Beköszöntő

Linuxvilág

■ ***Albert Einstein*** szerint: ***„Két végtelen dolog létezik: az univerzum, meg az emberi hülyeség. Bár az univerzumban nem vagyok egészen biztos...”*** Aki meg akar róla győződni, hogy az öregnek mennyire igaza volt, olvassa el ***Halusz Léna*** legújabb „pingvin cikkét”, amiből megtudhatja – vagy nem – hogy mi köze ***Jézusnak***, a liberalizmusnak, a melegeknek, meg a pingvinek szaporodásának egymáshoz. Mélyen filozofikus téma erős idegzetűeknek...
Állandó szerzőnk, ***Komáromi Zoltán*** – két új könyv bemutatása mellet – az ***Eclipse*** fejlesztőrendszer használatáról indít sorozatot. ***Fülöp Balázs*** ugyanakkor ***Java*** sorozatának hetedik, befejező részéhez érkezett. Búcsúzóul a folyamatok közti kommunikáció kerül terítékre.
Az alkalmazásokat bemutató rovatunk ***Párkányi Péternek*** a ***Beagle*** keresőről szóló cikkével indul. ***Péter*** nem egyszerűen egy újabb szerző, hanem minden valószínűség szerint a ***Linuxvilág*** történetének legifjabb cikkírója is egyben.
A fax manapság jószerével a digitális kőkorszak itt felejtett darabjának számít, néha mégis vannak előnyei az e-maillel szemben. Persze ahhoz, hogy ezeket ki is tudjuk használni, nem árt egy hatékonyan működő faxközpont, különösen akkor, ha nem egyéni felhasználásról, hanem vállalati szolgáltatásról van szó. Ehhez nyújt segítséget most debütáló szerzőnk, ***Csuhai Imre*** két részes, a ***Hylafax*** rendszerről szóló leírása.
Sütő János kedvenc témája a spam elleni küzdelem. Ezúttal a ***Razor*** rendszerről tanulhatunk tőle.
Kovács Levente neve szintén új a magazinban. Ő az ***NVU*** webszerkesztő bemutatásával lép színre decemberi számunkban, míg ***Apagyi György (killall)*** ezúttal a ***VLC*** videólejátszóról írt nekünk.
A ***DTP*** a ***Linux*** számára sokáig fehér foltnak számított, mivel az igazán használható programok csak ***Windowson*** vagy ***Macintoshon*** futottak. Aztán megjelent a ***GIMP***, ami a vele kapcsolatos kezdeti lelkesedés ellenére a mai napig sem érte el az igazán profi kereskedelmi termékek színvonalát, viszont a tudása különböző bővítményekkel korlátlanul bővíthető. Ezeket fogja nekünk bemutatni ***Juhász Attila*** egy cikksorozat formájában.
A ***DTP*** „másik frontja” a szöveg illetve az egész kiadvány szép formázása, vagyis a tördelés. Ehhez szintén sokáig nem volt linuxos eszköz, de aztán jött a ***Scribus,*** amelyről ebben a számunkban indít ***Lelovics Zoltán*** egy sorozatot.
Ami a ***3D***-t illeti, ***Auth Gábor*** folytatja a ***PovRay***-ről, ***Szalai András*** pedig a ***Blenderről*** szóló sorozatát. ***Novák Áron*** ugyanakkor az ***SVG*** formátumról szóló sorozat végéhez érkezett.
Végül laza levezetésképpen ***Medve Zoltán*** hív meg bennünket egy kicsit biliárdozni.

Jó szórakozást, kellemes időtöltést kíván a Linuxvilág stábja!

Hírek

Linux alapú tábla PC

Az *EmperorLinux* bemutatta első *Linux* operációs rendszert futtató, teljes értékű tábla *PC*-jét. A nagy gyártók hordozható gépeit *Linuxszal* kínáló cég új *Raven* modellje a *Lenovo X41*-es jelzésű gépére épül, az *EmperorLinux* érdeme ebben az esetben az, hogy a gép összes szolgáltatását *Linux* alól is elérhetővé tette. Ilyen például az, hogy a kijelző a tábla *PC*-kre jellemző módon elfordítható, majd ráhajtható a billentyűzetre – ezt követően a gép érintőképernyős készülékként használható, vagyis cerkával vezérelhető, az adatbeviteli lehetőséget pedig kézírásfelismeréssel biztosítja. Mivel az alapmodell ujjlenyomat-felismerővel is rendelkezik, ennek szolgáltatásait egyesítették a *GDM* beléptetési eljárásával. Természetesen a *Wi-Fi*-csatoló, az energiatakarékossági funkciók és a beépített eszközök, hálózati csatolók használhatósága biztosított. A vásárlók *Fedora*, *Red Hat Enterprise*, *Ubuntu*, *Debian*, *Slackware* és *SuSE* terjesztést választhatnak gépükre. A *Raven* ára 2200 dollártól, nagyjából 460 ezer forinttól indul.

➲ *www.emperorlinux.com*

Chipsetet még tudnak

Hamarosan az üzletekben is kapható lesznek az *Intel* legújabb, *975X* jelzésű lapkakészletére épülő alaplapok. Az új lapkakészlet új memóriavezérlőt hoz magával, valamint támogatja *x16*-os *PCI Express* sávok *x8*-as felosztását, amelynek köszönhetően a több *GPU*-t tartalmazó grafikai kártyák is használhatók vele. A *975X* legfeljebb 8 *GB DDR2* memóriát tud kezelni, 1066 MHz-es processzorbusza pedig a legújabb, kétmagos processzorok megfelelő kiszolgálását hivatott biztosítani. Várhatóan párosul hozzá a legújabb be- és kiviteli vezérlő, az *ICH7DH*, melynek nevében a *DH* a *digital home*, digitális otthon rövidítése, és a továbbfejlesztett multimédiás képességekre utal.

Fél kilónyi mozi

A *Toshiba* bemutatta a kivetítők legújabb generációját képviselő készülékét. A nagyjából 500 grammos projektor újdonsága abban rejlik, hogy a korábbi modellektől eltérően *LED*-es fényforrásokat használ a három alapszín előállítására. A készülék fogyasztása is a korábbiak tizede, mindössze 15 watt, így akkumulátorról is gond nélkül lehet majd üzemeltetni. A *LED*-es megoldás hátránya, hogy meglehetősen szerény fényerőt szolgáltat, így a készülék csak a hozzá mellékelt vetítővászonnal használható.

Fizikából jeles

Már az *AGEIA* fizikai gyorsítóprocesszorának bemutatásakor is felvetődött, hogy a játékokban szükséges fizikai modellezést a grafikai kártya processzorára is rá lehetne bízni, így nem kellene újabb szereplőt befogadni erre a gyilkos versengéssel jellemezhető piacra. A fizikai és animációs motorokat fejlesztő *Havok* ennek nyomán be is jelentette *Havok FX* termékét, amely pontosan ilyen feladatokat képes elvégezni – feltéve, hogy a gépben található grafikai kártya támogatja a *Shader Model 3.0*-t. A fizikai gyorsítás két területet céloz, egyrészt a játékmenethez kevésbé kötődő, elsősorban a látványt próbálja fokozni, ide tartozik például a szereplők ruházatának finomítása; másrészt magának a játékmenetnek a valószerűségét próbálja javítani, együttműködve a játékok mesterséges intelligenciájával és a hangeffektusokat előállító alrendszerével. A *Havok* motorja értelemszerűen főként a játékfejlesztőket érdekelheti, a játékosok szempontjából sokkal fontosabb kérdés, hogy ha a grafikai processzorok teljesítménye folyton kevésnek bizonyul, akkor vajon honnan kell majd órajeleket lecsippenteni a fizikai gyorsításhoz? Egyértelmű válasz nincs, de az, hogy maga a fejlesztő cég is a legdrágább videokártyákat ajánlja, felér egy cinkos kacsintással.

➲ *www.havok.com*

BSD-özön

A *FreeBSD* legújabb változata immár a hatos számot viseli. A fejlesztők csak szerényen nyilatkoznak az újdonságokról, azok közt leginkább kisebb-nagyobb hibajavításokat, biztonsági finomításokat találunk, de a szerénykedés mögött olyan fontos fejlesztések húzódnak, mint például a fájlrendszer teljesítményének komoly javítása és többszálúvá tétele, amelynek révén maradéktalanul képes kiaknázni a többprocesszoros rendszerek által kínált lehetőségeket; a vezeték nélküli hálózatok és csatolók, illetve a *WPA* protokoll támogatása; valamint az illesztőprogramok választékának bővítése, javítása. Újabb változat jelent meg az *OpenBSD*-ből is, amely immár a 18. kiadását érte meg. Az elsősorban a biztonságáról hírek rendszer újdonságait szintén leginkább az illesztőprogramok listájának bővülése, a hardverkompatibilitási listán szereplő hálózati csatolók sokasodása, valamint egyéb, elsősorban a kiszolgálói jellegű környezetben való használatot segítő fejlesztések jellemzik.
➲*www.freebsd.org;* ➲*www.openbsd.org;* ➲*www.freebsdmall.com;*

Képzelt gépek

parallels

Újabb szereplő hallatja hangját a virtuális gépek futtatását lehetővé tévő alkalmazások piacán. A *Parallels* nevű cég a *Parallels Workstation 2.0* programból immár a harmadik próbaváltozatot tette közzé, és bár a kínálat egyelőre ebből az egyetlen alkalmazásból áll, jövőre újabb változatot ígérnek belőle, valamint a választék két kiszolgáló oldali megoldással is kiegészül majd. A *Parallels Workstation 2.0 Windows* és *Linux* alapú gazdákon futtatható, vendég operációs rendszerként pedig a *Windowsokat*, a fontosabb *Linux* terjesztéseket, valamint a *FreeBSD*-t és az *OS/2* vonalat támogatja. Ára 99 dollár, ami a vezető szereplőnek mondható *VMWare Workstation* 189 dolláros ára mellett versenyképesnek számít. A *VMWare*-t sem kell azonban félteni, a cég *VMWare Player* névvel új, lebutított változatot adott ki virtualizációs szoftveréből. A *Player* ingyenesen letölthető, és mint neve is utal rá, csak virtuális gépek futtatására lehet használni, létrehozásukra nem alkalmas.
➲ *www.parallels.com*

WiMAX-hírek

A *Samsung* bemutatta *m8000* jelű mobiltelefonját, talán ez első olyan mobilkészüléket, amely a *CDMA* hálózatok mellett a *WiMAX* hálózatok használatára is képes. A *Windowst* futtató mobil léte már csak azért is furcsaság, mert a nagyobb távolságokra is több tíz megabites átviteli sebességet nyújtó, vezeték nélküli kommunikációs szabványt csak nemrég véglegesítették – a rejtély megoldása abban rejlik, hogy a dél-koreai piacra szánt készülék a szabvány egy előzetes, kifejezetten a távol-keleti ország számára kidolgozott, *WiBro* nevű változatát használja. Az *m8000* a *WLAN*-okat és a *Bluetooth* kapcsolatokat is támogatja, valamint az elérhető sávszélesség teljes értékű kihasználása érdekében videótelefonálásra és videokonferencia rendezésére is alkalmas. Ez a bizonyos sávszélesség letöltési irányban akár a 3 Mb/s-t is elérheti, míg a feltöltések legfeljebb 1 Mb/s sebességgel száguldhatnak. A technológia érdekessége, hogy – bár kisebb sebességgel – mozgó járműből is jól használható. A *WiMAX* környékén más gyártók részéről is nagy a nyüzsgés, a *Motorola* és az *Intel* például szövetséget kötöttek a technológia népszerűsítésére. Míg a nagyok a stratégián gondolkodtak, addig a *Wavesat* már el is készítette első *mini-PCI* formátumú referenciakártyáját. A cég várakozásai szerint hamarosan beindulnak a *WiMAX* telepítések, és szükség lesz a megfizethető végfelhasználói berendezésekre – például a hasonló kártyákra. Magyarországon a *WiMAX* szolgáltatást a *GTS-Datanet* már április óta teszteli, kereskedelmi elindítását – akkor – fél éven belülre ígérték.

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

Mi újság a rendszermag fejlesztése körül

Linus Torvalds összeállított egy olyan ***git*** tárat a 2.6-os rendszermag fejlesztéséhez, amely az összes anyagot tartalmazza egészen a ***BitKeeper*** bevezetéséig visszamenőleg. A mag fejlesztése továbbra is egy új fában folyik, de mostantól a régi fa is hozzáférhető a tájékozódás végett, illetve bármilyen más célból, ami a fejlesztés során felmerülhet. Igaz ugyan, hogy ezek a foltok nem most először tűnnek fel egy ***git*** tárban, viszont ez az első olyan alkalom, amikor egy ilyen, minden régebbi információt, és az elmúlt 3 „***git***-es hónap" termését tartalmazó tár mérete már elfogadhatónak tekinthető. A ***git*** első bejelentése óta tapasztalható fejlődése amúgy egyenesen elképesztő, és immár az 1.0-ás változat megjelenése is a küszöbön van. Bár a fejlesztők többsége valószínűleg ki fogja várni, amíg az eddig használt kedvenc változatkezelő rendszerének összes szolgáltatása elérhető lesz a ***git***-ben is, magának a ***git***-nek a fejlődése szempontjából ezek az új képességek tulajdonképpen már csak a habot jelentik a tortán. Azok az alapfunkciók ugyanis, amelyek lehetővé teszik hogy a rendszer robusztus módon támogassa az elosztott fejlesztést, kezdettől fogva megvannak.

Úgy fest, a ***RelayFS*** hamarosan bekerül a fő kernelfába. Ez a modul régóta jelen van az ***Andrew Morton*** által fenntartott ***-mm*** fában, és az elmúlt hónapokban egyáltalán nem kellett az alapfunkcióit foltozni. Amit nyújt, azt eleddig számos felhasználó találta már hasznosnak a legkülönbözőbb projektekben, és a kernelbe való bevezetésével kapcsolatos egyetlen érv tulajdonképpen csak az volt, hogy a ***DebugFS*** hasonló funkciókat lát el. Utóbbival igazából nincs is semmi gond, hacsak az nem, hogy egy ilyen nevű fájlrendszert senki nem akar semmi másra használni, csak nyomkövetésre. Így aztán ***Andrew*** nemrég kijelentette, hogy a ***RelayFS*** kódját be fogja nyújtani ***Linus Torvaldsnak***, különösen hogy a fejlesztők maguk is elérkezettnek látják az időt erre a lépésre.

Timothy R. Chavez és mások elkészítettek egy olyan, az auditálási képességeket kiegészítő foltot a ***VFS***-hez (***Virtual Filesystem***), ami lehetővé teszi, hogy egy fájlrendszer-objektum adatait annak neve és neve alapján kérdezhessük le. A ***VFS*** jelenlegi megvalósításában ezt lehetetlen megoldani. Amikor ***Timothy*** először előállt az ötlettel, a kernelfejlesztők részéről némi ellenállással találkozott, mondván az ***inotify*** szolgáltatást pontosan ezért találták ki, és egészen hasonló lehetőségeket nyújt.

A ***Timothy*** által kezdeményezett projekt indulásakor ugyanakkor az ***inotify*** fejlesztése még meglehetősen az elején járt, és csak egy külső foltként volt elérhető. Ennek megfelelően több értelme volt az auditálással kapcsolatos szolgáltatást egy külön, önálló kódrészletként megvalósítani. Most, hogy az ***inotify*** legalább már az -mm fába bekerült, kicsit jobban védhető a korábbi álláspont, miszerint talán mégis célszerűbb ezt használni. Ugyanakkor ***Timothy***, és a folt más fejlesztői, egyetemben olyan korábbi kritikusokkal mint ***Greg Kroah-Hartman*** kidolgoztak egy tervet, amelynek lényege az, hogy a két rendszer közös szolgáltatásait megvalósítják egy absztrakt felületként, amit aztán mindkét projekt közösen használhat. Az azonban, hogy ez a nagy egyesítés mikorra készül el, egyelőre nyitott kérdés.

Adrian Bunk „eltávolítandóként" jelölt meg néhány az ***OSS***-ben található meghajtót. Annak eldöntése, hogy mi menjen, és mi maradjon soha nem könnyű. A cél nyilván az, hogy megmaradjon a támogatás minden létező, forgalomban vagy használatban levő hardverelemhez, így ***Adriannak*** minden egyes döntés előtt meg kell vizsgálnia, hogy létezik-e a megfelelő ***ALSA*** meghajtó, és az valóban megfelelően működik-e. Mármost a dolog hátulütője az, hogy ezt gyakorta csak olyasvalaki segítségével lehet megtenni, aki rendelkezik az érintett, esetenként kőkorszaki eszközzel, az ilyen emberek felkutatása pedig nem éppen egyszerű. A munka dandárját természetesen ***Adrian*** végzi, ő az, aki levélben megkeresi a potenciális segítőket, és begyűjti tőlük a megfelelő kérdésekre adott válaszokat. Az ilyen „háztáji tisztogatás", mint amit ő végez gyakorta igen hálátlan feladat, sőt sokszor mindenféle ellenállásba is ütközik azok részéről, akik kedvenc hardverük támogatásának elvesztésétől tartanak. Köszönettel tartozunk tehát ***Adriannek*** és a hozzá hasonlóknak azért, hogy minden héten órákat áldoznak arra, hogy „tisztán és szárazon" tartsák a kernelt.

A különböző ***WSL (Wireless Security Lock)*** „kütyük" egyre nagyobb támogatást élveznek ***Linux*** alatt is. Ezek a készülékek érzékelik, ha egy vezeték nélküli eszköz egy bizonyos ponttól túlságosan távolra került, és ilyenkor aktiválnak bizonyos védelmi funkciókat, például zárolják a monitort. ***Brian Schau***, aki élete első kernelmodulját készíti, éppen egy ilyen eszközhöz ír meghajtót. Igaz ugyan, hogy egy ***Bluetooth*** telefon is képes betölteni hasonló funkciót, meg abban is van valami, hogy egy a felhasználói térben futó alkalmazás jobban megfelelne erre a célra, mint ***Brian*** kernelmodulja, a projektnek mégis van értelme, hiszen egy létező hardverelemhez nyújt támogatást. A jelek szerint tehát, így vagy úgy, de a ***WSL*** egyszer csak támogatott lesz ***Linux*** alatt is.

Andrew Morton fűzött némi magyarázatot ahhoz a jelenséghez, hogy egyes felhasználók miért használnak újabban inkább cserefájlt (swap) cserepartíció helyett. A 2.6-os kernel esetében szerinte gyakorlatilag nincs különbség a két megoldás között. Akár a teljesítményt, akár a megbízhatóságot nézzük, a cserefájl és a cserepartíció ugyanazt nyújtja, egy apró kivétellel: ha a cserefájlt erősen töredezett (fragmentált), az a teljesítmény rovására megy. De mivel a cserefájl fragmentáltságának mértéke nem változik a használat során, ezt a problémát egész egyszerűen kiküszöbölhetjük úgy, hogy a telepítéskor létrehozunk egy egybefüggőt. Ugyanakkor a 2.4-es kernelnél még a cserepartíció a győztes, mert ennek a használata nem igényli néhány olyan memóriaterület lefoglalását, amelyek a cserefájl kezeléséhez szükségesek.

