfigurációs állományba, a másiknál md5 titkosítást alkalmazunk (egy kis „sóval (salt) fűszerezve"). Az *md5*-ről és a *salt*-ról később részletesen fogok írni.
A jelszó titkosítás nélküli tárolásához a következőket írjuk a */boot/grub/menu.lst* fájl elejére:

```
timeout 10
password jelszavam
```

Az első paraméterrel megadjuk, hány másodpercet várakozzon, míg a `default` bejegyzéshez tartozó paraméterekkel elindítja a betöltést.
A második paraméternél a „jelszavam" helyére írjuk be saját jelszavunkat. Természetesen nem túl célszerű ennyire könnyen olvasható módon megadni egy jelszót, ezért titkosíthatjuk azt.
Ehhez adjuk ki a következő parancsot, ami bekéri a titkosítandó jelszót, majd kiírja az eredményt:

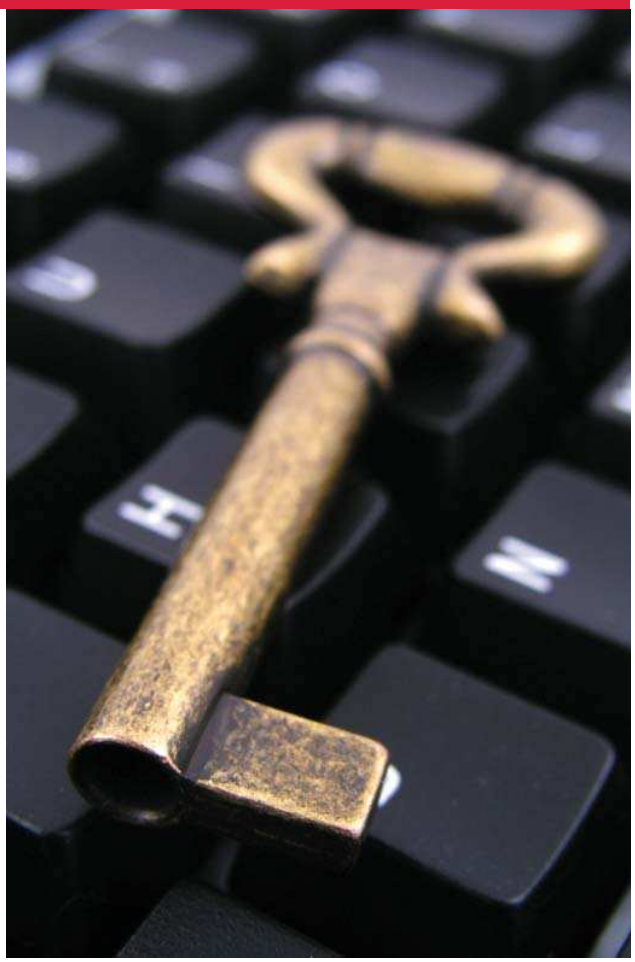
```
grub-md5-crypt
Password: *********
Retype Password: *********
Encrypted: $1$j9fu8/
➥Ha$5JM4n2wZOR.Zuo9iI0vJAS1
```

Másoljuk ki a titkosított karaktersorozatot. A */boot/grub/menu.lst*-be pedig illesszük be:

```
password --md5 $1$j9fu8/
➥Ha$5JM4n2wZOR.Zuo9iI0vJAS1
```

A védeni kívánt menübejegyzésekhez szúrjuk be a `lock` paramétert. A `grub-install` paranccsal pedig véglegesítsük a változásokat.
Legközelebbi indításkor a *GRUB* menüjében a védett elemnél a *P* billentyűt kell lenyomnunk, ekkor beírhatjuk a jelszót, ami feloldja a védelmet.
LILO-nál nem tárolhatjuk titkosítva a jelszót. Két lehetőségünk van: globális és menüpontonkénti védelem.
Globálisnál az */etc/lilo.conf* fájlban a következőket kell megadnunk:

```
password=jelszavam
restricted
delay=10
```



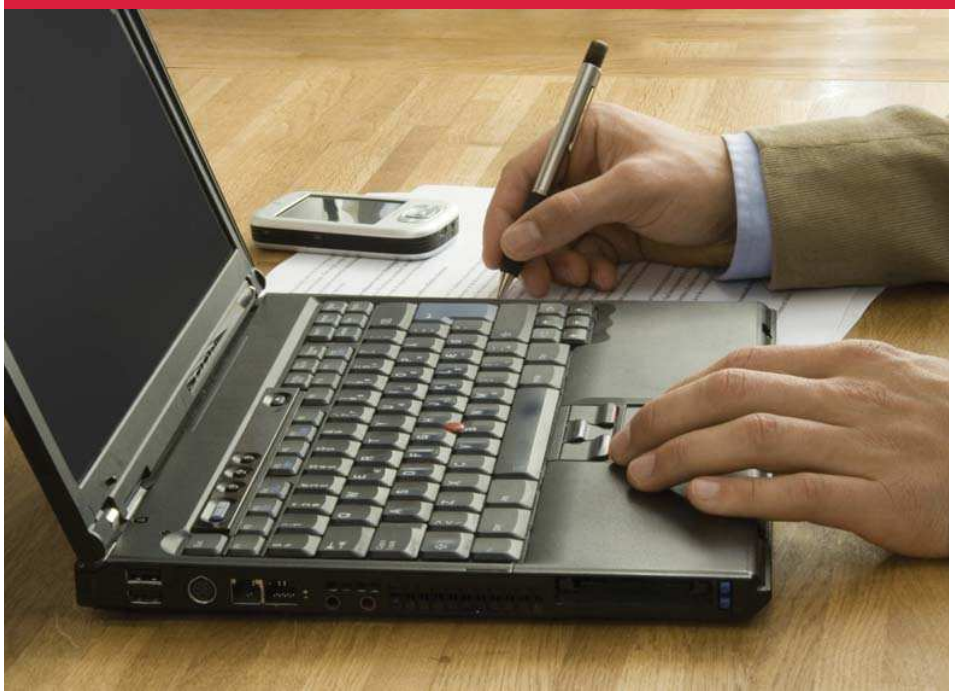
Minden egyes menüpontnál megadhatunk akár eltérő jelszavakat is:

```
image=/boot/kernel
        password=jelszavam
        restricted
```

A változtatás után ne felejtsük el kiadni a lilo parancsot.
A boot manager jelszóvédelme természetesen csak nehezítés támadónk számára. Ha például el tud indítani egy live Linux disztribúciót *CD*-ről, *DVD*-ről, floppyról vagy *USB* diszkről, akkor nem sokat ért ez a védelem, hiszen a boot managerre nincs is szüksége meghajtóink tartalmának eléréséhez. Viszont igen nehézzé tehetjük dolgát, ha ezeket nem képes végrehajtani, mert mondjuk egyik eszköz sem áll a rendelkezésére. Ez esetben már gyakorlatilag csak a merevlemezek ellopásával tudja adatainkat megszerezni.

Védelem a bejelentkezésnél

A bejelentkezés történhet szöveges terminálról, desktop menedzserből, távoli eléréssel, sőt akár



a soros portról is. Mindezeket megfelelően biztosítanunk kell, hogy ne érhessen bennünket meglepetés. Ha megnézzük az */etc/passwd* fájlt, valami ilyen fogad bennünket:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/
➥bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/
➥sync
games:x:5:60:games:/usr/games:/
➥bin/sh
```

Az egyes mezők (kettősponttal elválasztva) a következők:

- bejelentkezési név
- jelszó titkosítva vagy x
- felhasználói azonosító
- csoport azonosító
- felhasználó neve vagy megjegyzés
- felhasználó alapkönyvtára
- felhasználó parancsértelmezője

Ha a jelszó helyén x-et találunk, akkor szerencsére *shadow*, azaz árnyékjelszavakat használunk, amik nem itt, hanem az */etc/shadow* fájlban találhatóak. Fontos, hogy ne itt tároljuk a jelszavakat, mert a *passwd* fájl mások számára is olvasható. Látható, hogy a legtöbb bejegyzés nem valódi felhasználót takar, hanem a rendszerfolyamatok számára kialakított, speciális felhasználói azonosítókat. Ezeknek nem szabad hogy jelszava legyen. De ajánlatos még a „root" felhasználó jelszavát is törölni a *shadow* fájlban:

```
root:*:13043:0:99999:7:::
daemon:*:13043:0:99999:7:::
bin:*:13043:0:99999:7:::
sys:*:13043:0:99999:7:::
sync:*:13043:0:99999:7:::
games:*:13043:0:99999:7:::
```

Ha a *shadow* fájlban csillag vagy felkiáltójel van a jelszó helyén, úgy azzal a felhasználói azonosítóval nem lehet belépni. A létező jelszavak természetesen titkosítva vannak:

```
felhasznalo:$1$O7T8tF83$V.kJdI5
➥P3iNPQjHWy6LpN.:13044:0:
➥99999:7:::
```

A speciális azonosítóknak is általában van alapértelmezett shellje, ami így furcsának tűnik, hiszen „ők" sohasem igényelnek ilyet. A biztonság kedvéért egyesek javasolják, hogy ezeket cseréljük ki `/bin/false`-ra, ami biztosítja, hogy ezen rendszerfolyamatok feltörése által sem lesz képes senki shellt szerezni.

A root jelszavának törlése pedig külön biztonságot ad arra, hogy a próbálkozók ne tudjanak teljes jogosultságot szerezni. Ilyenkor ne felejtsünk el a */etc/sudoers* fájlban saját magunknak minden jogot megadni:

```
felhasznalo  ALL=(ALL) ALL
```

Root jogokkal így a jövőben a `sudo` paranccsal futtathatunk valamit, root shellt pedig a

```
sudo su -
```

paranccsal szerezhetünk.
Amennyiben mégis jónak tartjuk, hogy a root-nak van jelszava, akkor érdemes szabályozni, hogy melyik terminálokról léphet be valaki root-ként. A */etc/securetty* fájlban szabályozhatjuk ezt, itt meg kell adni azoknak a tty terminálóknak a neveit (a /dev nélkül), amelyeknek engedélyezzük a root logint. Érdemes csak az első terminált megadni, a (`tty1`), a többit kikommentezni. Ha nem szeretnénk engedni a soros portról való root belépést, a `ttyS0`-t is kommentezzük ki, ha szerepelne benne:

```
#ttyS0
tty1
#tty2
#tty3
#tty4
...
```

Ezzel megakadályozzuk, hogy egyszerre több terminálon is be lehessen jelentkezni root-ként, így csak az első terminálra kell figyelnünk, ha otthagyjuk a gépet, hogy kijelentkezzünk a root shellből.
Ennél finomabban szabályozhatjuk a bejelentkezéseket a *PAM (Pluggable Authentication Modules)* segítségével. Az egyes alkalmazások autentikációjához különféle *pam* modulokat alkalmazhatunk. Ezek betöltését szabályozhatjuk a */etc/pam.d* könyvtárban lévő konfigurációs fájlokkal. Szöveges felületen, helyi terminálba a `login` alkalmazással tudunk bejelentkezni, ezért ehhez a *login* fájl kell szerkesztenünk. Keressük meg a következő sort az `/etc/pam.d/login` fájlban:

```
account required pam_access.so
```

és töröljük az elejéről a # jelet (vagy kézileg is beírhatjuk). A bejelentkezések szabályozása azután az */etc/security/access.conf* fájlban történik. Az egyes sorok formátuma a következő:

```
engedélyezés:felhasználók:
➥honnan
```

Az `engedélyezés` mező + vagy - lehet attól függően, hogy megadni vagy elvenni akarjuk-e a felhasználóktól, akik a „honnan” mezőben megadott helyről jelentkeznének be a bejelentkezés jogát. Használható az `ALL` kifejezés a mezőnevekben, ez az összes jogosultságot vagy az összes felhasználót, illetve az összes helyet jelentheti attól függően, melyik mezőben alkalmazzuk. Működik még az `EXCEPT` is, amivel kivételeket definiálhatunk. A `honnan` paraméter értéke lehet `LOCAL` is, ez a nem távoli elérést jelenti, azaz egy helyi konzolt.
Például tiltsunk meg minden helyi bejelentkezést a root kivételével az összes helyi terminálra:

```
-:ALL EXCEPT root:LOCAL
```

A távoli bejelentkezésekhez pedig csak a `csaba` nevű felhasználót engedjük meg:

```
-:ALL EXCEPT csaba:ALL EXCEPT
➥LOCAL
```

Egész bonyolult szabályrendszert is összeállíthatunk így.
Aki *SSH*-val jelentkezik be, az bizonyára észlelni fogja, hogy ezek a szabályok az *SSH*-ra nem érvényesek, ugyanis külön *SSH PAM* (*libpam_ssh* csomagban) modulunk van (*pam_ssh.so*). Ha az előbbihez hasonlóan akarjuk az ssh bejelentkezéseket is szabályozni a */etc/security/acces.conf*-ban, akkor (persze a *pam_ssh.so* modul megléte mellett) írjuk az alábbi sort a */etc/pam.d/ssh* fájlba:

```
auth required pam_access.so
```

És máris ugyanaz a szabályrendszer lesz érvényben az *SSH*-t használó felhasználókra is, mint a helyi bejelentkezésekre.

Az SSH-t saját konfigurációs fájljában is tudjuk némileg szabályozni (*/etc/ssh/sshd_config*). Nézzük az *sshd_config* fontos beállításait!
A `PermitRootLogin no` sorban megtilthatjuk (vagy engedélyezhetjük) hogy root-ként bárki *SSH*-n keresztül bejelentkezzen. Ajánlott ezt megtiltani. Természetesen a `PermitEmptyPasswords no` sornál is a tiltás az ésszerű: ne lehessen jelszó nélkül belépni.
A hálózati bejelentkezésekkel (és az *SSH*-val) később még részletesen foglalkozunk. Most térjünk vissza a helyi géphez és nézzük meg, milyen lehetőségünk adódik még a bejelentkezés korlátozására.
Az */etc/login.defs* fájlban adhatunk meg különböző paramétereket. Ezekből álljon itt néhány.

```
FAIL_DELAY   3
```

Ezzel szabályozzuk, hogy hány másodpercig várakozik a hibás bejelentkezés után. Nyugodtan emeljük meg ezt az értéket, a próbálkozóknak hadd menjen el a kedve a jelszók beírogatásától!

```
FAILLOG_ENAB yes
```


Természetesen engedélyezzük a hibás bejelentkezések naplózását (a */var/log/faillog* fájlba).

```
PASS_MAX_DAYS     90
PASS_MIN_DAYS     3
PASS_WARN_AGE     7
```

Ezekkel a jelszavak élettartamát szabályozhatjuk (`PASS_MAX_DAYS`: hány napig érvényesek, `PASS_MIN_DAYS`: hány napot engedélyezünk még a jelszó lejárta után, `PASS_WARN_AGE`: hány nappal jelezzen a jelszó lejárta előtt).
Természetesen a grafikus bejelentkezésnél – *gdm*, *kdm* – is van mit beállítanunk. *gdm*-nél a */etc/gdm/gdm.conf* fájlt kell szerkesztenünk.
Az `AutomaticLoginEnable` értéke feltétlen `false` legyen, mert egyébként aki bekapcsolja gépünket, mindjárt az `AutomaticLogin` változó által beállított felhasználóként léphet be.
A biztonsági beállítások a `[security]` szekcióban vannak elkülönítve. Nézzük meg a legfontosabbakat!

- `AllowRoot`: Ajánlott, hogy `false` legyen az értéke, hogy még véletlenül se léphessünk be rootként, így ha valaki ezután odaülne gépünkhöz, korlátlan jogosultsággal garázdálkodhasson.
- `AllowRemoteRoot`: `False`, hogy távolból se jelentkezhessen be senki root-ként.
- `AllowRemoteAutoLogin`: Természetesen távolból se lehessen automatikus login – `false`.
- `RelaxPermissions`: értéke 0, ha csak a tulajdonos, 1 ha a csoport tag is és 2, ha bárki írhatja a fájlokat a gdm által. Hacsak nincs rá egyéb okunk, ennek értéke legyen 0.
- `CheckDirOwner`: Az előző változóhoz kapcsolódik, ellenőrizze-e a home könyvtár tulajdonosát. Ha `true` az értéke, akkor ellenőriz. Személyes tapasztalatom az, hogy ha az előző változó értéke 0, már akkor sem indul el a grafikus felület, amikor megváltozik a home könyvtár tulajdonosa.

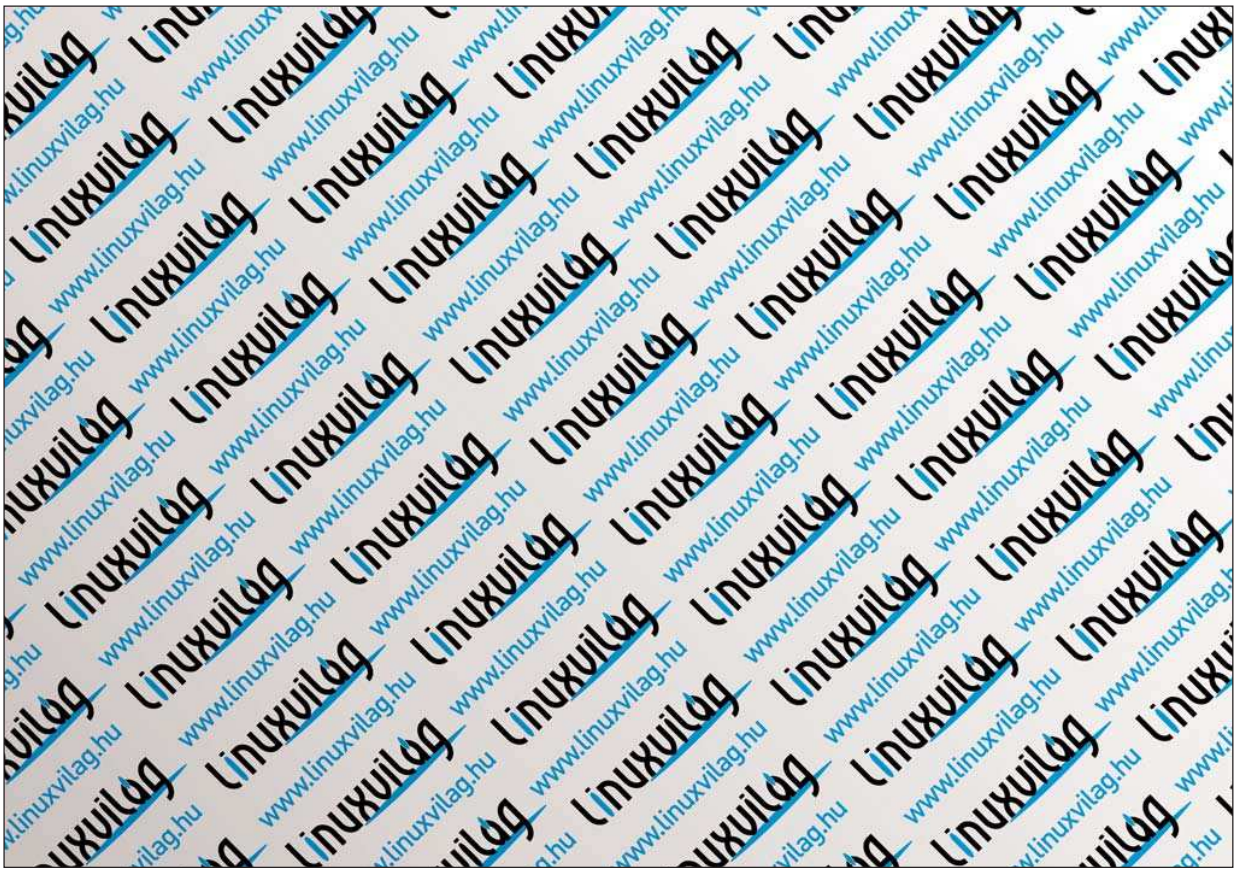
A bejelentkezés védelmében még nagyon sok lehetőséget említhetnék, de most csak a legalapvetőbb intézkedésekre jutott hely, így nem szóltam a különböző fizikai eszközökről sem, amelyek a biztonságos azonosítást szolgálják, valamint a *PAM* rendszerről is sokat lehetne még írni. Akinek ez kevés, a témáról nagyon sok és jó információt találhat az interneten.

A következő számban a fájlrendszer biztonságáról lesz szó.

Molnár Norbert
(molnar.norbert@gmail.com)

35 éves, rendszergazdaként dolgozik, 5 éve foglalkozik Linuxszal. Főként a szabad szoftverek és a számítógépes biztonság érdekli.

Budapesten él feleségével és 2 éves kisfiával. Hobbija a csillagászat és a filozófia – lehetőleg jó vörösbor mellett.



Grsecurity – Az én házam, az én váram

A Linux kernelen alapuló operációs rendszerek sokféle módot eszeltek ki a rendszer biztonságossá tételére, különféle tűzfalakat tartalmaznak, igyekeznek azonnal foltozni a hibás csomagokat, erre aztán egyre eszesebb csomagkezelőket gyártottak, egyszóval igyekeztek a rabló-pandúr versenyfutásban nem nagyon lemaradni. Most itt a kernelszintű biztonsági megoldásokról lesz szó, hogy miképpen tudjuk a Linux elemi lehetőségein túl teljességgel uralni rendszerünket.

A birodalom helyzete

Elsősorban most a kiszolgálói feladatot ellátó rendszereket tárgyaljuk, de az itt leírtak alkalmazhatók minden változtatás nélkül munkaállomásnak használt környezetekre is. Mi is az a birodalom, ahol várat kap egy frissen telepített ***Linux*** teljhatalmú ura? Nem más ez, mint az internet, ahol a viszonyok elég puskaporosak. Ha az offline világban valahol autókat gyújtogatnak fel, első oldalas hír lesz, ha viszont egy szervert vernek szét virtuálisan (ahol a befektetett munkaóra esetleg többszörösen meghaladja egy autó árát), akkor esetleg valamelyik internetes portál lehozza hírként. A virtuális vandalizmus mindennapos esemény. Így aztán a kacsalábon forgó várunkhoz nem árt, ha védőfal és testőrség is jár. Ez minden internetre kapcsolt számítógépre igaz, azonban azokra, melyek kiszolgálói alkalmazásokat is futtatnak, különösen érvényes. A pingvines rendszerrel pedig kimondottan előszeretettel valósítanak meg efféle feladatokat. A ***Linuxnak*** vannak ugyan elemi megoldásai a biztonság fokozására, de szerencsére ezeken túlmutató eszközök is rendelkezésre állnak. Itt van mindjárt a ***grsecurity***. Ha a váras analógiát még nem unja a kedves olvasó, akkor úgyszólván a ***grsecurity*** elsősorban arról gondoskodik, hogy a várfalakon belül minden rendben legyen.

Grsecurity dióhéjban

A ***grsecurity*** egy biztonsági lehetőségeket megvalósító kernelfolt (***patch***). Használatához új kernelt kell üzembe helyezni, cserébe sok új eszköz pottyan az ölünkbe. Kivédjük a telepítésével a legtöbb buffer overflow-n alapuló támadást, megerősítjük a ***TCP/IP***-t és kapunk egy nagyon finoman hangolható hozzáférés-vezérlést (***ACL***). Ezen túl persze számtalan változás lép életbe a kernel színfalai mögött, hogy nagyobb biztonságban legyünk. Az ***ACL*** rendszerrel folyamatokra lebontva meg lehet mondani, hogy ki mit tehet a rendszerben, az erőforrásokkal és a hálózaton. Például szabályozhatjuk folyamatonként a processzor és memóriahasználatot. Ahhoz hogy egy kernelpatchet vezérelni tudjunk, szükség van egy ***felhasználói térben (userland)*** futó alkalmazásra is. Ez jelen esetben a ***gradm***. A módosított kernellel és a ***gradm*** segítségével fogjuk egy példán keresztül megnézni, hogy miben is képes a ***grsecurity*** támogatni bennünket a biztonságért folytatott sziszifuszi küzdelemben. A kipróbálást gyorsítják azok a virtualizációs technikák, melyekről a szeptemberi számban kiváló cikkek jelentek meg (***qemu***, ***vmware***). Az itt leírtak előfeltétele egy tesztrendszer, akár valódi, akár egy virtuális számítógépen. Ez a tesztrendszer a ***Debian GNU/Linux 3.1*** volt, de más rendszerekre is átvihetők az alkalmazott megoldások. Lehetőség szerint disztribúciófüggetlen lépéseket teszünk.

A tesztkörnyezet

Tegyük fel, hogy van egy internetes kiszolgálónk, amin egyetlen ***Apache 2*** (a továbbiakban az egyszerűség kedvéért

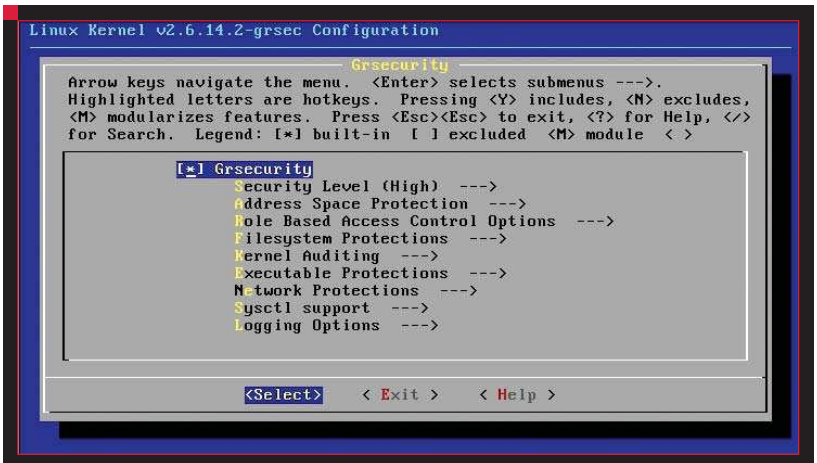
csak *Apache*) webkiszolgáló fut.
Ez a kiszolgáló képes *PHP* alkalmazásokat futtatni, de mivel a webmesteri feladatokat nem mi látjuk el, kiemelten fontos, hogy az Apache a legminimálisabb jogokkal bírjon. Hiszen ekkor, ha a webmester (aki az *Apache* által kiszolgált fájlokat/programokat gondozza) rosszindulatú kódot tölt fel, akkor a *www-data* felhasználó (amivel a webszerver fut), jogosultsága szerinti pusztítást tud végezni.
Ez ellen lehet védekezni, mondjuk *chroot* használatával, de az ellen például nem, ha a kiszolgált weboldalunk hallatlanul népszerű és például áldozatul esik a *Slashdot* effektusnak. Ekkor a gép minden erőforrását leköti a www szolgáltatás és esetleg egyéb kritikus alkalmazások leállhatnak.
Számtalan módon tudjuk szabályozni a folyamataink jogait a fájlrendszerhez, a hálózathoz és a gép erőforrásaihoz. A már előbb említett *Debian 3.1*-re az alaptelepítésen kívül csupán a *libncurses5* és a *libncurses5*, az *apache2* és a *php4* lett feltelepítve. Az utolsó kettő a feladathoz, az első kettő pedig a kényelmes majdani kernelfordításhoz (`make menuconfig` a `make config` helyett). Az *Apache 2* konfigurálva lett, hogy ténylegesen végrehajtsa a *.php* kiterjesztésű fájlokat. Ezek után felkerült a rendszerre a 2.6.14.2-es kernelforrás és az ahhoz való 2.1.7-es *grsecurity* kernelpatchnek és a *gradmnak* a fájljai.
Ezeknek a fájloknak a nevei letöltési címekkel a cikk végén megtalálható. Itt kapcsolódik be a cikk a történetbe. A cél egy olyan webkiszolgáló létrehozása, mely bár a *www-data* jogaival fut, egészen beszűkített jogai vannak, pontosan a feladathoz szabva. Igaz, hogy az *Apache*-hoz és a *PHP*-hez is számtalan olyan kiegészítés elérhető, mellyel a fentiek többé-kevésbé elérhetőek, viszont a most következők bármelyik alkalmazásra átvihetőek. Lássunk munkához!

Üzembe helyezés

A