Zack Brown

Linux Journal 2005. 139. szám

A háló kalózai – Hogyan lopjunk kontinenst

E könyv formájában egy új műfaj mutatkozik be Magyarországon: a „hackerkrimi”. Abban különbözik a sima krimitől, hogy itt szereplők nem fegyverekkel harcolnak emberek ellen, hanem ésszel, fortéllyal próbálnak meg gépeket, rendszereket az uralmuk alá hajtan, feltörni, tönkretenni, stb. valamilyen cél érdekében.

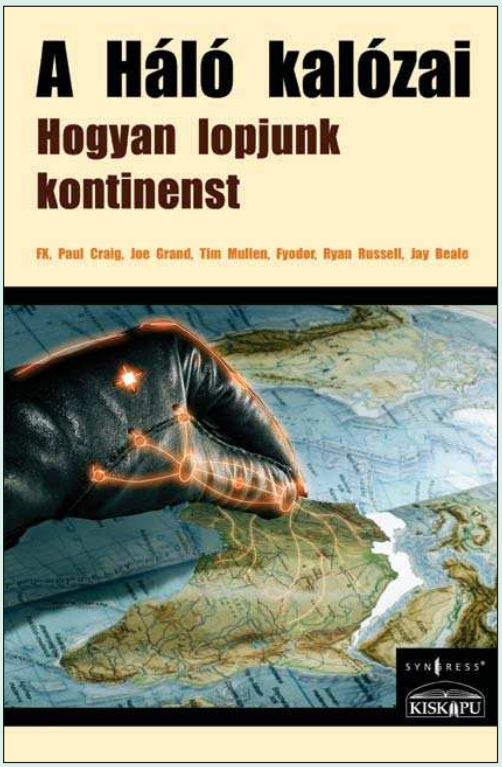
A történet elején megismerkedhetünk a főhőssel, aki felteszi magának a kérdést: mennyi pénzre volna szüksége ahhoz, hogy élete végéig kényelmesen éljen a világ egyik eldugott sarkában, anélkül, hogy egy percet is dolgoznia kellene. Számításai szerint úgy 180 millió dollárra, amelyet úgy akar megszerezni, hogy a világhálón keresztül különböző afrikai pénzintézetek számláit csapolja meg egy-egy „kisebb” összeggel. Adott tehát a terv, amelynek megvalósításához egy hackerekből álló csapatot szervez, amelyet teljes inkognitóban irányítva próbálja elérni célját. A csapattagok nem ismerik egymást, és úgy tudják, különböző projekteken dolgoznak, amely valójában mind egy célt szolgál: óriási mennyiségű pénzt meglovasítani a fekete kontinensről.

A regény több kisebb, látszólag különálló epizódból áll, amelyek mindegyike a nagy balhé egy-egy fázisának, mondjuk úgy akcióijának történetét meséli el, főszerepben a csoport azon tagjával, aki az adott akciót végrehajtja. Minden fejezetben más-más technológiával, betörési módszerekkel, feladatokkal célpontokkal találkozhatunk. Kicsit olyan, mint egy *Tarantino* film, ahol a történetet mindig más szereplő szemszögéből nézve szemlélhetjük, csak az utolsó epizód megismerése után áll össze fejünkben a teljes történet.

Ezen felül minden fejezet végén van egy „Utóhatás” alcímű rész, amely az addig történteket egy harmadik személy (általában valamilyen nyomozó, vagy károsult) mesél el, sok apró adalékot adva a történet későbbi minél teljesebb megértéséhez.

Minden fejezetet más-más szerző írt, önmaga által „megszemélyesítve” az adott epizód főszereplőjét, egy választott hackert. Nekünk, olvasóknak azonban nem kell hackernek lennünk ahhoz, hogy élvezni tudjuk a regényt. Sokszor nem is a technikai folyamat a lényeges, számtalan helyen az emberi tényezőre építenek: hogyan kell átverni valakit, hogy eláruljon nekünk néhány bizalmas információt. Illetve ott a pénzmosásról szóló fejezet, ahol kisebb gazdasági képzésben vehetünk részt, világosan bemutatva számunkra a pénz tisztításának számos működő módszerét. Ha azonban minket ez sem érdekelne, akkor még mindig koncentrálhatunk nyugodtan a történetre, amely azáltal, hogy utólag kell fejben összeraknunk, garantálja, hogy nem fogunk elaludni olvasás közben.

S bár a könyv csak kitalált eseményeken alapul, az egyes epizódok nagyon is valóságos, egészen konkrét módszereket mutatnak be, amelyekkel illetéktelenül férhetünk hozzá elméletileg védett anyagokhoz. Egy-egy betörés leírása olyan részletes és pontos, hogy azt akár az életben is simán ki lehetne próbálni. (Nem tartom kizártnak, hogy a bemutatott kódrészletekbe egy-egy szándékos hibát belecsúsztattak a szerzők, nehogy ebből később még bajuk legyen.)



A történet bevezet minket a kibervilág kapui mögé, végre megtudhatjuk, mi is irányítja mindennapjainkat, és hogy ezek a mindennapok mennyire sebezhetőek, s mi, mindennapi emberek micsoda mértékben függünk ettől a sokak szemében misztikus és bonyolult technológiától. Épp ezért ajánlom különösképp a technikában kevésbé jártas olvasóknak is: aki egy kicsit is kedveli a számítógépeket, bizonyosan örömét leli benne. A megszállottaknak pedig azért ajánlom, mert ettől még inkább megszállottnak érezhetik magukat, de nem kizárt az sem, hogy rájönnek, mennyire keveset értenek az informatikához...

Komáromi Zoltán

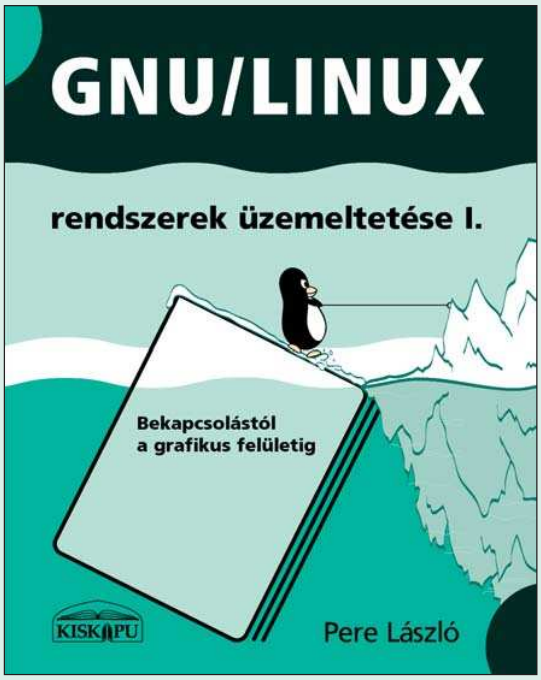
Pere László: GNU/Linux rendszerek üzemeltetése I., II.

Linuxos rendszergazdaként az egyik legnagyobb fájdalmam, hogy az üzemeltetés során felmerülő problémák esetén csak az adott területre vonatkozó elszórt ismeretanyagot találtam...

Általában is elmondható, hogy átfogó ismeretanyagot kevés helyen találni, még idegennyelven is. Átfogó alatt nem csak azt értem, hogy az összes problématerületet érinti, de egyfajta gondolkodásmódot is kialakít az olvasóban a ***UNIX*** rendszerekkel kapcsolatban. A fentiek tudatában igen megörültem, amikor a ***GNU/Linux*** rendszerek üzemeltetése első kötete megjelent. Végre az első olyan magyar nyelvű könyvet tarthattam a kezemben, amely távolról sem hasonlít a ***HOGYAN***-ok és szakácskönyvek családjára, ugyanakkor a „pontszerűen" feltűnő helyi problémák megoldása esetén is kiválóan alkalmazható.

Örömöm tovább folytatódott, ugyanis a napokban érkezett meg a könyvesboltok polcaira a könyv második kötete, amely a ***Linux/Unix*** hálózatok üzemeltetésének bemutatását tűzte ki céljául. Ezen alkalomból szeretném röviden ismertetni a könyvvel kapcsolatos benyomásaimat.

A könyv nem kifejezetten azoknak szól, akik csak „kattintgatni" szeretnének, azaz akár otthon, akár a munkahelyen egy már telepített ***Linuxon*** dolgoznak, hanem inkább azoknak, akik telepíteni, üzemeltetni szeretnék, azaz elsősorban rendszergazdáknak, illetve megszállottaknak. Megszállottként vagy rendszergazdaként sem csak azok számíthatnak arra, hogy okosodnak a könyvből, akik már most is profiként tengetik mindennapjaikat: a könyv alapfilozófiája, hogy minden esetben részletesen tárgyalja az alapokat. Ha ezt megértettük, csak ezután jöhet az oda vonatkozó konkrét ismeretanyag.



Eddig azt hittem magamról, hogy „tapasztalt" rendszergazda, sőt linuxfüggő vagyok, de a könyv olvasása során rá kellett döbbennem, hogy tévedtem: ismereteim hiányosak, lyukasak. Amint tudom, más is hasonlóan áll ezzel, ugyanis a legfőbb hiányosság, amit már az elején is említettem az, hogy nem áll rendelkezésre átfogó ismeretanyag. Magam is úgy tanultam, hogy naprólnapra folyamatosan szembekerültem mindenféle problémával, megkerestem rá az arra (és csak arra) vonatkozó választ, majd továbbrepültem.

Aztán egy idő után már elegendően szélesek voltak az emlékeim, hogy azt (jóindulattal) ismeretanyagnak lehessen nevezni. Egy ilyen folyamatnak azonban sajátossága, hogy a megszerzett tudás hiányos, illetve számtalan területen megalapozatlan. A könyv kiválóan alkalmas ezeknek a területeknek a befoltozására, az ember ismereteinek valóban teljes körűvé tételére.

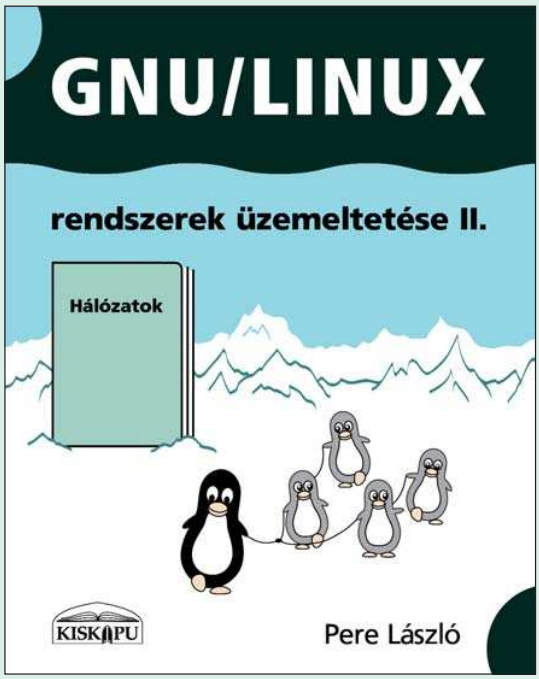
Pere László írása egyébként nem specifikus. Ez egyrészt azt jelenti, hogy nem tér ki mindenféle kisebb-nagyobb csomagok, ***GNU/Linux*** megoldások ismertetésére, ehelyett inkább a tradicionálisabb, ***UNIX*** alatt is elérhető lehetőségeket ismerteti. A dolog másik része az, hogy a bemutatás során a rendszer vázára koncentrál a szerző. Ez nem csak azért fontos, hogy megértsük az alapokat, de ha ezeket a rendszereket ismerjük, szinte egyáltalán nincs szükségünk a „specialitásokra".

Az olvasó további szerencséje, hogy a könyv nem igazán épít más, előző anyagok ismeretére, illetve megemlíti a hátlapon, hogy feltételezi ***Pere László*** egyik korábbi könyvének, a ***Linux*** fel-

használói ismeretek c. könyv első kötetének ismeretét, aki azonban látott már *Linuxot*, tudja mi az az egy gyökerű fájlrendszer, vagy a befűzés (*mount*), az enélkül is elboldogul.
Ha mégsem sikerülne azonnal megértenünk, mit is akart mondani nekünk tanítónk, még mindig ott a második védelmi vonal, méghozzá konkrét kidolgozott feladatok formájában. Természetesen mindkét kötet teljesen közérthető, de sok esetben a feladatok lerövidíthetik a megértés folyamatát.
Mielőtt a könyvek tartalmából szemezgetnénk, még egy utolsó megjegyzést: ez az eddigi első könyv, amely ésszerűen és pontosan használja a magyarításokat. Szinte mindegyik szakkiadó törekszik arra, hogy minden egyes kifejezést magyarra fordítson. Ezzel csupán annyi a baj, hogy csak a legritkább esetben sikerül eltalálniuk a célt, jellemzően a lefordított szó nem fedi az eredetit, de ha fedi is, általában nem lehet tudni, hogy az mire utal, minek lehet a fordítása. *Pere László* azonban kitűnő érzékkel él a lefordított kifejezésekkel, az általam angolul ismert kifejezésekre azonnal ráismerek a magyar megfelelő (esetleg szövegkörnyezet) elolvasása után.

Az első kötet

Az első kötet tehát a *Linux* rendszerek általános felépítését és belső működését mutatja be.
A lemezrészek (partíciók), azok linuxos viselkedésének bemutatása után a szerző egy hasonló terjedelmű részt szán a lemezrészek felett lévő állományrendszerek ismertetésére szót ejtve a leggyakoribb típusokról, kellően részletesen elmagyarázva azok felépítését, működését, amely elengedhetetlen a későbbi problémák megértése érdekében.
Ezt követi az állományrendszerek beillesztésével kapcsolatos leírás.
Itt egy kicsit „operatívabb síkra lép" a szerző: megkezdődik a rendszer működésének bemutatása, első lépésként a rendszerindítás folyamatával, a rendszermag betöltésével, valamint a gyakoribb rendszertöltők ismertetésével. Az életben is úgy van, hogy a rendszertöltés következő eleme az egyes rendszerszolgáltatások indítása. Ezt a szerző is követi a *Linuxra* jellemző futási szintek fogalmának, jellemzőinek bemutatásával. Ha már elindult a rendszerünk, nyilván használni szeretnénk, aminek elengedhetetlen feltétele, hogy legyenek felhasználói fiókjaink.
A szerző nem akármilyen részletességgel veti bele magát a téma ismertetésébe: nem (csak) arról ír, hogy hogyan kell felhasználót létrehozni, de módszeresen elemzi a felhasználói adatbázisokat, a csoportok kezelését, az árnyékjelszórendszert, stb. És akkor még nem is beszélt semmi többről, ugyanis a bejelentkezés folyamata, az azonosítás, a *PAM* rendszer, a felhasználó lehetőségeinek beállítása a következő terjedelmes fejezet témája. Az ezt követő, mintegy száz oldal a programcsomagnak lett szentelve. Nemcsak a két legelterjedtebb csomagkezelőrendszert (*rpm*, *deb*) ismerhetjük meg, de megtanulhatunk forráskódot fordítani, és a programcsomagokkal végezhető egyéb műveletekben is jártasságot szerezhetünk. Ma már tudjuk, hogy az operációs rendszer nem is nevezhető annak grafikus felület nélkül, ezért az utolsó fejezet az *X* kiszolgáló és a grafikus felület működését, működtetését taglalja, tetemes részt szentelve az *X* beállításainak.



A második kötet

A második kötet a bevezetést követően a hálózatok üzemeltetéséhez szükséges alrendszerek ismertetésével kezdődik. A következő fejezet egy kicsit kakukktojásként hat, de mindenképp nagyon fontos: a rendszermag újrafordításának fortélyaival és a magmodulok kezelésének módjaival találkozhatunk.
A negyedik fejezet pedig már igazán a hálózatokról szól, méghozzá a protokollok ismertetésének szintjéről indulva, az útválasztó táblázatokon keresztül egészen a konkrét, terjesztésfüggő beállítóprogramokig.
Ha már van hálózatunk, következhet a hálózati szolgáltatások ismertetése, itt is egészen a protokollszintről indulva, megismerve a *TCP*, *UDP* protokollok lelkivilágát, az *inetd* szuperkiszolgálót, a névkiszolgálók szerepét, valamint a *DHCP* rendszer működését. Bár nem hackereknek szóló kézikönyvet tart az olvasó a kezében, mégis egy teljes fejezet van benne a hálózati lehallgatás gyakorlatáról. Ez természetesen egyrészt védelmi célzatú, másrészt szükségünk lehet rá hibakeresés, illetve a működés megfigyelésének céljából.
A hálózat természetesen attól hálózat, hogy több gép, illetve több alhálózat alkotja. Az ilyen „kisebb" hálózatok „logikai" összekapcsolásának módjait ismerteti az utolsó előtti fejezet, nem feledve a különböző biztonsági megfontolásokat. Végezetül a leggyakoribb hálózati szolgáltatásokat ismerhetjük meg, mint például az *NFS* kiszolgáló, *FTP*, *HTTP* kiszolgálók, *SMTP* alapú levelezés, vagy épp a *NIS* felhasználói adatbázis.
Tanítani nehéz... *Pere László* azonban már bizonyított a felsőoktatásban, és az ott szerzett tapasztalatát láthatjuk viszont a *GNU/Linux* rendszerek üzemeltetése c. sorozat minden egyes lapján.
Előismeretektől függetlenül mindenkinek bátran ajánlom.

Komáromi Zoltán

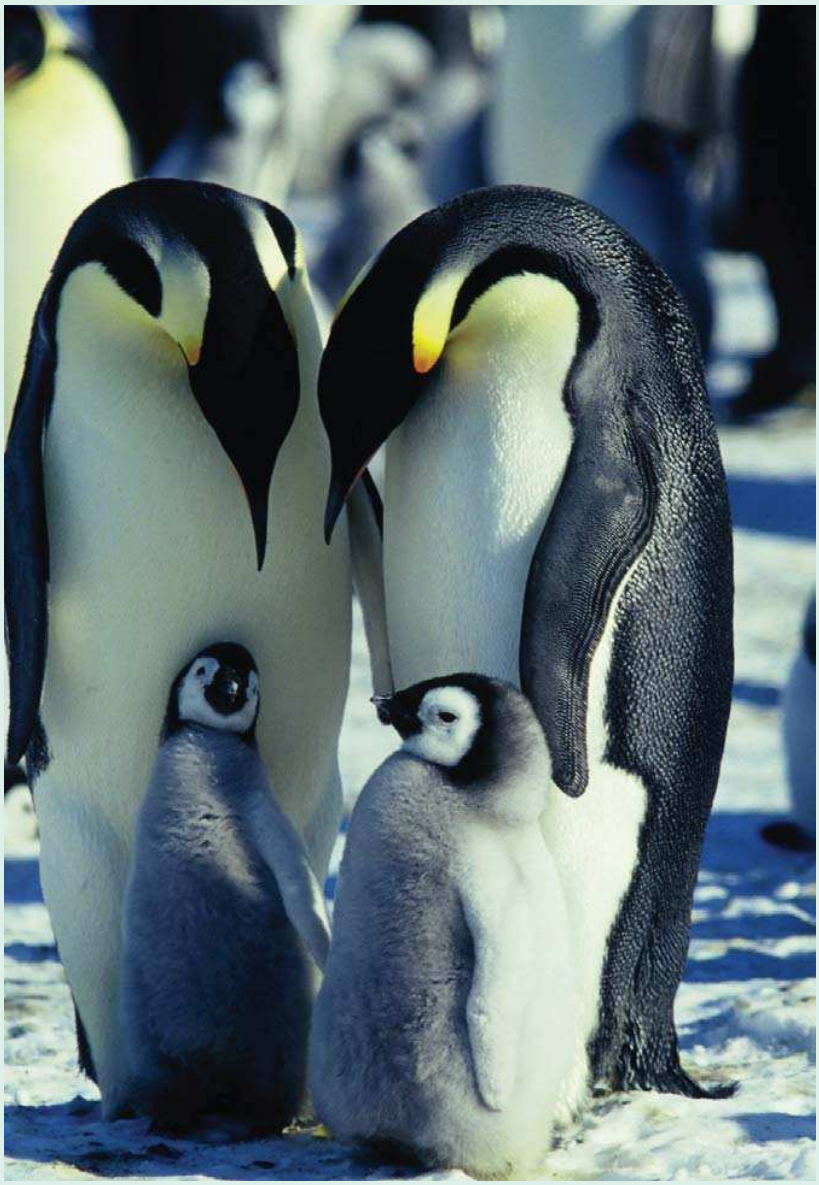
Premier – La Marche de L'Empereur

November 3-án mutatták be a hazai mozikban A pingvinek vándorlása című francia természetfilmet. Hogy a zord időjárású sarkvidéken forgatott 13 hónapnyi pingvinélet dokumentje miért verte félre a politikai, vallási és tudományos harangokat a tengerentúlon és miként telítette több tízmillió dollárnyira a kasszákat? Nos kérem, ez legalább annyira érdekes történet, mint az alternatív Nobel...

A filmszínház székébe vetve magam izgatottan vártam, ragadjon át reám is az amerikai nézők áhítata és harciassága, s példájukat követve a vetített képsorok hatására nyomban családvédelmi lelkigyakorlatokat szervezhessek, vagy éppen liberálisok zászlajával tüntethessek. Ugyanis elterjedt a híre annak, hogy e produkció következtében egyházközösségek a ***Szentlélek*** kifürkészhetetlen útjára lelve a szívek érintését hirdetik, míg például a melegek kirekesztettség-érzetükben inkább a rendező nyakát érintenék, többek spirituális jelképeket lobogtatnak, mások prostitúcióról, ***Krisztusról***, hűségről és hermafroditizmusról regélnek... Mármost lássuk, mi is történik a filmben. Megtudhatjuk, hogy bizony az antarktiszi pingvineknek nemcsak ***Nobel-díjas*** széklet-röppályája érdemes a világ figyelmére – hanem az a nyomóerejéhez méltó eltökéltsége és kitartása is, amely a fajfenntartással kapcsolatos tevékenységüket jellemzi. Dióhéjban a történet: ha eljő az idő, pingvinek ezrei kelnek ki az óceán habjaiból, s hóban-fagyban-libasorban masíroznak a Déli-sark kietlen belsejébe,

párzási helyükre. Pingvinlányok csőrpárbajjal megvívnak legényeikért, párokba rendeződve szerelmes tekintetek villanásában röpke tömegszex-jelenetnek lehetünk tanúi, majd ennek okán tömeges tojásgyártásnak.
Az immáron asszonnyá vált madarak fajfenntartó golyóbisaikat általadják hímpárjuknak őrzésre és kikeltésre, ők pediglen eltotyognak az óceánhoz halhabzsolásra.
Hastelve sprintelnek vissza a csapathoz, hol a fiókák világrajövetele, no meg a viszontlátás feletti össznépi családi örvendezés után a kiéhezett pingvinpapák ájuldoznak s indulnak élelem után.
Az újabb családi szétválás és egymásra találás újabb örvendezést szül, s ez az utanként több hónapos vándorlás mindaddig folytatódik, míg a született pingvincsemete fel nem cseperedik; ekkor mindenki el és szét, azaz vissza az óceán habjaiba. S mindez minden évben megismétlődik, véres, fókafalta, ragadozómadár-támadta drámákkal, folyvást tomporra esetlenkedő humorral, szívbemarkoló, megríkató fagyhalálokkal, has-összedörzsölgető *l'amour*-okkal.
Hogy ez miért merített mély, szinte végbélig nyúló visszértágulatos meditációba ily sok embert, „más" csoportokban hogyan keltett néhány pingvincsalád nyilvánosan megalázó, övön aluli érzéseket? Mi módon indított el egy gyönyörű, még sosem látott képekkel teli és remek zenével kísért természetfilm kultusszá nőtt hisztérikus politikai, vallási, szexuális hovatartozási és egyéb csatákat? Talán mert igaz lehet a mondás, miszerint nagyobb a különbség ember és ember, semmint némely ember és állat között...

Halusz Léna



Illusztrációk

Napfogyatkozás szabadon

Eclipse

Az Eclipse az egyik, ha nem „a” legnépszerűbb nyílt forrású, platformfüggetlen integrált fejlesztőkörnyezet, amely a több száz elérhető bővítmény segítségével szinte az összes ma használatos fejlesztési technológiát támogatja. A rendszer már a 3.1-es változatszámnál tart, legfőbb ideje tehát, hogy a Linuxvilág hasábjain is bemutatkozzék.

A projekt célja igazából nem pusztán egy integrált fejlesztőkörnyezet (***IDE – Integrated Development Environment***) létrehozása volt, hanem egy olyan keretrendszer megalkotása, amely a hozzá kapcsolódó bővítmények révén képes bármilyen programozó igény kielégítésére. S bár az ***Eclipse*** hivatalosan csak „keretrendszer”, a gyakorlatban egy teljes értékű környezetről van szó, amely képes bármilyen technológiához idomulni. Sok esetben nem csak a kódolást segíti, de például az ***UML/UML2*** modellező nyelv használatával a programtervezés folyamatát is támogatja. De ne szaladjunk ennyire előre, előbb ismerkedjünk meg a „csupasz” ***Eclipse*** legfontosabb jellemzőivel: a telepítés folyamatával, a használati alapfilozófiával és a nagyszerű lehetőségekkel. Rendszerünk, ha nem is teljesen platformfüggetlen, mindenképp többplatformos. Letöltéskor kiválaszthatjuk, hogy milyen operációs rendszeren, milyen architektúra felett szeretnénk használni. Létezik windowsos változat, linuxos állomány – ez utóbbin belül az ***x86***-os, az ***x86-64bit***-es és a ***PowerPC*** architektúra támogatott. Ezeken kívül elérhető még ***Solaris 8***-ra, ***AIX***-re, ***Mac OS X***-re, és akkor még nem beszéltem az egyes rendszereken belül a különböző grafikus felületekről, amit a rendszer támogat. Alapjaiban mindezt úgy éri el az ***Eclipse***, hogy a szoftver lelke java alapú. Nem nehéz kitalálni, hogy a futtatásához minimum ***Java*** futtatókörnyezetre (***J2RE***) van szükségünk, de az alapértelmezett használathoz inkább a ***Java*** fejlesztőkörnyezet (***J2SDK***) megléte javasolt. A „csupasz” fejlesztőkörnyezet ugyanis a ***Java*** nyelven történő programozást támogatja. Ha mi más nyelvet szeretnénk használni, ahhoz bővítményre lesz szükség. A használat és a felépítés természetesen a bővítmények esetében is ugyanaz, ezért először az alapkörnyezetet fogjuk megismerni.

Telepítés

A telepítendő változat az ***Eclipse 3.1.1-Linux-x86-GTK2*** lesz. Ehhez szükségünk lesz ***GTK2*** környezetre, amely ma már az összes terjesztésnek alapértelmezetten része, mi több, általában telepítve is van, tehát ezzel a résszel nincs gondunk. Amire jobban oda kell figyelni, az a ***Java*** fejlesztőkörnyezet. Ez általában egyetlen ***Linux*** terjesztésnek sem része, ezt a ***Sun*** weboldaláról kell letölteni. Ha nekünk már van ***Java*** fejlesztőkörnyezetünk, a következő lépést kihagyhatjuk.

Java környezet

Az ***Eclipse*** a ***Java2***-t, vagyis az ***1.4***-es sorozatot javasolja, de tapasztalatom szerint a Java5-tel (1.5-ös sorozat) is kiválóan működik. Mindenesetre mi most maradjunk az első lehetőségnél. Látogassunk el a ➲ ***http://java.sun.com/j2se/1.4.2/download.html*** címre, és itt válasszuk a ***J2SE SDK*** letöltését, abból is a linuxos, önkicsomagoló változatot. Ha ez megvolna, indítsuk el az állományt. A licencfeltételek elfogadása

után a *Java* fejlesztőkörnyezet kicsomagolódik egy *jdk1.4.2_09* nevű könyvtárba. Ennek tartalmát helyezzük át a */usr/lib/java* könyvtárba, majd a */usr/lib/java/bin/java* illetve */usr/lib/java/bin/javac* fájlokról készítsünk szimbolikus linket a */usr/bin* könyvtárban. Ezzel a *Java* fordító illetve futtatókörnyezet telepítésével el is készültünk.

Eclipse

Az *Eclipse* telepítése ennél egyszerűbb. Mindenekelőtt töltsük le a ➲ *http://www.eclipse.org/downloads/index.php* címről az *Eclipse SDK 3.1.1*-es, linuxos, *GTK2*-t támogató változatát. (Az operációs rendszertől függően fel fogja ajánlani az oldal a nekünk megfelelőt.) Ha ezzel megvolnánk, nincs is más dolgunk, mint kitömöríteni a *tar.gz* állományt egy általunk választott könyvtárba, és készen vagyunk. A fejlesztőkörnyezetet a `<választott könyvtár>/eclipse/eclipse` paranccsal indul.

Az első lépések

A program az indulás után rögtön arra kérdez rá, hogy melyik könyvtár legyen a munkaterületünk. Alapértelmezett ezt a könyvtárat fogja használni mindenféle fájlművelet során. Nemcsak az indítás során tudjuk ezt megadni, de a későbbiekben is a *File* menü *Switch Workspace* menüpontjával is változtathatunk rajta. A munkaterületek használatával jobban elkülönülnek egymástól a különböző projektek.

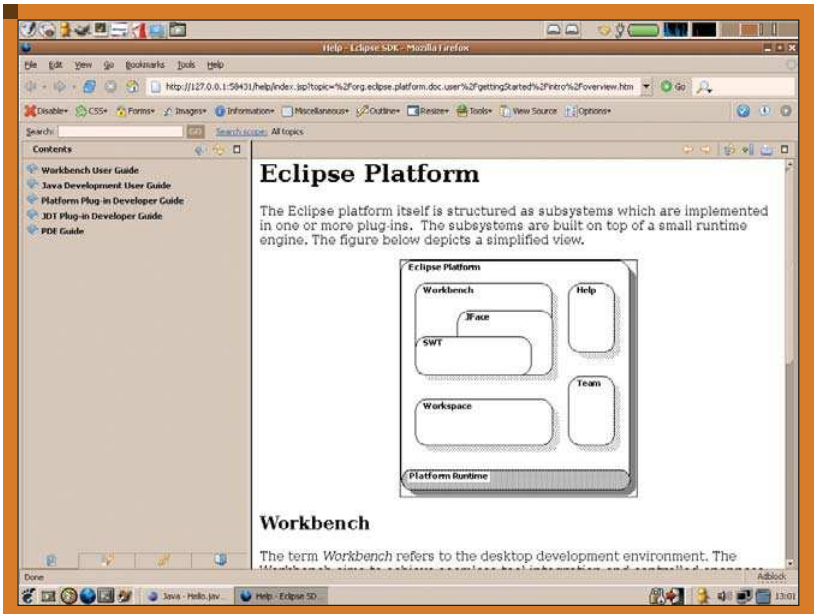
1. ábra Az indítási folyamat során adhatjuk meg a használni kívánt munkakönyvtárat

A program az első indítás során egy általános súgót jelenít meg. Innen érhetjük el többek között a teljes web alapú felhasználói dokumentációt, amely természetesen a helyi gépen található, tehát nincs szükség hozzá internetkapcsolatra. Ebben a bizonyos dokumentációban minden szükséges információt megtalálunk: rendkívül részletes és egyértelmű. Javaslom, hogy mielőtt bármilyen komolyabb munkába kezdenénk, olvassuk el legalább az áttekintő részeket, hogy képet kapjunk a program egészéről.

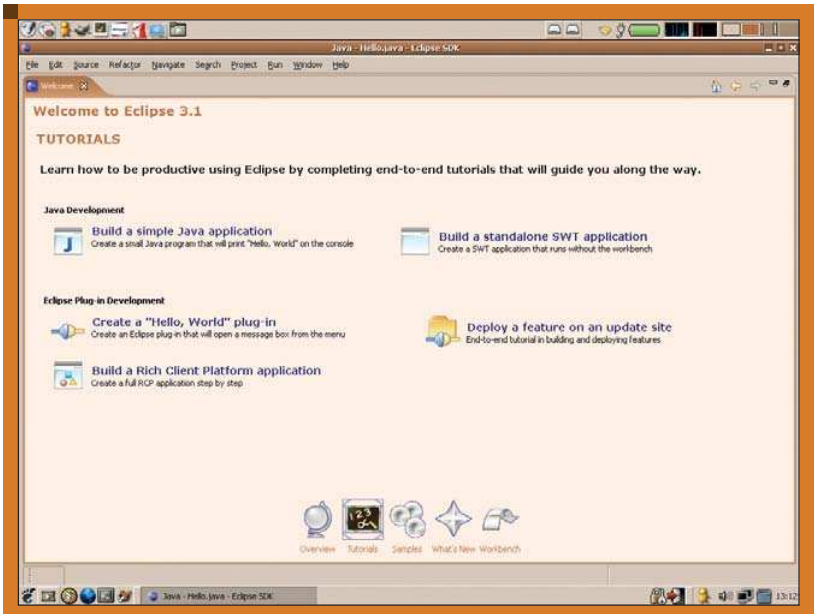
Az első indítás során az üdvözlőképernyőt láthatjuk. Itt rengeteg segítséget találunk az egyszerűbb folyamatok (például *Java* projekt létrehozása) bemutatásától kezdve egészen a már említett komplett dokumentációig. Érdemes barangolni itt is egyet. (Ez a felület egyébként a *Help* menü *Welcome* menüpontjának segítségével bármikor előhozható.)

Az üdvözlőképernyő bezárása után kapjuk meg a fejlesztőkörnyezet igazi felületét. A felület több alrekeszre tagolódik, amelyek elrendezése tetszőlegesen változtatható – ez a képlékenység egyébként az egész felületre is igaz.

A különböző fejlesztői tevékenységek ellátásához természetesen különböző felületi elemekre van szükségünk. Például néhol szükségünk van a szövegszerkesztőre, néhol nincs, néhol pedig egyenesen két darab kell. Ennek a helyzetnek a megoldására a rendszer különböző perspektívákat tartalmaz. Minden perspektíván más-más felületi elemek kaptak helyet, így ha például nyomkövetést szeretnénk végezni, elég csak „néze-



2. ábra A web alapú dokumentációból mindent megtudhatunk

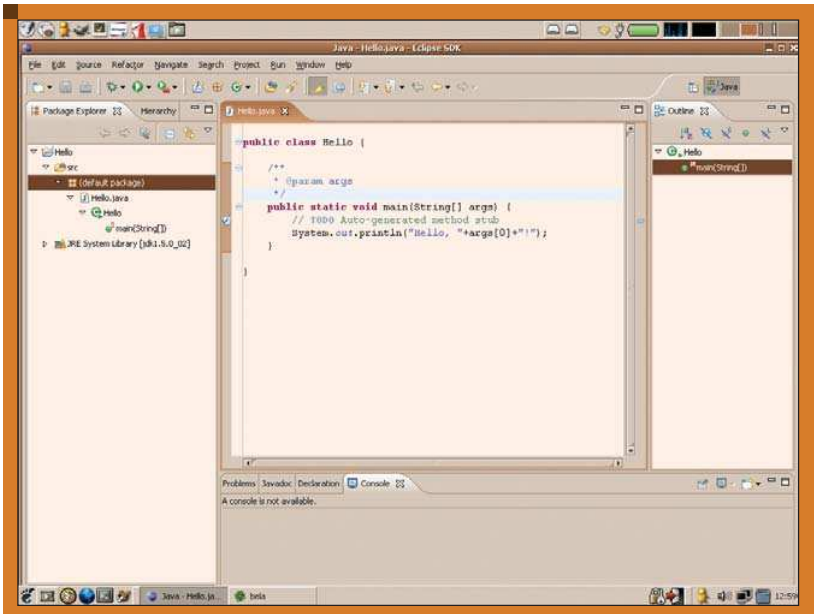


3. ábra Az első indítás során felkarol bennünket a segítőkész üdvözlőképernyő

tet váltani", és máris minden szükséges dolog a helyére kerül. Ha végeztünk, csak vissza kell mennünk az előző nézetbe.
Az alapértelmezett fejlesztői nézet minden programnyelv esetében nagyjából ugyanolyan: baloldalunk található a projekt böngésző, itt található a projekt minden eleme (lásd később) faszerkezetben mutatva.
Ez talán a környezet leggyakrabban használt eleme, egyfajta navigátor. Mellette található maga a szövegszerkesztő, amiről szintén később szólok. A szövegszerkesztő mellett jobb oldalon van az ***áttekintés*** (***Outline***) ablak, amely az éppen megnyitott fájlban, mint programkódban található osztályokat, tagfüggvényeket, változókat tartalmazza. A szerkesztő és az áttekintő ablak alatt található egy sok füles ablak, ahol megtalálhatjuk a programkódban lévő hibák listáját, változómeghatározásokat, valamint a program kimenetét (standard output).