```
gunzip grsecurity_file
```

utasítás után a kitömörített kernelforrásra a `patch` parancs kiadásával lehet a *grsecurityt* telepíteni. Lépjünk be abba a könyvtárba, ahol



1. ábra A kernel grsecurity-specifikus beállításai

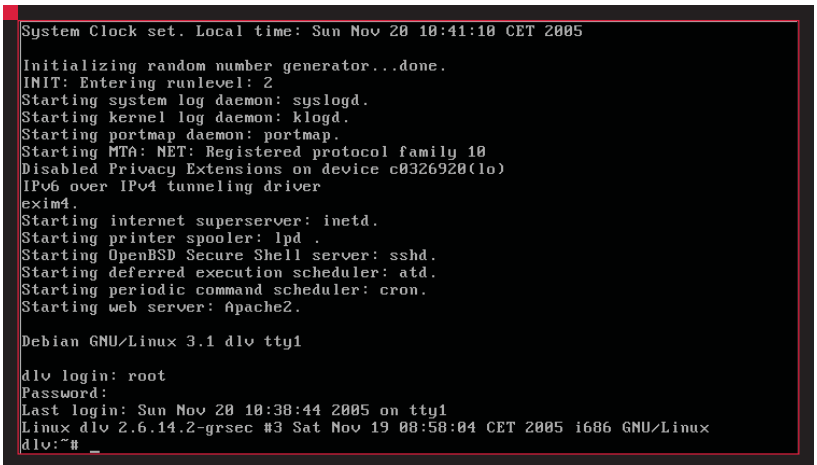
a *linux-2.6.14-2* könyvtár van, aztán hajtsuk végre:

```
patch -p0 < grsecurity_file
```

A `grsecurity_file` helyére a *grsecurity* kernelpatch teljes elérési útját írjuk. Ezután belépve a kernelforrás könyvtárába a

```
make menuconfig
```

következik. Bár a kernelfordítás minden csínja nem tartozik szorosan a cikk témájához, azért néhány dologgal érdemes tisztában lenni. Működő, éles rendszeren a kernel lecserélése nem várt problémákhoz vezethet, rosszabb esetben az új kernellel el sem indul a rendszer. A kernel beállításánál csak azzal foglalkozunk, ami ténylegesen pluszt jelent egy szokásos újrafordításhoz képest. Ha nem így tennénk, akkor a cikk címe „Hogyan fordítsunk kernelt?” lenne.
Mivel a *grsecurity* patchet alkalmaztuk, a `make menuconfig` által előcsalogatott menü kiegészül egy olyan alponttal, hogy *Security options* / *Grsecurity*. Itt kezdjük a munkát, miután a kernelforgatást saját szájunk íze szerint beállítottuk, ahogy egyébként is tettük volna. Az első opció a *Security Level*. A *grsecurity* nagyon összetett rendszer, akár minden elemét beállíthatjuk egyenként, vagy választunk az előre definiált szintek közül. A gyors beállítás érdekében egyelőre válasszuk a *High*-t, de azért nézzük át, miket lehet állítani! A *Role Based Access Control* alatt megadhatjuk, hogy hány elrontott jelszó után zárjon ki a rendszer, valamint hogy hány percig tartson a kitiltás, ha elérjük az előzőekben megadott küszöböt. A *Filesystem Protections* alatt le lehet tiltani, hogy a felhasználók lássák a */proc*-ban a nem hozzájuk tartozó folyamatokat és azok adatait, valamint szigorítani lehet a *chroot*-os környezeteket. A *Kernel Auditingban* számtalan naplózó/ellenőrző funkció kapcsolható be. Az *Executable Protections* / *Trusted Path Execution* által le tudjuk tiltani, hogy egyes felhasználók olyan programokat futtassanak, amik nem root tulajdonosú és csak root által írható könyvtárakban vannak. Ezzel nehezíthető rosszindulatú kód végrehajtása. Természetesen nem tökéletes a megoldás, hiszen egy nem megbízható felhasználó által felmásolt gonosz.pl-t a perl ezek után is vígan le fog futtatni. A *Networking Protections* alatt, ha a *High* biztonsági szintet választottuk, akkor csupán egy aktív lehetőség van, mégpedig a *socketek* korlátozására csoportazonosítók (*GID*) szerint. Ezen túlmenően is sok opció van, amire még érdemes vetni egy pillantást, az a *PaX*. Ez egy menüszinten található a *GrSecurityval*, a *grsecurity* patch-csel együtt felkerül. A *PaX* alapgondolata, hogy az általunk futtatott szoftverek hibásak, ezért ki kell védeni operációs rendszer szinten a gyakori hibalehetőségeket. Ide tartoznak a buffer overflow (túlcsordulás) hibák is. Többek közt ezekre gyógyír a *PaX*. Miután a menüpontokon átrágtuk magunkat (az egyes opciókon állva a *Help*-et választva egész kis dokumentációt



2. ábra Elindult a Grsecurityval felvértezett új rendszer

lehet előcsalni, melyek jól érthetőek), nem kell mást tenni, mint elmenteni a beállításokat és elindítani a fordítást. Ha virtuális gépen dolgozunk, akkor a kernelfordítás ideje alatt, géptől függően egy kávé, tea, sütemény vagy az egész e havi *Linuxvilág* áttanulmányozása is belefér. Persze nem kötelező virtuális gépen fordítani, megtehetjük, hogy a valódi operációs rendszer fordítójával lefordított kernelt felmásoljuk a virtuális hálózaton keresztül. A kernel a

```
make
make modules_install
```

parancsokkal települ, ezek után *Debian 3.1* esetén a */boot* alá másoljuk be az *arch/i386/boot/bzImage*-t (a kernelforráshoz képesti relatív útvonal). A *LILO*-t illetve a *GRUB*-ot módosítsuk úgy, hogy az új kernelt is ajánlja fel rendszerindításnál. Ne vegyük ki a régit, hiszen nem biztos, hogy olyan kernel jött létre, ami el is indul az adott gépen. Persze ha virtuális géppel dolgozunk, az ebben rejlő kockázat elég csekély. Miután végeztünk ezzel, jó esetben az újraindítás után semmi változást nem észlelünk, éppen úgy be tudunk jelentkezni, mint eddig.

A rendszabályok életbe léptetése

Az előzőekben megtett lépésekkel már sokkal előrébb jár a rendszerünk biztonságosságban, azonban az egyes programok személyre szabott megregulázása csak most fog következni. Először is ki kell tömörítenünk a *gradm* csomagot, aztán a *gradm2* könyvtárba belépve a

```
make
make install
```

parancsokkal fel tudjuk telepíteni az adminisztrációs felületet. A folyamat közben meg kell adnunk egy jelszót, mely a *grsecurity* beállításait védi. Miután ezzel végeztünk, újabb jelszó beállítására van szükségünk, hogy bekapcsolhassuk a rendszabályainkat. Ezt a

```
gradm -P admin
```

paranccsal tehetjük meg. Ezután ha a

```
gradm -E
```

utasítást kiadjuk, akkor életbe lépnek a gyári *grsecurity* korlátozások, melyek elég szigorúak. Például az *Apache*-ot nem lőhetjük le még rootként sem és az */etc/grsec/* könyvtárhoz sem férhetünk hozzá, ahol a beállítások lakoznak. Ügyes! Ha meg akarunk szabadulni az őrmestertől, aki a *grsecurity* képében megszállta a rendszert, a

```
gradm -D
```

kiadásával és a jelszó beírásával megtehetjük. Jó tudni, hogy engedélyezett *RBAC* (ez a szabályozó rendszer neve) mellett a gyári beállításokkal a `reboot` parancs sem működik rendesen, mert fájlrendszereket leválasztani sincs jogunk például. Talán már látszik, hogy a *grsecuritynek* megadott jelszavak mennyire fontosak. Egyfelől ha elvesztettük, és az *RBAC* engedélyezve van, akkor nagy bajban vagyunk, másrészt, ha kitalálható, akkor majdnem ott vagyunk, mintha fel se raktuk volna a *grsecurityt*. A teszt kedvéért készült két apró *PHP* szkript: az egyik a paraméterül kapott értéket beírja a */var/www/adat* fájlba, a másik pedig a */tmp/zagyva* fájlba ír véletlen számokat a végtelenségig. Nyilvánvaló, hogy az első szkripttel nincs semmi baj, de a másodikat jobb lenne letiltani. Egyrészről megtehetjük, hogy csak azokat a helyeket engedjük írni a webszervernek, amit majd ténylegesen kell neki, másrészről pedig letilthatjuk, hogy a */var/www* könyvtárban lévő dolgok közül csak azokat olvashassa el, amik ténylegesen rá tartoznak. Akár megtehetjük, hogy azt is, hogy az egész fájlrendszert csak olvashatónak lássa az *Apache* és mondjuk a *MySQL* socketet engedélyezni neki. A telepítés után a tesztrendszer /var/www könyvtárába bekerült a két bugyuta *PHP* szkript (*1. Lista*). A *grsecurity* rendszabályok nélkül és a *grsecurity* alapértelmezett rendszabályaival mindkét szkript kifogástalanul működik. A *rossz.php* 30 másodperc múlva leáll, mivel a *PHP*-ban az az alapértelmezett maximális futási idő, de addig kíméletlenül próbálja megtölteni(nem túl okos módszerrel) a */tmp* lemezrészt. Ezután a gyári */etc/grsec/policy* fájl a következő bejegyzéssel egészült ki:

```
subject /usr/sbin/apache2
        / r
        /var/www x
        /var/www/jo.php r
        /var/www/adat rw
        +CAP_NET_BIND_SERVICE
        /etc/ x
        /etc/apache2 rx
        RES_NPROC 2 3
```

Mit is mondanak ezek a sorok? A / könyvtár és minden ami alatta van (a tulajdonságok öröklődnek) csak olvasható az `/usr/sbin/apache2` számára, kivéve a `/var/www`, ebbe a könyvtárba beléphet ugyan, de nem olvashatja. A `/var/www/jo.php`-t olvashatja, de nem írhatja például. A `/var/www/adat`-ot írhatja és olvashatja. A `+CAP_NET_BIND_SERVICE` sor

1. Lista Egy jó és egy kevésbé jó PHP szkript

jo.php

```
<?
$fajl = fopen("adat", "w");
fwrite($fajl, $_GET["adat"]);
fclose($fajl);
?>
```

rossz.php

```
<?
$fajl = fopen("/tmp/zagyva",
➥w);
while(1)
{
  fwrite($fajl, rand(-500,
  ➥500));
  fflush($fajl);
}
fclose($fajl);
?>
```

mondja meg, hogy a folyamatnak lehetősége van a hálózaton szolgáltatásként üzemelni: lefoglalhat egy portot, címet magának. Az `/etc x` azért szükséges, hogy ne tudja olvasni a folyamat az */etc* alatt lévő fájlokat, hiszen ott lakozik például az */etc/passwd*, mely nem lenne jó, ha a tréfás kedvű webmester miatt publikussá válna. Viszont az */etc* alatt vannak az ***Apache 2*** beállítófájlok, ezért a következő sorban mindjárt megmondjuk, hogy ott szabad a gazda. Végül, mivel egy kis forgalmú teszthelyről van szó, a maximálisan elindítható ***Apache*** folyamatok számát 3-ban határozzuk meg. Minden ilyen `RES_VALAMI` korlátozás két számot vár el. Egy puha (első szám) és egy kemény korlátot. A puha korlát elérésekor a folyamat kiadhatja a `setrlimit` hívást, hogy többet szeretne az adott erőforrásból, amit egészen a kemény korlátig növelhet. A példából látszik egy bejegyzés általános szerkezete, annyit érdemes még hozzátenni, hogy a `subject` vonatkozhat könyvtárra és végrehajtható állományra egyaránt. Az */etc*-re vonatkozó bejegyzésnek következtében, ha a webmesterünk gonosz tréfát akarna velünk űzni, akkor a `print_r (file ("/etc/passwd"));` sorral semmire sem megy, hiszen az ***Apache*** ezt nem olvashatja. Egyébként is szerencsés, hogy a rendszerbeállításokat olvasni sem tudja egy ***PHP*** szkript. Feltűnhet, hogy az Apache most a saját naplóit sem tudja írni. Ez így igaz, a naplózási elérési utak mindenütt */dev/null*-ra lettek állítva. Valódi webkiszolgáló alkalmazásnál a naplózás elengedhetetlen, így megírhatjuk azt a sort is, ami engedélyezi, hogy a */var/log/apache2/access.log* illetve *error.log* írható legyen a folyamatnak. Jelen pillanatban a folyamatban ott tartunk, hogy sikerült a *rossz.php*-t működésképtelenné tenni, valamint a *jo.php* is csak a */var/www/adat* fájlba tud írni, sehova máshova.

A szabályok létrehozása nagy munka összetett alkalmazás esetén, ezért a ***gradm*** képes helyettünk összeállítani szabályokat! A

```
gradm -F -L /root/tanul
```

élesíti a tanuló módot és az eredményt a `/root/tanul` fájlba rakja. Ide azok az események fognak kerülni, melyeket a jelenlegi szabályrendszer megtiltott volna. Így aztán a parancsot kiadva érdemes végrehajtani az összes műveletet, futtatni az összes olyan programot, amire szükség van. De vigyázat, a szolgáltatások leállítása/elindítása nem ide tartozik, hiszen azt feltételezzük, hogy a támadó már rendszergazdai jogokkal bír a rendszer felett és éppen azon ügyködünk, hogy minél kevesebb kárt tehessen. Miután végeztünk a tevékenységekkel, adjuk ki a

```
gradm -F -L /root/tanul -O
➥/etc/grsec/acl
```

parancsot, így a ***gradm*** legyártja a naplókból a nekünk szükséges szabályokat. Amit végrehajtottunk, azokat a későbbiekben is futtatni tudjuk így. Nincs más teendő ezután, mint élesíteni az új szabályokat. Ha kiadjuk a

```
gradm -E
```

parancsot, akkor kapunk egy valamivel biztonságosabb Apache kiszolgálót. Be lehet állítani azt is, hogy a `gradm -E` már a futási szint részeként elinduljon. Az engedélyezéshez nem kell jelszó, így le kell gyártani a disztribúciónak megfelelő indítószkriptet, mely a `gradm -E` parancsot kiadja a betöltési folyamat azon pontján, miután a szigorítások nem akadályozzák a rendes üzemet. A cikkben beállítottakat (alapbeállítások és az ***Apache***-ra vonatkozó sorok) nem lehet a rendszerindítás során alkalmazni, a webkiszolgáló el sem indul úgy. Látható, hogy a legnagyobb munka a ***grsecurityval*** egy alkalmas szabályrendszer felállítása, amely aztán jobb esetben megvédi a fontos rendszert. Nem csodaszerről van szó: az alattomos szoftverhibákkal a továbbiakban is számolni kell, az emberi tényező pedig kikerülhetetlen. Ha rossz szabályrendszert dolgozunk ki vagy a ***gradm***-hoz használt jelszó gyenge, egyezik a root jelszóval vagy a monitor oldalára van felírva, akkor hiába dolgoztunk.

A részletes dokumentáció megtalálható a ➲ ***www.grsecurity.net*** oldalon a ***Papers*** menü alatt. A ***grsecurity*** most már a várunk őrzője, hatékonysága csak rajtunk múlik! Találjuk meg az arany középutat, amely a használhatatlan, de biztonságos (vö. kihúzott ***UTP*** csatlakozó) és az átjáróház (vö. ***open-relay*** levelezőszerver) között van valahol. Ennek feltérképezése végképp egyedi, minden rendszergazda saját környezetében kerülhet egyre közelebb az ***aurea mediocritashoz***.

Novák Áron
(aaron@szentimre.hu)
BME-VIK-es gólya, műkedvelő rendszergazda. Jelenleg leginkább a NetBeans-szel és mindenféle hordozható eszközzel foglalkozik, legalábbis mindazokkal amelyeket meg lehet szólaltatni Linux alatt.

KAPCSOLÓDÓ CÍMEK

A szükséges fájlok letölthetők a következő helyekről:

- ➲ http://www.hu.kernel.org/pub/linux/kernel/v2.6/linux-2.6.14.2.tar.bz2
- ➲ http://www.grsecurity.net/grsecurity-2.1.7-2.6.14.2-200511150641.patch.gz
- ➲ http://www.grsecurity.net/gradm-2.1.7-200511041858.tar.gz

Digitális őrszem – IPTraf

Az IPTraf egy nagyon kényelmesen és egyszerűen használható hálózat-elemző program, amely segítséget nyújt a hálózatba kötött gépek adatforgalmának közvetlen, vagy közvetett figyelésére.

Az *IPTraf* telepítése nagyon egyszerű. A hardverkövetelménye szerény, ezért gyakorlatilag az összes ma használatos gépen képes működni, amiben van legalább 16 MBájt memória. Szoftverkövetelményére sem lehet panaszunk, hiszen legalább 2.2-es kernelt követel magának, de természetesen működik a 2.4-es és a 2.6-os sorozattal is. A fordításhoz sincs szükség különleges alkalmazás telepítésére, arra ügyeljünk, hogy az ***ncurses*** a fejlesztői eszközökkel együtt telepítve legyen.

A programot legegyszerűbben az ➲ *ftp://iptraf.seul.org/pub/iptraf* címről tölthetjük le ***tar.gz*** formátumban. A fájlnévben szerepel a verziószám, ami a cikk írásakor a 2.7.0. Az ***FTP*** helyen megtalálhatjuk a már lefordított bináris fájlt is, amely az ***iptraf-2.7.0.bin.i386.tar.gz*** nevet viseli. A letöltés biztosan nem vesz igénybe hosszú időt, hiszen alig több, mint 350 kB a mérete a fájloknak. Sok Linux terjesztésben is megtalálható már, ezért azt is leellenőrizhetjük, hiszen ilyen esetben a telepítés sokkal egyszerűbb lesz.

Miután letöltöttük, ki kell csomagolni egy ideiglenes könyvtárba, amihez használjuk a

```
tar -xzvf iptraf-2.7.0.tar.gz
```

illetve a már lefordított fájl esetében a

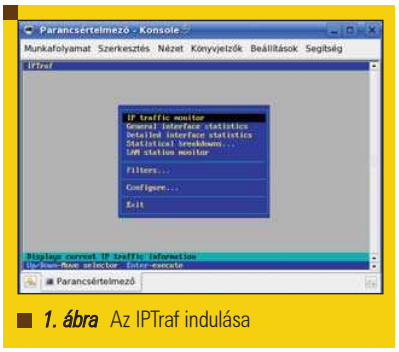
```
tar -xzvf iptraf-
➥2.7.0.bin.i386.tar.gz
```

parancsot.

A kicsomagolás után következhet a telepítés, amely ugyanúgy zajlik, mind a forrás-, mind a lefordított program esetén, csak a háttérben zajló folyamatok lesznek mások. Lépjünk be a könyvtárba, amit a tar létrehozott, majd root jogosultságokkal (ha nem root néven lépnénk, be, akkor használjuk a su parancsot) adjuk ki a `./Setup` parancsot. Ez a parancsfájl felismeri, hogy bináris, vagy forrás állományból dolgozunk és ez utóbbi esetben automatikusan lefordítja a programot. A telepítés tulajdonképpen a létrejött parancsok megfelelő helyre másolását jelenti. Arra figyeljünk, hogy a fordítás rendeben lefusson, hiszen a parancsfájl nem ír ki semmilyen hibaüzenetet. Ezért menjünk vissza a terminál ablakban és ellenőrizzük, hogy nincs hibát tartalmazó sor. A telepítés alapértelmezett könyvtára a */usr/local/bin*. Persze ez eltérhet, ha nem a fent említett módon telepítjük, hanem a ***Linux*** terjesztésünk csomagját használjuk. A ***SuSE Linux*** alatt például a */usr/sbin* könyvtárba kerül a futtatható fájl.

Miután végeztünk a telepítéssel, máris elindíthatjuk a programot, amihez egyszerűen rendszergazdaként adjuk ki az `iptraf` parancsot. Meg kell jelennie a az *IPTraf* köszöntő képernyőjének amiből bármilyen billentyű lenyomásával átléphetünk a főmenübe (*1. ábra*).

Az *IPTraf* programnak szüksége van a ***terminfo*** adatbázisra, amit a */usr/share/terminfo* útvonalon keres. Ha `Error opening terminal` hibaüzenetet kapunk, amikor megpróbáljuk elindítani a programot, akkor valószínűleg az adatbázis más útvonalon található. Ilyen esetben keressük meg, hogy melyik könyvtárban van, majd adjuk ki a



1. ábra Az IPTraf indulása

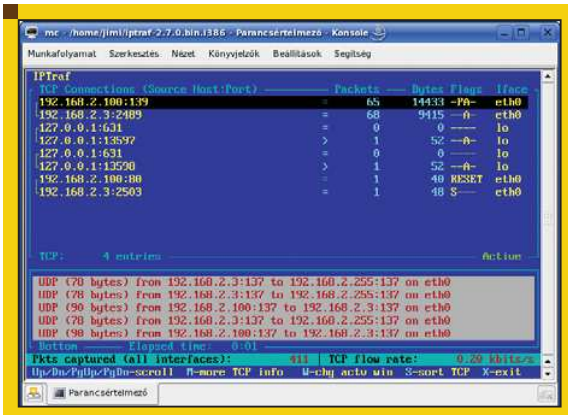
```
TERMINFO=/usr/lib/terminfo
```

majd az

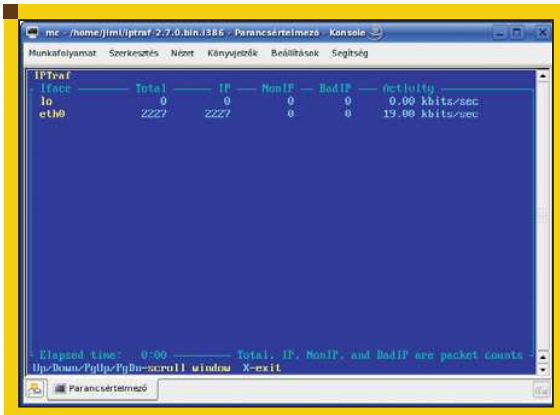
```
export TERMINFO
```

parancsokat. Persze az elérési útvonalat ne felejtsük el átírni. Ahhoz, hogy ne kelljen minden újraindítás után begépelni ezeket a parancsokat, berakhatjuk őket a *~/.profile* fájlba, ami minden bejelentkezésünkkor fog futni. Használhatjuk a rendszer */etc/profile* fájlját is, ugyanis ez minden operációs rendszer betöltődéskor le fog futni. Ha bonyolultnak találjuk ezt a műveletsorozatot, akkor hozzunk létre a könyvtárra egy linket az alábbi paranccsal:

```
ln -s /usr/lib/terminfo
➥/usr/share/terminfo
```



2. ábra Az IP forgalom figyelése



3. ábra Általános csatoló statisztika

Persze ne felejtsük el átírni a megfelelő könyvtárneveket. A program indulása után a menüből tudjuk kiválasztani, hogy az öt üzemmód közül melyiket szeretnénk használni. Az üzemmódok a következők:

IP forgalom figyelése (IP traffic monitor)

Ebben az üzemmódban a kiválasztott csatolón folyó *TCP* adatforgalmat vehetjük szemügyre. Az ablak két részből áll, a felső részben láthatjuk a *TCP* kapcsolatok legfontosabb adatait: a forrás és a cél címét, valamint a szolgáltatás portszámát, a csomagok számát (*Packets*), méretét (*Bytes*), a kapcsolatnál használt állapotjelző biteket (*Flags*), valamint hogy a kapcsolat melyik csatolón keresztül jött létre. Az állapotjelző bitek meghatározzák, hogy az éppen feldolgozott keretek milyen állapotban vannak.

Ezt az üzemmódot vehetjük szemügyre a *2. ábrán*. A címek mindig két részből, egy célból és egy forrásból állnak, amelyek között az összetartozást egy kapocs mutatja. A listában fel-le kurzormozgató billentyűkkel lépkedhetünk, így az ablaknak az a tartalma is elérhető, amely nem fér el a képernyőn. Az *M* billentyű lenyomásával a kapcsolat csomag- és az ablakméretét is ellenőrizhetjük. Fontos, hogy melyik ablakrész az aktív, ezek között a *W* billentyűvel válthatunk.

Az alsó ablakban a kapcsolatban használt csomagok adatai láthatók, a méreten túl a forrás- és célcím, valamint a hálózati csatoló neve szerepel. Az *S* billentyűvel a sorba rendezés szempontját választhatjuk ki, mégpedig két lehetőségből, a csomagok számából, illetve a csomagok méretéből.

Az *IPTraf* jelen verziója már helyesen kezeli és jeleníti meg azoknak a forgalomirányítóknak a forgalmát, amelyek hálózati címfordítást (*Network Address Translation, NAT*) és *IP* cím elrejtést (*IP Masquerading*) végeznek. Ilyen vizsgálatakor minden csomagot kétszer látunk, a bejövő csomagok között éppúgy, mint a kimenők között.

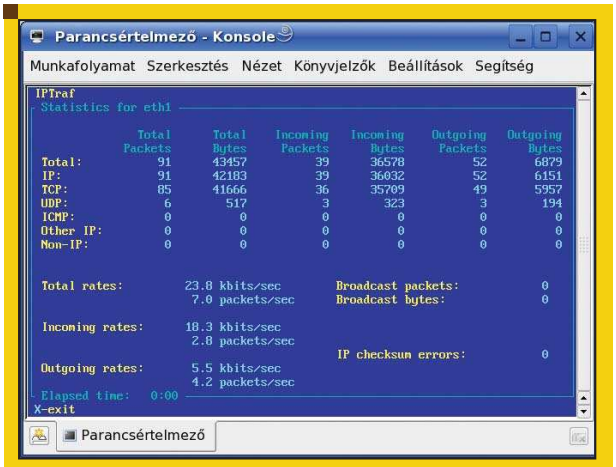
Általános csatolóstatisztika (General interface statistics)

Ebben az üzemmódban a számítógépükbe telepített csatolók összefoglaló adatait látjuk. A táblázatban olvashatjuk a csatoló nevét (*Iface*), a program jelen üzemmódjának indítása óta forgalmazott csomagok számát (*Total*), a használt *IP* címeket (*IP*), a nem *IP* csomagokat (*NonIP*), a hibás *IP* csomagok számát (*BadIP*), valamint az éppen aktuális adatátviteli sebességet. Ezt az üzemmódot mutatja be a *3. ábra*.

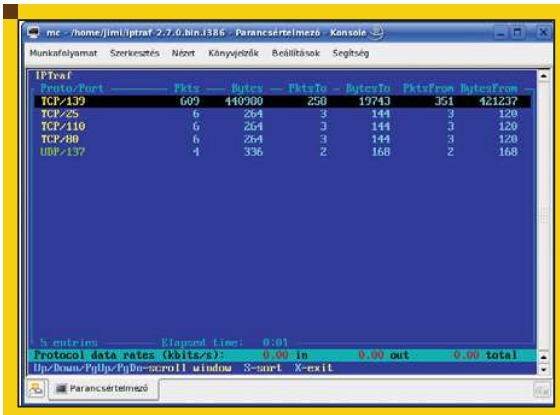
Részletes csatoló statisztika (Detailed interface statistics)

Ez az előző üzemmód bővítése, amit már csak egy csatolóra választhatunk ki. A megjelenő ablakban egy kétdimenziós táblázatban tanulmányozhatjuk, a csomagok számát (*Total Packets*), méretét (*Total Bytes*), a bejövő csomagok számát (*Incoming Packets*) és méretét (*Incoming Bytes*), valamint a kimenő csomagok számát (*Outgoing Packets*) és méretét (*Outgoing Bytes*). Mindezeket az adatokat megmutatja a program az összes csomagra, az *IP* csomagokra, a *TCP*, *UDP* és *ICMP* keretekre, valamint a más *IP* és hibás *IP* kapcsolatokra vonatkoztatva.

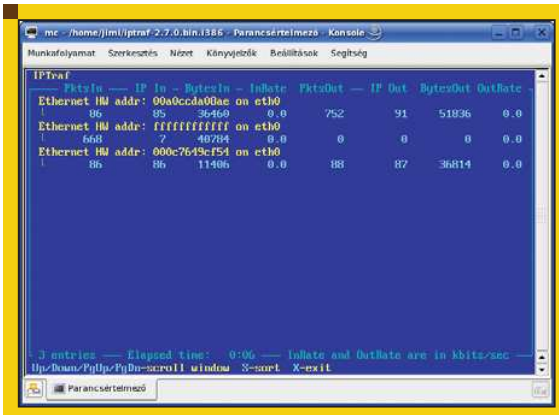
A táblázat alatt sebesség adatokhoz is hozzájuthatunk, hiszen a program megadja a teljes, a bejövő és a kimenő, valamint az adat szórt csomagok számát és méretét, ráadásul még azoknak az *IP* csomagoknak a számát is, amelyek *IP* ellenőrző összege érvénytelen volt. Ezekből az adatokból már egy csatoló működésére következtetni tudunk. Ha sok a hibás ellenőrző összeges csomag, akkor az utalhat hálózati hibára. Az ablak felépítését mutatja az *4. ábra*.



4. ábra Részletes csatoló statisztika



■ *5. ábra* TCP/UDP statisztikai üzemmód



■ *6. ábra* Helyi hálózat forgalmának figyelése

Statisztikai üzemmódban (Statistical breakdowns)

Mint a nevében is benne van, statisztikákat kapunk a képernyőre. A menüpontnak két további almenüje van, amivel kiválaszthatjuk, hogy a statisztika csomagméretre, vagy *TCP/UDP* portra vonatkozzon-e. Az előbbi esetben egy olyan táblázatot kapunk, amelyben csomagméret határok vannak felsorolva és mindegyik után látható, hogy hány darab csomag volt a felügyeleti időszakban abban a tartományban. A csomagméretbe nem tartozik bele az adatkapcsolati fejrész, és a maximális méretet az ***MTU (Maximum Tranfer Unit, maximális adatátviteli egység)*** határozza meg.
A *TCP/UDP* statisztikában egy táblázatban felsorolja a program a portokat (protokoll/portszám), valamint az ezeken keresztülment csomagok számát és méretét, valamint, hogy ezek közül mennyi volt bejövő és mennyi volt kimenő csomag. Ha nem tudjuk, hogy melyik port milyen szolgáltatáshoz tartozik, segítségünkre lehet a */etc/services* fájl, amit a

```
less /etc/services
```

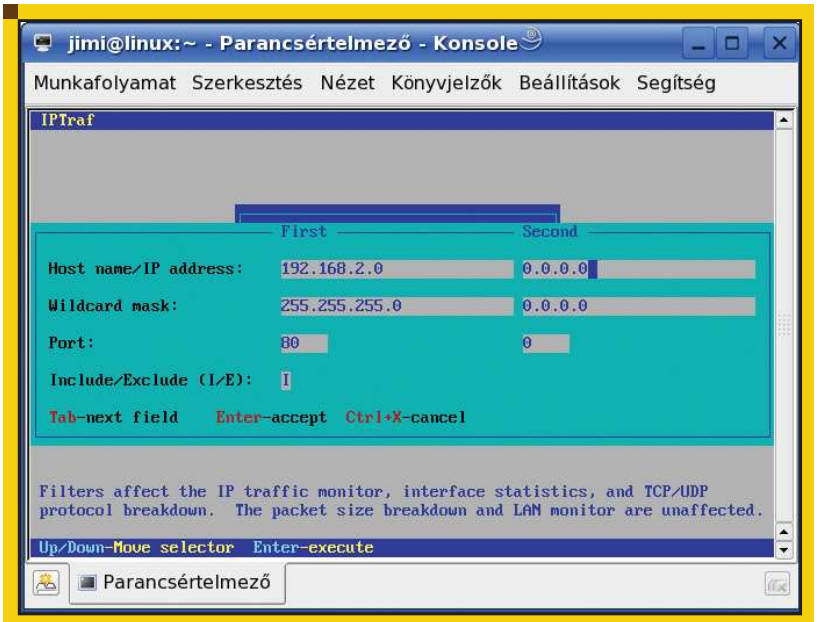
paranccsal írathatunk ki a képernyőre, vagy esetleg a listát szűrhetjük is a `grep` paranccsal, amihez például a 80-as számot tartalmazó sorokhoz az alábbi formában használhatjuk

```
less /etc/services | grep 80
```

A listát sokféleképpen sorba rendezhetjük, amit az *S* billentyű lenyomásával tudunk eldönteni. Itt gyakorlatilag minden oszlopot kiválaszthatunk és a program a csomagokat az szerint fogja sorba rendezni. Az ablak felépítését az *5. ábrán* vehetjük szemügyre.

Helyi hálózati felügyelet (LAN station monitor)

Ebben az üzemmódban a hálózat forgalmát tudjuk felügyelni. Ez akkor hasznos, ha például lelassul a hálózatunk, akkor ebben az üzemmódban kideríthetjük, hogy melyik számítógép bonyolít nagy forgalmat, ami esetleg okozhatja a lassulást.
Az egyéb okokból kialakuló lassulást más, az eddig ismertetett üzemmódokban kereshetjük meg. Két dologra kell felhívnom a figyelmet. Az egyik, hogy a számítógép, amelyen elindítottuk az *IPTraf*-ot, is szerepelni a fog a listában. Ha ez a számítógép egy forgalomirányító, akkor valószínűleg ennek lesz a legnagyobb a forgalma, hiszen a teljes hálózati adatkapcsolat rajta keresztül zajlik.
Ha a listát sorban szeretnénk látni, használjuk az *S* billentyűt, ahol kiválaszthatjuk, hogy milyen szempont szerint szeretnénk ezt megtenni.
A másik, amit megemlítenék, hogy a kártyák fizikai, vagyis *Ethernet* címe látható a listában. Ebből még nem tudjuk, hogy melyik gépben található a hálózati kártya, ezért ehhez használhatjuk a forgalomirányítón az */sbin/arp* parancsot, amely kiírja, hogy melyik *IP* címhez milyen *Ethernet* cím párosul, valamint azt is, hogy ez melyik hálózati csatolón keresztül érhető el. Ennek az üzemmódnak a segítségével már nem tudnak a nagy hálózati forgalmat generáló számítógépek megbújni.
Ezt az üzemmódot mutatja be a *6. ábra*.
Ha nagy a forgalom a hálózatunkban, szükség lehet a hosszú listát valamilyen szempont szerint rövidíteni, vagyis szűrni. Erre szolgál a ***Filters*** menüpont, amely alatt szűrési feltételeket adhatunk meg, törölhetünk szűrőt, illetve alkalmazhatjuk azt valamelyik üzemmódban megkapott listára.
Ha a menüt kiválasztjuk, akkor a választhatunk a ***TCP (TCP...)***, az ***UDP***, a más ***IP (Other IP...)***, az ***ARP***, a ***RARP*** és a nem ***IP (Non-IP...)*** lehetőségek közül. Az almenü jobboldalán látható, hogy milyen szűrési feltételek vannak beállítva a különböző, előbb felsorolt területeken. A ***TCP...*** pont alatt új szűrési feltételt tudunk felvenni (***Define custom TCP filter...***), érvényesíteni tudjuk a szűrőt (***Apply custom TCP filter...***), szerkeszthetjük a szűrőinket (***Edit custom TCP filter...***), valamint törölhetjük is azokat (***Delete custom filter...***).
A szűrő létrehozásnál, tartozzon bármelyik protokollhoz is, kell adnunk egy nevet a szabálynak, majd meg kell adnunk a ***forrás (First)*** és a ***cél IP címét (Second)***, a ***hálózati maszkot (Wildcar mask)***, valamint a portot. Nem árt tudnunk, hogy a forrás- és célcím szerepkörök felcserélődhetnek, a kétirányú kapcsolat következtében, valamint, hogy konkrét gépek címéhez a hálózati maszk `255.255.255.255`. Ha valamelyik címet nem szeretnénk megadni, akkor használjunk ott `0.0.0.0` *IP* címet és a hálózati maszk is `0.0.0.0` legyen.
Az is meghatározhatjuk, hogy azokat a csomagokat lássuk, amelyek megfelelnek a szűrési feltételnek (***Include, I***), vagy azokat, amelyek nem (***Exclude, E***). A mezők között



7. ábra Új TCP szűrő definiálása

a tabulátorral mozoghatunk, míg az *Enter* billentyűvel elfogadhatjuk a beállításokat, a *CTRL+X* kombinációval pedig kiléphetünk. A *7. ábrán* egy minta látható, ahol azt vizsgáljuk, hogy a `192.168.2.0/255.255.255.0` hálózatban milyen web-re irányuló forgalom van.

Az *UDP...* menü alatt hasonló lehetőségeink vannak, azonban itt még két további pontot is kiválaszthatunk, az egyikkel az összes *UDP* csomagot engedélyezhetjük (*Show all UDP packets*), míg a másikkal az összes nem *UDP* csomagot jeleníthetjük meg a képernyőn (*Show no UDP packets*).

Érdekes lehet az *Other IP...* pont, ugyanis itt azokat a csomagokat szűrhetjük, amelyek nem az aktív adatátvitelben vesznek részt, hanem egyéb adatvezérlési szerepük van, mint például az irányító protokollokban használt csomagok.

Éppen ezért a szűrő meghatározásánál azt is megadhatjuk, hogy milyen protokoll látszódjon. Választhatunk az *ICMP (Internet Control Message Protocol, internet üzenettovábbító protkoll)*, az *IGMP (Internet Group Management Protocol, internet csoport menedzselő protokoll)*, az *OSPF (Open Shortest Path First, nyílt legrövidebb út először protokoll)*, az *IGP (Interior Gateway Protocol, belső átjáró protokoll)*, az *IGRP (Interior Gateway Routing Protocol, belső forgalomirányító átjáró protokoll)*, a *GRE (General Routing Encapsulation, általános zárt forgalomirányítás)* és más *IP* protokoll (*Oth IP*) közül. Ha mindegyiket szeretnénk vizsgálni, megtehetjük az *Y* billentyűvel.

Az *ARP (Address Resolution Protocol, címfeloldó protokoll)* és a *RARP (Reverse Address Resolution Protocol, inverz címfeloldó protokoll)* üzeneteit is megjeleníthetjük (*Visible*), vagy éppen elrejthetjük (*Not visible*) a különböző üzemmódokban.

A két lehetőség közül az *Enter* billentyűvel válthatunk.

A *nem IP csomagok (Non-IP)* láthatóságát is tilthatjuk (*Non visible*), vagy engedélyezhetjük (*Visible*). A program beállításait finomhangolhatjuk a *Config* menüpontban. Kissé összetett ablakot kapunk, ha kiválasztjuk ezt a menüpontot.

A *Reverse DNS lookup (inverz DNS névfeloldás)* bekapcsolt állapotában az *IPTraf* megpróbálja a tartományneveket és az *IP* címeket feloldani. Ehhez persze DNS szerverre van szükség, illetve a */etc/hosts* fájlban megfelelő módon be kell jegyezni a számítógépeinket.

A *TCP/UDP service names (TCP/UDP szolgáltatás nevek)* bekapcsolásával a portok helyett a szolgáltatás nevét láthatjuk a képernyőn.

A *Force promiscuous (válogatás nélküli csomagvizsgálat)* bekapcsolásával a kártyát ebbe az üzemmódba kapcsolhatjuk. Ahhoz, hogy megértsük, mit is jelent ez, egy kicsit a hálózati csatolók és a hálózatok működésébe is be kell tekintenünk. Alapesetben a hálózati csatoló csak azokat a kereteket veszi, amelyekben az ő *Ethernet címe (MAC cím)* szerepel, mint célcím. Pontosabban a bemeneti átmeneti tárba minden keret beírásra kerül, de a felsőbb rétegek felé már csak azokat küldi el, amely ténylegesen a mi számítógépünknek szól.

Ha a kártyát átállítjuk *promiscuous* üzemmódba, akkor minden keretet továbbítani fog a felsőbb rétegek felé. Ezzel gyakorlatilag akár a szegmens teljes hálózati forgalmát le lehet hallgatni. Ha egy szerveren futtatjuk a programot, akkor alapvetően minden forgalom rajta keresztül zajlik, viszont ha nem, akkor is rendkívül hasznos lehet ennek a módnak a használata. Azt is meg kell említenem, hogy nem minden hálózati kártya képes ennek az üzemmódnak a kezelésére.

A *Color (szín)* magért beszél, ha engedélyezzük, akkor a program színesben fut, egyéb esetben pedig szürke árnyalatosan.

A *naplózás (Logging)* bekapcsolásával az a forgalom, amit a program a képernyőn mutat, egy fájlban is tárolásra kerül. A naplófájlnak a nevét és az elérési útvonalát a hálózatfigyelés előtt minden üzemmódnál megkérdezi tőlünk a program. A /var/log/iptraf könyvtár szolgál ezeknek az állományoknak a tárolására.

Azt is beállíthatjuk, hogy az átviteli sebességnél *kbit/s* vagy *kBájt* mértékegység közül melyiket szeretnénk használni. Erre szolgál az *aktivitás mód (Activity mode)* menüpont.

Amennyiben szeretnénk a hálózati forgalom figyelésénél a forrásgép fizikai (*MAC*) címét is látni, akkor állítsuk át ezt a *Source MAC addrs in traffic monitor* menüpontban.

A *Timers* menüpont alatt időzítési beállításokat végezhetünk el. A *TCP timeout... (TCP időtúllépés)* pont alatt percekben tudjuk megadni azt az időt, aminek letelte után a a kapcsolatot megszakadtnak tekintjük. Alapértelmezés szerint ez 15 perc.

A *Log Interval... (naplózási időtartomány)* menüben felülbírálhatjuk az a 60 perces értéket, amíg az *IPTraf* a különböző üzemmódok adatait elhelyezi egy fájlba a megadott helyre. Természetes azt is beállíthatjuk, hogy milyen gyakorisággal frissüljön (*Screen update interval...*) a különböző üzemmódokban összeállított lista. Itt másodpercekben adhatjuk meg az időt. A *TCP closed/idle persistence...* menüpontban azt tudjuk meghatározni, hogy az *IPTraf* mennyi idő elteltével távolítsa el a *TCP* ablakból a megszakadt, lezárt vagy forgalommentes kapcsolatokat.
Az *Additional ports...* menüpont alatt meg tudjuk adni annak a porttartománynak az elejét és a végét, amelyet vizsgáltatni szeretnénk az *IPTraf* programmal. Alapértelmezés szerint az *IPTraf* csak a jól ismert szolgáltatások portjait vizsgálja, amelyek portszáma 1024 alatti.
Ha szeretnénk a magasabb portokat forgalmát is megjeleníteni, akkor itt megadhatjuk ezt. Ha már nincs szükségünk a kiegészítő porttartományra, akkor azt törölni a *Delete port/range...* menüponttal lehet.

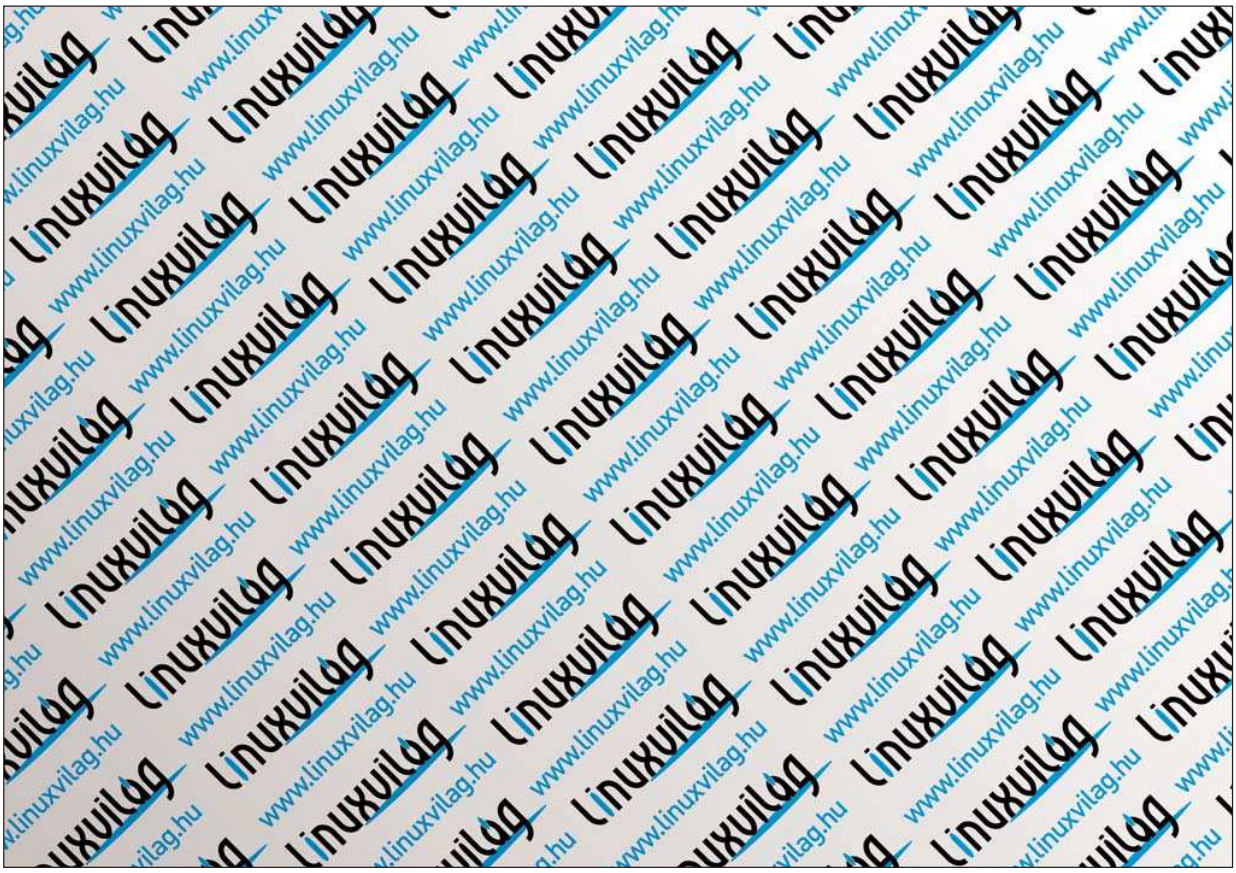
Az *IPTraf* rendelkezik egy nagyon hasznos lehetőséggel, amely főleg rendszeres használat mellett hálálja meg magát. Amint már láttuk a hálózatban adatokat cserélő gépeket a fizikai cím alapján azonosítja és jeleníti meg. Ahogy már említettem az *ARP* táblát kiírathatjuk, amiből megtudhatjuk az IP címet és ezzel együtt már beazonosíthatjuk a számítógépet. Az *Ethernet/PLIP host descriptions...* menüpontban lehetőségünk van a fizikai címhez leírást párosítani, amivel az *ARP* tábla alapján történő címfeloldást megtakaríthatjuk magunknak. Ha ezt elvégezzük, akkor a forgalom felügyeletnél a MAC cím mellett a gép leírása is olvasható lesz. Ezt a funkciót a gyűrű topológiájú hálózatokban is használhatjuk, ebben az esetben az *FDDI/Token Ring host descriptions...* menüpontban adjuk meg a címek leírását.
Az *IPTraf* nem csak a menürendszerén keresztül kezelhető, hanem a fenti üzemmódokba parancssori kapcsolókkal beléptethető, amivel akár automatizálni is lehet a futását. A kapcsolókról az

```
iptraf --help
```

paranccsal kaphatunk információt. Mit láthattuk, az *IPTraf* programnak nagyon sok lehetősége van és megfelelő tudással nagyon sok mindent el tud árulni a számítógéphálózatunkról és az abban folyó adatforgalomról. A rendszergazdáknak egy nagyon hasznos eszköz, akár automatizálva a teljes forgalom felderíthető. Arra azonban ügyelünk, ha a naplózást bekapcsoltuk, akkor elég tetemes mennyiségű adatot képes eltárolni, ami pedig akár pár nap alatt is betölthet egy /var partíciót. Ennek a kezelésére is megvannak a módszerek, de azt már nem ennek a cikknek a feladata bemutatni.

Markó Imre
(linux@akribisbt.hu)

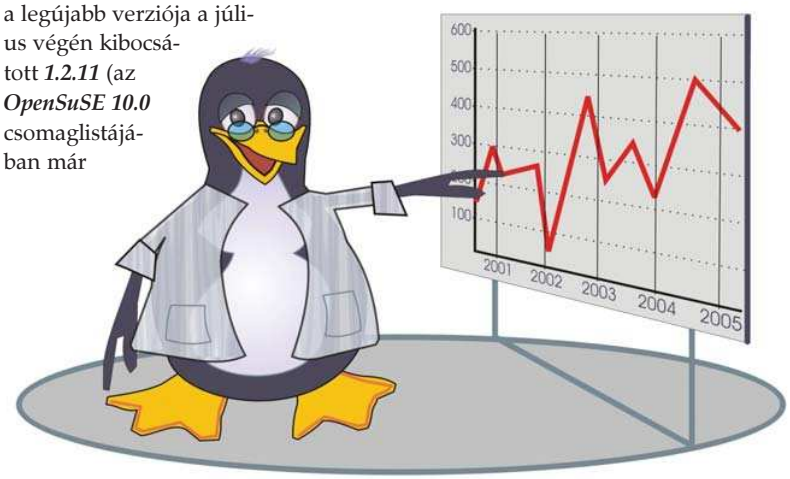
Hardvermérnök és mérnöktanár végzettsége van. Saját cégében Linux rendszerek tervezésével és üzemeltetésével foglalkozik. Ezen kívül egy főiskolán oktat, elsősorban hardveres tantárgyakat.



Grafikonok minden mennyiségben – RRDTool (1. rész)

A rendszergazdáknak előbb-utóbb ingere támad arra, hogy a rendszereik minden mérhető jellemzőjét – szeretve dédelgetett rendszerük minden szívdobbanását – naplóba vezessék. Ezek az adatok azonban nem igazán látványosak, s nem látszik bennük vizuálisan a gépeik minden rezdülése. A begyűjtött és megőrzött – akár percre pontos – adatok többnyire feleslegesen foglalják a helyet, mivel hónapok vagy évek távlatában nem igazán gyakori igény pontosan utánanézni a szerver átlagos terhelésének.

Tobias Oetiker keze nyomán 1998 januárjában megjelenő *RRDTool* első verziója rendkívül hiánypótló, s rendkívül kezdetleges program volt. Alapkoncepciója azonban ***zseniálisan egyszerű***: egy-egy időszakról (órák, napok, hetek, stb.) csak a neki megfelelő részletességgel tárol adatokat, s ezeket is körbeforgatja: az adatbázis mérete így állandó marad. Ez a módszer adatvesztéssel jár, ugyanis a körbeforgatás esetén az időszak legrégebbi értékét a legújabb felülírja. Az *RRD* név is ezt a jellegzetességet takarja: ***Round-Robin Database***.
Az *RRDTool* közel egy évtizedes fejlődése remek eszközöket ad a kezünkbe, teljesen belesimul a ***UNIX KISS*** (***Keep It Simple & Stupid***) filozófiába. A legtöbb terjesztés több éve tartalmazza ***rrdtool*** néven, amelynek a legújabb verziója a július végén kibocsátott ***1.2.11*** (az ***OpenSuSE 10.0*** csomaglistájában már benne van). Ha mégsem találjuk meg ezt a programot csomag formájában, akkor a http://www.rrdtool.org oldalról indulva le tudjuk tölteni a forrását, amelyet lefordítva és telepítve már használhatjuk is.
Bármilyen adatokat szeretnénk beletölteni az adatbázisba, tudnunk kell, hogy az *RRDTool* nem képes összegyűjteni azokat. Az *RRDTool* kész adatokat vár bizonyos időközönként, amely lehetnek rendszertelenek, de leggyakrabban rendszeres időközönként kell beletölteni ezeket. Ha tudunk rendszeres adatokat prezentálni, akkor nagyobb kihagyás esetén az *RRDTool* képes interpolálni köztes értékeket, ha így hozzuk létre az adatbázist.
Általában az *RRDTool* működése nem szakad meg, mivel egyszerűségénél fogva nem kell sok hibalehetőségtől tartanunk: a program rendkívül stabil és robusztus működésű. Mivel az ördög nem alszik, érdemes rendszeresen bináris és szöveges (*XML*) mentést készíteni az összes *rrd* állományunkról. A mentést leszámítva a program egyetlen kimeneti módja a grafikonok készítése, amelyek tetszetős és átlátható formába öntik az adatbázis aktuális állapotát.

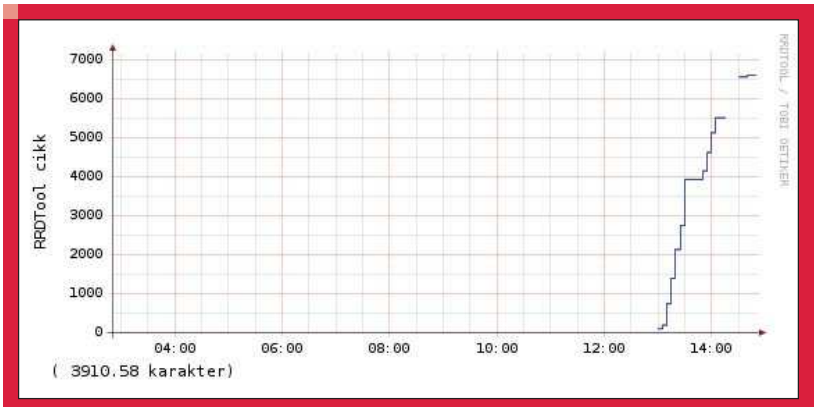


RRD állomány készítése

Ha grafikonon akarjuk ábrázolni valamilyen rendszeresen előálló változó értéket, akkor legelső dolgunk azon adatok meghatározása, amelyeket ábrázolni szeretnénk. Nagyon gondoljuk át, hogy milyen adatokat rögzítünk, mivel – jelenleg még – nincs lehetőségünk egyszerű módszerekkel a működő adatbázisunkat kibővíteni.
Kezdeti ujjgyakorlatként hozzunk létre egy egyszerű *RRD* adatbázist, amely e cikk írásának egyetlen jellemzőjét tárolja, mégpedig a leütött karakterek számát. Ezzel szépen követhetjük az időtengelyen, hogy miképp is állt össze a teljes cikk.
Első látásra kissé átláthatatlan betűkből és számokból álló parancsot kell kiadnunk, hogy ez az adatbázis létrejöjjön:

```
$ rrdtool create rrdcikk.rrd
➥--step 300 DS:karakter:GAUGE:
➥600:0:100000
➥RRA:AVERAGE:0.5:1:1200
```

A paraméterlista tömör és egyszerű, mint ahogy maga a program is. Látható, hogy az első várt paraméter egy tevékenység neve, amely most a létrehozás (`create`). Minden ilyen tevékenységnek más a paraméterlistája, más-más adatokat kell megadnunk.
A létrehozás esetén a második paraméter az adatbázis állomány neve, amelynek általában ***rrd*** a kiterjesztése. Nagyon fontos paraméter a `--step`, amellyel a legkisebb kezelt lépésközt adhatjuk meg másodpercben. Ezt a feladat függvényében kell megválasztanunk, nekem nem volt kedvem percenként megnézni, hogy hány karaktert írtam eddig (mivel aljas módon ***OpenOffice.org*** alatt dolgozom, és nem konzolon): így közel öt percenként elég megadnom új adatot (ez gyakorlatilag egy-egy bekezdés végét jelenti).
A `DS` szöveg a ***Data Source*** rövidítése, ezzel kell bevezetnünk minden egyes adatforrást és annak paramétereit, ezek kettősponttal van elválasztva.
Az első paraméter az adatforrás neve, majd ezt követi az adatforrás típusa. Ez többféle lehet, nekünk most a `GAUGE` kell, amely egyszerűen csak a kapott számot tárolja. Lehetőségünk van még számláló (`COUNTER`) típusra is, amely folyamatosan növekvő számokkal dolgozik (ideális a hálózati forgalom ábrázolására). Innen ismét elágazik a paraméterek feladata, mivel minden egyes adatforrás típusnak saját opciói vannak. A `GAUGE` opciói sorra a `heartbeat`, a minimum és a maximum.
Az első érték (szívdobbanás) másodpercben értendő, és rendszeres adatfeltöltés esetén általában duplája a megadott lépésnek, hiszen nem kell túl nagy kieséstől tartanunk. Ha nem érkezik adat a megadott időtartam alatt, akkor `UNKNOWN` (ismeretlen) kerül érték helyett a következő adat helyére.
Ez kihagyásként jelentkezik a grafikonon, s nem néz ki túl szépen (*1. ábra*). Ha a lépésköz többszöröse ez az idő, s két adatfeltöltés között több lépésköz is kimarad, de még nem telt le egy szívdobbanásnyi idő, akkor nem jelenik meg a fenti kihagyás, a program interpolál az előző és jelenlegi adatfeltöltés értékeiből. Úgy válasszuk meg ezt az időt, hogy a kisebb hibákat elfedje, de a nagyobb hibák kiderüljenek.
A minimum és a maximum érték önmagáért beszél, itt tudjuk megadni az adott adatforrás legkisebb és legnagyobb lehetséges értékét. Ha valamelyiket nem ismerjük, egy U betűt is írhatunk bármelyik helyen.



1. ábra Szakadás a grafikonban

A `DS` blokk után megadhatunk újabb adatforrás definíciókat, egyszerűen csak szóközzel elválasztva. Nekünk jelenleg nem kell több adatforrás, tehát következhet az eltárolt adatok körforgásának meghatározása (***RRA***, ***Round-Robin Archives***).
Az adatok eltárolásának van négy különböző módja: `AVERAGE`, `MIN`, `MAX` és `LAST`. Ezek rendre az adott lépésközben a beírt adatok átlagát, minimumát, maximumát vagy az utolsó értéket tárolják. Gyakori az átlag használata, mivel ez nagyobb léptékű (havi vagy éves) grafikon esetén is szépen ábrázolja az értékeket, s a kisebb idők esetén is jó eredményt ad. Természetesen eltárolhatjuk a többi értéket is, amelyek jól jönnek majd a grafikonkészítés során.
Az eltárolás típusát követi három szám, amelynek az `xff`, a `steps` és a `rows` nevet adta a program készítője. Ezek közül az első paraméter értéke általában 0.5, s a programot az ismeretlen értékek kezelésében befolyásolja. Sokkal fontosabb paraméter a lépésköz és a letárolt sorok száma.
A lépésköz itt nem másodperc, hanem az adatbázis létrehozásakor megadott lépésköz egész többszöröse. Esetünkben ennek értéke 1, így öt percenként tárol egy-egy értéket az adatbázisunk.
A sorok száma önmagáért beszél.
Az eltárolás megadásakor készíthetünk több lépésközt is, amelyeket érdemes a hozzáigazítani a grafikonok rajzolásához. Általában 1 órás, 6 órás, 12 órás, 1 napos, 1 hetes, 1 hónapos, illetve 1 éves intervallumra készítünk grafikont. A grafikonok felbontása gyakorlatilag tetszőleges méretű lehet, de általában képernyőhöz méretezzük, ahol a képpontok száma közelít az adatsorok számához. Egy egész napos grafikon esetén – egy perces lépésközzel – 1440 adatunk áll rendelkezésre, amelyet többnyire 600 vagy 800 képpont széles grafikonon ábrázolunk. Ez azt jelenti, hogy 60 másodpercel lépésköz esetén szükségünk lesz egy

```
RRA:AVERAGE:0.5:1:1440
```

paramétersorra, amely egy perces gyakorisággal 1440 értéket rögzít, pont egy egész napot.
Egy hét adatainak grafikonon való ábrázolásához tíz perces rögzített adatok már szép eredményt adnak, így írhatunk egy

```
RRA:AVERAGE:0.5:10:1008
```

paramétersort is, ez egy teljes hét.
Egy hónap ábrázolásához elég fél óránként egy érték, s így is 1440 sort tudunk rögzíteni:

```
RRA:AVERAGE:0.5:30:1440
```

Az éves értékekhez hat óránként szükséges egy adat, s így rögzíthetünk 1600 sort, amely bőven sok a szép grafikonhoz:

```
RRA:AVERAGE:0.5:360:1600
```

Ha a két utolsó paramétert egymással összeszorozzuk, majd az adatbázis lépésközével is megszorozzuk,

megkapjuk az adott tárolás körbefordulási idejét. Például a legutolsó esetben ez ***360*1600*60***, ami ***34560000*** másodperc, s ez pontosan ***400*** napnak felel meg. Ha egy évnél túl szeretnénk grafikont rajzolni, akkor ahhoz megfelelő RRA értékeket kell készítenünk, különben az adatok elvesznek a körbefordulás miatt.
A példában csak egy ***RRA***-t hoztunk létre, amely az utolsó 100 órát rögzíti. Nem szeretném 100 óránál tovább írni ezt a cikket, így előre láthatólag elég lesz.

Adatok beírása

Az adatbázist azért hoztuk létre, hogy adatokat írjunk be, erre szolgál a frissítés (`update`) tevékenység. Ennek nagyon egyszerű formája van:

```
$ rrdtool update rrdcikk.rdd
➥N:9324
```

Az `N` paraméter annyit tesz, hogy az aktuális idő (***now***) szerint szúrjuk be az adatokat. A program a ***UNIX*** időmeghatározást is ismeri, amely az 1970. január 1-e óta eltelt másodpercek számát jelenti. Ez utóbbira akkor van szükség, ha egy új adatbázisba régebbi adatokat akarunk tölteni (a `'date +%s'` mutatja meg ezt).
Ha több adatot is meg akarunk adni egyszerre, akkor többször meg kell ismételnünk az utolsó paramétert (természetesen ekkor meg kell adnunk időt):

```
$ rrdtool update rrdcikk.rrd
➥1129988409:9452
➥1129988865:9512
➥1129989103:9643
```

Az adatok mentése

Az ***RRDTool*** program régebben szöveges állományba mentette a megadott adatbázis tartalmát, mostanában már ***XML*** a kimenet formátuma. Ezt sokkal egyszerűbb más programhoz illeszteni, illetve leírja a saját szerkezetét.
Az adatok mentésre a `dump` tevékenység szolgál, amely minden esetben a kimenetre írja az elkészült ***XML*** adatfolyamot:

```
$ rrdtool dump rrdcikk.rrd
<!-- Round Robin Database
➥Dump -->
<rrd>
        <version> 0003 </version>
        <step> 300 </step> <!--
        ➥Seconds -->
        <lastupdate> 1129989064
        ➥</lastupdate> <!--
        ➥2005-10-22 15:51:04
        ➥CEST -->
[...]
```

Az adatok visszatöltése

A kiexportált ***XML*** állományból `restore` tevékenységgel tudunk ***rrd*** formátumú bináris adatbázist készíteni. Ehhez a következő parancsot tudjuk használni:

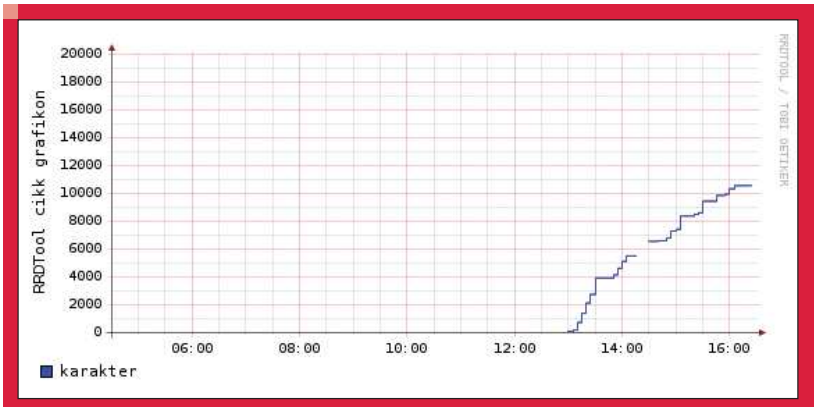
```
$ rrdtool restore rrdcikk.xml
➥rrdcikk2.rrd
```

Kénytelenek vagyunk más nevet megadni a bináris formának, mivel létező adatbázist nem ír felül a program.

Grafikon készítése

A program egyik fő feladata a grafikonok készítése, így kellően összetett és bonyolult első látásra a kiadandó parancs (*2. ábra*):

```
$ rrdtool graph cikkRRD.png
➥--imgformat PNG --width 464
```



2. ábra Az elkészült grafikon

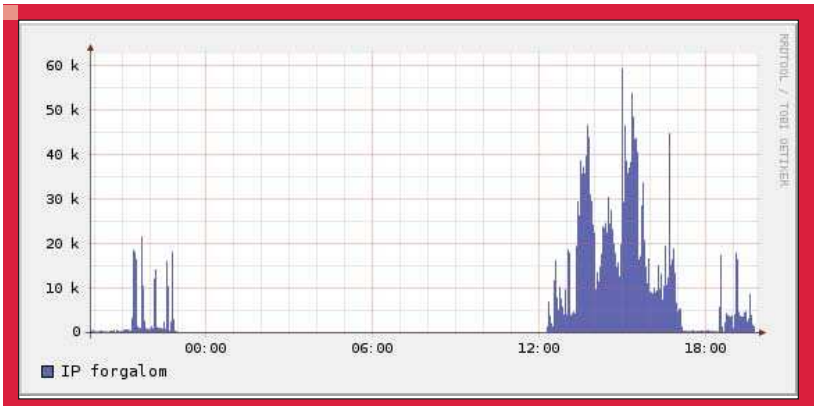
```
➥--height 200 --end now
➥--start end-12hours
➥--vertical-label "RRDTool
➥cikk grafikon" --lower-limit
➥0 --units-exponent 0 --lazy
➥--color "SHADEA#ffffff"
➥--color "SHADEB#ffffff"
➥--color "BACK#ffffff"
➥"DEF:karakter=rrdcikk.rrd:
➥karakter:AVERAGE"
➥"LINE1:karakter#0000FF:
➥karakter"
```

Néhány példa után már nem fog olyan bonyolultnak és átláthatatlannak látszani a fenti parancs, azonban vannak olyan mély rejtelmei az ***RRDTool*** programnak, amelyek még a profikat is zavarba hozhatják.
A legbiztosabb, hogy meg kell adnunk a rajzolás (`graph`) tevékenységet és ezt követően annak a grafikus állománynak a nevét, amely a grafikont fogja tartalmazni. Ezek után rendet kell vágnunk a „kusza” parancsok között.
Kezdjük az ismerkedést olyan egyszerű paraméterekkel, mint amilyen a grafikon mérete és a típusa. A `--width` és a `--height` után megadott szám lesz a grafikon mérete, amely nem tartalmazza a tengely feliratait, a díszítéseket és a magyarázó szövegeket. Ezen viselkedése egy kicsit kiszámíthatatlanná teszi a program által készített képek végső méretét, ezért a készítés során a kimeneten láthatjuk a kép tényleges méretét (a fenti esetben ez ***561x268***).
A `--imgformat` alapértelmezett értéke a `PNG`; de ezen kívül a program képes `SVG`, `EPS` vagy `PDF` állományt is készíteni.

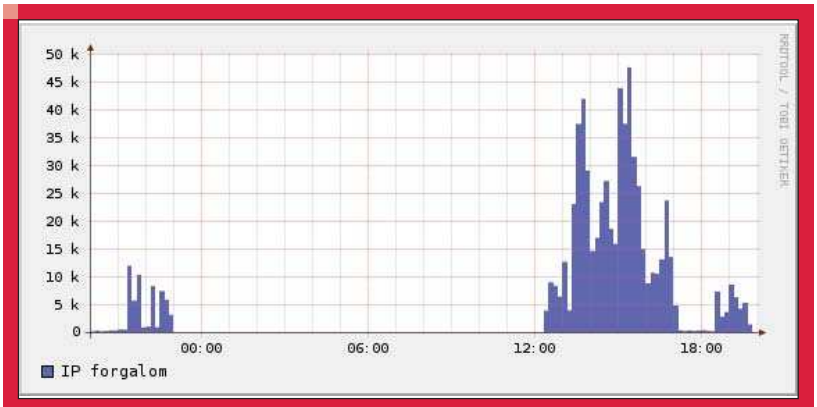
A `--end`, a `--start` és a `--step` a grafikon léptékét határozza meg. Ezek közül a `--end` a grafikon végének időpontját határozza meg, amely általában a now (most) szó. A `--start` ebből következően grafikon kezdetének időpontja lesz, s a végponthoz képest szoktuk meghatározni. Ez gyakorlatilag egy `end±n` forma lesz, ahol az n egy úgynevezett időeltolás (***time offset***). Ezt simán angol nyelven tudjuk megfogalmazni, például az

```
end-2years3months14days14hours
➥56minutes23seconds
```

jelenti, hogy a grafikon kezdőpontja végpontjához képest ***két*** évvel, ***három*** hónappal, ***14*** nappal, ***14*** órával, ***56*** perccel és ***23*** másodperccel korábbi időpontban van. A `--step` paraméter a grafikon felbontását határozza meg, amelyet nem szükséges megadnunk, mivel a program saját maga meghatározza a grafikon felbontásától és a megadott kezdő- és végponttól függően.
Az időtengelyt már rendbe tettük, azonban nem foglalkoztunk még a függőleges iránnyal, amelyen az értékeket ábrázoljuk. A program alapértelmezésben automatikusan meghatározza a szükséges határokat, azonban vannak esetek, amikor jobb belenyúlni ebbe a folyamatba. A legalapvetőbb beállítási lehetőség a minimális és a maximális ábrázolt értékek közvetlen megadása az `--upper-limit` illetve a `--lower-limit` paraméterrel. Ezen túl a program saját automatizmusát egy alternatív módszerre tudjuk cserélni a `--alt-autoscale` paraméterrel, amely akkor hatásos igazán, ha nagy érték körül van kis változás (a program kézikönyve a `260*0.001sin(x)` függvényre hivatkozik).
Ha a két tengelyt megfelelően beállítottuk, akkor beállíthatjuk a grafikon rácsozatát, amely megvezeti szemünket az értékek követésénél. A dokumentáció szerint *„**quite complex**”*, de szerencsére csak viccel a program írója, bár el kell ismerni kicsit ködösen fogalmaz. Nos, a lényeg, hogy az `--x-grid` paraméter után meg kell adni a rácsozatot `MINUTE:30` formában, amely azt takarja, hogy félóránként lesz egy vonal a grafikonon. Ezt egy kettőspont után a fő rácsozat követi, amely például `HOUR:4` esetén minden negyedik órában lesz egy vastagabb vonal a grafikonon.
Ezt követi (szintén kettőspont után) a feliratok pozíciója, amely például lehet `DAY:1`, s ekkor minden eltelt nap alatt lesz egy-egy felirat. A felirat pozícióját és egy formátumszöveget



3. ábra Hálózati IP forgalom



4. ábra 10 perces lépésköz – nem túl szép, de pontos

kell végül megadni; így a példánk a következőképpen néz ki:

```
--x-grid MINUTE:30:HOUR:4:DAY:
➥1:0:%X
```

A függőleges tengely paraméterezése más sokkal egyszerűbb, hiszen csak meg kell adni azt a két számot, amelyek a rácsvonalak és a fővonalak sűrűségét meghatározzák:

```
--y-grid 40:10
```

Itt a program 40 egységenként tesz egy vonalat, s 400 egységenként egy fő rácsvonalat. Ezzel a függőleges tengely paraméterezése nincs teljesen kész, hiszen lehetőségünk van logaritmikus ábrázolásra is, amelyet különösen nagy változások esetén érdemes használni:

```
--logarithmic
```

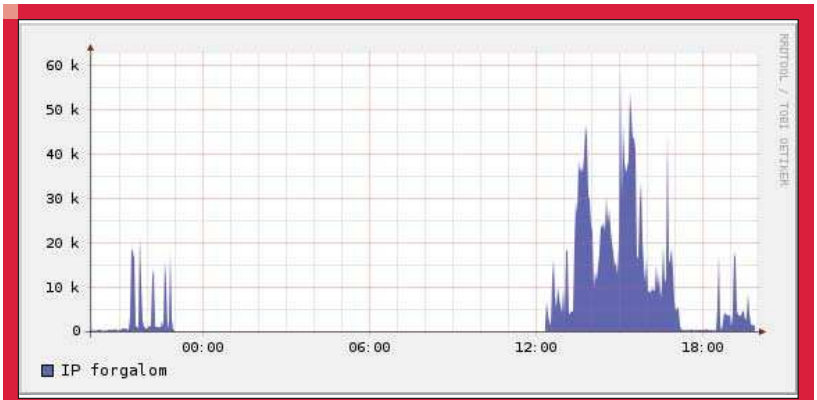
A grafikonra tehetünk címkéket, amelyeket a maradék két szabad helyre tudunk tenni: a grafikon fölé vagy bal oldalára. A jobb oldalt ugyanis a program készítője lefoglalta magának, a grafikon alatt pedig a jelmagyarázat kapott helyet. A főcímet a `--title` paraméterrel tudjuk beállítani, a bal oldali szöveget a `--vertical-label` paraméter segítségével. Mindkét esetben idézőjelek között érdemes megadni a kívánt szöveget.
Módosíthatjuk a grafikon színeit, ha erre támad kedvünk, így például zökkenőmentesen beilleszthetjük egy weboldal arculatába a kész képeket. Ehhez a

```
--color KOMPONENS#RRGGBB[AA]
```

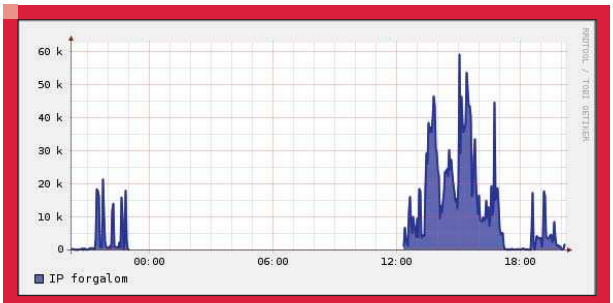
használható, ahol a kettőskereszt után megadott hexadecimális számok a szín vörös, zöld és kék összevevőjét jelölik, utána megadhatjuk az adott komponens átlátszóságát is, ha szeretnénk ilyet. A komponens lehet a `BACK`, mint háttér; `CANVAS`, mint a grafikon háttere; `SHADEA` és `SHADEB`, mint a grafikon kerete; `GRID` és `MGRID`, mint a rácsozat; `FONT`, mint a betűk; `AXIS`, mint a koordináta rendszer; `FRAME`, mint a keret; s végül `ARROW`, mint a koordináta tengelyek lezáró nyilai. Ezzel meg is volnánk a közös paraméterekkel, így jöhet a rázósabb része a grafikon definíciónak, amellyel meghatározzuk, hogy milyen adatsorokból milyen grafikont szeretnénk rajzolni. Ehhez `DEF` kezdetű sorokat kell megadnunk, amelyet minimálisan három megadandó paraméter követ: az érték elnevezése, az ***rrd*** állomány neve, végül az adatforrás neve és adattárolás a módja zárja a sort (természetesen minden egyes paramétert kettőspont választ el). Ezeken a paramétereken túl megadhatunk más léptéket, kezdő- illetve záróidőt; ezen paramétereket ritkán szoktuk használni, a program automatikusan beállítja helyettünk.
Ha több ***rrd*** állományunk van, és ezekben több adatforrásunk, akkor is tudunk egy grafikonra adatokat gyűjteni, ugyanis minden DEF sornak egyedi nevet kell adnunk, s meg kell neveznünk az adatbázist az adatforrással együtt. Az eddigi sok munkával még nem rajzoltunk semmit, egyszerűen csak meghatároztuk azon adatsorokat, amelyeket ábrázolni szeretnénk... a rajzolás még teljes egészében hátravan.
A rajzoláshoz érdemes keresni olyan ***rrd*** állományokat, amelyek sok összefüggő adatot tartalmaznak. Igazi állatorvosi ló lehet például az ***Ntop rrd*** modulja által készített hálózati adatbázis, amely rengeteg információt rögzít a hálózati forgalmunkról.
Nézzünk egy egyszerű példát, ahol is szeretnénk tudni a teljes ***IP*** forgalom alakulását egy napra visszamenőleg, mégpedig kitöltött területtel ábrázolva (*3. ábra*):

```
DEF:ip0=ipBytes.rrd:counter:
➥AVERAGE AREA:ip0'#6666FF':'
➥IP forgalom'
```

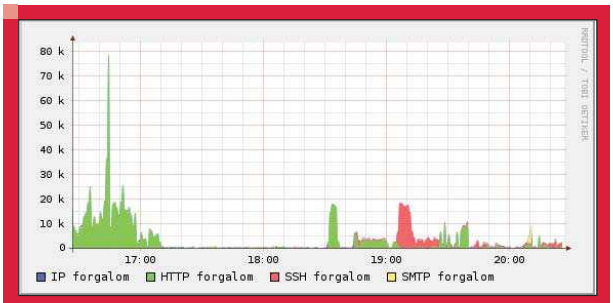
A kitöltött területet az `AREA` vezeti be, amelyet az adatsor neve, a színkódja, s végül a megnevezése követ. A színkód az őt megelőző kettőskereszt miatt van idézőjelben, mivel a # jel utáni szöveget a ***UNIX*** parancsértelmezők egy része figyelmen kívül hagyja, mint megjegyzést (a ***BASH*** csak akkor, ha a # jel előtt egy szóköz is áll).



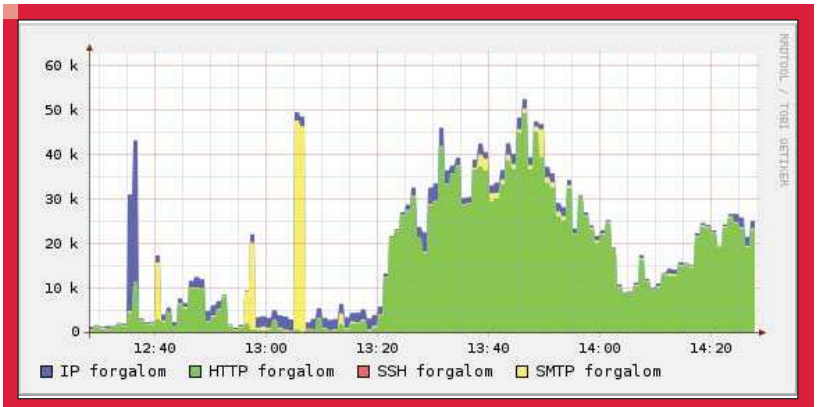
5. ábra Lejtős átlagolás – szép, de nem pontos



6. ábra Beburkolva



7. ábra Halmozott területek I.



8. ábra Halmozott területek II.

Látszik, hogy nem valami szép a grafikon, ha az értékek nagyon ingadoznak, érdemes egy picit simítani a képen, amelyhez két lehetőségünk van: vagy a `--step` paramétert hívjuk segítségül (*4. ábra*) egy nagyobb (például tíz perces lépésközzel) vagy bekapcsoljuk az úgynevezett lejtő-módot (`--slope-mode`) (*5. ábra*). Még szebb eredményt érhetünk el, ha egy vastagabb vonallal beburkoljuk a görbét (*6. ábra*):