Mindenek lelke: a projektek

A fejlesztőkörnyezet projekt alapú. Ez azt jelenti, hogy bármi, amit csinálunk, fejlesztünk, egy projekt része, eleme kell legyen, s ezt nem lehet megkerülni. Nem tudunk például olyat csinálni, hogy van 2-3 *Java* vagy *PHP* fájlunk, és azokat szerkesztgetjük... muszáj létrehozni egy projektet (egy afféle tartályt), amihez a fenti fájlokat hozzárendeljük. Ez természetesen néha hátrány: nem mindig van szükségünk ugyanis ilyen tartalmazó objektumra, amit a ***projekt*** testesít meg az ***Eclipse***-ben.
Habár az is igaz, hogy ilyen esetekben nincs szükségünk magára a fejlesztőkörnyezetre sem.
Az ***Eclipse*** célja, hogy segítségével össze tudjuk fogni a fejlesztést, keretet adjon a munkának, letisztult képet mutasson a bonyolult fejlesztésekről, egyszóval az összetett, nagy projektet támogatása, ami viszont elképzelhetetlen lenne a fenti megszorítások nélkül. Ennek megfelelően persze a fejlesztőkörnyezet lesz kissé bonyolult, amint az a beállítási lehetőségek mennyiségéből is látszik.
Ha más rendszerekről térünk át, netán egy fejlesztőrendszert teljesen nélkülöző (csak forrásfájlokat tartalmazó) kódot szeretnénk ***Eclipse*** segítségével továbbvinni, ahhoz elég egy üres projekt, ahová később néhány kattintással importálhatjuk a „külső" kódot. Így biztosítja a rendszerünk, hogy a más felépítésű fejlesztőkörnyezetekről is zökkenőmentesen térhessünk át ***Eclipse***-re. Később látni fogjuk, hogy nemcsak helyi fájlok, de távoli ***CVS***-ben tárolt projektek is játszva fejleszthetők tovább a segítségével. Látszik, hogy az ***Eclipse***-t fejlesztő közösség nagyon jól kitalálta a rendszernek ezt a részét: bármilyen távoli projekthez igen egyszerűen hozzáférhetünk.
Egy-egy ilyen projekt további sajátossága, hogy egyéni beállításokat adhatunk meg mindegyik számára. Ezek alapértelmezetten a globális beállításokból öröklődnek, de lehetőségünk van a különböző elvárásoknak megfelelően, úgymond



4. ábra Ilyen az alapértelmezett nézet

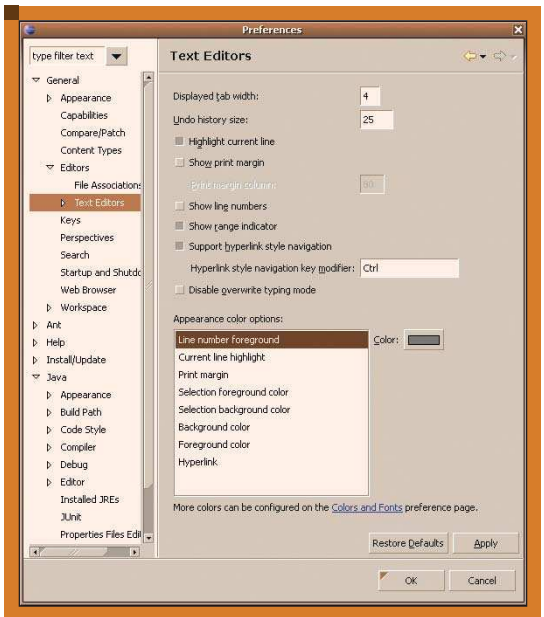
testreszabni egy projekt esetére a fejlesztőkörnyezet beállításait, amely változtatások azonban nem terjednek ki a többire.

A kódolás alapeleme: a szövegszerkesztő

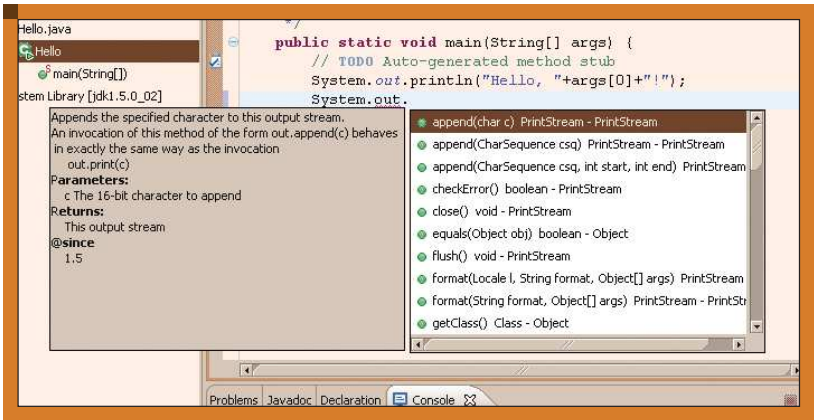
Ha programfejlesztés, akkor kódolás, ha kódolás, akkor valamilyen szövegszerkesztő. Az ***Eclipse***-ben található szövegszerkesztő méltó feladatára: elegendően sokat tud, hogy valóban alapeleme legyen a fejlesztőkörnyezetnek, de semmivel sem bonyolultabb annál, mint ami feltétlenül szükséges. Ez utóbbi akkor nagy előny, ha valaki találkozott már olyan szerkesztőkkel, mint például a ***JEdit***, amely vitathatatlanul nagyszerű, de egyrészt bonyolult és nehezen átlátható, másrészt az kódoló a számtalan nyújtott szolgáltatásnak mindössze töredékét használja, tehát bátran mondhatjuk, hogy felesleges.

A szerkesztő legfontosabb ismérvei:

- Támogatja a programkódon belüli blokkok elrejtését. Ilyen blokk elrejtés lehet egy már megírt, jól működő tagfüggvény „bezárása” oly módon, hogy a szerkesztőben csak a függvénydefiníció látszik, a függvény tartalma helyén három pontocska jelenik meg egy négyzetben, amire kattintva újra „lenyílik” az adott függvény kódja. Hasonló a helyzet a hozzáfűzött komment alapú dokumentációval, amelyet szintén egy sorra csukhatunk össze, amennyiben szükséges.
- Ismeri az adott programnyelvre jellemző szintaktikát, a szintaktikai kiemelést végez, amelyet szabadon testreszabhatunk, és nem utolsó szempont, hogy ha hibázunk, figyelmeztet. Nem csak a gépelési hibát, pontosvessző hiányát, vagy zárójel hibákat vesz észre, de például Java nyelv esetén a nyelvi sajátosságoknak megfelelő problémákat is megjeleníti. Ilyen például, ha egy statikus változót dinamikus környezetben szeretnénk használni, vagy ha nem létező, nem megfelelő típusú változót, objektumot akarunk használni, ha mellőzzük a szükséges ***try-catch*** blokkokat, elavult (***deprecated***) függvényeket próbálunk használni stb.
- Ismeri a programban használt gyári és saját fejlesztésű osztályokat, így a gépelés közben a gépelt szöveg alatt lenyíló menüben felajánlja a választható tagfüggvényeket, még a paraméterek kitöltésében is segít. Ez nagyon hasznos akkor, ha nem ismerjük a függvény paramétereinek pontos sorrendjét, típusát, stb. Ez is bővítményfüggő természetesen, tehát nem csak Java nyelven áll rendelkezésre, de a megfelelő bővítmények megléte esetén *C/C++* vagy akár ***PHP*** nyelven is.
- Tartalmaz kódolás segítőt (***code-assist***). Ez leggyakrabban abban nyilvánul meg, hogy bezárja helyettünk a zárójelet, idézőjelet, sortörés esetén az épp szerkesztett karaktersorozatot (***string***) megfelelő szintaktika szerint töri meg. Ide tartozik még a fejlesztő által megadott sablonok kezelése. Ezek többnyire ***XML/HTML*** címkék lehetnek, amelynek így elég elkezdeni beírni a nevét, a szerkesztő felajánlja a teljes kódrészletet, és segít az egyes jellemzők (***attributes***) kitöltésében.



5. ábra Az Eclipse környezet átfogó beállításait egy helyről módosíthatjuk



6. ábra Így segít nekünk a szövegszerkesztő kódolni

Csapatmunka

Ma már a legtöbb fejlesztés elképzelhetetlen csapatmunka és változatkezelés nélkül. Ennek fényében az ***Eclipse***-nek szerves része a ***CVS (Concurrent Versioning System)***változatkezelő. Nem csak azt teszi lehetővé, hogy kezeljük a változatokat, változtatásokat, de segítségével egyszerre több ember is dolgozhat ugyanazon a projekten, ugyanazokon a fájlokon. Nagy előnye a rendszernek, hogy nincs szükség külön szoftver telepítésére, a szolgáltatás az ***Eclipse***-be van építve. Ha létrehozunk egy projektet, aztán

■ *7. ábra* Új projekt létrehozása ablak (1. lépés)

■ *8. ábra* Új projekt létrehozása ablak (2. lépés)

■ *9. ábra* Új „főosztály” létrehozása ablak

azt „megosztjuk” úgy, hogy egy *CVS* tárba helyezzük, akkor minden egyes projekt-elem (fájl, erőforrás, stb) mellett megjelenik a fájl állapota, változatszáma, stb. A fájl helyi menüjében egyszerűen kattintásokkal lehet összehangolni a helyi változatot a *CVS* tárral mindkét irányban – mindenféle nehézség nélkül.

Annak érdekében, hogy ez a művelet sok fájl, sok változtatás esetén kényelmes legyen, rendelkezésünkre áll egy kifejezetten erre a célra készült nézet (*CVS Repository Explorer*). Itt láthatjuk a fájlokat, összehasonlíthatjuk a központi tárral, megnézhetjük, hogy a fájl mely sorai és hogyan változtak – mindezt természetesen grafikus formában, két egymás mellé helyezett, szinkronban mozgó szövegszerkesztővel, ahol színek, nyilak mutatják, hogy mi miről mire változott, akár visszamenőleg az összes „történeti” változtatásra. Ugyanitt lehetőség van a fizikai szinkronizációra is: a változtatások központi tárban történő rögzítésére. Véleményem szerint talán a rendszer egyik legnagyobb előnye, hogy ilyen jó *CVS*-kezelőt tartalmaz (külön szoftver esetében sem láttam még ilyen fejlett változatot).

■ *10. ábra* A „Run...” menüpont eredményezte beállítóablak

Egy példa

Egy alapvetően grafikus felületet természetesen igen nehéz írásban bemutatni. Ezt a nehézséget enyhítendő, megpróbálom egy egyszerű *Java* alapú projekt létrehozásával bemutatni a fent leírtakat: a rendszer alapvető felépítéséts.

A példa egy *„Hello Világ!”* program készítése, amit aztán később paraméterfüggő köszöntéssé alakítunk át. Indítsuk el az *Eclipse*-t.

A *File* menü *New->Project* menüpontjával kezdjük el létrehozni a projektünket. A megjelenő ablakban válasszuk a *Java Project* lehetőséget, majd a megjelenő ablakban adjuk a projektnek a *„Hello”* nevet. A többi beállítást hagyjuk változatlanul. Ismét egy ablak jelenik meg, amit szintén változatlanul hagyjunk jóvá.

Baloldalon, a böngészőben megjelenik a *Hello* projekt. Jobb gombbal kattintva a *New->Class* menüvel adjuk hozzá az egyetlen, fő osztályunkat. Adjuk neki a *„Hello”* nevet. Az alatta lévő csomagnévhez is nyugodtan írjuk ugyanezt. Ha nem tesszük, egy *default* nevű csomagba rakja az

osztályunkat. (A csomagok használatát a projektekéhez hasonlóan nem lehet megkerülni.) Ha ezzel megvolnánk, alul jelöljük be, hogy szeretnénk a `public static void main(String[] args)` nevű tagfüggvénycsonk létrehozását, majd egy bátor ***Finish*** gombnyomással nyugtázzuk az elvégzetteket.
lépés: miután a szövegszerkesztő megnyitotta számunkra a létrehozott osztályt, elkészítette a belépési pontot, a main függvény vázát, csak annyi a dolgunk, hogy beírjuk az alábbi sort a függvény törzsébe:

```
System.out.println("Hello
➥Világ!");
```

Futtassuk le az elkészült programot. A projekt böngészőben válasszuk ki a *Hello.java* fájlt, majd jobb gombbal elérhető helyi menüben válasszuk a ***Run as -> Java Application*** menüpontot. Ennek hatására a program lefordul, majd lefut, és alul, a ***Console*** feliratú fül fogja tartalmazni a program kimenetét:

```
'Hello Világ!';
```

Érdekes dolog a futtatás panelje a jelenlegi *Hello* alkalmazásunknak. A feladat most az, hogy a köszöntést tegyük személyhez szólóvá: a programnak átadott paraméterben szereplő néven köszöntsön.
A szövegszerkesztőben a *Hello.java* main függvényének törzsében írjuk át az egyetlen sort az alábbira:

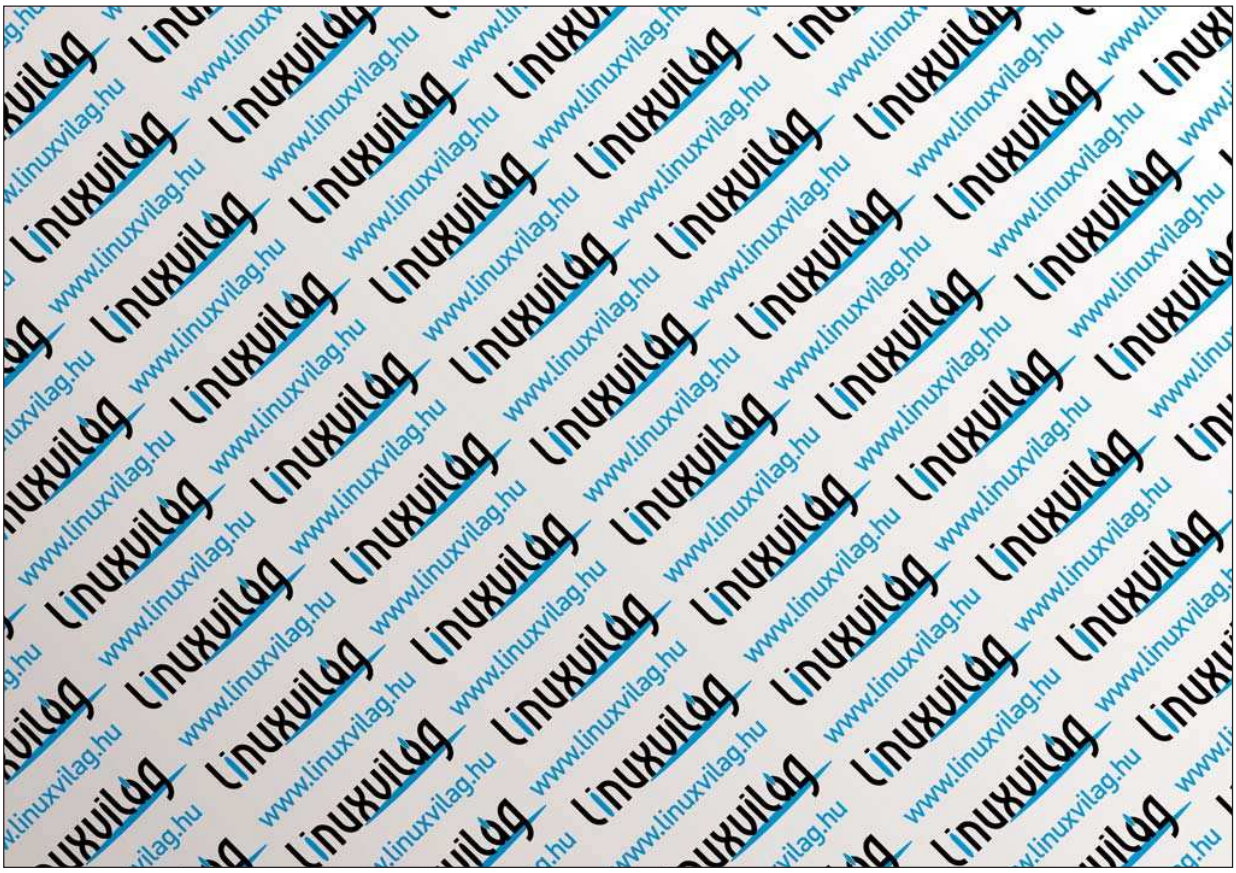
```
System.out.println("Hello,
➥"+args[0]+"!");
```

A ***Run*** menü ***Run...*** menüpontját választva láthatjuk a környezethez tartozó futtatási beállításokat, szabályokat. A ***Java Application*** csoport alatt található a mi *Hello* nevű futtatási szabályunk, amely az előző feladat utolsó lépéseként jött létre. Kattintsunk rá, és a jobb oldalon megjelenő panel ***Arguments*** címkéjű fülén a ***Program arguments*** mezőbe írjuk be a saját nevünket. Az ***Apply***, majd a ***Run*** gombok megnyomása után láthatjuk az eredményt. A továbbiakban a következő módosításig ez a futtatási szabály lesz érvényben, ez az alapértelmezett, de természetesen tetszőleges számú futtatási szabályt alkalmazhatunk egy adott projektre, ekkor a futtatáskor ki kell választani, hogy melyik szerint szeretnénk a programunkat lefuttatni.

Összefoglalás

A fent leírtak tényleg csak ízelítő jellegűek. Egy ilyen fejlesztőkörnyezet bemutatása egy teljes tankönyvnyi helyet emésztene fel – ez inkább csak kedvcsináló némi kezdő tanáccsal, hogy az esetleges kudarcok ne vegyék el kedvünket a továbbiaktól. Számtalan dolog kimaradt a cikkből, például a Java fejlesztés részletesebb ismertetése konkrétumokkal, a *Javadoc*, mint eszköz alkalmazása, a különböző bővítmények használata, és még sorolhatnám. Ezen hiányok pótlása cikkünk egy következő részében esedékes, és ahogy a showman mondaná: ***Kérjük maradjanak továbbra is velünk!***

Komáromi Zoltán
(komi@kiskapu.hu)
25 éves, mérnök-informatikus. Rendszermérnökként dolgozik a Kiskapu Kft.-nél, szabadidejében szívesen fotózik.



Kávéfőzés lépésről lépésre (7. rész)
Két folyamat beszélget. Az egyik azt mondja...

Az igazat megvallva még egy viccet sem hallottam, ami így kezdődött volna. Ez egyrészt szerencse, mivel némiképp alátámasztja azt a magamról alkotott képet, hogy nem vagyok arany fokozatú kocka. Másrészt furcsa, hogy a folyamatok közötti kommunikációt (IPC, inter process communication) biztosító eszközök gazdag tárháza eddig nem jelentett termékeny táptalajt az ismerten magas színvonalú informatikus humornak.

Az *IPC*, ahogy az az előző mondatból sejthető, egy gyűjtőfogalom. Az ide sorolható programozási eszközök nagyobb része ugyanazon a számítógépen futó folyamatok közötti kommunikációt tesz lehetővé. Ilyen például a jól ismert csővezeték, ami akár parancshéjból is könnyen létrehozható. Nem kisebb azonban azoknak az eszközöknek a jelentősége, melyek távoli folyamatok kommunikációjára használhatóak. Ezek közül a legszélesebb körben használt megoldás a foglalat.

A foglalat nagy népszerűségét leginkább annak köszönheti, hogy programozói szemszögből nem sokban különbözik egy közönséges állományleírótól. Az első lépés a megnyitás, ami a kapcsolat felépítését jelenti. Ezt követheti az írás és olvasás, azaz egy foglalat biztosítja mindkét irányban a kommunikációt. Végül le kell zárni, vagyis a kapcsolatot meg kell szüntetni. Ez tehát újabb ékes példája az unalomig ismételt bölcseletnek, mely szerint ***UNIX*** alatt minden fájl. Mi több, a módszer annyira jól működik, hogy nem kizárólag ***UNIX*** rendszereken honosodott meg. Valójában minden olyan operációs rendszer, ami képes ***TCP/IP*** alapú kommunikációra, lehetőséget ad valamilyen formában foglalatok létrehozására. Így nem véletlen, hogy a Java szabványos osztálykönyvtárában is találunk olyan osztályokat, melyekkel foglalatok hozhatók létre. A sorozatot megkoronázó, befejező cikkben ezek használatát vizsgáljuk meg közelebbről.

Mi az a foglalat?

A foglalat két folyamat közötti két irányú kommunikációnak egy végpontja. Jellemzően kiszolgáló-ügyfél felépítésű adatcserét biztosít. A kiszolgáló létrehoz egy ***foglalatot (socket)***, és ***hozzáköti (bind)*** egy úgynevezett kapuhoz (port). Ez egy egész szám, ami azonosítja a számítógépen futó folyamatot. A kiszolgáló ezután várakozik az ügyfélre. Amint egy ügyfél jelentkezik, annak kapcsolódási kérelmét ***elfogadja (accept)***, és ezzel kezdetét veszi az adatcsere. Az ügyfél ugyanúgy egy foglalatot hoz létre. Általában a kapuhoz való kötést az operációs rendszerre bízza, mivel ennek a foglalatnak a kapu száma érdektelen. Ismeri a kiszolgáló IP címét és a távoli folyamat kapu számát, ehhez ***kapcsolódik (connect)***. A kapcsolódással a kiszolgáló egy új foglalatot kap egy másik kapuhoz kötve, így a kommunikáció már egy másik kapun keresztül folyik. Eközben, mivel a kiszolgáló meghirdetett kapuja felszabadult, új ügyfélre várakozhat.

Java-ban mindez igen egyszerűen működik. A megoldáshoz két osztályt használhatunk a `java.net` csomagból. Az egyik a `Socket`, ami a fent vázolt kommunikációnak egy végpontját jelenti. A `ServerSocket` pedig egy olyan foglalat, amit kiszolgálók használhatnak ügyfelek csatlakozási kérelmeinek figyeléséhez és elfogadásához. Ezek az osztályok elrejtenek minden, operációs rendszertől függő megvalósítási kérdést, így a ***Java***-tól jól megszokott platformfüggetlen módon programozhatók.

Mennyi az idő?

Helló világ helyett ismét kísérletet teszünk valami olyasmi készítésére, aminek haszna is van. Az *1. Listában* egy egyszerű idő-kiszolgáló kódja látható.

1. Lista Egy egyszerű időkiszolgáló

```
import java.net.ServerSocket;
import java.net.Socket;
import java.io.PrintWriter;
import java.io.IOException;
import java.util.Date;

/**
* Egy ido kiszolgalot hoz letre, ami egy egyetlen szalon futo
* TCP szolgaltatast jelent. A csatlakozo ugyfel foglalatara
* kiirja a pillanatnyi datumot, majd lezarja a kapcsolatot.
* Jelenleg a kiszolgalot csak Ctrl+C-vel lehet leallitani.
* @see #run
*/
public class IdoSzolgaltato implements Runnable {

        /**
        * A kiszolgalo foglalata.
        */
        private ServerSocket kiszolgaloFoglalat;

        /**
        * A kiszolgalo egyetlen szala.
        */
        private Thread kiszolgaloSzal;

        /**
        * Letrehozza a parameterkent kapott kapun a szolgaltatast.
        * Kivetelt dob, ha a muvelet sikertelen volt. Ezt kovetoen
        * letrehoz egy szalat, es elinditja a szolgaltatast.
        * @param kapu
        * a TCP kapu szama
        * @throws IOException
        * ha a foglalatot nem sikerul letrehozni
        */
        public IdoSzolgaltato(int kapu) throws IOException {
                // foglalat letrehozasa
                kiszolgaloFoglalat = new ServerSocket(kapu);
                // szal letrehozasa, inditasa
                kiszolgaloSzal = new Thread(this);
                kiszolgaloSzal.start();
        }

        /**
        * A kiszolgalo szalat jelento metodus. Ez egy vegtelen ciklus,
        * amelybol a kilepes jelenleg nem megvalositott. Ezert a
        * <code>kiszolgaloFoglalat</code> soha nem kerul lezarasra!
        * A szal varakozik egy ugyfelre, majd sikeres kapcsolodas utan
        * kiirja annak foglalatara a pillanatnyi datumot, es lezarja
        * a kapcsolatot.
        */
        public void run() {
                while (true) { // vegtelen ciklus
                        try {
                                System.out.print("Varakozas ugyfelre...   ");
```

1. Lista (folytatás)

```
                        // a varakozas az ugyfel foglalatara
                        // blokkolja a szalat!
                        Socket ugyfelFoglalat =
                                kiszolgaloFoglalat.accept();
                        // az ugyfel foglalatanak kimeneti adatfolyama
                        PrintWriter iro = new PrintWriter(
                                ugyfelFoglalat.getOutputStream(), true
                        );
                        // pillanatnyi datum kiiratasa
                        iro.println(new Date());
                        // kimeneti adatfolyam lezarasa
                        iro.close();
                        // ugyfel foglalatanak lezarasa
                        ugyfelFoglalat.close();
                        System.out.println("siker");
                } catch (IOException kivetel) {
                        System.out.println("hiba");
                }
        }
    }

    /**
    * Az alkalmazas belepesi pontja letrehoz egy uj peldanyt az
    * osztalybol, <code>6666</code> parameterrel (kapu szam). Elkapja
    * a konstruktor kivetelet es hibauzenetet jelenit meg a stderr
    * csatornan, ha hiba tortent.
    */
    public static void main(String[] args) {
            try {
                    // uj ido szolgaltato a 6666-os kapun
                    new IdoSzolgaltato(6666);
            } catch (IOException kivetel) {
                    // reszletes hibauzenet egyebkent
                    System.err.println(
                            "Nem sikerult letrehozni a foglalatot: '" +
                            kivetel.getLocalizedMessage() + "'."
                    );
            }
    }

}
```

Az osztály látható módon példányosítás után létrehoz egy új szálat, és ebben kezeli a kapcsolódó ügyfeleket. Ez viszont nem azt jelenti, hogy a kiszolgáló több szálon fut! Egy ügyfél kiszolgálása alatt más ügyféllel nem tud foglalkozni. Jelen helyzetben a szálkezelés akár el is maradhatott volna, ám fontos hangsúlyozni, hogy a `ServerSocket` osztály `accept()` metódusa addig nem adja vissza a vezérlést, amíg nem érkezett új ügyfél. Az ilyen tulajdonságú műveleteket pedig érdemes külön szálba helyezni, így később könnyebben bővíthető az alkalmazás.

A belépési pont egy példányt hoz létre a kiszolgálóból a 6666-os kapun. Mivel a konstruktor kivételt dobhat, ezt itt lekezeljük. Hiba esetén a szabványos hibacsatornára írunk ki egy üzenetet. Megjegyzem, hogy egy kivétel `getLocalizedMessage()` metódusától ne várjunk sokat. Általában ugyanazt adja vissza, mint a `getMessage()`. Például az alkalmazást kétszer indítva nekem az alábbi jelent meg:

```
Nem sikerult letrehozni
➥a foglalatot: 'Address
➥already in use'.
```

A konstruktor egy `ServerSocket` objektumot hoz létre. Ezzel egy lépésben létrehoztunk egy foglalatot és hozzákötöttük a megadott kapuhoz. Ez a művelet okozhat

2. Lista A finalize működése

```
public class Alma {

        public Alma() {
                System.out.println("letrejott egy alma");
        }

        protected void finalize() throws Throwable {
                System.out.println("megszunt egy alma");
        }

        public static void main(String[] args) {
                new Alma();
        }

}
```

`IOException` kivételt, aminek a kezelését a hívóra bízzuk. Ezt jelezzük a metódus fejlécében. Továbbá létrehozzuk a fő szálat is, és azonnal el is indítjuk. A szál kódja jelen osztály `run()` metódusában található.
A `run()` egy végtelen ciklust tartalmaz. Ez nem programozási hanyagság, egy folyamatosan figyelő kiszolgálót így szokás megvalósítani. A ciklus minden lefutásban lekezel egy ügyfelet. Előbb fogadja a csatlakozási kérelmet, majd a kapott foglalathoz készít egy `PrintWriter` objektumot a kényelmes íráshoz. Ennek konstruktorában a második paraméterként megadott `true` jelenti az ***azonnali írás (autoflush)*** bekapcsolását.
A foglalatra kiíratjuk a pillanatnyi dátumot, majd lezárjuk az adatfolyamot és a foglalatot.
Ebben a helyzetben nincs szükségünk arra, hogy külön szál foglalkozzon minden egyes ügyféllel, mert egy ügyfél lekezelése nagyon rövid ideig tart. Komolyabb alkalmazások esetén ez azonban elkerülhetetlen. Ilyenkor a kapcsolat fogadása után azonnal el kell készíteni egy új szálat az ügyfélnek, majd annak átadni a kapott foglalatot. Ehhez egy másik osztály készítése szükséges, aminek a feladata az, hogy egyetlen ügyféllel foglalkozzon. Ez esetben pusztán az alkalmazás leállítása is szinkronizációs kérdéseket vet fel, melyek megoldása túlmutat a cikk keretein.

Az alkalmazás leállításánál maradva, be kell ismernem, hogy ez az egyetlen szállal működő program sem tökéletes. Látható, hogy a `kiszolgaloFoglalat` objektum `close()` metódusát sehol sem hívjuk meg. Ez nem jelent nagy problémát, mert a foglalatokat a nyitva maradt állományokhoz hasonlóan a ***JVM*** kilépéskor lezárja. Mégsem kimondottan szép ez a megoldás.

Azért hagytam mégis így, mert nem találtam egyszerű és rövid utat a foglalat bezárásához.
Egy rövid kitérő erejéig bemutatom, hogy milyen módszereken gondolkoztam, mielőtt erőt vett rajtam a lustaság. Pontosabban fogalmazva, mielőtt úgy döntöttem, hogy a foglalatok használatát bemutató példaprogramot nem szeretném annyira elbonyolítani, hogy a megvalósítás apró részleteinek sokasága eltakarja a lényeget.
Mivel a `ServerSocket` objektum a konstruktorban jön létre, adódik, hogy azt egy destruktor metódus zárja le. Java-ban nincs destruktor, mivel a ***szemétgyűjtő (garbage collector)*** eljárás gondoskodik az objektumok megszüntetéséről. Van azonban egy destruktor jellegű metódus, amelyet minden osztály ősosztálya, az Object definiál. Ez a finalize(), melynek működését a *2. Listában* olvasható rövid példa mutatja be.
A programot futtatva az alábbi kimenetet láthatjuk:

```
letrejott egy alma
```

Hogy-hogy nem szűnt meg? A szemétgyűjtő láthatólag nem volt hajlandó meghívni a `finalize()` metódust.

3. Lista Saját ügyfélprogram

```
import java.net.Socket;
import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.io.IOException;

/**
* Ugyfel alkalmazas az
* <code>IdoSzolgaltato</code>-hoz.
*/
public class IdoUgyfel {

        /**
        * A tavoli szamitogep neve
        *(lehet IP cim is).
        */
        private String szgepNev;

        /**
        * A szolgaltatas kapu szama.
        */
        private int kapuSzam;

        /**
        * Beallitja a szamitogep nevet es
        * a szolgaltatas kapu szamat.
        * Nem vegez semmilyen muveletet
        * a halozaton.
        * @param szgep
        * a szamitogep neve (lehet IP cim is)
        * @param kapu
        * a szolgaltatas kapu szama
        */
        public IdoUgyfel(String szgep,
        ➥int kapu) {
                szgepNev = szgep;
                kapuSzam = kapu;
        }

        /**
        * Lekerdezi a pillanatnyi idot
        * a beallitott szolgaltatotol.
        * Kivetelt dob, ha a muvelet kozben hiba
        * tortent.
        * @throws IOException
        * ha a lekerdezes iras/olvasas miatt
        * meghiusult
        */
        public void lekerdez() throws
        ➥IOException {
                // foglalat letrehozasa,
                // csatlakozas
                Socket foglalat = new
                ➥Socket(szgepNev, kapuSzam);
                // pufferelt olvaso keszitese
                BufferedReader olvaso = new
                ➥BufferedReader(
                   new InputStreamReader
                   ➥(foglalat.getInputStream())
                );
                // lekerdezes, kiiratas
                System.out.println
                ➥(olvaso.readLine());
                // olvaso lezarasa
                olvaso.close();
                // foglalat lezarasa
                foglalat.close();
        }

        /**
        * Az alkalmazas belepesi pontja letrehoz
        * egy uj peldanyt az
        * osztalybol, <code>"localhost"</code>
        * es <code>6666</code>
        * parameterekkel. A lekerdezes igy
        * a helyi szamitogeptol, a
        * 6666-os kapun tortenik. Hibauzenetet
        * jelenit meg a stderr csatornan,
        * ha a muvelet sikertelen volt.
        */
        public static void main(String[] args) {
                // ugyfel letrehozasa
                IdoUgyfel ugyfel = new IdoUgyfel
                ➥("localhost", 6666);
                try {
                        // lekerdezes
                        ugyfel.lekerdez();
                } catch (IOException kivetel) {
                        // reszletes hibauzenet,
                        // ha nem sikerult
                        System.err.println(
                            "A lekerdezes
                            ➥nem sikerult: '" +
                              kivetel.
                              ➥getLocalized
                              ➥Message() + "'."
                        );
                }
        }
}
```

Egészítsük most ki az `Alma` osztály `main(String[])` metódusát az alábbi sorral:

```
System.runFinalizersOnExit(true);
```

Fordításkor az alábbi figyelmeztető üzenetet kapjuk:

```
Note: Alma.java uses or
➥overrides a deprecated API.
Note: Recompile with -Xlint:
➥deprecation for details.
```

A második futtatás eredménye már a várt eredményt adja:

```
letrejott egy alma
megszunt egy alma
```

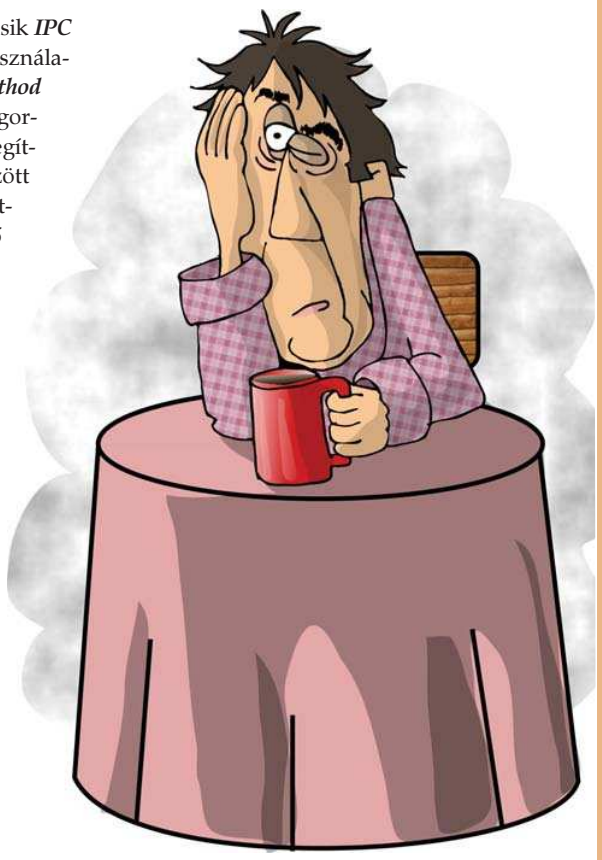
Ettől azonban nem lehetünk teljesen elégedettek. A figyelmeztető üzenetet azért kaptuk, mert a `runFinalizersOnExit(boolean)` metódus az új Java változatokban érvényét vesztette (deprecated). Használható ugyan, de nincs garancia arra, hogy a jövőbeni változatok is tartalmazzák, mivel nem minősül biztonságosnak. Egy többszálas alkalmazásban előfordulhat ugyanis, hogy a `finalize()` metódus még élő objektumok esetén is meghívódik, melyeket más szálak még használnak, ami akár holtpontot is okozhat.
A destruktor megközelítés tehát nem hozott eredményt. Nyilvánvaló, hogy valamilyen egyéb eseményt kell találni, aminek fellépésekor a foglalat bezárható. Egy vad megközelítésben rábízhatnánk az ügyfelekre a bezárást. Ha az ügyfél foglalatáról olvasnánk is, figyelhetnénk, hogy kapunk-e olyan üzenetet, ami a szolgáltatás bezárását kérelmezi. Ez nem csak nem biztonságos, de felveti azt a fontos kérdést is, hogy miért van szükség hálózatra egy szolgáltatás leállításához.
Linux alatt jellemzően ***jelzésekkel (signal)*** adhatjuk tudtára egy démonnak, ha be szeretnénk zárni. A jelzések mellesleg szintén az *IPC* csoportjába tartoznak. Ezt a módszert viszont tisztán a ***Java API***-ra támaszkodva nem használhatjuk. Valószínűleg a Java fejlesztői úgy ítélték meg, hogy a jelzések túlságosan platformfüggők ahhoz, hogy helyet kapjanak a szabványos osztálykönyvtárban. Szerény véleményem szerint létezik olyan értelmes keresztmetszete a különböző rendszereken használt jelzéseknek, amelyre alapozva nincs elvi akadálya a jelzéskezelés megvalósításának. Jelenleg csak különböző kerülő utakon, jellemzően nem teljesen platformfüggetlen módon lehetne ezt megoldani.
További megoldás a billentyűzetről, vagy valamilyen *GUI* felületről kezdeményezett bezárás. Egy ilyen alapokon nyugvó alkalmazás azonban soha nem kerülhet igazán háttérbe, ezért az ilyen megoldásokat szokás a programozói szakzsargonban gagyinak hívni. A félreértések elkerülése végett megjegyzem, hogy a probléma nem megoldhatatlan. Egy másik *IPC* eszközzel, a *Java*-ban használatos *RMI*-vel (***Remote Method Invocation***) átvágható a gordiuszi csomó. Az *RMI* segítségével két folyamat között távoli eljáráshívás valósítható meg. Így készíthető olyan alkalmazás, ami bezárja a főprogramot. Ez érzésem szerint egy kicsit hasonlít arra, amikor ágyúval lövöldözünk verebekre. Fogadjuk tehát el, hogy a ***Ctrl+C*** billentyűkombinációval lépünk ki. Ez egyébként egy *SIGINT* jelzés a *JVM*-nek, csak az nem hajlandó erről értesíteni a futtatott alkalmazást. Búsulásra viszont semmi ok, hiszen a virtuális gép kilépéskor lezár minden nyitott állományt, illetve foglalatot.
Térjünk vissza az időkiszolgálónkhoz. Fordítás és futtatás után a következőt láthatjuk:

```
Varakozas ugyfelre...
```

Elsőre elég, ha saját gépünkről próbáljuk ki a hálózati alkalmazást. Ha tűzfalunk nem akadályoz meg ebben, egy egyszerű ***telnet*** ügyféllel csatlakozhatunk a 6666-os kapura. Ekkor az alábbit olvashatjuk:

```
Trying 127.0.0.1...
Connected to
➥localhost.localdomain.
Escape character is '^]'.
Tue Nov 01 22:07:58 CET 2005
Connection closed by foreign
➥host.
```

A 4. sor bizonyítja, hogy a kiszolgáló kifogástalanul működik. Mi több, a kiszolgáló konzolján megjelenik a `siker` felirat, majd ismét várakozó állásba helyezkedik. Viszont ne érjük be ennyivel, írjunk saját ügyfélprogramot! Ennek kódja látható a *3. Listában*.
Az osztály kódja igazából önmagáért beszél. A konstruktor beírja a paraméterként kapott számítógép nevet és kapu számot a tagváltozókba.