```
DEF:ip0=ipBytes.rrd:counter:
➥AVERAGE AREA:ip0'#6666FF':'
➥IP forgalom' LINE2:ip0'
➥#0000AA':''
```

Érdemes néhány másik jellemzőt is felrajzolni a grafikonra, például az *IP* forgalom tételes megoszlását: szeretnénk látni, hogy mennyi ebből a *HTTP*, az *SMTP* és például az *SSH*. Ehhez csak annyit kell tennünk, hogy más-más *rrd* állományokból gyűjtünk össze adatforrásokat:

```
DEF:ip0=ipBytes.rrd:counter:
➥AVERAGE DEF:http0=
➥IP_HTTPBytes.rrd:counter:
➥AVERAGE
DEF:ssh0=IP_SSHBytes.rrd:
➥counter:AVERAGE DEF:mail0=
➥IP_MailBytes.rrd:counter:
➥AVERAGE
```

Majd vonalakkal ábrázoljuk az egyes mennyiségeket:

```
AREA:ip0'#6666FF':'IP forgalom'
➥LINE:http0'#66FF66':'HTTP
➥forgalom' LINE:ssh0'#FF6666'
➥:'SSH forgalom' LINE:mail0'
➥#FFFF66':'SMTP forgalom'
```

Ez olyannyira nem lett szép, hogy nem is érdemes megmutatnom, a vonalak ugyanis egymást keresztezik, átfedik... sokkal szebb eredményt tudunk elérni, ha nem vonalakat, hanem területet használunk, s ezek nem a függőleges tengely legalsó pontjáról indulnak, hanem egymásra halmozzuk őket, s így a legfelső pontjuk pont az összes halmozott érték összegénél fog járni:

```
AREA:ip0'#6666FF':'IP forgalom'
AREA:http0'#66FF66':'HTTP
➥forgalom'
AREA:ssh0'#FF6666':'SSH
➥forgalom':STACK
AREA:mail0'#FFFF66':'
➥SMTP forgalom':STACK
```

Az összegzés mindig az előző értékhez képest működik, így a teljes IP forgalomhoz nem adjuk hozzá az összetevőit. Egy picit belenéztem a grafikonba, s találtam olyan pontot, ahol a különféle adatforgalmak szépen látszanak (*7. ábra; 8. ábra*).
A második (befejező) rész az ***RRDTool*** olyan extra tulajdonságait taglalja, mint az értékek és szövegek írása a grafikon alá, illetve az adatsorokkal való matematikai és logikai műveletek.

Auth Gábor
(auth.gabor@enaplo.hu)

Egy pécsi középiskolában informatikát és programozást oktat. Tíz éve botlott először a UNIX rendszerekbe, 7 év Linux használat után kapta el a FreeBSD lázat, amiből máig nem tudott kigyógyulni.

KAPCSOLÓDÓ CÍMEK

Az RRDTool honlapja
➲ http://www.rrdtool.org

A cikkben említett példák forrása
➲ http://user.enaplo.hu/~auth.gabor/rrd/

Való életből vett példa
➲ http://www.enaplo.hu/index.jsp?page=visitor.loadStat

Adataink biztonságos tárolása és mentése

Akivel már megtörtént, hogy fontos adatot veszített bármilyen okból kifolyólag, az egészen biztosan kínosan ügyel arra, hogy még egyszer ilyen elő ne fordulhasson vele. Akivel ez még nem történt meg, az egyfelől nagyon szerencsés, másfelől jobb ha okos ember módjára más hibájából tanul, nem a sajátjából.

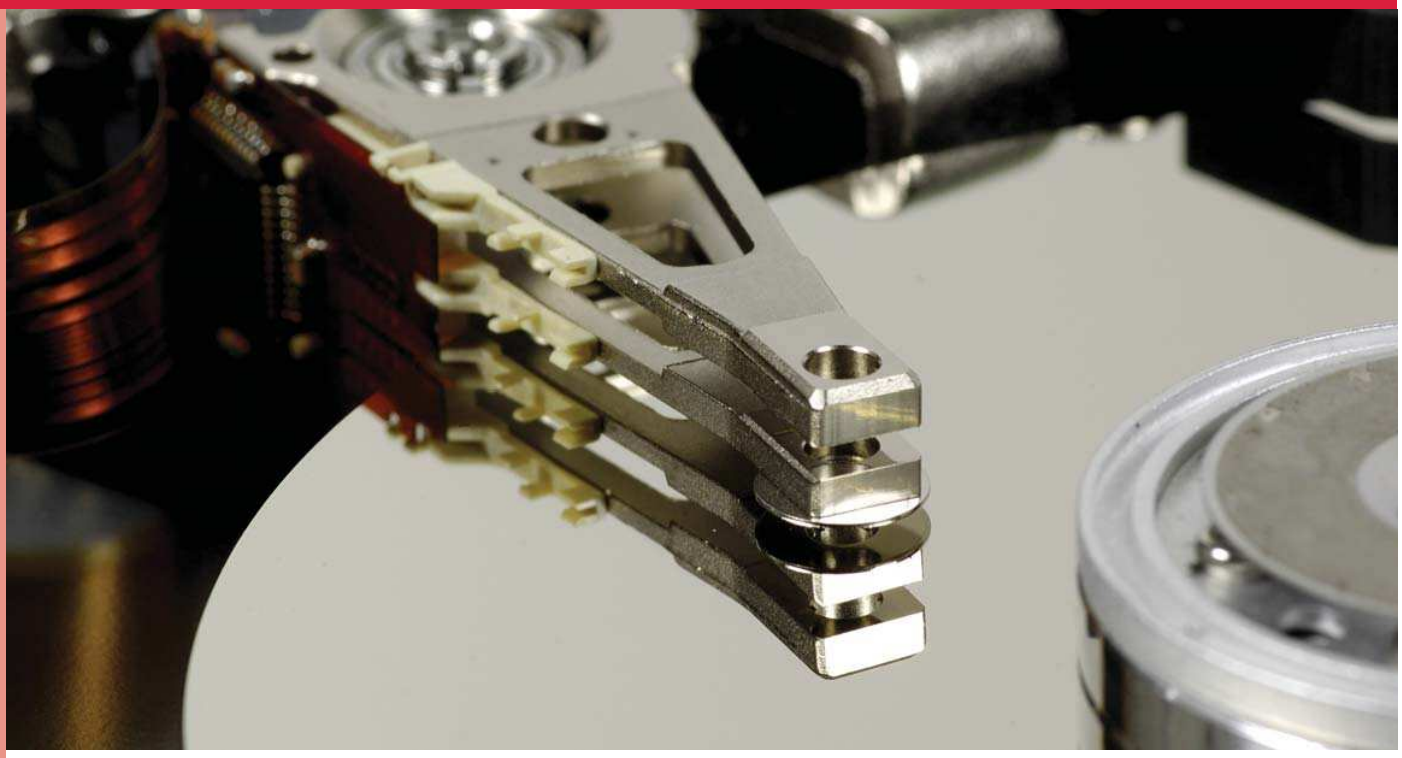
Statisztikailag ugyanis minél többet foglalkozunk a számítógépünkkel, annál nagyobb az esélye annak, hogy egyszer velünk is megtörténik a katasztrófa és erre jobb felkészülni, mint akkor kapkodni fűhöz, fához.

Adatbiztonság

Az adatbiztonság egy meglehetősen összetett kérdés, több pusztán adataink tárolásánál. Amikor adatbiztonságról beszélünk, akkor az adatok tárolásáról, azok hozzáférhetőségéről beszélünk annak a kockázatnak a vizsgálata mellett, hogy illetéktelenek miként tudnak az adatainkhoz hozzáférni, miként tudják azokat ellopni, módosítani, mások felé minket megszemélyesíteni. Adataink biztonságos tárolása ugyan nem az egyik legfontosabb pontja a klasszikus értelemben vett adatbiztonságnak, ám a hétköznapi számítógép használat alkalmával legalább olyan fontos kérdéssé válik, mint az illetéktelen hozzáférés kiszűrése.

Biztonságos adattárolás – módszerek és technológiák

Adataink biztonságos tárolásáról ma már hétköznapi eszközök igénybevételével is meglehetősen egyszerűen gondoskodhatunk. Gondoljunk csak a legegyszerűbb megoldásra: ma már minden háztartásban fellelhető egy *CD*-író eszköz. Ha ezt rendszeresen használjuk adataink archiválására, már sokat tettünk állományaink biztonsága érdekében.
Érdemes tehát legalább a legegyszerűbb eszközöket arra használni, hogy adatainkat biztonságba helyezzük. Azt azonban nem árt átgondolni, hogy melyik médiára milyen típusú adatokat írunk ki. Ha a fényképalbumunkat szeretnénk archiválni, akkor arra nem érdemes többször írható médiát használni, ezeket ugyanis egyszer kiírjuk, utána maximum visszanézzük a képeket. Itt jegyezném meg, hogy az írható *CD*-n és *DVD*-n elhelyezett adatainkat is érdemes néha újra és újra megnézni, mert sajnos az idő múlásával ezek a lemezek veszítenek minőségükből, így elképzelhető, hogy egy-egy fényképgyűjtemény a kukába kerül, mert négy-öt év múlva egyszerűen nem fogjuk tudni visszaolvasni az állományokat. Persze okos ember erre is felkészül, néha újra kiírja az ilyen fontos anyagokat, továbbá az is jó megoldás lehet, ha több másolatot is készítünk egy-egy lemezből.
A fényképekkel, családi videókkal ellentétben az általunk készített dokumentumokat érdemes olyan médiára írni, amelyik többször írható. Ez azért jó megoldás, mert általában a dokumentumokat tartalmazó könyvtárunk nem olyan struktúrában készül el, hogy magán a könyvtárstruktúrán követni lehessen az idő változását, így két adott időpont között készült állományok mentése problémás lehet. De ha ügyesek voltunk és lementettünk, akkor egy ilyen formában készült mentésen bármit megtalálni az egy újabb szép kihívás lesz.
Érdemes tehát ilyenkor az adott könyvtárstruktúrát, vagy annak részleteit egyben időnként egy többször írható lemezre menteni. Ez a többször írható lemez lehet egy *CD-RW*, egy *DVD-RW* lemez, de talán praktikusabb egy *DVD-RAM*-ot, vagy külső merevlemezt, esetleg flash meghajtót használni. Hogy miért? Rengeteg időt megspórolhatunk több gigabájt mentésekor, ha úgy készítjük a mentést mondjuk egy második merevlemezre, hogy a mentett dokumentumok mappánkat oly módon írjuk felül az eredeti dokumentumok mappával, hogy ott csak a módosult



állományok kerüljenek mozgatásra. Ezzel rengeteg időt megspórolhatunk például egy *USB*-n csatlakoztatott külső merevlemez esetén.

A RAID

Ugyan nem kapcsolódik szorosan mostani témánkhoz, de mindenképpen meg szeretném említeni, hogy a biztonságos adattárolás bizonyos esetekben semmit nem ér anélkül, hogy magán a gazda számítógépen is biztonságban legyenek az állományok. Erre nyújt megoldást a redundáns lemezkezelés, ismertebb nevén a *RAID (Redundant Array of Independent Disks)*. A *RAID* egy nagyon jó megoldás abban az esetben, ha a gazda számítógépünkön – például egy kiszolgálón – elviselhetővé szeretnénk tenni azt a problémát, amit egy merevlemez meghibásodása, kiesése okozhat. A *RAID* éppen arra nyújt különböző konfigurációkban megoldást, hogy egy, vagy több lemez kiesése esetén is működőképes maradjon a rendszer.
Gyorsan nézzük át, hogy melyek a legnépszerűbb *RAID* konfigurációk:

RAID 0

Napjainkban már sok középkategóriás alaplap is támogatja a *RAID 0* konfigurációt, ám attól eltekintve, hogy a neve *RAID*, nem sok haszna van a biztonságos adattárolás területén. A *RAID 0* konfiguráció arra használható, hogy több független lemezt úgy fűzzünk össze, hogy azok a végén az operációs rendszer számára egy logikai lemezként látszódjanak. Így a sok kis merevlemezünkből tudunk egy nagy logikai lemezt készíteni. Ám bármelyik lemez kiesése esetén az egész tömb használhatatlanná válik.

RAID 1

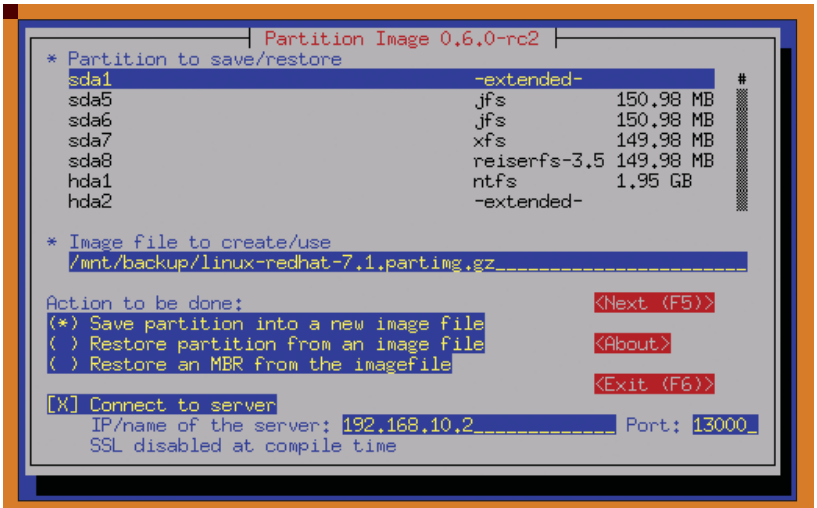
Talán a legegyszerűbb *RAID* konfiguráció. Szintén sok középkategóriás alaplap támogatja, valamint nagyon jó támogatással rendelkezik a Linux rendszermagba építve is. A módszer lényege, hogy veszünk legalább két ugyanolyan lemezt, vagy szoftveres megoldás esetén ugyanakkora partíciót és ezeket úgy használjuk, hogy tükrözzük egymásra az adatokat. Így tehát egy elmentett állomány nem csak az egyik lemezen kerül elmentésre, hanem a tömb minden lemezén. Az eredmény, hogy attól függően, hogy hány lemezt tettünk a tömbbe, n-1 lemez kiesés esetén a rendszer még használható. Ellenben a tömb teljes tárterületeként annyit látunk csak, mintha egy lemezt használnánk csak. Ez a biztonság ára.
A *RAID 1* egy nagyon egyszerű és könnyen használható megoldás, akár hardveres, akár szoftveres megoldás esetén.

RAID 10

A *RAID 10* a *RAID 0* és *1* ötvözése, oly módon, hogy a *RAID 0* által létrehozott logikai meghajtókat tükrözéssel teszi redundáns állományrendszerré. A *RAID 10* egy jó megoldás volna, de van ennél egy sokkal frappánsabb és olcsóbb megoldás.

RAID 5

A *RAID 1* mellett a legelterjedtebb redundáns megoldás. Óriási előnye a *RAID 1*-hez képest, hogy amellett, hogy redundáns, tetszőlegesen nagy logikai lemezek hozhatók létre a segítségével. A *RAID 5* konfigurációhoz legalább három darab egyforma lemezre van szükségünk és ezt a három lemezt fűzzük össze egy logikai meghajtóvá oly módon, hogy a teljes tárterület kétharmada fog rendelkezésre állni, mint szabad terület. A működés egy zseniálisan egyszerű ötleten alapul, a logikai *XOR* (kizáró vagy) műveletet használja. Vesz két bitet, amit el akar tárolni, ezeket kiírja az első és a második lemezre, majd a két bit *XOR* művelettel vett eredményét a harmadikra. Ennek a megoldásnak óriási előnye, hogy a *XOR* művelet miatt bármelyik lemez kiesik, a másik két lemezről egyértelműen előállítható az eredeti tartalom.
A *RAID 5* rendszerbe természetesen nem csak három, hanem annál több lemez is összefogható egy logikai egységgé, valamint a redundancia



1. ábra Partimage – Mentési folyamat indítása

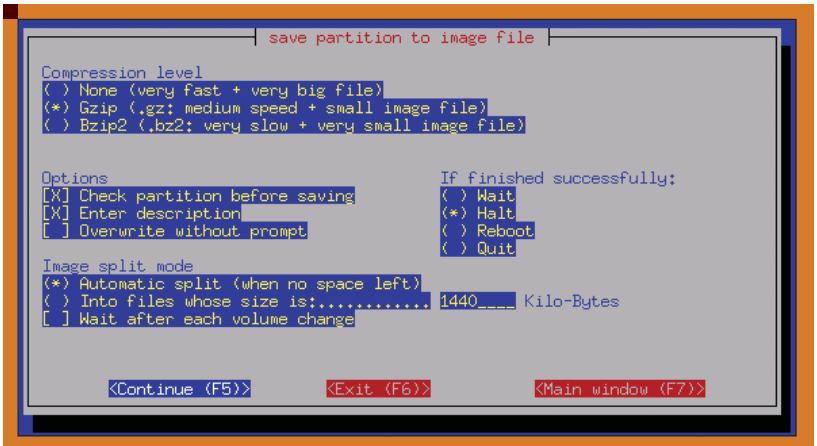
mértéke növelhető úgynevezett spare, tartalék lemezek beállításával. A *Linux* kernel ezt a *RAID* megoldást is támogatja rendszermag szinten, így szoftveres *RAID 5* lemezeket is használhatunk. Ilyenkor mindössze arra kell figyelni, hogy a *Linux* rendszerek jelenleg nem tudnak szoftveres *RAID 5*-ről indulni, ezért létre kell hoznunk egy kisebb partíciót a rendszertöltő és a rendszermag számára. Számolni kell továbbá azzal a plusz terheléssel, amit a bitenkénti *XOR* művelet jelent a processzor számára.

Biztonsági mentés Linux alatt

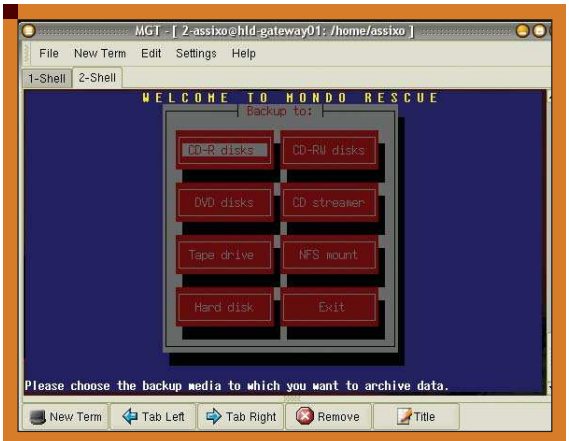
Térjünk rá a cikk főtémájára, a linuxos adatmentő szoftverekre. Két fő szoftvercsoportot fogunk megvizsgálni, mindkettőnek egy-egy képviselőjét. Az első csoport azokat a szoftvereket foglalja magában, amely szoftverek segítségével egy fizikai lemezről olyan képállományt tudunk készíteni, amelyet utána bármely másik számítógépre vissza tudunk tölteni. A második csoportba pedig azok a programok tartoznak, amelyek segítségével élő rendszereket tudunk lementeni úgy, hogy ahhoz semmilyen szolgáltatást nem kell kell leállítani.

Partimage

Bizonyára sokan ismerik és talán többen használták is már korábban a *Symantec*-féle *Ghost*-ot arra, hogy számítógépükről biztonsági mentést készítsenek. Ezzel a programmal könnyedén meg lehetett csinálni, hogy időnként az ember elmentette a számítógép teljes tartalmát és amikor valami probléma előjött, akkor gyorsan, percek alatt visszavarázsoltunk egy korábban készített mentést és ezzel megúsztunk egy órákig, vagy éppen napokig tartó újratelepítést. Nos a *Ghost* annak idején egy nagyon jópofa szoftver volt, amelyből aztán egy meglehetősen komoly üzleti rendszert fejlesztett a *Symantec*, csak éppen a magánszemélyek és kisebb vállalkozások számára nem biztos, hogy megfizethető áron.
Persze elkeseredésre most sincs semmi okunk, mert itt van nekünk a *Partimage*! Ez a program tökéletesen helyettesíti az előbb említett termékcsaládot és mint az egy jó linuxos rendszerhez illik nyílt forráskódú és ingyenesen használható.
Működését tekintve a *Partimage* legfontosabb szolgáltatása, hogy számos különböző állományrendszerről képes úgynevezett lenyomatot (image) készíteni, ezt tetszőleges lokális, vagy hálózati erőforráson elhelyezni, illetve ilyen image-ből a lementett rendszert percek alatt visszatölteni.
A programhoz tartozik egy nagyon jól használható konzolos felhasználói felület, így lépésről lépésre tudjuk beállítani a mentés folyamatát, de természetesen a program a megfelelő paraméterekkel parancssorból is futtatható, ennek megfelelően nagyon jól szkriptelhető.
A program indítása után kiválasztjuk, hogy melyik partíciót szeretnénk menteni és megadjuk, hogy hol szeretnénk ezt az állományt elhelyezni. Erre rögtön két mód is kínálkozik, az első, hogy egy a lokális állományrendszeren megadunk egy könyvtárat és egy állományt, vagy pedig egy távoli kiszolgálón helyezzük el a mentést. Utóbbi esetben több dologra is szükségünk van. Először is telepíteni kell a *partimaged* csomagot a kiszolgálóra, ahová a mentést tenni szeretnénk, valamint rendelkeznünk kell az adott gépre olyan hozzáféréssel, hogy ezt a mentési műveletet végrehajtsuk. Távoli kiszolgáló használata esetén lehetőség van arra, hogy *SSH* segítségével titkosítsuk a kommunikációs csatornát. Tulajdonképpen ez a megoldás pofon egyszerűen működik, bárki percek alatt elsajátítja a használatát, ám mégsem biztos, hogy ez a legjobb megoldás, ugyanis meglehetősen lassú ez a mentési mód. Amennyiben lehetőségünk van rá, kiszolgálóra való mentéshez lokális hálózaton inkább *NFS* megosztást használjunk, csatoljuk be a megosztást a megfelelő helyre



2. ábra Partimage – Mentési paraméterek beállítása



3. ábra Mondoarchive – Válasszunk médiát

tömegesen állítsunk vissza, vagy telepítsünk fel munkaállomásokat. Olyan helyen, ahol sok egyforma *PC* van, nagyban megkönnyíti a program a rendszergazda munkáját.
És hogy miként tudjuk a rendszert például *Windows* munkaállomások mentésére használni anélkül, hogy *Linux* telepítenénk a gépekre? A megoldást a *Knoppix LiveCD* jelenti, ezen megtalálható a `partimage` program és egy megfelelő *NFS* kiosztással már indulhat is a telepítés.

Éles rendszer mentése

A *Partimage*-el az előbb megtanultunk könnyen és gyorsan biztonsági mentést készíteni a munkaállomásunkról, most viszont nézzük meg, hogy hogyan tudjuk lementeni a szervereinket úgy, hogy rendszerösszeomlás esetén is perceken, de legalábbis órákon belül újra működőképesek legyünk.
Linuxról biztonsági mentést készíteni nem egy nagy ördöngösség, egy egyszerű másolással is meglehetősen jó mentést lehet készíteni, a tar program használatáról nem is beszélve. Azonban vannak már olyan programok, amelyek az esetlegesen zárolt állományok mentését, vagy a mentés alatti változásokat is jól kezelik, arról nem is beszélve, hogy akár egy tucatnyi mentési médiát is támogatnak.
Az én kedven ilyen programom a *Mondo*. Amellett, hogy egy nagyon barátságos és könnyen használható felhasználói felület tartozik hozzá, amely mellett természetesen nagyon jól kezelhető parancssorból is, rengeteg mentési módot ismer. Tud menteni szalagos meghajtóra, *CD*-re, *DVD*-re, *ISO* állományba, *NFS*-re és természetesen lokális állományba is. Emellett elég jól állítható a menteni kívánt és a mentésből kihagyni kívánt állományok és könyvtárak listája.
A program két részből áll, a `mondoarchive` és a `mondorestore` parancsokkal indítható. A `mondoarchive` rész való a rendszerünk mentésére, míg a `mondorestore` a visszaállítási folyamatokhoz.
A `mondoarchive` indítása után azonnal kiválaszthatjuk, hogy hová szeretnénk a mentést készíteni. Én legtöbb esetben a kiszolgálókat szalagos meghajtóra mentem, szerintem ez a legjobb módja a biztonsági mentés készítésének, de természetesen bármely más módozat is választható.
Miután kiválasztottuk a médiát a mentéshez, a program rákérdez, hogy milyen tömörítési szintet szeretnénk használni. Itt is elmondható, amit a `partimage`-nél már említettem, ha van idő a mentésre, akkor érdemes a nagyobb tömörítést választani.
Ha ezzel megvagyunk, akkor meg kell adni a menteni kívánt könyvtárstruktúra kezdőpontját. Egy teljes mentés esetén ez a gyökérkönyvtár, tehát a /.
Következő lépésben megadhatjuk azokat a könyvtárakat, amelyeket az előbb megadott struktúrán belül nem szeretnénk menteni. Ilyenek lehetnek az ideiglenes állományok tárolására szolgáló könyvtárak, vagy a töménytelen mennyiségű zenét, filmet tartalmazó könyvtárak.
Ha ezzel is végeztünk, a program rákérdez, hogy szeretnénk-e a mentés végeztével ellenőrzést tartani a biztonsági mentés állapotáról. Ez ugyan tovább növeli a mentés idejét, de mindenképpen megéri elvégezni ezt a folyamatot, ugyanis rossz dolog, ha egy mentés használhatatlanságával akkor szembesülünk, amikor használni kéne.
Következő lépésben arról kell nyilatkoznunk, hogy szabványos

4. ábra Így indul az Unison – Válasszunk profilt

rendszermagot használunk-e. Mivel én a legtöbb gépen saját általam fordított kernelt szoktam használni, én itt általában nemmel felelek, de aki gyári kernelt használ, nyugodtan nyomjon igent. Erre azért lesz a későbbiekben szükség, mert szalagos mentés készítése esetén a `mondo` készít egy indítólemezt, amellyel egy rendszerösszeomlás esetén a gépet indítva szinte egy gombnyomásra vissza lehet állítani a rendszert. Így tehát fontos, hogy az indítólemezre olyan rendszermag kerüljön, amivel a gép el tud indulni. Ezután már nincs is más hátra, mint elindítani a mentést. Miután lefutott a mentésünk, a rendszer végrehajtja az ellenőrzést a mentett állományon és az esetleges eltéréseket, amelyek a mentés közben keletkeztek kilistázza nekünk.

CD-re, vagy *DVD*-re való mentéskor lehetőségünk van arra, hogy a `mondo` olyan lemezeket készít, amelyekről a gépünk el tud indulni, így a lemezt a gépbe helyezve szintén egy gombnyomásra indítható a teljes rendszer-visszaállítás.

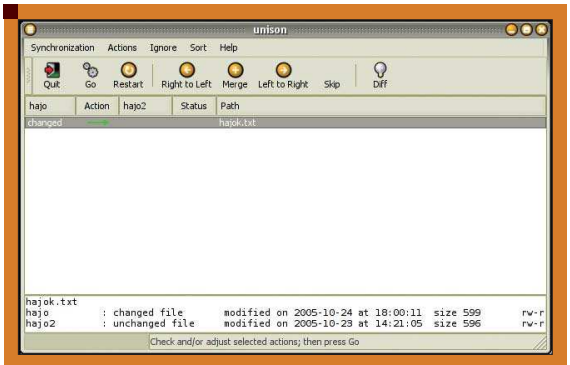
A rendszer visszaállítása

Az előbb már említettem, hogy kazettás és *CD*/*DVD*-s mentés esetén pofon egyszerűen tudunk teljes rendszer-visszaállítást indítani. Ám mi van akkor, ha csak néhány állományt szeretnénk visszatölteni a kazettáról, vagy a *CD*-ről. Nos, ehhez indítsuk el a `mondorestore` programot.

Itt megint, ahogy a mentésnél is, meg kell adnunk, hogy milyen médiát szeretnénk használni. Ezek után meg kell adni, hogy melyik könyvtárba szeretnénk a visszaállított állományokat tárolni és egy fa struktúrából ki kell választani, hogy melyik könyvtárakat szeretnénk visszaállítani. A könyvtárak kiválasztása után lehetőségünk van egyesével megadni, hogy az adott könyvtárakon belül melyik állományokat szeretnénk visszanyerni a mentésből. Ha ezzel is megvagyunk, akkor már csak idő kérdése és az állományok ott lesznek a megadott könyvtárban. Egyszerű és tényleg nagyszerű. Szerencsére eddig összesen egyszer kellett a mentésekhez nyúlnom rendszer-visszaállítás céljából, de akkor nagyon jól jött, hogy ilyen egyszerűen használható rendszert választottam.

Egyéb megoldások adatok mentésére

Most, hogy megnéztük, hogy milyen megoldások vannak munkaállomások és szerverek mentésére, be szeretnék mutatni még egy megoldást, amelyet a hordozható számítógépek tulajdonosai használhatnak előszeretettel.



5. ábra Unison munka közben

A program neve ***Unison*** és arra használható, hogy egy hálózati erőforrás és egy lokális mappa állományait szinkronban tartsuk vele.

Tegyük fel, hogy van egy céges hálózati dokumentumok mappánk, amit az egész cég használ arra, hogy abban közös használatú állományokat helyezzen el. Azonban mi szeretnénk, ha dokumentumaink nem csak akkor állnának rendelkezésünkre, ha az irodában vagyunk, hanem kapcsolat nélkül is szeretnénk azokat használni. Mit tudunk ilyenkor tenni? Mondjuk minden alkalommal lemásoljuk a mappa teljes tartalmát és amikor visszaérkezünk az asztalunkhoz, visszamásoljuk a módosított állományokat. Végül is ez is egy megoldás, ám elég körülményes lekezelni azt, ha közben a hálózati mappa tartalma is megváltozott.

Az ***Unison*** éppen erre nyújt megoldást, végignézni a két könyvtár tartalmát és ahol eltérést talál valamelyik oldalon azt jelzi a felhasználónak. Ahol pedig olyan eltérést talál, hogy mindkét oldali állomány megváltozott, ott lehetőségünk van arra, hogy megnézzük a két állomány közötti különbséget. Természetesen ennek legnagyobb haszna a szöveges állományoknál van, mint például egy ***HTML*** kód, vagy egy ***XML*** fájl. Egy ***OpenOffice*** dokumentum, vagy egy ***PDF*** állomány kódjának összehasonlításából kevés ember tud mélyreható következtetéseket levonni.

A működés elve nagyjából hasonló a *CVS*-hez, mintha minden állomány bináris formában töltenénk be egy *CVS*-be.

A program ismeri a lokális könyvtárak szinkronizációját, így egy becsatolt ***NFS*** mappával való szinkronizáció pofon egyszerű. A program ezen felül azonban tud több különböző protokoll felett is szinkronizálni, így ismeri például az *SSH*-t. Ez abból a szempontból nagyon jó dolog, hogy így például otthonról is biztonságos csatornán le tudjuk szinkronizálni az állományainkat.

A program meglehetősen stabilan működik, beállítása és használata nagyon egyszerű. ***Gnome***-ot használó felhasználóinak pedig azzal kedveskedik, hogy egy nagyon kulturált ***GTK2***-es felhasználói felülettel együtt is telepíthető. Szerintem érdemes kipróbálni ez a segédprogramot is.

Zárásként megjegyezném, hogy természetesen a fenti három programnál jóval több megoldás létezik mentések készítésére, én ezt a hármat azért emeltem ki, mert rendszeresen használom őket és működésükkel nagyon meg vagyok elégedve. Természetesen most is érvényes a mondás, érdemes minél több dolgot kipróbálni, hogy ki tudjuk választani azokat az eszközöket, amelyek számunkra és az adott feladathoz a legjobban használhatóak.

Kezdésként talán érdemes az ***LVM*** állományrendszerrel és annak a segédprogramjaival is megismerkedni, mert sok érdekes megoldást rejtenek, akárcsak a ***CODA*** fájlrendszer az ***NFS*** és az előbb említett ***Unison*** kiváltására. Bár megjegyezném, hogy a *CODA*-val még voltak rossz tapasztalataink, ami a rendszert stabilitását illeti. Mindenesetre ezek is olyan projektek, amelyekre érdemes odafigyelni és fejlődésüket követni.

Illés Viktor
(illes.viktor@assixo.com)
Mérnök informatikus, az Assixo Kft. munkatársa.

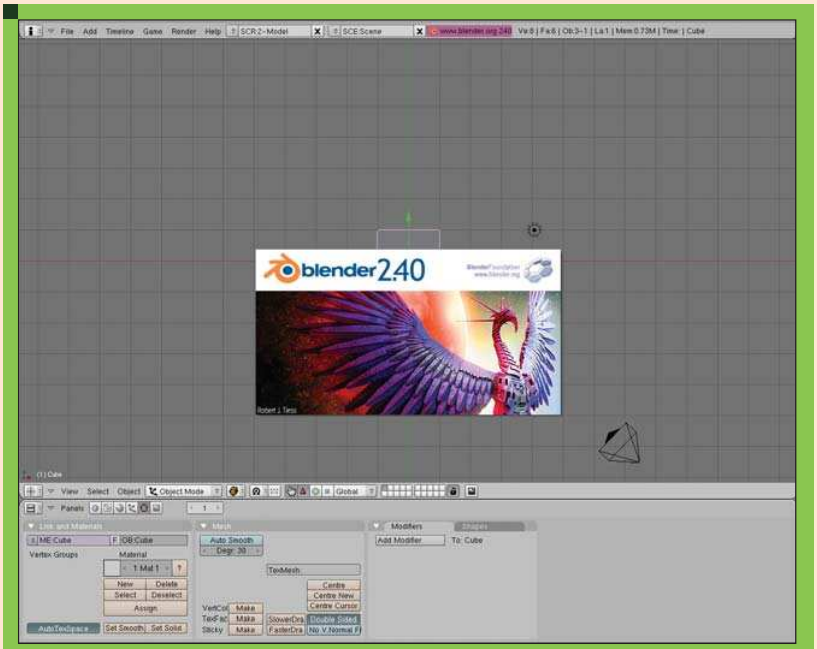
Térhatás

A Blender használata (6. rész)

Modellezési technikák haladóknak

Új év, új kiadás. Nem is akármilyen. A Blender 2.40-es verziója talán az eddigi legtöbb újítást és hibajavítást tartalmazó kiadás, mióta a világ több dimenziós. Bár első ránézésre nem sokat változott, alaposabban körülnézve észrevesszük, hogy sok apróságot megváltoztattak, és belül szinte teljesen újraírtak egyes részeket – ezt köszönhetjük többek közt a Google Summer Of Code fedőnevű akciójának is – így azt hiszem nem hazudok, ha azt mondom: a 2.40-es Blender az eddig megjelent legstabilabb és leggyorsabb verzió.

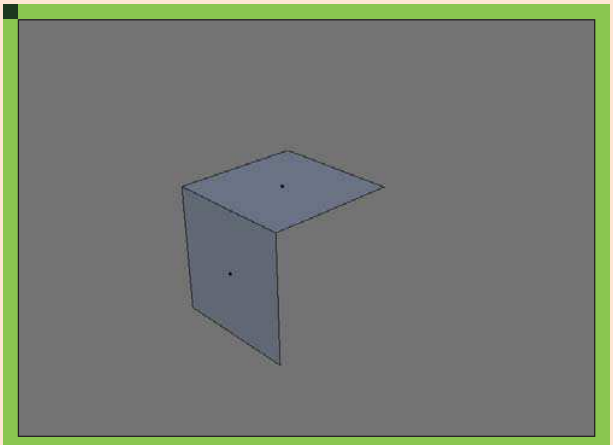
Aki még nem töltötte le, mindenképp tegye meg, mert megéri. Ezt megtehetjük a ***Blender*** honlapján (➲ ***http://www.blender.org***), ahol megtalálhatjuk az újítások teljes listáját is. Folytatjuk a januárban megkezdett modellezgetést, további eszközöket, módszereket próbálok majd bemutatni, remélhetőleg a hasznosabbak közül néhányat. Ehhez mindenképp ajánlott letölteni az új verziót, az itt leírtakat ugyanis én már abban fogom végezni.



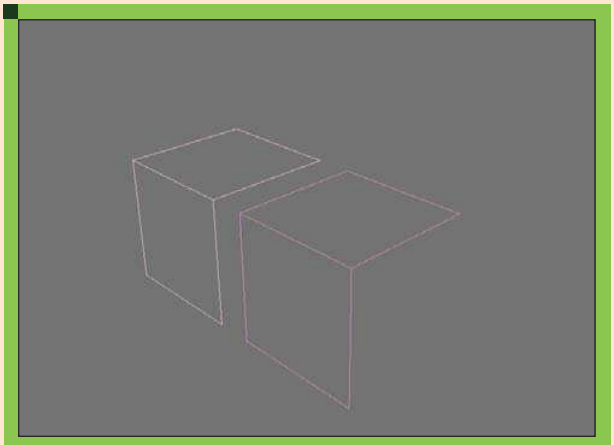
1. ábra Az új Blender

Szimmetrikus modellezés

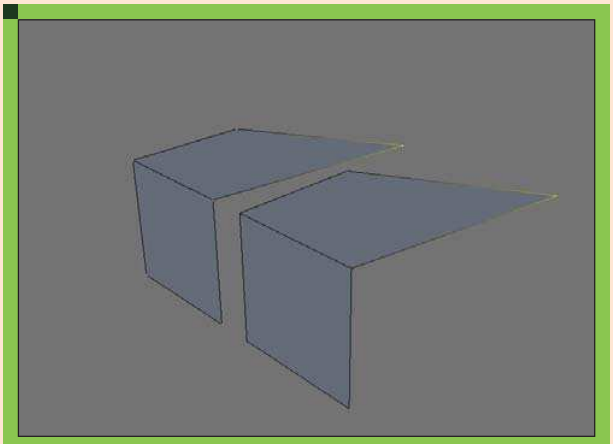
Gyakran előfordul, hogy szimmetrikus alakzatokat szeretnénk készíteni. Ha egyszerűbb modellről van szó, könnyen megoldhatjuk: lemodellezzük az egyik oldalt, majd duplikáljuk, vagy körszimmetrikus alakzatoknál használjuk a ***Spin Dup*** funkciót. Mi azonban nem szeretjük az egyszerű alakzatokat, (ugye nem?) inkább azt



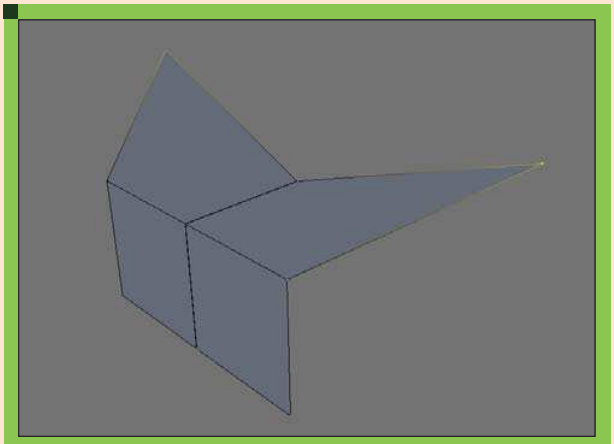
2. ábra Egy fél akármi



3. ábra Két fél akármi



4. ábra A két mesh ugyanaz

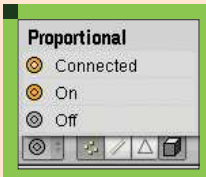


5. ábra Kész a szimmetrikus alakzatunk