A `lekerdez()` metódus kapcsolódik a távoli folyamathoz, létrehoz egy pufferelt olvasó folyamot, kiírat egy olvasott sort, majd bezárja a folyamot és a foglalatot.
A `main(String[])` metódus egy példányt hoz létre az osztályból, majd egy lekérdezést hajt végre. Ha ez meghiúsult, a keletkezett kivételt elkapja és hibaüzenetet jelenít meg a szabványos hibacsatornán.
A *Java* sorozat ezennel véget ért. Remélem, hasznos olvasmányt jelentett mind a kezdők, mind a haladóbbak számára ez az utolsó rész is. Ha a kedves Olvasónak bármilyen *Java*-val kapcsolatos kérdése van, írjon bátran!

Fülöp Balázs
(bigwig42@gmail.com)
21 éves, imádja a Túró Rudit, a Debian Linuxot és a teheneket. Kedvenc írója Slawomir Mrožek. Leginkább a számítógépes hálózatok biztonsága érdekli. A BME VIK műszaki informatikus szak hallgatója.

Keress, szimat!

A Beagle kereső

A Beagle egy igen okos, jó szaglású vadászkutya. Nyilván ez a tulajdonsága ihlette az azonos nevű keresőprogram készítőjét is, hiszen az alkalmazás szinte mindent megtalál, ami érdekelheti az embert.

Kicsit hasonlít az *Apple* új operációs rendszerében bemutatott *Spotlight* funkcióhoz: fájlokon belül is tud keresni, amibe beletartoznak címjegyzékeink, levelezésünk, internetes beszélgetéseink, még képeink és zenéink is.
Ahhoz, azonban, hogy ezt a szaglást ki tudjuk használni, teljesülnie kell néhány feltételnek.

Működési feltételek

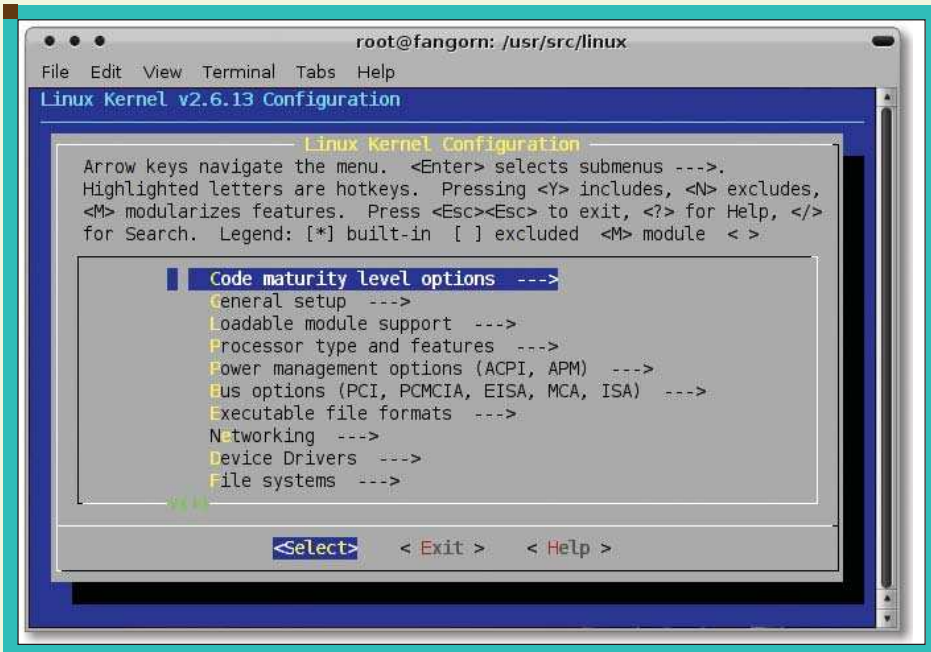
Először is kell egy fájlrendszer, amelyik tartalmazza a *„extended attributes”* funkciót. Ez az *SGI XFS*-ében és az *IBM JFS*-ében alapból megtalálható, *ext2*, *ext3*, *Reiserfs* esetén azonban külön kell engedélyezni a kernelben. Erre azért van szükség, mert a meta-adatokat a *beagled* démon ebben a térségben tárolja, mivel ezt a tért a lemezen gyorsabban el tudja érni, mintha a fájlok közé írná őket.
A másik dolog, amire szükségünk van a kernelből az az *inotify* funkció. Egészen új szolgáltatásról lévén szó, mindenképpen legalább *2.6.13*-as rendszermagot kell használnunk. Ez szintén a fájlrendszerek rész alatt érhető el. Azért van rá szükség, hogy új állományok létrehozásakor, azok módosításakor a kernel értesítse a *beagled* démont, így az a lehető leggyorsabban tudja frissíteni adatbázisát. Ha azonban nem szeretnénk a kernelfordítással bíbelődni, lehetőségünk van különböző disztribúciókhoz előre elkészített csomagokat letölteni a ➲ *http://beaglewiki.org/Inotify_Kernel* címről, vagy egyszerűen kihagyni ezt a lépést, ekkor azonban a démon nem lesz annyira hatékony, mint ha kernel szinten is támogatva van.
Ha a rendszermaggal végeztünk, tegyük fel a *Beagle*-t disztribúciónk csomagkollekciójából, melyhez nagy segítséget nyújt a ➲ *http://beaglewiki.org/Installing_Beagle* oldal.

A Beagle beüzemelése

Miután a *Beagle*-t telepítettük, nincs más dolgunk, mint futtatni a `beagle-settings` parancsot, melynek segítségével jelöljük ki, milyen könyvtárakban szeretnénk keresni sajátunkon kívül, vagy hol nem a megadott útvonalakon belül. Ugyan a *beagled* démon elég okos ahhoz, hogy tudja, nekünk mikor jó, és mikor nem, ha ő merevlemezünket vizsgálja, azért első indítása előtt érdemes beállítani a `BEAGLE_EXERCISE_THE_DOG` héjváltozó értékét 1-re, az alábbi paranccsal:

```
export
➥BEAGLE_EXERCISE_THE_DOG=1
```

Ezzel azt érjük el, hogy ránk való tekintet nélkül létrehozza a *Beagle* kezdeti adatbázisát, melyből aztán gyorsan tud keresni. Ezt az értéket, ha feltétlen nincs rá szükség, ne állítsuk be, mert kellemetlen lassulást okozhat, valamint a folyamat véget ér, ha munkájával végzett! Miután ezzel készen vagyunk, használjuk keresésre a `best` parancsot. Ez egy grafikus alkalmazás, mely nagyon kezesen tud működni, és ha a fejlécen zárjuk be, a tálcára kicsinyíti magát, és az *F12* gomb megnyomásával rögtön megjelenik újra.
Keresési szintaxis rendkívül egyszerű: például, ha azt írjuk a keresési mezőbe, hogy *„Linus Torvalds”*, megtalálja az összes olyan fájlunkat, e-mailünket, dokumentumunkat, amelyben szerepel a ez a szópár, ilyen sorrendben. Ha viszont idézőjelek nélkül adjuk meg ezt a keresési kulcsnak, kilistázza



1. ábra Fordítsunk új kernelt!

nekünk az összes előbb felsorolt adatunkat, melyben vagy a ***Linus***, vagy a ***Torvalds*** szó, vagy mindkettő megtalálható legalább egyszer.
Ha azonban nem szeretnénk hogy a ***Torvalds*** szó szerepeljen a talált elemekben, hiába van mellette a ***Linus*** szó, használjuk a – tagot a szó előtt, például ***Linus -Torvalds***. A ***Beagle*** csak úgy tud keresni, hogy a kis-, és nagybetűket megkülönbözteti, erre figyeljünk!
Létezik még egy keresési interfész, aminek használata azonos a BEST-tel, ez pedig a webes felület. Ennek eléréséhez a *„localhost:8888”* hivatkozást kell beütnünk böngészőnk címsorába, ha fordításnál engedélyeztük a ***WebServices*** támogatást a `--enable-webservices=yes` opció kapcsolásával a `./configure` parancs futtatásakor. Ha ezt megtettük, kapunk egy új fület is a `beagle-settings` beállítóalkalmazáshoz *„Networking”* címmel.

Parancssori eszközök – beagle-config

Parancssori beállításra a `beagle-config` használható. A beállításokat a készítők – csakúgy, mint a grafikus felület esetében – kategóriákon belül végezhetjük el:

- *deamon*: démonnal kapcsolatos beállítások
- *webservices*: a webes felület beállításai
- *indexing*: A fájlok indexelésével kapcsolatos beállítások
- *searching*: Kereséssel összefüggő beállítások. Nálam nincs semmilyen lehetőség.

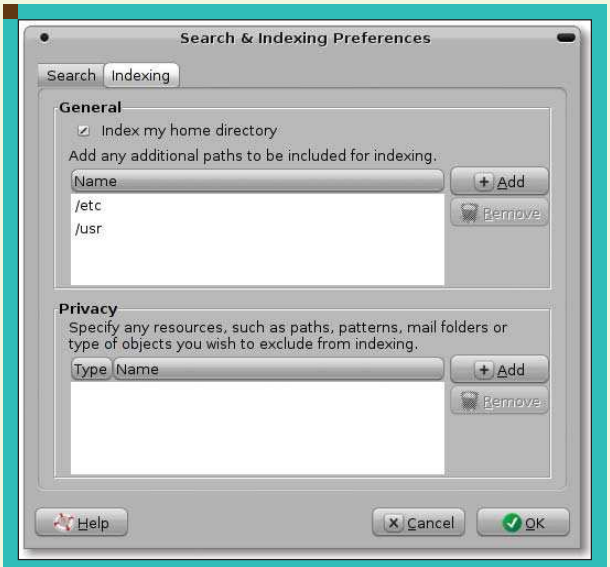
A grafikus beállítóeszköz kevesebb beállítási lehetőséggel rendelkezik, mint a `beagle-config`, de azt jóval egyszerűbb használni.

beagle-query

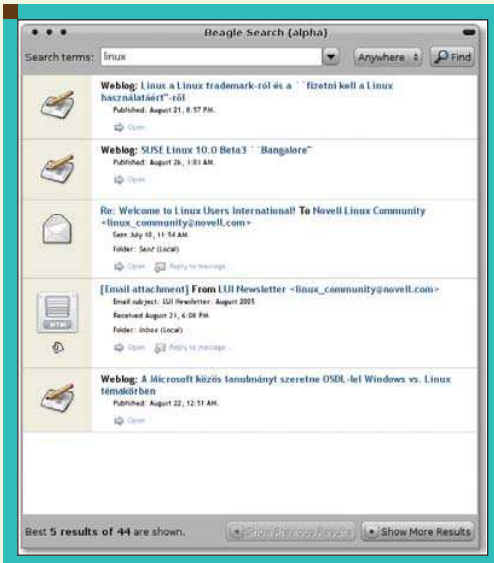
A `beagle-query` alkalmazás szintén parancssori, de keresésre szolgál. Legerősebb funkciója a `--live-query` kapcsolóban rejlik, amivel is nincs vége a keresésnek, tehát ha új fájlt hozunk létre, vagy már meglévőt módosítunk, azt azonnal észleli, és megjeleníti a képernyőn. Ha ezt az opciót nem alkalmazzuk, a kimenete hasonló a minden ***UNIX*** szerű rendszeren régóta megtalálható `slocate` program kimenetével, tehát a program futtatásának *„van vége”*, és csak fájlneveket ír ki.
A `--verbose` poció segítségével sok további információhoz juthatunk az adott fájlokról, míg a `--max-hits` kapcsolóval a találatok számát korlátozhatjuk.

Segédeszközök

Telepítés után hozzájutunk egy másik parancssori alkalmazáshoz is, ami a `beagle-info` nevet viseli. Az alkalmazás segít nyomon követni, hogy éppen mit is csinál a démon.



2. ábra A Beagle beállításai képernyő: kevés lehetőség, egyszerű felület



3. ábra A BEST front-end tényleg a legjobb

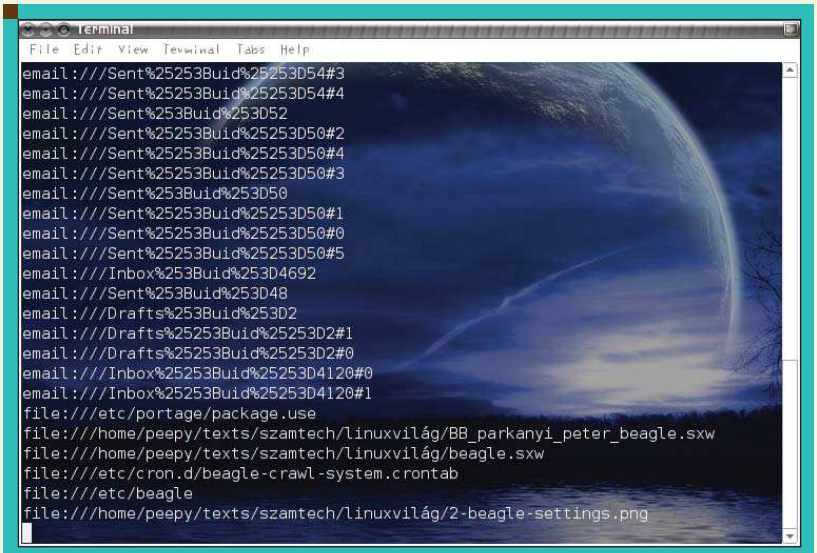
Különböző opcióinak kimenetére azonban léteznek különálló alkalmazások is, mint a `beagle-index-info` a `--index-info` kapcsoló megfelelője, és közli velünk a *Beagle* különböző statisztikáit.

Milyen fájlokban kereshetek?

Szimatunk képes megtalálni legrejtettebb adatainkat *Gaim* logjainkban, *Evolution* és *KMail* levelezésünkben, *Firefox* és *Epiphany* könyvelzőink között, a *Blam* és *Liferea* hírolvasóink adataiban, minden egyszerű szövegfájlban, *OpenOffice.org* által készített dokumentumokban, *AbiWord* írásokban, *PDF* állományokban, *Microsoft Office* fájlokban, *Java*, *Python*, *C*, *C#* és még sok más forráskód közt, *Texinfoban Monodocban*, *Docbookban*, *man* oldalakban, *Windows help* állományokban, *jpeg*, *png*, *svg* képeinkben, *mp3*, *ogg* és *flac* formátumú zeneink közt, és végül, de nem utolsó sorban alkalmazásindítókban.

Összegzés

A *Beagle* egy nagyon hasznos eszköz tud lenni azoknak, akik sosem tudják, mi hol van számítógépükön. Akik nem ebbe a csoportba tartoznak, hát, ők sem tudhatják mindig, hogy éppen melyik *IM* beszélgetésükben említette valaki az *„agresszív kismalac”*-ot, ami egy rendkívül humoros viccben volt benne, amit azonban azóta harmadik alkalommal felejt el. Erre pedig kitűnően alkalmas a *Beagle*, olyan hiányosságai ellenére, mint például, hogy rejtett könyvtárakban, vagy fájlokban nem találja meg, amire vágyunk, és az *X-Chat* naplóiban sem tud keresni jelenleg. A fejlesztők azonban felhívják rá a figyelmet: a program még *alpha* állapotú, ezután még béták lesznek, aztán pedig *RC*-k is szép számmal, ami viszont már nem görög betű. Nem tökéletes, de ígéretes, és nagyon jól használható. A *Mono* környezet használata viszont nem túl szerencsés a sebesség miatt, főleg, hogy a démon is ebben lett írva, ugyanis néha elég későn veszi észre a fájlrendszerben történt változásokat.



4. ábra A beagle-query akció közben

Párkányi Péter
(peter.parkanyi@gmail.com)
1991-ben születtem, és idén elsős vagyok egy pécsi gimnáziumban. Szeretem a jó zenét, szeretek kerékpározni, és kosárlabdát is szívesen játszom. Jelenleg Gentoo Linuxot használok.

LDAP Account Manager

Ha cégünk microsoftos rendszergazdáinak szeretnénk megadni a lehetőséget a Samba kiszolgálók felhasználóinak kezelésére, próbáljuk ki ezt a kiválóan tervezett, webes eszközt.