szeretnénk, ha szerkesztés közben látszódna minkét (mindhárom, vagy amennyit épp szeretnénk) oldal, sőt, egyik oldalt szerkesztve a másik is magától módosuljon. Hozzunk létre egy „fél akármit” (*2. ábra*), majd hogy ne legyen egyedül, rögtön szaporítsuk (*3. ábra*). Az egyszerű ***duplicate*** funkció (***Shift+d***) helyett most azonban használjuk az ***Alt+D*** billentyűkombinációt. Talán még emlékszünk rá, hogy egy mesh több objektumhoz is tartozhat, ilyenkor csak az objektumok mérete és helye/helyzete változik külön-külön, ha azonban az egyik mesh-t módosítjuk, a másik is változik. (*4. ábra*) Ezek után tulajdonképpen le sem kellene írnom, olyan egyszerű a megoldás, de jószívű leszek és megteszem. Az egyik objektum X (vagy éppen Y) irányú méretét a -1 (mínusz 1)-szeresére változtatjuk. Ezt többek között úgy is megtehetjük, hogy megnyomjuk az ***S*** – mint ***Scale*** – majd az X (vagy Y) gombot és beütünk -1-et. Már csak annyi maradt hátra, hogy egymás mellé toljuk a két „fél akármit”, hogy legyen egy egész akármink, aminek egyik felét szerkesztve a másik fél is változik (*5. ábra*).

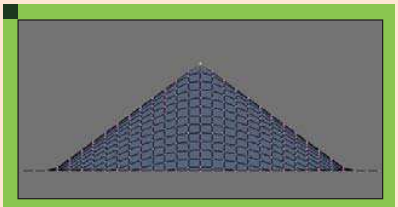
Proportional editing

Talán „arányos” szerkesztésnek lehetne fordítani, de bárhogy is nevezzük, a funkciója akkor is könnyen meghatározható: „hepék” illetve „hupák” szerkesztése (testvérem definíciója szerint a hupa befelé süllyed, a hepe pedig domborodik). Bonyolultabb mesh-ek módosítása elég nehézkes lehet. Sok vertexet kell megmozgatni, egyiket jobban, másikat kevésbé, mindezt egyesével, ráadásul kézzel, végül a felület nem lesz elég sima. Ezt a fajta problémát hivatott megoldani ez az eszköz. Leginkább egy mágneshez lehetne hasonlítani. Mikor elmozdítunk egy vertexet, annak „mágneses tere” magával húzza a szomszédos vertexeket. A közelieket jobban, a távoliakat kevésbé, a még távolibbakat egyáltalán nem. Bekapcsolni ***edit mode***-ban, a ***3D View*** fejlécén tudjuk (*6. ábra*). A ***On*** opciónál a mozgatás az összes közeli vertexre hatással lesz, ha viszont a ***Connected***-et választjuk, csak azokra, amelyek valamilyen módon kapcsolódnak, épp elmozduló vertexhez. Amint bekapcsoljuk, megjelenik egy újabb opció, az úgynevezett ***falloff*** (*7. ábra*). Itt megadhatjuk, hogy a mozgatott vertextől távolodva hogyan csökkenjen a „mágneses tér” ereje. Csökkenhet lineárisan, szinuszosan, stb. A *8-11. ábrákon* ezek közül az opciók közül láthatunk néhányat. Ennek a bizonyos „mágneses mezőnek” természetesen van határa is, nem akármilyen messze működik. Szerkesztés közben az egér görgőjével állíthatjuk a méretét.

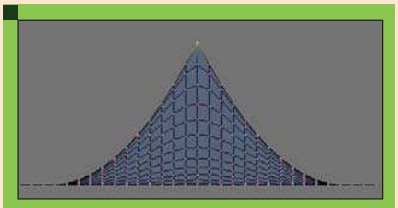


6. ábra A proportional edit bekapcsolása

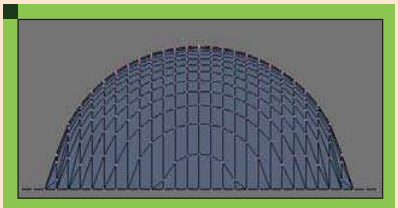
7. ábra A Falloff beállítása



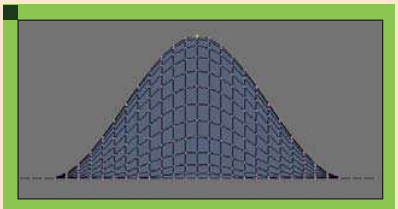
8. ábra Linear falloff



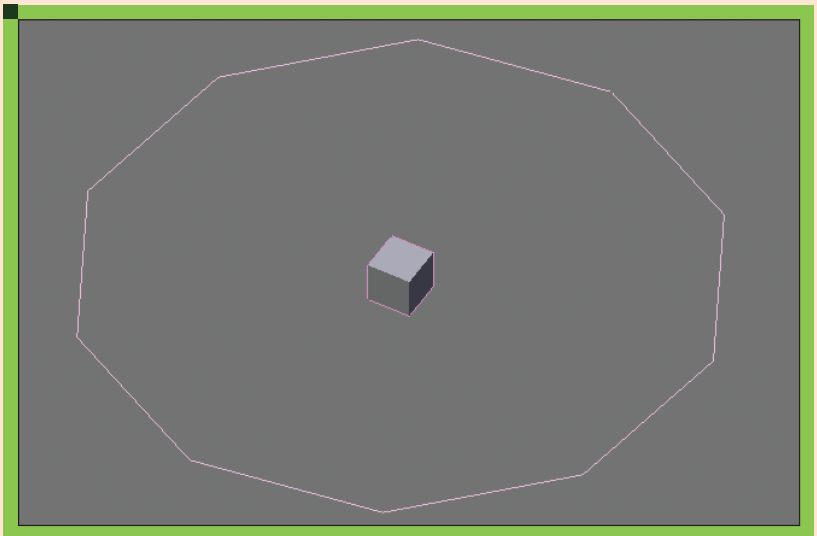
9. ábra Sharp falloff



10. ábra Sphere falloff



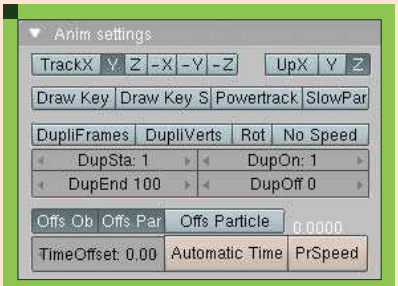
11. ábra Smooth falloff



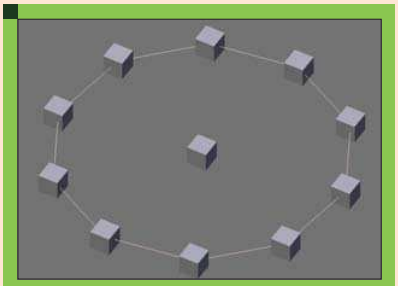
12. ábra Előkészületek

DupliVerts

Ha más a többszörözésnél tartunk, vessünk egy pillantást a ***DupliVerts***-re. Nem kell megijedni, a név körülbelül annyit tesz *„**DUPLIcate at VERTiceS**”*, magyarul vertexenkénti többszörözés. Mindjárt meg fogjuk érteni. Hozzunk létre egy kört (***Space billentyű -> Add -> Mesh -> Circle***). Ne álljon túl sok vertexből, 10-15 bőven elég lesz. Tegyünk mellé egy kockát is (***Space -> Add -> Mesh -> Cube***), majd a kocka szülőjeként (parent) állítsuk be a kört. Ehhez jelöljük ki mindkettőt – a kockával kezdve – majd nyomjuk meg a ***Ctrl+P*** billentyűt. Ha ez is megtörtént, csökkentsük a kocka méretét körülbelül a tizedére. Az előkészületekkel meg is vagyunk (*12. ábra*). Jelöljük ki a kört, keressük meg a gombok között a *13. ábrán* látható panelt, majd nyomjuk meg a ***DupliVerts*** gombot. Az végeredmény a *14. ábrán* látható. A következő történt: A ***DupiVerts*** bekapcsolásával arra utasítottuk a ***Blendert*** hogy a kör minden vertexe helyére rajzolja ki a objektum gyermekobjektumait, jelen esetben a kockát. Ilyenkor egyik eredeti objektum sem látszik a renderelt képen, sem az amelyiket többszörözzük, sem annak szülője, csak a duplikátumok. Ha valami nem úgy működne, ahogy szeretnénk, ellenőrizzük a szülő-gyermek viszont – akit többszörözni szeretnénk, ő legyen a gyermek, nem pedig fordítva – és ne felejtsük el, hogy a szülő objektumon kell bekapcsolni a ***DupliVerts*** gombot.

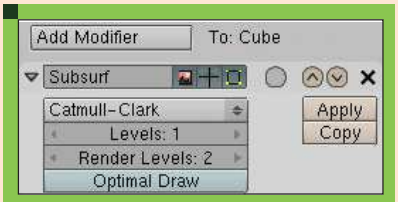


13. ábra Anim Settings panel

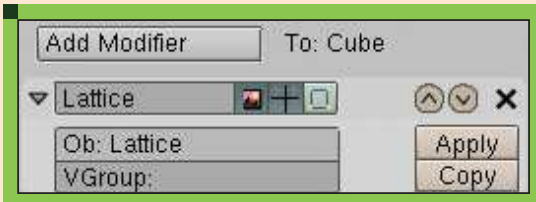


14. ábra DupliVerts

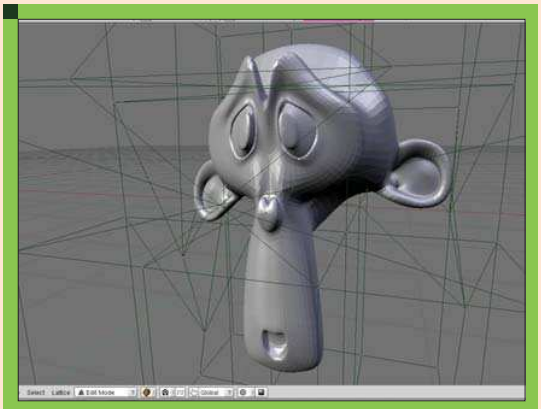
A ***DupliVerts*** mellett találhatunk egy másik gombot, a ***DupliFrames***-t is. Ennek funkciója hasonló (többszörözés), azonban nem vertenként többszöröz, hanem animációkor képkockánként, ezért itt nem kell szülő objektum sem. Mivel az animációval részletesebben csak később foglalkozunk, ezért a funkció részletes bemutatása most nem vállalkozom. Azonban ha van elég fantáziánk, a ***DupliFrames*** az egyik legjobban használható funkcióvá válhat a kezünk alatt.



■ ***15. ábra*** A SubSurf modifier



■ ***16. ábra*** A Lattice modifier



■ ***17. ábra*** A Lattice modifier működés közben

Modifiers

A ***Blender*** legnagyobb hiányosságának legtöbbször azt hozzák fel, hogy nehéz a kezelőfelületen eligazodni vagy egy adott gombot megtalálni. A 2.40-es verzió egyik újítása, hogy az egymáshoz hasonló funkciókat megpróbálták csoportosítani. Így jött létre például a ***Physics buttons***, a ***Contraints*** panel és a ***Modifiers*** panel is. Most csak ez utóbbiról lesz szó, mivel ez tartozik szorosan a modellezés témaköréhez. A ***modifier*** szó magyarul módosítót jelent, tehát objektumainkhoz különböző módosítást végző funkciókat rendelhetünk hozzá. A régebben már bemutatott ***SubSurf*** is ebbe a menübe került, így a régi helyén hiába is keressük. Egy objektumot kijelölve, majd az Add ***Modifier*** gombra kattintva válasszuk ki a ***SubSurf*** opciót. Rögtön meg is jelenik a *15. ábrán* látható néhány gomb. Felül található a ***modifier*** neve, amit szabadon megváltoztathatunk. Tőle jobbra három gomb található, amivel bekapcsolhatjuk, mikor legyen érvényes ez a bizonyos módosító. Az elsővel a rendereléskor, a másodikkal ***object mode***-ban, a harmadikkal pedig ***edit mode***-ban kapcsolhatjuk be-illetve ki. A bal felső sarokban található kis háromszöggel egy sorra összeszűkíthetjük a modifiert. Erre a jobb átláthatóság miatt van szükség, ugyanis egy objektumra egyszerre többet is alkalmazhatunk. Ilyenkor olyan sorrendben hajtódnak végre, amilyen sorrendben a panelen találhatóak. A fel és lefele nyilacskákkal a sorrendet módosíthatjuk, az x-et ábrázoló gombbal pedig törölhetjük a ***modifiert***.

A módosítások nem véglégesek, a ***modifier*** törlésével a mesh visszaáll a módosítás előtti állapotba. Ha mégis szeretnénk véglegesíteni a változást, az ***Apply*** gombbal tehetjük meg. Figyelem! A módosítás tényleg végleges, a ***modifier*** ilyenkor törlődik, csak a módosult mesh marad.

Lattices

A ***SubSurf*** részletezése már egy korábbi számban megtörtént helyette egy másik ***modifiert*** mutatnék be. Ő a ***Lattice*** keresztnévre hallgat – teljes nevén ***Lattice Modifier*** – és ő fog nekünk segíteni eltorzítani néhány mesh-t. Első lépésben szükségünk van egy eltorzítandó objektumra. ***Susan*** (a majom) tökéletesen megfelel a célnak. A következő egy ***Lattice*** típusú objektum hozzáadása. (***Space -> Add -> Lattice***) Ha ez is megvolt, helyezzük mindkettőt középre, majd a ***Lattice*** objektumot növeljük meg annyira, hogy ***Susan*** beleférjen. De mi is ez a lattice? Ha átváltunk edit módba, láthatjuk, hogy valamiféle kockával van dolgunk (egyelőre), első ránézésre csak a színe különbözteti meg egy sima mesh-től. Keressük meg a ***Lattice*** panelt a gombok között, majd az U, V és W beállításokat növeljük meg 4 re. Ezzel a lattice (ami egyébként magyarul rácsot jelent) sűrűbb lett, és mindjárt nem hasonlít annyira egy egyszerű mesh-re. Jelöljük ki ***Susant***, majd adjunk hozzá egy ***Lattice modifiert*** a más ismert módon. Az ott található *„**Ob:**”* nevű mezőbe írjuk be a ***Lattice*** nevét, ami nagy valószínűséggel *„**Lattice**”*. Ezzel meg is volnánk, nincs más dolgunk, mint kijelölni a rácsot, ***edit mode***-ba váltani, és elmozgatni a vertexeket. Ha mindent jól csináltunk, ***Susant*** kedvünkre (de)formálhatjuk (*17. ábra*).

A modifiers menüben még számos eszköz megtalálható, ezek egy része az animációhoz, más része a görbékhez, kapcsolódik. Ezek egyikével sem foglalkoztunk még, ezért ezen eszközök bemutatása valószínűleg kudarcba fulladna. Ennyi tehát egyelőre, a közeljövőben szó lesz a nem mesh alapú modellezésről (görbék, nurbs objektumok, text, stb), majd két hónap múlva végre nekiesünk az animációnak is. Addig viszont még várni kell egy kicsit, és modellezgetni, hogy aztán legyen mit animálni.

Szalai András
(sly87@freestart.hu)

Jelenleg középiskolába jár, ahol informatikát tanul. Jövőre érettségizik. Hobbija a programozás és a biztonságtechnika, és a továbbtanulási szándékai is ilyen irányúak.

KAPCSOLÓDÓ CÍMEK

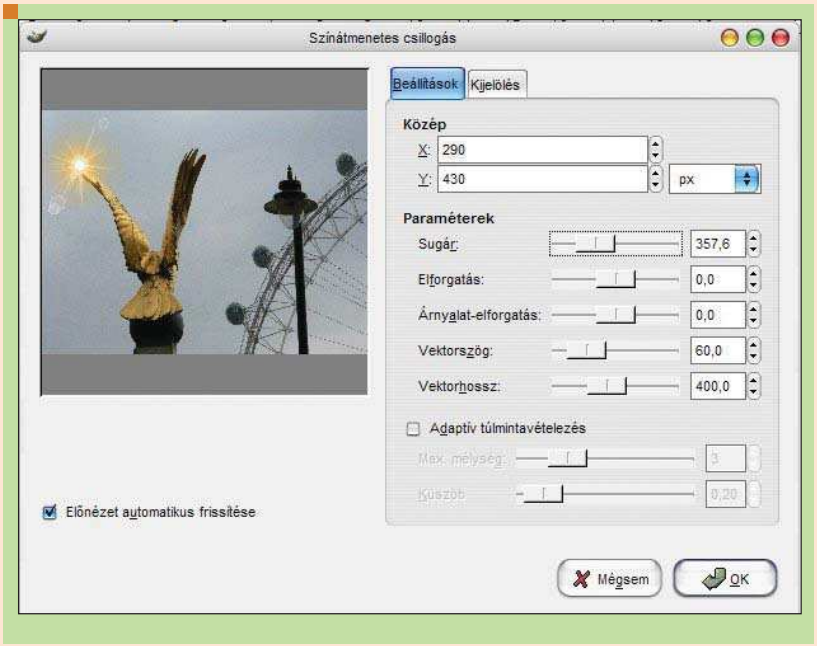
A Blender 2.40 letöltése:
➲ http://www.blender.org/cms/Blender.31.0.html

Az újítások listája (angolul):
➲ http://www.blender.org/cms/Blender_2_40.598.0.html

Gimp bővítmények (2. rész)
Siker, pénz, csillogás

Az elmúlt alkalommal a Gimp kiterjesztései közül a művészi ihletésűekkel foglalkoztunk. Most a fény és árnyék világával, vagyis a fényhatások széles tárházával fogunk elkezdeni ismerkedni.

Kezdjük a *csillogás()* szűrővel a mai ismerkedést (***Szűrők>Fényhatások>Csillogás***)! Valószínűleg nem ez lesz az az eszköz ami megváltoztatja az életünket, de ha egy egy tárgyra vagy fotóra szeretnénk egy becsillanást helyezni utólag, mégis jól jöhet. Fotók esetén inkább annak szoktunk örülni, ha nem csillan be rajtuk semmi.
Ezért joggal merül fel a kérdés, mikor használható ez a szűrő?
Ha egy képünk fényviszonyait az alább ismertetett fényhatások szűrő segítségével átalakítjuk, a térbeli hatás fokozásához hasznos lehet egyegy becsillanást helyezni a tárgyak élére. Szerencsére a csillogás szűrő egyszerű volta miatt beállításokkal nem lesz sok dolgunk, mindössze a becsillanás x és y koordinátáit kell megadnunk és már készen is vagyunk.
Látható, hogy nem a csillogás szűrő lesz az amit a legtöbbet használunk majd. Sokkal inkább szükségünk lenne egy jobban testre szabható eszközre, mellyel a fényhatásokat alakíthatjuk át. Kívánságunkat már valóra is váltották a *Gimp* fejlesztők, hiszen a fényhatások szűrő jóval tágabb teret ad a képzeletünknek.
(***Szűrők>Fényhatások>Fényhatások***)
Elhelyezhetünk képünkön különböző lámpákat, irányíthatjuk a fényeket, megadhatjuk intenzitásukat és színüket is. Mikor lehet hasznos ez az eszköz? Képzeljünk el egy fotót, mondjuk egy csoport képet, amelyen a fények nagyon változatosak.



Szeretnénk ha mindenkit egyformán jól lehetne látni, tehát módosítanunk kéne az egyes részek világosságát. Persze nem ígérem, hogy könnyű ügy lesz részletesen beállítani a megvilágítás paramétereit, de egy próbát biztosan megér!
A szűrő bőséges beállítási lehetőségekkel rendelkezik. A szűrő ablakában bal oldalon találjuk az előnézetet, jobb oldalon pedig fülek segítenek eligazodni a lehetőségek között. Az első fül cselesen ***Beállítások (Options)*** névre hallgat. Azt hiszem az itt található dolgok nem fognak nehézséget okozni senkinek. Ha az átlátszó háttér kiválasztót megjelöljük, akkor a szűrő egy átlátszó hátteret vesz alapul. Ennél sokkal használhatóbb az ***Új kép (Create New Image)*** és a ***Jó minőségű előnézetet (High Quality Preview)*** lehetőség.
A ***Fény*** fül ***(Light)*** alatt már jóval izgalmasabb dolgokat találunk, hiszen itt van lehetőség létrehozni a fényforrásokat. Válasszunk ki egy fényforrást először! Döntsük el, hogy irányított vagy pontszerű fényforrást szeretnénk-e. Ezután jelöljük ki a szívünkhöz legközelebb álló színt és intenzitást. A további paraméterekkel a fényforrásunk helyét változtathatjuk meg. Ha irányított

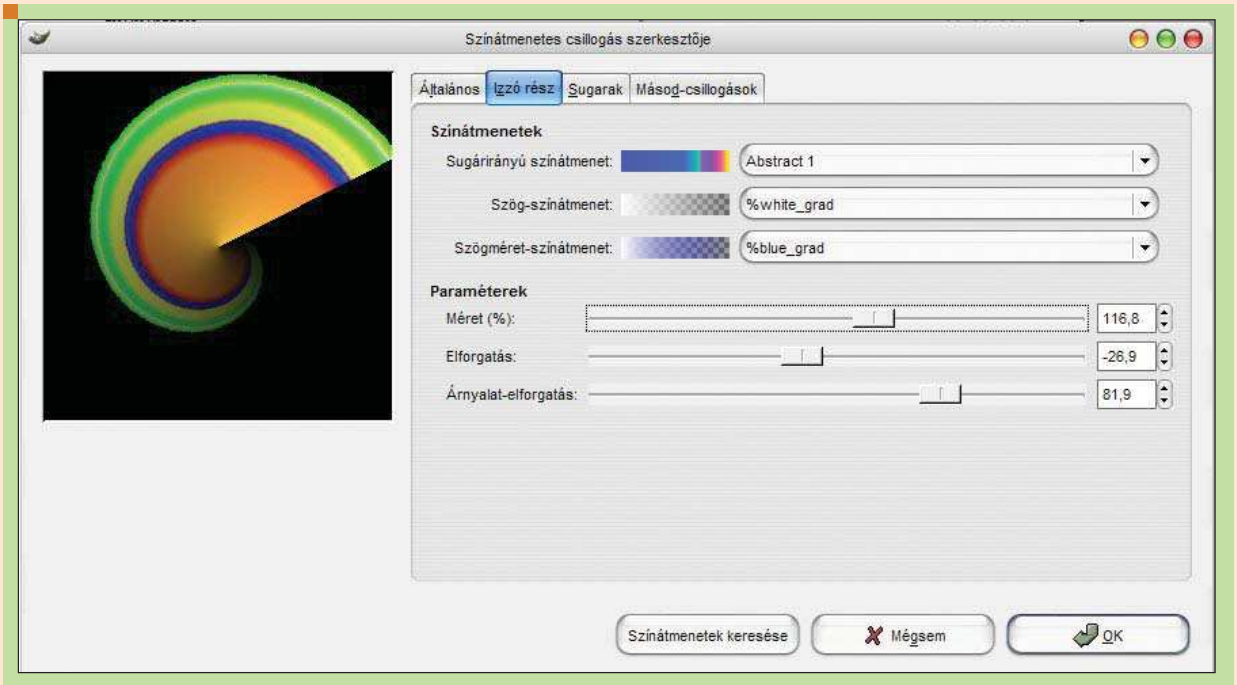
fényforrást választottunk, akkor meghatározhatjuk még az irányt is amerre világítani fog. Az elszigetelés kiválasztót bejelölve utasíthatjuk a *Gimp*-et, hogy az előnézetben csak az aktuális fényforrást mutassa. Ez megkönnyíti a munkánkat ha sok fényforrással dolgozunk már.

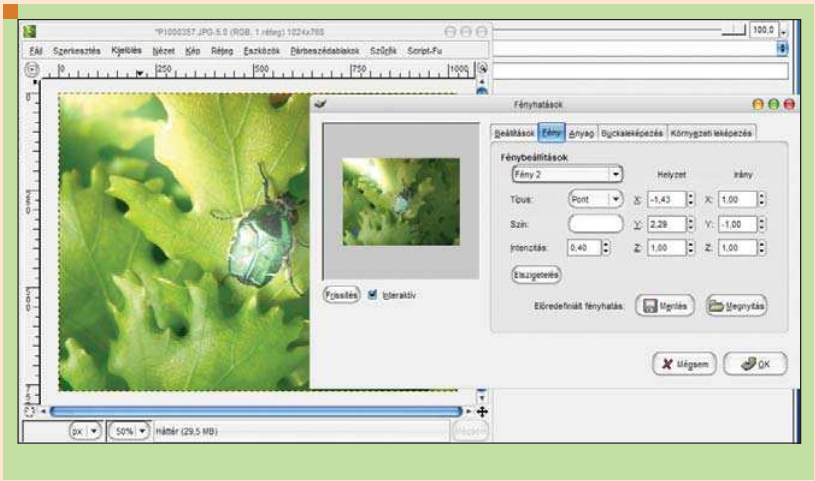
Immár vannak fényeink. A következő fülön találjuk a képünk anyagának beállításait. Ezek mind a fények játékát befolyásolják méghozzá úgy, hogy meghatározzák miként fognak visszacsillanni a fények a képről. A ***Ragyogással (Glowing)*** adhatjuk meg, hogy az eredeti színek mennyire maradjanak meg ott, ahol csak szórt fény vetül a képre. A ***Fényesség (Bright)*** értékkel befolyásolhatjuk az eredeti színek teltségét amennyiben a lámpák direktben megvilágítják a területet. ***Kifényesítettség (Shiny)*** jelenti a világos részek intenzitását és a ***Csiszoltság (Hell)*** magas értékei még jobban kiemelik az amúgy is világos részeket.

Amennyiben a fémszerű felületet szeretnénk látni, jelöljük be a ***Fémszerű (Metallic)*** kiválasztót.

Az első akadályokon immár túl is vagyunk, viszont most jönnek azok a részek, amikre földi halandóként nem biztos hogy egyből ráérzünk minden magyarázat nélkül. A ***Buckaleképzés (Bump Map)*** segítségével

kisebb buckákat hozhatunk létre a képen, vagyis domborúvá tehetjük. Mindössze egy másik rétegen lévő fekete-fehér képet kell megadnunk, amely alapján a szűrő az alacsonyabban fekvő illetve a magasabban fekvő részeket felismeri. Ha már domború képünk van, akkor viszont máshogy fognak megcsillanni rajta a fények. Most már remélem látszik hogy hova lehet kilyukadni ezzel a lehetőséggel! Például ha létrehozunk egy labirintus alakú mintát a (***Szűrők>Megjelenítés>Minta>Labirintus***) szűrővel egy új rétegen, majd pedig ezt a réteget használjuk





fel mint ***Buckaleképzési képet (Bump Map Image)***, akkor az eredeti képünk úgy fog kinézni, mintha egy 3 dimenziós labirintusra lenne rányomtatva, köszönhetően a visszacsillanó fényeknek.

A ***Környezeti leképzés (Environment Map)*** a környezetből visszacsillanó fényt hivatott hasonló módon reprezentálni. Itt már azonban nem egy fekete-fehér, hanem egy ***RGB*** színterű képet vár a program. Ezzel végére is értünk a Fényhatások szűrőnek. Remélem sikerrel gazdagítja majd mindenkinek az eszköztárát.

Jól használható a ***Színátmenetes csillogás (Gflare)*** extrém sport fotókhoz, amikor imitálni szeretnénk a napfény becsillanását az objektíven. Sokszor megtörténik ez imitáció nélkül is, de ha mégsem jött volna össze élőben... Szóval lássuk a szűrő beállításait (***Szűrők>Fényhatások>Színátmenetes csillogás***)!

A beállítások fül alatt adhatjuk meg a csillogás x és y koordinátáit, a sugarat, mely a becsillanás méretét befolyásolja. Itt találjuk továbbá az elforgatás csúszkát mellyel a becsillanás sugarait tudjuk elforgatni. Ennek a beállításnak sok hasznát nem látom, megköthették volna kezünk annyival a fejlesztők, hogy ezt automatikusan választja a program. Az ***Árnyalat elforgatással (Hue Rotate)*** a színeket tudjuk változtatni. A vektorszög beállítással, a becsillanás és a vízszintes tengely közötti szöget állíthatjuk. Vannak különböző fajtájú becsillanások, van olyan, amely egyre kisebb körökből áll. Ezek a körök egy szakaszon elszórva helyezkednek el. Ennek a szakasznak a hosszát állíthatjuk a vektorhosszal. Ha bejelöljük az ***Adaptív túlmintavételezést***, akkor az alatta lévő beállításokkal az élsimítást finomhangolhatjuk.

A ***Kijelölés (Selector)*** fület lehetett volna kicsit szerencsésebben is fordítani, ugyanis ezen a fülön választhatunk a különböző típusú csillogás fajták közül. Megéri végigpróbálni valamennyit! A meglepetés azonban még hátra van! Magunk is készíthetünk új színátmenetes csillogást! Készítsünk is egyet rögtön, majd kattintsunk a ***Szerkesztés*** gombra.

A szerkesztőben hamar rájöhetünk, hogy minden színátmenetes csillogás három különálló dologból áll össze. Először is van maga az ***Izzó mag (Glow)***. Fontos összetevők még a ***Sugarak (Rays)***, majd legvégül a ***Másodcsillogás (Second Flares)*** következik. Az általános beállításoknál megadhatjuk a három rész átlátszóságának mértékét és a módot, ahogy a szűrő az adott részt megrajzolja. ***Normal*** módban a csillogást nem befolyásolja az alatta lévő képrész. ***Hozzáadás (Addition)*** mód választásakor a színek összegződnek, ***Átfedés (Overlay)*** mód esetén a megfelelő sötét illetve világos részek kölcsönösen erősítik és gyengítik egymást. Végezetül ***Képernyő (Screen)*** módban a kép sötét részei felerősítik a csillogás megfelelő világos részeit.

Ezután következik az egyes alkotóelemek viselkedésének aprólékos finomhangolása. Nézzük az Izzó részt!

A beállítási lehetőségek a színekkel kezdődnek. Segítségükkel gyorsan rá lehet jönni, hogy az előnézetben látható körlap hogyan épül fel. Állítsuk a sugárirányú ***színátmenetet()*** ***Abstract1*** értékre, majd a ***Szög-színátmenetet %white_grad*** és végül a ***szögméret-színámnetet %blue_grad*** értékre. Így már tisztábban látható, hogy a három paraméter mit is jelent. A paramétereket is könnyebben értelmezhetjük miután láthatóvá tettük az izzó rész felépítését. Gyakorlatilag a szűrő elkezd rajzolni egy körszeletet. A kiinduló szöget mi adhatjuk meg az ***Elforgatás (Rotation)*** paraméterrel. Az egyes pontok színét pedig a megadott színátmenetek alapján számolja a program. Logikusan minden pont színét két adata határozza meg: a középponttól való távolsága és a kiinduló szöghöz képesti szöge. Hogy azonban tényleg teljes legyen a szabadságunk, a paraméterek között található Árnyalat-elforgatással az egész színtartományt kedvünkre eltolhatjuk.

Ha megértettük az izzó rész beállításait, akkor nem lesz meglepő mindaz amit a sugarak szekcióban találunk. Mindössze két paramétert emelnék ki, mint hasznos segítséget: az ***Ágak-száma (Number of spikes)*** és az ágvastagság fontos szerepet tölt be abban, hogy látványos végeredményt kapjunk.

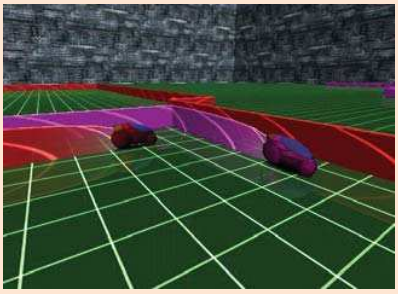
A másod csillogásoknál ne feledjük, hogy a színátmenetek az egyes kis csillogásokra vonatkoznak! A paraméterek segítségével pedig az összes kis csillogás felett rendelkezünk. Mégis, a legizgalmasabb kérdés ezek után következik, ami nem más mint a másod csillogások alakja. A körön kívül választhatunk sokszöget is. Érdekes eredményt kapunk azonban ha például 1-et írunk a szövegmezőbe... A szűrő meglepően értelmezi az egy oldalú sokszöget!

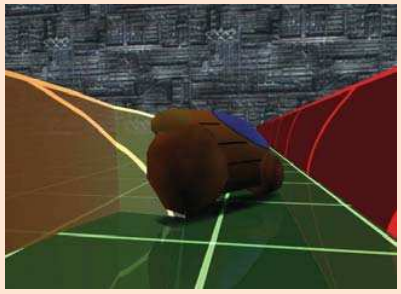
Juhász Attila
(rabszolga@goraffe.hu)

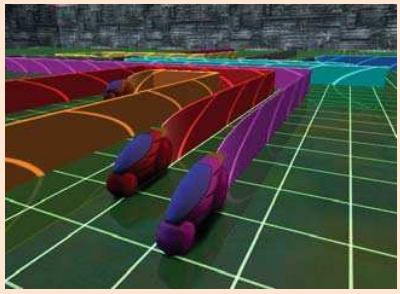
Az Információ Technológiai Kar hallgatója a Pázmány Péter Katolikus Egyetemen. Érdeklődik a bioinformatika és a neurális hálózatok iránt. A fotózás és a tánc mellett öt éve foglalkozik webgrafikákkal. A linux terjesztések közül a Gentoo és az Ubuntu áll legközelebb a szívéhez. Fotós oldala a http://people.goraffe.com/attila címen található.

Armagetron

Nyilván sokan ismerik a Kígyó című játékot a régebbi Nokia telefonokból. Nos a játék alapötlete ennél sokkal régebbről, pontosan 1982-ből származik. Ekkor láthatta ugyanis a nagyközönség a Tron című sci-fi-t. A filmben volt egy játék, a fénymotor. Ezt valósította meg a Nokia – síkban.



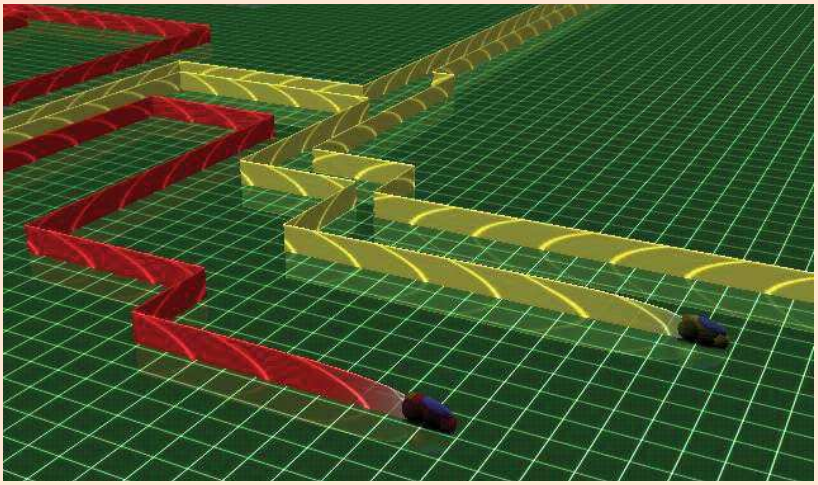




Nos a szabad szoftveres világban sok *Tron* hívő van. Ennek köszönhető, hogy elkészült az *Armagetron* (a *Tron* klón), és ennek köszönhető az is, hogy sokan – köztük én is – szívesen játszok eme remek játékkal akár gép ellen, akár a haverokkal hálózaton. A játék elég régóta többplatformos, tehát van windowsos változat is. A hardverigénye is elhanyagolható, hiszen a mai *PC*-ken gond nélkül fut, persze nem árt egy erős *OpenGL*-es videókártya, mert ez azért mégiscsak egy 3D-s program. A játék célja, hogy minél tovább állva maradjunk, vagyis ne menjünk neki se saját, se mások csíkjának. Tudniillik a fénymotorok menet közben csíkot húznak, ezért is mondtam, hogy a *Nokia Kígyója* kicsit hasonlít az *Armagetronra*... azonban a *Nokia Kígyójával* ellentétben, itt akárhány játékos száguldozhat, akár gép és ember vegyesen is. Nem kell más, mint pontos reflexek, és kidolgozott stratégia.

Az irányítás nagyon egyszerű. A programban mindenki saját maga állíthatja be kedvenc kiosztását. Alapvetően elég, ha balra és jobbra kanyarodunk, meg néha fékezünk. Persze gyorsíthatunk is, de ehhez már nagyobb gyakorlat szükséges, ugyanis ha nagyon közel megyünk egy már meglévő csíkhoz, úgy abból energiát nyerünk, aminek segítségével gyorsulunk...

Jó szórakozást!



Medve Zoltán
(e-medve@e-medve.hu)

2001-ben kezdett „Linuxolni", de már korábban is ismerkedett a szabad szoftverek világával. Ha éppen nem a gép előtt ül, akkor fotózgat, olvasgat vagy bicajozik.

KAPCSOLÓDÓ CÍMEK

Armagetron honlap
➲ http://armagetron.sourceforge.net/

Ablak a világra – TV kártyák használata Linux alatt

Manapság a TV-re sokszor mondják, hogy ablakot nyit a világra, hisz ha kíváncsiak vagyunk mondjuk, hírekre, nem kell messzire menni, elég ha van a közelben egy TV.

Na de mi van akkor, ha nem szeretnénk beruházni méregdrága *TV* készülékbe, mert már van mondjuk egy jó 19 colos *LCD* monitorunk? Ilyenkor az ember elballag a sarki számítástechnikai üzletbe, és kér egy *TV tuner kártyát*. Hazamegy, beilleszti a helyére, és konstatálja, hogy a hőn szeretett operációs rendszere nem vagy nem megfelelően kezeli... Ilyenkor morogva visszamegy az üzletbe és ott vagy kicserélik egy másikra, vagy nem.

A kulcsszó: Brooktree

Hogy az említett eset lehetőségét minimalizáljuk, érdemes megnézni még a boltban a kártyán lévő legnagyobb – legtöbb kivezetéssel rendelkező – *integrált áramkörön* lévő feliratokat. Ha van rajta ilyesmi, hogy *Brooktree* vagy *BT*, akkor már majdnem helyben vagyunk. Ha még egy *848-as* vagy *878-as* szám is szerepel, akkor jó eséllyel fogjuk tudni *Linux* alatt is használni. Ez azonban csak a *PCI-os* kártyákra áll, hiszen az *USB-s tunereket* nem kaphatjuk szét. Tehát a biztosabb siker érdekében *PCI-os* kártyát javasolnék (nekem egy *Kworld 878 RF-Pro* van), arról nem is szólva, hogy olcsóbb is.

Megjelenítés

A jó TV-kártyához jó grafikus kártya és alaplap is kell. Miért? Egy *tvtuner* például elég jól megy *PCI-os* grafikuskártyával, de eléggé megfogja a gépet, ezért mindenképp ajánlott *AGP-s* videókártya, abból is lehetőleg valami nevesebb. Persze ez nem jelenti azt,

1. ábra A kártya lelke: egy Brooktree 878-as chip

hogy anélkül nem megy, hiszen nekem már egy őskövületnek számító *PA-RISC* processzoros *HP* masinán is sikerült tévézni, aminek a videókártyája csak 8 bites (256) színt tudott, azt is csak *FrameBuffer* eszközként.

A kernel beállítása

A *Debian* gyári kernele alapból támogatja a *Brooktree* alapú tunerkártyákat. Amennyiben azonban szeretnénk saját kernelt, úgy pár dolgot a kártyatámogatásán túl bele kell fordítani.
Ilyen pl. az *I2C támogatás* (2.4.31-es kernelben:

```
Character devices -> I2C
➥Support),
```

ezen belül:

```
 I2C support
```

és

```
I2C bit-banging interfaces
```

Na és persze szükséges a Tvtuner *hardveres* támogatása is (2.4.31-es kernelben):

```
Multimedia devices -> Video for
➥linux),
```

ezen belül:

```
V4L information in proc filesystem
```

és

```
BT848 Video for Linux
```

A hangkártyák között is van, ami szükséges:

```
BT878 audio dma
```

és a

```
TV card (bt848) mixer support
```

A kernel konfigurálásakor a súgó bővebben elmondja, melyik miért kell. A *2.6.x* kerneleknél hasonlóan kell eljárni, ott azonban van egy aprócska különbség, éspedig hogy a kernel támogatja a *V4L2*-t.
Ha megvagyunk a kernelfordítással, akkor az lspci parancsra valami ilyesmi tárul elénk:

```
....
0000:00:10.0 Multimedia video
➥controller: Brooktree
➥Corporation Bt878 Video
➥Capture (rev 11)
0000:00:10.1 Multimedia
➥controller: Brooktree
➥Corporation Bt878 Audio
➥Capture (rev 11)
....
```

A modulok beállítása

Amennyiben modulként konfiguráltuk az említett kernel opciókat, úgy érdemes a */etc/modules* fájlba bevinni a szükséges modulokat: *tuner* és *bttv* (ilyen sorrendben), a többi almodult (*I2C*) már magától behúzza a rendszerünk. A modulok konfigurálását a */etc/modules.conf* fájlban tehetjük meg. Nálam a fájl elejére ez került:

```
options bttv card=78
options tuner type=28
```

A 2.6.13.4-es kernel viszont már ilyen formában várja:

```
options bttv card=78 tuner=28
```

Ezzel a *bttv* modulnak megmondom, hogy a 78-es típusú kártyát használja, illetve a *tuner* modullal tudatom, hogy a kártyán a 28-as típusú tuner chip van. Azt, hogy melyik szám mit jelent, a kernel forrás dokumentációjából deríthetjük ki, nevezetesen a *Documentation/video4linux/bttv/CARDLIST* fájlból. A fájl végén a tuner típusok vannak felsorolva. Ilyenkor kell egy marék szerencse, általában a tuner típusa leolvasható arról a *fém dobozról*, amibe az *antenna* megy be a kártyán. Nálam sajna egy *Kworld* címke van. Marad hát a jó öreg `modprobe`/`rmmod` páros:

```
modprobe tuner type=xx
modprobe bttv card=yy
rmmod bttv
rmmod tuner
```

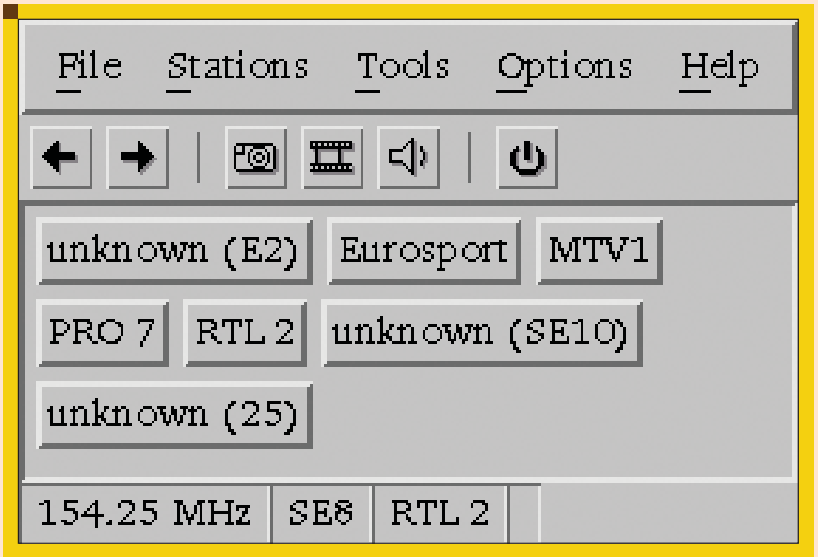
A sorrend mindkét esetben fontos, ugyanis a *tuner* modulra épül a *bttv* modul. Ha van kép, de nem tudunk hangolni, akkor a tuner típusa nem jó, ha kép sincs, akkor a kártya típusa nem jó. A kép meglétét a később ismertetett programokkal tudjuk ellenőrizni. Ha eltaláltuk a tuner és a chip típusát, akkor jegyezzük fel valami biztos helyre... Előfordulhat, hogy egy kártya például többféle *card numberrel* is megy. Az enyém például megy 10-esként és 78-asként is. Ez azért fontos, mert régebbi kerneleknél (mondjuk 2.4.18) pl. még nem biztos, hogy van 78-as.
Súgó a tunerchip fajtájához: az európai tunerek nagy része a *PAL-BG* szabvány szerint megy. Esetleg meg lehet próbálni még a *PAL-DK*-t.

Scantv

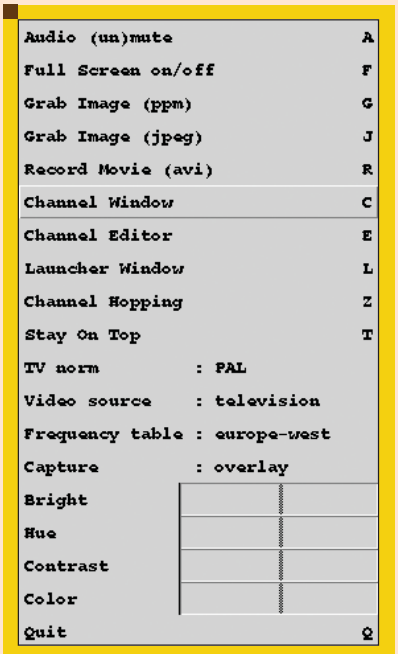
Ezt a programot akkor hívom segítségül, ha nem tudom, helyesen működik-e a tuner chip, vagy esetleg az *xawtv*, *fbtv*, *motv*, stb. egyikét szeretném használni, ezekhez gyárt ugyanis beállítóállományt. Ha csak szimplán kapcsolók nélkül futtatjuk, úgy rákérdez a normára (*PAL*) és a sáv kiosztás típusára (*europe-west*), a végeredmény pedig ilyesmi:

```
scanning channel list
➥europe-west...
E2    ( 48.25 MHz): ???
[unknown (E2)]
channel = E2
E3    ( 55.25 MHz): no station
E4    ( 62.25 MHz): Eurosport
[Eurosport]
channel = E4
S01   ( 69.25 MHz): no station
S02   ( 76.25 MHz): no station
S03   ( 83.25 MHz): no station
E5    (175.25 MHz): no station
E6    (182.25 MHz): no station
E7    (189.25 MHz): no station
E8    (196.25 MHz): no station
E9    (203.25 MHz): MTV1
[MTV1]
channel = E9
E10   (210.25 MHz): no station
E11   (217.25 MHz): PRO 7
[PRO 7]
channel = E11
E12   (224.25 MHz): no station
SE1   (105.25 MHz): no station
....
```

Ez kb. 1 perc alatt fut le. Ahhoz, hogy a fent említett programok tudják használni az adatokat, ki kell írni valamilyen fájlba. Ezt a `-o` kapcsolóval érhetjük el. Ha valamiért nem találja meg az *összes csatornát* a szolgáltató által kínáltak közül, úgy érdemes a *-a kapcsolóval* futtatni a programot. Ekkor azonban készítsünk valami hideg élelmet is, hiszen a program a kártyától függően akár negyed mHz-enként ellenőrzi a csatornákat. (Nálam ez úgy 20 perc szokott lenni.)



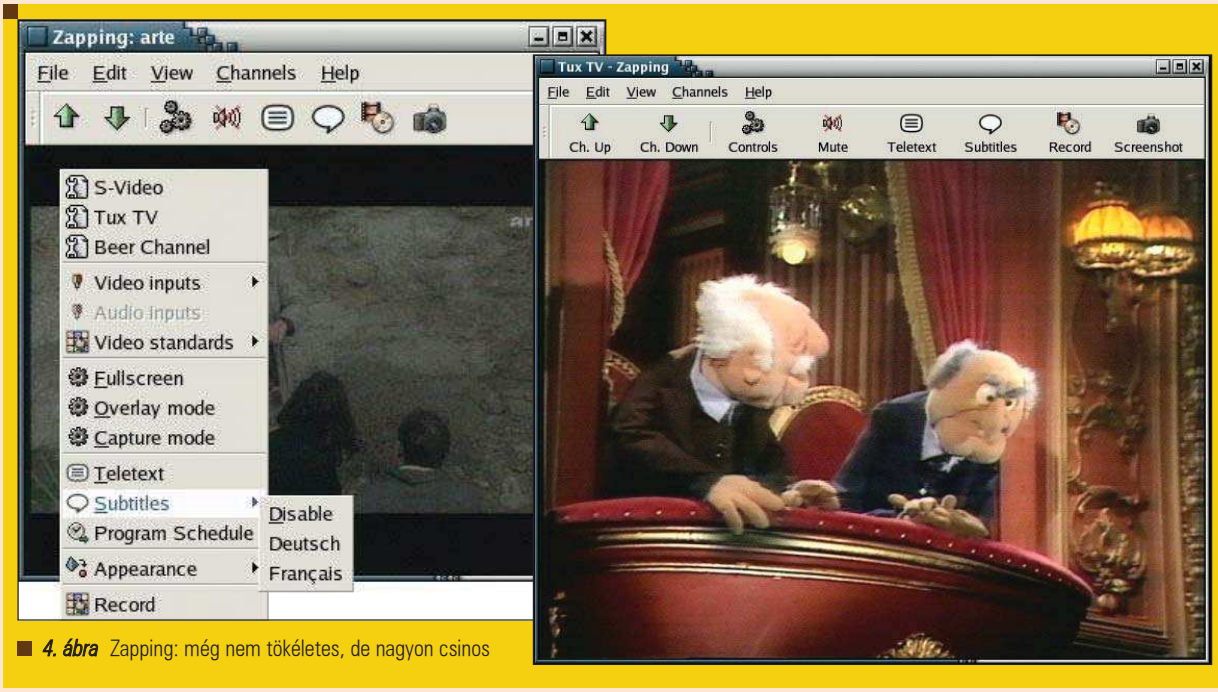
3. ábra A Motif alapú motv menüje kissé kifinomultabb



2. ábra Az xawtv elég egyszerű darab

Xawtv, fbtv, motv

Scantv segítségével ezekhez a programokhoz gyárthatunk beállító állományt. Három program, három különböző személyiség.
Az *fbtv* a már korábban említett *PA-RISC* processzoros *HP* masinán ment, hisz mint a neve is mutatja framebufferes eszközzel működik. A framebuffer például jó arra, hogy a *TV*-s masinára ne kelljen a teljes *X11*-es pereputtyot feltelepíteni. Elég csak az *fbtv*, de ekkor a kernelbe is bele kell fordítani a framebuffer támogatást.
Az *xawtv* is elég egyszerű darab: a bal egérgombra előjön a már korábban a *scantv* által megtalált és eltárolt csatornák listája, a jobb egérgombra pedig előjön a menü.
A *Motif*-alapú *motv* hasonlóképp viselkedik, bár ennek kissé intuitívebb a menüje és több minden kalibrálható benne.
Mind a *motv*, mind az *xawtv* az *F gombbal* vált az ablakos és a teljes képernyős nézet között, illetve a *Q* gombra lép ki.
Bármelyiket is használtam közülük, meglepő módon a */etc/X11/XF86Config-4* állományom *Modules*



4. ábra Zapping: még nem tökéletes, de nagyon csinos

szekciójába be kellett írjam, hogy `Load "v4l"`, máskülönben nem jelent meg kép a zárt forráskódú, hivatalos *Nvidia* meghajtóval.

Zapping

Gnome-os felületű, így csak akkor ajánlom, ha egyébként is fent vannak a *Gnome*-hoz szükséges osztott könyvtárak. A *Zapping* szintén meglehetősen intuitív, az pedig kifejezetten pozitív az előzőekhez képest, hogy van benne *teletext*. A *Zapping* sajnos még közel se tökéletes. Nekem a tesztelés során párszor kiakadt szegmens hibával, illetve az adók közül se talált meg jónéhányat... Ha szeretnénk a programot teljes képernyőn használni, úgy az alapértelmezett *overlay* módot – ekkor a kép megjelenítéséhez minimális rendszererőforrás kell – le kell cserélnünk *capture* módra. Ezt – mókás kombináció – a *CTRL + C* párossal tehetjük meg.

TvTime

Ő a személyes kedvencem. Nem azért, mert kevés rendszererőforrást fogyaszt, nem azért mert tele van extra funkciókkal, még csak felvenni se tud – a program készítője szerint nem is fog tudni. Egyszerűen azért kedvelem, mert a *Tvtime* kezelése hasonlít a legjobban egy *TV* készülékéhez, és az *OSD*-je – a képernyőre kirakott kijelzője – is ezt támasztja alá. Például ki tudja rakni a pontos időt a jobb felső sarokba. Ami szintén megdobogtatta a szívemet, az az, hogy nagyon szép a képe még gyenge képminőségű adásnál is, hisz több fokozatú szűrője van (*deinterlacer*), amit a helyzet és a rendelkezésre álló gép erejének megfelelően állíthatunk be.

5. ábra Tvtime: ez hasonlít leginkább egy igazi TV készülékhez

Aatv

A név az *Ascii-art TV* rövidítése. Ezt igazán csak azoknak szoktam megmutatni, akik azt hiszik, abszolút szöveges konzolon – tehát még framebuffer se – nem lehet TV-zni. Hát lehet. Ez a program betűkből és azok fényességével machinálva rakja ki a *TV* képkockáit. Legalább egyszer érdemes kipróbálni...

Mplayer/Mencoder

Az *Mplayer* igazán a filmlejátszási képessége miatt lett méltán híres, azonban megfelelően felparaméterezve *TV*-t is nézhetünk vele.

```
mplayer tv:// -tv driver=
➥v4l:width=640:height=480:outf
➥mt=i420 -vc rawi420 -vo xv
```

Mi is a Teletext?

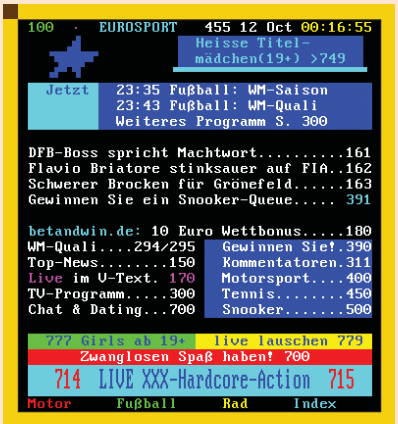
A *Teletext* nem más, mint átmenet az egyirányú *TV* és a kétirányú Internet között. A kedves felhasználó nem érhet el akármilyen tartalmat, sőt, grafikus tartalmat se, de bizonyos mértékű interaktivitást ad. *Teletexten* keresztül például megnézhetjük egy csatorna műsorát anélkül, hogy elő kellene keresnünk a *TV* magazint. De interaktívvá teszi a hirdetést is, hiszen a néző dönti el, meg szeretne-e nézni egy adott hirdetést vagy sem. Az is igaz, hogy a hirdető nem értesül arról, hogy hányan nézték meg a hirdetését, hiszen a *Teletext* technológia ezt nem teszi lehetővé.

A másik amire jó lehet a *Teletext*: segítségével feliratozni lehet egyes tv műsorokat. Ezt úgy érik el, hogy átlátszóvá lehet tenni a teletext fekete hátterét, így az alsó két-három sorba kerülhet a felirat, ami időnként frissül. Ez a technológia lehetővé teszi, hogy hallássérültek is információhoz jussanak a tv-n keresztül.

Az *Mencoder* pedig egy rugalmasan konfigurálható *PVR*-t (*Personal Video Recorder – személyi videó rögzítő*) ad a kezünkbe, amivel egy tv adás felvétele se nehéz feladat:

```
mencoder -tv driver=v4l:width=
➥768:height=576 -ovc lavc
➥-lavcopts vcodec=mpeg4:
➥vbitrate=$RATE -oac mp3lame
➥-lameopts cbr:br=128 -o
➥$FILE.avi tv:// -endpos $TIME
➥> /dev/null 2>&1
```

Ezzel a paranccsal például a *TV*-ből lehet felvenni, ahol az általunk megadható változók: `RATE` (a videó sávszélessége), `FILE` (a felvételt tartalmazó fájl neve), `TIME` (a felvétel hossza).



6. ábra Alevt: a legjobb teletext program

Alevt

Manapság az internet korában eltörpül a teletext jelentősége, ám ha nincs éppen net, jó ez is. Az *Alevt* szerintem az egyik (ha nem a) *legjobb teletext* program. Egy hiányossága van: nem lehet kihúzni teljes képernyőre. Ha valamiért nem akar elindulni, érdemes a jogosultságokat megnézni, ugyanis a program a */dev/vbi* eszközt használja.

Radio, Gradio

Hogy kerül egy TV tuneres cikkbe rádiós program? Számos *TV tuner használható rádió tunerként is*, hiszen az elektronika megvan, éppen csak kép nem kell. A *Radio* program akár szöveges konzolon is használható, míg a *Gradiohoz* már *GTK* környezet szükséges. A beállítás nem bonyolultabb, mint az asztali rádiónk behangolása.

Konklúzió

Hogy ma a digitális TV-zés hajnalán érdemes-e analóg TV tuner kártyába beruházni? Jó kérdés. Szerintem igen, ugyanis a jelek szerint még minimum 3 év (egyesek szerint több, mások szerint kevesebb) év szükséges, hogy csak digitális jeleket tudjunk fogadni Magyarországon. Ha pedig eljön a teljesen digitális korszak, még akkor se lesz remélhetőleg gond, hiszen nagy valószínűséggel kaphatóak lesznek úgynevezett *set-top-box*-ok, amik a digitális jelet analóggá konvertálják.

Medve Zoltán
(e-medve@e-medve.hu)

2001-ben kezdett „Linuxolni”, de már korábban is ismerkedett a szabad szoftverek világával. Ha éppen nem a gép előtt ül, akkor fotózgat, olvasgat vagy bicajozik.

KAPCSOLÓDÓ CÍMEK

Linux kernel
➲ ftp://ftp.kernel.org/pub/linux/kernel/

Xawtv, Motv, FBTV
➲ http://linux.bytesex.org/xawtv/

Zapping
➲ http://zapping.sourceforge.net/cgi-bin/view

Tvtime
➲ http://tvtime.sourceforge.net/

Mplayer/mencoder
➲ http://www.mplayerhq.hu

Mplayer/mencoder (mirror)
➲ http://www2.mplayerhq.hu

Alevt
➲ http://www.goron.de/~froese/

Újjászületett legendák, avagy remake játékok

Nemrég mutattuk be a Quake4 linuxos változatát. Most a fejlesztést irányító id Software két örökifjú játékával foglalkozunk, legalábbis linuxos remake projektjeik gyors bemutatásának erejéig.

Az *id Software* csapatáról dióhéjban annyit érdemes tudni, hogy első embere a világ egyik legismertebb programozó guruja *John Carmack*, aki kezdettől fogva a *C* nyelv megszállottja, így jellemzően *FPS* játékaik motorjáért felelős. Társfejlesztőként, ötletgazdaként *John Romero* neve is felmerült a korai időkben, ám ő időközben kilépett a cég kötelékéből. Mivel az *id* nem idegenkedik a *Linux* rendszerektől – az említett *C* nyelv meg ugye ennek egyik alappillére – így programjaik mindegyikének van natív linuxos változat is. A binárisok elkészültét jelenleg *Timothee Besset* neve és szaktudása garantálja. Időközönként a kifutó, jelentős potenciállal már nem bíró játékmotorjaik forrását *GPL* licensz szerint megnyitják, ennek köszönhetően a *Doom 1* / *Quake (1-3)* sorozat alapkódjai ma mindenki számára szabadon elérhetőek és felhasználhatóak. Emiatt kultuszprogramjaik megújulásának nincs akadálya: elképesztő újjászületéseknek lehetünk szemtanúi, fanatikus és szakértő fejlesztőcsapatoknak hála.

1. ábra A Doom Legacy javított képi világa

Doom

Ez a játék 1993-ban született, és a – szintén *id* fejlesztésű – *Wolfensteinen* túllépve gyakorlatilag megadta a kezdőlökést e máig töretlen sikerű műfajnak. Megjelenésekor „felbolydult" a világ: volt, aki a nap 24 órájában *Doom* addikciójának hódolt (80386 alapú, DOS rendszerű gépén...), de olyan is akadt, aki tiltakozott a játék brutalitása és „sötét hangulata" ellen.
Megjelenése után több millióan töltötték le az akkor hatalmasnak számító 10 Mbyte méretű, egy pályasort tartalmazó demót ami nem csoda, hiszen remek hangulatú, nagyon jó zenei aláfestéssel tálalt klasszikus játékról volt szó. Hivatalos, linuxos binárisa igencsak régi állapotot mutat: 320x200 felbontás, szoftveres renderelés, szűkös beállítási lehetőségek, minimális hangeffektek.
A *Doom Legacy* korszerűsített játékmotorja az eredeti *doom.wad* adatállomány köré építi világát, melyből *3D* grafikai hardverünk is kiveszi a részét: a modellek textúráinak bilineáris szűrése, a képzett fényeffektusok emészthetővé teszik a régi játék szegényes képét. Természetesen nem szabad „csodagrafikát" várni a programtól, viszont a tiszteletre méltó remake magában foglalja a *Quake 2* menürendszerét és a fejlettebb kezelési lehetőségeket is (ugrás, célzás stb).
A *Legacy* hivatalos honlapján bináris formában érhető el.
A ➲ *http://legacy.newdoom.com* címről letöltött *legacy.verzió.tgz* fájl kicsomagolása után a *Doom* (első és második rész, *Ultimate* egyaránt életképes) adatállományát a mappa gyökerébe másolva rögtön indítható az újjászületett játék. Két startállományt tartalmaz az említett könyvtár: egyikük szoftveres renderelésre (természetesen *X11* alapon), a másik pedig *OpenGL API*-ra épít. Mentéseit kulturált módon személyes mappánkban tárolja. A projekt

2. ábra Doom, mint Doom3 mod: apja fia. Vagy inkább fordítva?

kiforrottsága miatt igen népszerű. Az eddigi legelképesztőbb próbálkozás, a „nagy öreg” újjáélesztését illetően ***Classic Doom for Doom3*** névre hallgat. ***Doom3*** modként, újraírt adatcsomagokkal adja vissza a régmúlt érzéseit (mindezt a több mint 10 éves játék pályáinak teljes újraképzésével és „újrafutózott” zenével). A módosított változat a friss játékmotorra épít, felhasználva az új textúrákat és fizikát. Hardverigénye ***Doom3*** szintű, így csak erős géppel érdemes nekiesni, különben ez a varázslatos világ egy szaggató és idegesítő játék benyomását fogja kelteni.
A játék a ➲ ***http://cdoom.d3files.com*** oldalról letöltve, a telepített ***Doom3*** gyökerébe kibontva abból indítható, mods almenüjét használva. Az öreg ***Doom*** itt ultramodernnek hat – azt el sem merem képzelni, mennyi munkába telt a pályák adatait újraképezni... Mentéseit személyes mappánkban tárolja. Kihagyhatatlan darab, felemelő élmény – béta állapotát meghazudtolóan. Bár mod jellege miatt nem klasszikus remake, de személy szerint minden idők modifikációinak mindenkori királyát tisztelem benne.

Quake

1995-ben, az elődök „hamis” ***3D*** világán túllépve jelent meg a gótikus világban játszódó ***Quake***. Mind a mai napig a „legaddiktívabb” ***FPS*** játékok egyike: az aktív szervereken futó bajnokságok állandó résztvevője. Az engine valódi ***3D*** leképezést használ, kezdetben csupán a nagynevű ***3Dfx*** cég ***Glide API***-ját támogatta a szoftveres renderelés felett. Nagyon jól eltalált pályák, félelmetes ellenfelek, jó kezelhetőség, adrenalinnövelő zene jellemzi – igen előkelő helyet foglal el a ***PC***-s játékok között.

A ➲ ***http://www.fuhquake.net*** oldalon elérhető ***FUH Quake*** projekt a ***GPL*** licensz szerint publikált ***Quake*** játékmotor forrását foltozza és egészíti ki újabb képességekkel. Szoftveres és ***OpenGL*** rendereléssel egyaránt elboldogul, természetesen az eredeti ***Quake*** adatállományaiból renderel. A remake játék rendkívül látványos, képi effektekben gazdag, miközben hardverigénye mindehhez képest szerény. A ***Loki Installers for Linux Games*** csapata a kódot disztribúciófüggetlen, előre fordított binárisként terjeszti javított textúrákkal és egyéb nyalánkságokkal „összedrótozva”, valójában ***EQuake*** projektként.
Telepítése után (rootként elindítjuk a ➲ ***http://liflg.org*** oldalról letöltött ***equake.verzió.run*** állományt) látható lesz, hogy egyaránt tartalmaz szoftveres és ***OpenGL*** rendereléshez tartozó indítókat is. Igény szerint (még futtatás előtt) a telepítési út id1 almappájába másolandó az eredeti játék két ****.pak*** állománya, az ott lévő demó adatokat felülírva. A ***FUH*** a mai ***Quake*** világ egyik alappillére, minden remake projektek egyik iskolapéldája. A ➲ ***http://tenebrae.sourceforge.net*** oldalon elérhető csapat az eredeti játékmotort és kiegészítő állományait úgy programozza újra, hogy ezáltal a kód képességeinek listájában fellelhető a friss ***Doom 3*** kódhoz tartozó paraméterek némelyike is. A ***Tenebrae***

3. ábra A FUH színgazdag és effektdús világa

4. ábra A Tenebrae Doom3-szerű grafikai effektekkel dolgozik

néven ismert kód jellemzően fémes látványú, bump mapping effektekkel és valós időben számított árnyékokkal dolgozik – ennek megfelelően hardverétvágya is magas: programozható árnyaló egységet nélkülöző videókártyán el sem indul. Telepítése szintén gyerekjáték: rootként indítva a – ***FUH*** gyanánt már említett – ➲ ***http://liflg.org*** oldalról letöltött, telepítőbe ágyazott tenebrae.verzió.run állományt (mely előre fordított binárist, kiegészítő override textúrákat, demó pakot tartalmaz), igény szerint a *Quake* pakjainak */$path/id1* mappába másolásával éleszthető fel a játék, mely felhasználóként kiadott ***tenebrae*** paranccsal indítható. A ***Tenebrae*** a ***FUH*** igen komoly alternatívája, az újjászületett programkód pedig egy szemet gyönyörködtető technikai brillírozás.

Írásom messze nem törekszik teljességre, mivel jelen téma a cikk többszörös terjedelméig elemezhető – nem beszélve arról, hogy természetesen léteznek egyéb életképes kódok is (***Doom'sDay***, ***OpenQuartz*** stb). Azonban az itt említett megoldások kiforrottságuk által a régi játékok hangulatára szomjazó ***Linux*** megszállottakat minden tekintetben képesek kiszolgálni. Fanatikus fejlesztőik profi hozzáállása és az ***id Software*** szabadon elérhető forráskódjai együttesen komoly potenciált hordoznak: bármely megnevezett játék észrevétlenül csempészi vissza a hőskort, ***Linuxot*** futtató gépünk képernyőjére.

Kovács Zsolt (kovi@linuxforum.hu)
Quake fanatikus