Az *LDAP Account Manager (LDAP fiókkezelő, LAM)* *POSIX* fiókok és felhasználókhoz, csoportokhoz és *Microsoft Windows* alapú gépekhez rendelt *Samba SAM* fiókok kezelésére használható alkalmazáscsomag. A *LAM* tetszőleges webkiszolgálóval használható, amennyiben az támogatja a *PHP4*-et. Az *LDAP* kiszolgálóval titkosítatlan vagy *SSL* feletti kapcsolatokat létesít.
A *LAM PHP* nyelven készült, és a *LAM* honlapról, a *sourceforge.net/projects/lam* címről tölthető le, illetve *GNU GPL* hatálya alatt futtatható.
Az alapértelmezett jelszó: *lam*.
Lehetőleg minden a *LAM* használatával kapcsolatos távoli művelethez *SSL* felett kapcsolódjunk a webkiszolgálóhoz. Ha biztonságos kapcsolatokat akarunk létrehozni, akkor *Apache* webkiszolgálónkat úgy állítsuk be, hogy a *LAM* kapcsolatainak felépítését csak *SSL* felett engedélyezze.
A *LAM* használatához a következőkre van szükség:

■ *1. ábra* Részlet a LAM beállító lapjából

- *PHP4* támogatással ellátott webkiszolgáló
- *PHP4* (letölthető a *PHP* honlapról, *www.php.net*)
- *OpenLDAP 2.0* vagy újabb változat
- *CSS*-támogatással rendelkező webböngésző
- *Perl*
- A *gettext* csomag
- *mcrypt+mhash*
- *SSL*-támogatás (nem kötelező, de jó, ha van)

A telepítéssel kapcsolatos, amúgy könnyen követhető útmutatást a terjesztés .tar fájlja tartalmazza. Ha végeztünk a *LAM* telepítésével, indítsuk el webkiszolgálónkat, majd böngészőnkkel nyissuk meg a *LAM URL*-jét. Kattintsunk a *Configuration Login (Beállítási bejelentkezés)* hivatkozásra, majd a *Configuration Wizard (Beállító varázsló)* hivatkozásra kattintva indítsuk el az alapértelmezett profilt.
A *LAM* beállításakor az *LDAP* kiszolgálónak már üzemelnie kell, így a helyes műveleteket jóvá tudjuk hagyni. Azt is megtehetjük, hogy a *lam.conf_sample* fájlt *lam.conf* névvel a *config* könyvtárba másoljuk, majd kedvenc szerkesztőnkkel a helyi igényekhez és körülményekhez igazítjuk a beállításokat. A varázsló által létrehozott profilfájlban szereplő megjegyzések és egyéb tudnivalók rendkívül hasznosak, segítségükkel számos gyakori hibát el tudunk kerülni.
A *LAM* beállításszerkesztője számos megfelelő beállítási értéket tesz elérhetővé (*1. ábra*), ilyen például a rendszerünkben engedélyezett legnagyobb és legkisebb *UID/GID* érték. Lehet, hogy

■ *2. ábra* A LAM a leggyakrabban használt oldallal, a felhasználókezelővel indul

az alapértékek nem felelnek meg, ez esetben módosítanunk kell a jól ismert windowsos hálózati felhasználókhoz és csoportokhoz tartozó alapértelmezett fiókértékeket. A legjobb megkerülő megoldás, ha az alsó határértékeket ideiglenesen nullára állítjuk, így meg tudjuk adni a kezdő beállításokat. Később a *LAM* felhasználók és csoportok hozzáadására történő használatba vétele előtt persze ne feledjük ezeket a fontos értékeket visszaállítani.
A *LAM*-nak vannak különleges szolgáltatásai is. Érdekes és meglepő lehetőség például, hogy a legtöbb képernyőről készíthetünk egy a beállítási adatokat összegző *PDF* fájlt. Nem rossz ötlet. Amikor bejelentkezünk *LAM* alá, a nyitó képernyőről a felhasználókezelőbe jutunk (*2. ábra*) – ésszerű lépés, hiszen így a legtöbbet igényelt szolgáltatást azonnal használatba vehetjük. A meglévő felhasználók módosításának és az új felhasználók

3. ábra A LAM segítségével könnyedén megoldható a Windows tartományi tagok kezelése

hozzáadásának folyamata könnyen követhető, az oldalak elrendezése, megjelenése is ezt segíti. Első lépésként az általános beállításokat, majd a unixos beállításokat, végül a ***Samba*** fiók jellemzőit kell megadnunk. Minden lépésnél rá kell kattintanunk egy gombra, amely végigvezet bennünket a folyamaton. Amikor végeztünk, egyszerűen csak a ***Final (Végső)*** gombra kell kattintanunk.

A felhasználói és csoportfiókok szerkesztőképernyője hasonló gyorsasággal ismerhető meg. Az állomásfiókok kezelése önműködően, az ***smbldap-tools*** parancsfájlokkal történik. Ebből következően a ***Hosts (Állomások)*** szerkesztőképernyőt (*3. ábra*) elég ritkán fogjuk látni. A ***LAM*** egyik, esetlegesen bosszantó jellegzetessége, hogy bizonyos hagyományokat ráerőltet a rendszergazdára. A ***LAM*** például annak ellenére sem engedi olyan ***Windows*** felhasználó vagy csoport létrehozását, amelynek neve nagybetűt vagy szóközt tartalmaz, hogy az operációs rendszerek gond nélkül kezelik ezeket. Figyelembe véve, hogy még az alapértelmezett windowsos fióknevekben is szerepelnek ilyen karakterek, kicsit bosszantó tud lenni ez a hiányosság. Ettől eltekintve a ***LAM*** hasznos társ a rendszergazdák mindennapjaiban.

Linux Journal 2004. 125. szám

John H. Terpstra
a PrimaStasys, Inc. műszaki vezetője – azé a cégé, mely újfajta informatikai eszközök kipróbálásában segíti a szervezeteket, és a gyakorlati megoldásokban is megtérülő módosításokat vezet be. A Samba csapat régi tagja, részt vesz az Open Source Software Institute kuratóriumának munkájában, továbbá ő a hivatalos Samba-3 HOWTO and Reference Guide és a Samba-3 by Example leírások szerzője.

Faxoljunk számítógéppel – Hylafax

A faxolás ma már tulajdonképpen kőkorszaki technológiának számít. Ennek ellenére – főleg az üzleti életben – még mindig fontos módja a dokumentumok továbbításának. Előnye az internetes megoldásokkal szemben, hogy elterjedt, olcsó, egyszerű eszközökkel megvalósítható, a dokumentumok könnyen iktathatóak és nyomon követhetők, és hivatalos (aláírt-lepecsételt) dokumentumok küldésére egyelőre még alkalmasabb mint az internetes-digitális aláírásos megoldások.

A továbbítással szembeni előnye nyilvánvalóan a gyorsaság, és az olcsóság, de emellett faxoláskor a visszacsatolás is fontos előny, konkrétan hogy rögtön kapok visszajelzést arról, hogy elment-e a fax vagy nem.
A ***HylaFax*** egy komoly, céges szintű szolgáltatásokat nyújtó faxrendszer ***Linux/UNIX*** platformra, amely szabadon letölthető a ➲ http://www.hylafax.org helyről. Ugyanitt jelentős mennyiségű dokumentáció is rendelkezésre áll a használatához. A ***hylafax*** csomag két részből áll, a szerverből és a kliensből. Bár „csak" ***Linux/UNIX*** platformra elérhető, a kliens egyszerű felépítése miatt könnyen portolható más rendszerekre. Léteznek windowsos, mac-es, és platformfüggetlen ***HylaFax*** kliensek is.
A rendszer nem csak faxok küldésére és fogadására alkalmas, hanem kliens-szerver felépítése következtében lehetőség van arra, hogy a csomagot a hálózati szerverre telepítve a munkaállomásokról küldhessünk-fogadhassunk faxokat. A rendszer robusztus, működése kis forgalmú otthoni szerverként és többmodemes céges rendszerként is megbízható, stabil. A rendszer alapfunkciói közé tartozik a faxok küldése, fogadása, a faxok elosztott továbbítása e-mailben illetve lehetőség van a faxok automatikus nyomtatására is.
A fax bármilyen méretű lehet, és egyaránt lehet 98 vagy 196 lpi felbontású.
A program ***TIFF*** formátumban menti, ami közvetlenül megnézhető, e-mailben továbbítható és nyomtatható.
A működtetéshez használt modem szintén bármilyen, ***Linux*** alatt használható, faxolásra képes készülék lehet, mivel a ***Hylafax*** egyaránt támogatja a ***Class 1***, ***Class 2*** és ***Class 2.0*** szabványú modemeket.
A faxok küldése a `sendfax` programmal történik. A kimenő fax anyaga megint csak sokféle formátumú lehet, mivel a `sendfax` intelligens módon átalakítja a küldésre alkalmas ***PostScript*** vagy ***TIFF*** formátumra. Lehetőség van automatikus fedőlapgenerálásra és beillesztésre, illetve a `sendfax` képes egyszerű szöveges telefonkönyv kezelésére is. Egyszerűen konfigurálható e-mail-fax átjáróként is.
Kliens-szerver felépítéséből adódóan lehetőség van arra, hogy a kliensek helyi hálózaton vagy interneten keresztül érjék el a szervert. A rendszer a ***HylaFAX™ Client-Server Protocol*** és alfanumerikus oldalakhoz a ***Simple Network Paging Protocol (SNPP, RFC 1861)*** használja, mindkét protokoll szabad és jól dokumentált. A hozzáférés felhasználó és gépnév alapon beállítható.
Lehetőség van több modem egyidejű használatára, a szerver hatékonyan elosztja a feladatokat közöttük. Képes megkülönböztetni csúcsidőt és csúcsidőn kívüli időszakot, és telefonszám alapján szűrni a kimenő faxokat.
A rendszer konfigurálástól függően két szerverfolyamatot futtat. A ***faxq*** felelős a sorkezelésért és a munkák időzítéséért, míg a ***hfaxd a*** kliens-szerver protokoll kezelője, és a kliensekkel való kapcsolatért felel. Ezen kívül a ***faxgetty*** modemfigyelő fut még szerveroldalon.

Telepítés, konfigurálás

A rendszer beüzemelése a következő lépésekből áll:

- A ***Hylafax*** csomag telepítése
- A modem beüzemelése
- A szerver konfigurálása a ***faxsetup*** program segítségével
- Modem hozzáadása a ***faxaddmodem*** szkripttel
- A kimenő és bejövő szolgáltatások beállítása
- A hozzáférések megadása
- A karbantartási feladatok beállítása

A Hylafax telepítése

A csomag a ➲ ***http://www.hylafax.org*** honlapról (is) letölthető. A forrás mellett binárisok is vannak ***Red Hat***-re, ***Slackware***-re, ***Suse***-ra és ***Solarisra***.
A ***Red Hat/Fedora*** binárisok frissek, létezik Intel és AMD64 platformra fordított változat egyaránt. A többi sajnos nincs rendesen frissítve. Fordítása sem okoz különösebb gondot, hiszen a „szokványos" módon történik (`./configure; make; make install`, mindez természetesen rootként.)
A program legfontosabb része a ***spool*** könyvtár alatt található, ami általában a ***/var/spool/hylafax*** (*1. Lista*).

1. lista A /var/spool/hylafax könyvtár tartalma

```
root@magno:/var/spool/hylafax# ls -lai
total 23
82892 drwxr-xr-x  17 uucp uucp  528 2005-07-20 20:25 ./
   43 drwxr-xr-x  17 root root  504 2005-07-23 16:08 ../
82909 -r--r--r--   1 root root 5426 2004-11-01 14:00 COPYRIGHT
82908 prw-------   1 uucp uucp    0 2005-08-09 13:44 FIFO|
96497 prw-------   1 uucp uucp    0 2005-07-02 11:21 FIFO.ttxS1|
83130 prw-------   1 uucp uucp    0 2005-07-20 20:25 FIFO.ttyS1|
82907 drwx------   2 uucp uucp   48 2004-11-01 13:58 archive/
82893 drwxr-xr-x   2 uucp uucp  472 2005-07-20 20:25 bin/
82894 drwxr-xr-x   2 uucp uucp   48 2005-07-20 19:45 client/
82895 drwxr-xr-x   2 uucp uucp 1904 2004-11-01 14:00 config/
82896 drwxr-xr-x   2 uucp uucp   96 2004-11-01 14:05 dev/
82904 drwx------   2 uucp uucp  312 2005-07-07 13:49 docq/
82903 drwx------   2 uucp uucp  216 2005-07-20 18:15 doneq/
82897 drwxr-xr-x   2 uucp uucp  456 2005-07-20 20:25 etc/
82898 drwxr-xr-x   2 uucp uucp  208 2005-02-16 21:30 info/
82899 drwxr-xr-x   2 uucp uucp 4264 2005-08-09 13:42 log/
82906 drwx------   2 uucp uucp   48 2004-11-01 13:58 pollq/
82900 drwxr-xr-x   2 uucp uucp  264 2005-08-09 13:43 recvq/
82902 drwx------   2 uucp uucp   72 2005-07-20 18:15 sendq/
82901 drwxr-xr-x   2 uucp uucp   96 2005-07-02 11:22 status/
82905 drwx------   2 uucp uucp   48 2005-07-07 13:49 tmp/
```

A Hylafax saját könyvtárai közül a fontosabbak tartalmát az *1. táblázat*-ban foglaltuk össze.

A modem beállítása

Linux alatt ez a szokványos módon történik, ami viszont a modem fajtájától függ. Hogy mennyire bonyolult, az esetfüggő. Egy sima külső soros modemnél például valószínűleg nincs szükség semmi extra beállításra. Ellenőrizzük le hogy melyik `tty` porton érhető el modemünk és hogy üzemképes-e. Ezt például a

```
minicom /dev/ttyS1
```

paranccsal tehetjük meg, amennyiben a modem a második soros portra csatlakozik. (***Windows*** alatt ez a ***COM2***-nek felel meg.) Ha a modemterminálban kiadjuk az `AT` parancsot, `OK`-t kell kapjunk válaszként. Ha megcsörgetjük a modemet, a terminál ezt csengetésenként egy `RING` karaktersorozattal jelzi. Ekkor `ATA` paranccsal tudjuk „felvenni” a kagylót. Kitárcsázni az `ATDT1234567` paranccsal tudunk, ahol `1234567` a tárcsázandó telefonszám. Ha mindez sikerült, akkor modemünk mindkét irányban működőképes.

A faxkiszolgáló beállítása

A `faxsetup` szkript végzi el a a futtatás előtti konfigurációt, ez különösen binárisból telepített verzió esetén fontos, ekkor ugyanis az elérési utakat és egyéb paramétereket is ellenőrizni kell. Ellenőrzi a függőségeket (***TIFF*** könyvtár, ***PostScript***/***Ghostscript***), és hogy a futási környezet megfelelően be vane állítva (például felhasználók, bootoláskor indítás, ***FaxMaster*** stb...). A futás „eredménye” a program ***spool*** könyvtárán belül (***/var/spool/hyalafax***) az ***etc/setup.cache*** fájl, ami a ***Hyalafax*** kommunikációs szkriptjei számára szükséges paramétereket, valamint az ***/etc/setup.modem,*** ami a modemmel való kommunikációhoz szükséges paramétereket tartalmazza. A `faxsetup` ezután meghívja a `faxaddmodem` scriptet.

A modem hozzáadása

A ***faxaddmodem*** szkriptnek adhatjuk meg a modem által használt hívószámot, körzetszámokat, az előhívószámokat, a csöngetések számát és a modemhangszóró hangerejét, a faxdokumentumok, naplófájlok, stb hozzáférési jogait, és a modemspecifikus paramétereket. Az általános tárcsázási szabályokat a ***spool*** könyvtáron belül az ***etc/config*** fájl tartalmazza (*2. Lista*). A modemre vonatkozó paramétereket az ***etc/config.ttyS1*** fájlban találjuk, feltéve, hogy a modem a ***ttyS1***-en keresztül csatlakozik. Ennek a számunkra lényegesebb részét a *3. Lista* mutatja. A fájl a közvetlenül is szerkeszthető, de egészségesebb a szkriptet használni. Ami még érdekes lehet a `GettyArgs:` kezdetű sor, és az `AdaptiveAnswer:` kezdetű szakasz. Amennyiben a modemünk támogatja az ***adaptive answer*** módot, lehetőség van arra, hogy a bejövő hívásból a `faxgetty` megállapítsa, hogy hang, adat, vagy faxhívásról van szó, és amennyiben nem fax, egy külső `getty` programnak adja át a vezérlést, mint például a `mgetty+sendfax`. Egy ilyen kombinációval lehetőség nyílik komplett telefonos kommunikációs megoldás létrehozására, ami magában foglalja a faxküldést és -fogadást, a hangpostát, és emellett lehetőség van ***PPP*** vagy ***SLIP*** protokollon keresztüli betárcsázós internetszolgáltatás nyújtására is.

1. táblázat ***A Hylafax fontosabb könyvtárai***

docq	A kimenő faxok tárolója
recvq	A bejövő faxok
sendq	A kimenő faxok leírófájljai, tartalmazza a címzett és az átvitel főbb paramétereit
log	Az egyes munkák naplófájljai
config	Modem definíciós fájlok
etc	A legtöbb konfigurációs fájl itt található

A kimenő és bejövő szolgáltatások beállítása

A kimenő szolgáltatásokat kezelését a `faxq` végzi, modemenként egy példány fut belőle. A már csatlakoztatott modem használatát engedélyezni kell a sorkezelő számára, ezt a

```
faxmodem /dev/ttyS1
```

paranccsal lehet, de a `/dev/` rész akár ki is hagyható. Lehetőség van arra is, hogy a modemet csak kimenő kapcsolatok számára engedélyezzük, ha faxot fogadni nem akarunk. Ekkor azonban vigyázni kell, ilyenkor ugyanis

2. lista
A /var/spool/hylafax/etc/config fájl tartalma

```
LogFacility:          daemon
CountryCode:          36
AreaCode:             36
LongDistancePrefix:   06
InternationalPrefix:  00
DialStringRules:      etc/
                      dialrules
ServerTracing:        1
```

3. lista Kivonat a etc/config.ttyS1 fájlból

```
CountryCode:            36
AreaCode:               36
FAXNumber:              "+36 36 789 155"
LongDistancePrefix:     06
InternationalPrefix:    00
DialStringRules:        etc/dialrules
ServerTracing:          1
SessionTracing:         11
RecvFileMode:           0600
LogFileMode:            0600
DeviceMode:             0600
RingsBeforeAnswer:      5
SpeakerVolume:          off
GettyArgs:              "-x 6 -r -s 38400"
LocalIdentifier:        "FaxModem"
TagLineFont:            etc/lutRS18.pcf
TagLineFormat:          "From %%l|%c|Page %%P of %%T"
MaxRecvPages:           25
AdaptiveAnswer:         yes             # enable adaptive answer
AnswerRotary:           "fax data"      # answer for fax, then
                                        # data
ModemAnswerCmd:         AT+FCLASS=1;A   # default is to answer
                                        # as fax
ModemAnswerDataCmd:     ATH+FCLASS=0;A  # hangup and answer as
                                        # data
Class1RecvIdentTimer:   10000           # timeout fax answer in
                                        # 10 secs
```

nem fut a `faxgetty`, és így a ***Hylafax*** nem képes a modem állapotának folyamatos figyelésére. Tehát, ha faxot akarunk küldeni miközben más alkalmazás zárolja a modemet, nem fog sikerülni.
A bejövő szolgáltatások nyújtását és a modemfigyelést a `faxgetty` valósítja meg, ezt a rendszer a */etc/inittab* fájlban beállított módon hívja meg, egy ehhez hasonló bejegyzés alapján:

```
#FAX
m0:2345:respawn:/usr/local/sbin/
➥faxgetty ttyS1
```

Változtatás után az

```
init q
```

paranccsal tudjuk újraindítani az inicializáló rendszert és érvényt szerezni a változásoknak.

A hozzáférések megadása

A kliensekkel való kapcsolattartást a `hfaxd` démon végzi, bootoláskor indítandó, például a */etc/rc.d/rc.local* szkriptből, általában ilyen formában:

```
/usr/local/sbin/hfaxd -i 4559
➥-o 4557 -s 444
```

Indítható a `inetd` segítségével is, ekkor a következő sort kell a */etc/inetd.conf*-ban szerepeltetni:

```
snpp stream tcp nowait fax
➥/usr/local/sbin/hfaxd hfaxd
➥-S -d
```

Azt, hogy a hálózat mely számítógépei férhetnek hozzá a szerverhez, a Hylafax spool könyvtárán belül az *etc/hosts* fájlban adhatjuk meg, ilyen formában:

```
root@magno:/var/spool/hylafax
➥# cat etc/hosts.hfaxd
localhost                # <-
gépnév
127.0.0.1                # <-
hozzá tartozó ip
magno
192.168.0.1
```

A *etc/hosts* fájl tulajdonosa a faxfelhasználó kell legyen, jogai pedig 0600, különben a `hfaxd` nem engedélyezi a hozzáférést a klienseknek.

A karbantartási feladatok beállítása

Ezeknek a feladatoknak a beállítása nem feltétlenül szükséges, de nagy forgalmú szervereken célszerű.
Két segédprogramot tartalmaz a csomag erre a célra, a `faxqclean` és a `faxcron` szkripteket.
A `faxqclean` a szerver által generált dokumentum- és feladatleíró fájlokat törli, a `faxcron` pedig üríti a naplófájlokat, törli a régi naplókat, és a bejövő, már feldolgozott faxokat. A ***cron*** időzítő démon segítségével automatikusan futtathatjuk a két karbantartó alkalmazást, például naponta az */etc/cron.daily* könyvtárban létrehozott szkript segítségével:

```
#!/bin/sh
/usr/local/sbin/faxqclean
/usr/local/sbin/faxcron | mail
➥FaxMaster
```

Csuhai Imre
(csuhi@csuhi.homelinux.net)
Közalkalmazottként és egyéni vállalkozóként közbeszerzéssel és informatikával foglalkozom, főként szerverek és hálózatok érdekelnek. Legszívesebben Slackware-t és UHU-Linuxot használok, szórakozásképpen pedig Quakekel és egyéb FPS-ekkel játszom.

Razor

Spammerek borotvaélen

Bár az előző cikkemben (Linuxvilág 2005. november) bemutatott postgrey nevű szürkelista minden bizonnyal drasztikusan lecsökkentette az olvasók postafiókjába kerülő spamet, mégsem nyújt tökéletes védelmet.

A spammerek egy része ugyanis az *RFC* szabványok szerint működő levelező alkalmazással küldi el leveleit, így a spam nem az elküldés után azonnal, hanem 5 vagy 10 perc múlva, de megérkezik a postaládánkba. Ha az olvasó is „zéró toleranciával" viszonyul a spamhez, hadd mutassak egy újabb, a szürkelistát jól kiegészítő megoldást.
Ebben az írásban egy olyan kollaboratív spamszűrőt mutatok be, amely egy közösség erejével, kollektív tudásával harcol a spammerek ellen. A két legismertebb ilyen alkalmazás a *DCC* és a *Razor*. Cikkemben ez utóbbit ismertetem. Azért esett a választásom a *razor*-ra, mert viszonylag egyszerűen használható, és ha egyszer már bekonfiguráltuk, nem igényel további felhasználói karbantartást.

Telepítés

Töltsük le a *razor-agents* legutolsó változatát – ez a cikk írásakor 2.77 – az alkalmazás honlapjáról (➲ *http://razor.sourceforge.net/*).
A *razor Perl* nyelven írt alkalmazás, amely különféle *Perl* modulokat használ. Ezeket szerencsére nem kell egyenként letöltenünk, a *razor* készítői a *razor-agents-sdk* csomagban mindet összeszedték, így elég csak azt letölteni, szintén a *razor* honlapjáról. Azonban a modern *Linux* disztribúciók az *sdk*-ban található modulok többségét eleve tartalmazzák, nekem *Slackware Linux* alatt elég volt a *Digest-SHA1* és *URI* modulokat telepíteni a *CPAN*-ról (➲ *ftp://ftp.cpan.org/*), a többit tartalmazza a kedvenc disztribúcióm.

Csomagoljuk ki a *razor*-t:

```
tar jxvf razor-agents-
➥2.77.tar.bz2; cd
➥razor-agents-2.77
```

Készítsük el a *Makefile*-t:

```
perl Makefile.PL
```

Fordítsuk le, majd telepítsük:

```
make; su -c 'make install'
```

Beállítás

Első lépésként létre kell hoznunk a *razor* konfigurációs állományait, adjuk ki az alábbi parancsot:

```
razor-admin -create
```

Ez a home könyvtárunkban létrehozza a *.razor* könyvtárat, amelyben alapértelmezett paramétereket helyez el. Ezután regisztráljunk magunknak egy fiókot, vagy ha már létező nevet adtunk meg – ebben az esetben hibaüzenetet kapunk róla – akkor válasszunk másikat:

```
razor-admin -register -user
➥bela3 -pass anagyonnehezjelszo
```

Ha minden rendben ment (ebben az esetben a `razor-admin` 0 visszatérési értéket adott vissza, amit a `$?` héjváltozó tartalmaz), akkor létrejött a

```
~/.razor/identity-bela3
```

fájl és az erre mutató szimbolikus link:

```
~/.razor/identity
```

Próbáljuk ki!

Nézzük meg, mit tud a *razor*. Ehhez vegyünk egy levelet (teljes fejléccel), majd futtassuk az alábbi parancsot,

amely egy számot ad vissza. Ha az 0, akkor a levél spam, ha 1, akkor nem spam.

```
cat level | razor-check; echo $?
```

Mi történik a háttérben? A `razor-check` elemzi a levelet, és különféle ellenőrző összegeket (úgynevezett ***fuzzy checksums***) képez belőle. Ezután – amint a következő kódrészletből látható – gépünk felveszi a kapcsolatot a ***66.151.150.22*** IP című géppel (ez igazából a ***shock.cloudmark.com*** vagyis az egyik ***razor*** szerver), amely egy bannerrel válaszol a mi kapcsolatunkra, amire a mi kliensünk a program nevével és verziójával válaszol, ezután pedig elküldi a már említett összegeket. Ha a levél spam, azaz már szerepel a szerver adatbázisában, akkor a válaszban a `p=1` sztring szerepel, ha nem, akkor a `p=0`. (*1. lista*) Szeretném megnyugtatni a leveleik titkát féltő olvasókat, hogy a `razor-check` a levél tartalmát nem, csak az ellenőrzőösszegeket küldi el a ***razor*** szerverekre.

1. Lista

```
#ngrep -X port 2703
interface: eth0 (10.1.1.0/255.255.255.0)
filter: ip and ( port 2703 )
####
T 66.151.150.22:2703 -> 10.1.1.2:56089 [AP]
  sn=C&srl=5084&a=l&a=cg&ep4=7542-10..
##
T 10.1.1.2:56089 -> 66.151.150.22:2703 [AP]
  cn=razor-agents&cv=2.77..
##
T 10.1.1.2:56089 -> 66.151.150.22:2703 [AP]
  -a=c&e=4&ep4=7542-
  ➥10&s=o3dTCeHTppRgGT9B4M5wsdOOUoIA..a=c&e=4&ep4=
  7542-10&s=C6s6JzaIBPpnZQfOn4lTZqbpnUYA
  ➥..a=c&e=8&s=ySNqMCzAfDOA...
  ..
##
T 66.151.150.22:2703 -> 10.1.1.2:56089 [AP]
  -p=0..p=1&cf=100..p=1&cf=100.....
#
T 10.1.1.2:56089 -> 66.151.150.22:2703 [AP]
  a=q..
####
```

Élesítsük a borotvát

A ***razor*** leginkább a felhasználó oldali használatra felel meg, legegyszerűbben ***procmail***-lel, ***maildrop***-pal együtt lehet használni. Nézzünk egy példát ez utóbbira. Készítsük el a ***maildrop*** konfigurációs állományát (alapesetben ~/***.mailfilter***) az alábbi módon:

```
'/usr/bin/razor-check'

if($RETURNCODE == 0)
{
        to "mail/junk"
}

to "Mailbox"
```

Ennek hatására a ***maildrop*** minden egyes beérkező levelet átad a `razor-check` programnak, és annak visszatérési értékétől függően az adott levél vagy a kukába (***junk folder***) kerül, vagy a többi levél közé. Itt szeretném megemlíteni, hogy a `razor-check` és `razor-report` programok nem csak önálló leveleket, hanem ***mailbox*** formátumú folder-ben tárolt leveleket is fel tudnak dolgozni.

A ***razor*** előnye, hogy a tévesen spamnek ítélt levelek száma elhanyagolható, azaz, amire a ***razor*** azt mondja, hogy az spam, az – szinte – bizonyosan az. De a ***razor*** sem tévedhetetlen, ha a levél nem szerepel a ***razor*** szerveren, akkor azt nem ismeri fel spamként, még akkor sem, ha a levél valóban spam. Ez fordítva is igaz, ha a mi levelünk ellenőrző összege ütközik egy a már szerveren lévő másik levél összegével (***collision***), akkor a ***razor*** a mi levelünket ebben az esetben akkor is spamként értékeli, ha az valójában nem is az.

Ha a `razor-check` egy levelet tévesen ham-ként azonosított (***false negative***), akkor azt a `razor-report` programmal el tudjuk küldeni a ***razor*** szerverekre. Ebben az esetben azonban a kliensprogram – a hálózaton átmenő forgalom elemzése alapján – elküldi a teljes levelet. Úgy tapasztaltam, hogy néhány órának el kell telnie, mire a (fel)jelentésünk az éles adatbázisba kerül.

Teszt

Ideje megnézni, hogy milyen hatékonysággal, illetve pontossággal dolgozik a `razor-check`, amit 3 ***mailbox*** formátumú fájlon teszteltem. A vizsgálatban egy angol ill. egy magyar nyelvű levelező lista, továbbá a junk folderem vett részt. A két listán elvileg csak ham, míg a junk folderemben csak spam szerepelt – jellemzően

1. táblázat *Egy mérés eredménye*

	Levelek száma	Mailbox mérete [MB]	Feldolgozás ideje [sec]	Pontosság [%]
junk folder	319	1,9	24	73,04
angol nyelvű lista	309	2,5	18	98,7
magyar nyelvű lista	1121	15,7	75	98,82

angol és valami számomra értelmezhetetlen keleti nyelven. A mérés eredménye az *1. táblázatban* látható.
Futási idő viszonylatában – az arányokat tekintve – nincs különbség a két csoport, illetve 3 fájl között, átlag 15 levelet tudott kiértékelni a `razor-check` másodpercenként. Erről még annyit szeretnék megjegyezni, hogy ***qemu***-ban futtatott ***Slackware Linux*** alatt végeztem az említett teszteket. A számunkra érdekes különbség a pontosságban van. Míg a tiszta spamet tartalmazó junk folder levelei közül 86-t nem azonosított spamként, és így 73%-os pontosságot ért el, addig a levelező listákon meglepetésemre 2-4 levelet (körülbelül 1,3%) spamnek tekintett, tévesen. A magyar nyelvű listán elmarasztalt két levélben az a közös, hogy a levél törzse igen rövid (1 sor, ill. 1 betű), elképzelhető, hogy az erre kapott ellenőrző összegek hasonlítanak valaki valamelyik spam levelének lenyomatára. Hangsúlyozni szeretném, hogy ezek a számok az én leveleimre vonatkoznak, az Olvasó lehet, hogy ennél jobb vagy esetleg gyengébb eredményt ér el a ***razor*** használatával.

További gondolatok

Noha a ***razor*** leginkább kliensoldalon használható, de könnyen megoldható, hogy egy szerver oldali spamszűrő is használhassa. Ebben az esetben a `razor-admin` készítette konfigurációs fájlokat át kell helyezni a szűrőt futtató felhasználó home könyvtárába, majd azt kell elérni, hogy a spamszűrő alkalmazás például egy `exec()` rendszerhívással lefuttassa a `razor-check` programot, és annak visszatérési értékétől függően vagy automatikusan a spam folderbe mozgatja a levelet, vagy megjelöli valahogyan, ami alapján a kliensoldalon dönteni lehet róla, esetleg pozitív esetben extra büntetőpontot kap a levél.
Kezdetben az ezen az elven működő szűrők egyszerűen képezték a levél például ***MD5*** ellenőrző összegét. Azonban a spammerek elkezdték „testre szabni" a leveleket, azaz tettek bele például egy web bug-ot, ami egyedi azonosítót tartalmazott, így pusztán ettől már más lett ugyanannak a levélnek az ***MD5*** összege. Vagy tettek a levél végére néhány hottentotta „kifejezést", amellyel szintén az előbbi hatást érték el. Ezért ma már a levél ellenőrző összegét felhasználó szűrők akár több összeget is képeznek (mint például a most tárgyalt ***razor***), vagy igyekeznek kihagyni a számolásból a fenti módon perszonalizált részeket.
Szintén a szűrőket segíti néhány kevésbé ismert eljárás. A Rabin algoritmus hasonlóan működik, mint a jól ismert ***MD4***, ***MD5*** vagy ***SHA1***, ez is ujjlenyomatot képez a bementi adatból. Ebben eddig semmi új, de amíg például az ***MD5*** úgy van kitalálva, hogy az akár 1 biten is eltérő adatokhoz egymástól gyökeresen eltérő összegek tartozzanak, addig a Rabin algoritmus ezzel ellentétben két egymástól kevéssé eltérő adathoz hasonló összeget képez. Ha a spammerek minden levelet testre szabnak – manapság ezt automatikusan tudják a spamware alkalmazások – azaz a levelek gyakorlatilag viszonylag kis részletben térnek el egymástól, semmi gond, ezután is képezzük a szokásos ellenőrző összeget, és a szerver azt nézi meg, mekkora a „távolság" az ehhez leginkább hasonlító ujjlenyomattól. Ha ez egy határ alatt van, azaz a két szignatúra nagyon hasonlít egymáshoz, akkor nyilvánvalóan ugyanannak a spamnek két változatáról van szó. Ez az eljárás még másra is használható. Bár az írás témája a spamszűrés, de az elv a vírusok, elektronikus levélben terjedő férgek ellen is hatásos. Tegyük fel, hogy egy feketekalapos illető elkészíti a XXI. sz. szuperférgét, amely majd e-mailben fog terjedni. A vezető vírusírtó cégek nyilván csak egy bizonyos reakcióidő elteltével – ez a Commtouch szerint ➲ ***http://www.commtouch.com/site/OEM/zero_hour.asp*** akár 20-30 óra is lehet – tudnak hozzá vírusszignatúrát – vírusadatbázis frissítést – kiadni. Azonban az előbb tárgyalt módszer segítségével már akkor is védelmet élvezhetünk a „szuperféreg" ellen, amikor még nincs is rá „védőoltás", már a féreg elszabadulását követő pár perctől kezdve. De ehhez az is szükséges, hogy minél többen használják ezt a technológiát. A ***Commtouch*** 35 millió postafiókot említ a honlapján, amelyek szolgáltatói az ő technológiájukat licencelték.

Záró gondolatok

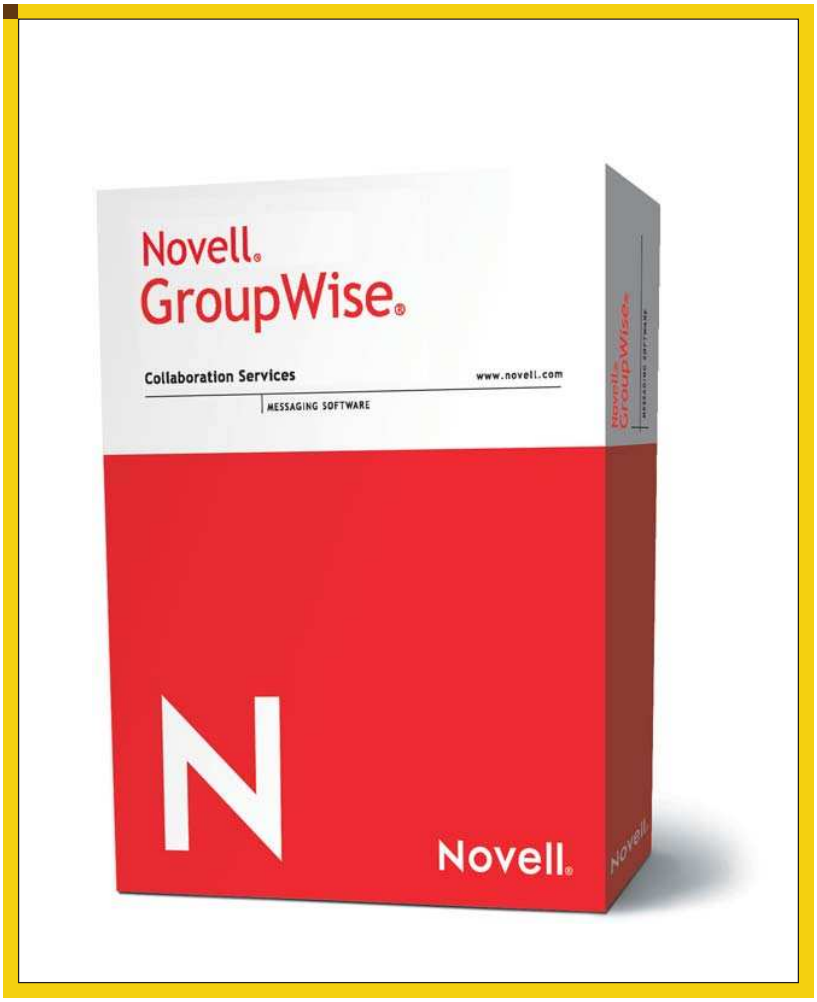
A ***razor*** hatékonyan használja ki az internet felhasználóinak kollektív erejét. Az Olvasónak is javaslom, hogy éljen ezzel a lehetőséggel, ha a levelezőszerverén használhatja a `procmail` vagy a `maildrop` programot. A ***razor*** által megjelölt leveleket bátran tehetjük a junk folder-be a `maildrop` segítségével. Érdemes továbbá a ***razor*** által fel nem ismert spamet egy külön spool fájlba gyűjteni, és azt rendszeres időközönként a ***razor-report*** programmal betölteni a ***razor*** adatbázisába.

Sütő János
(jsuto@freemail.hu)
1997 óta használ Slackware Linux-ot. Szabadidejében a postfix clapf nevű vírus- és spamszűrőjét polírozza.

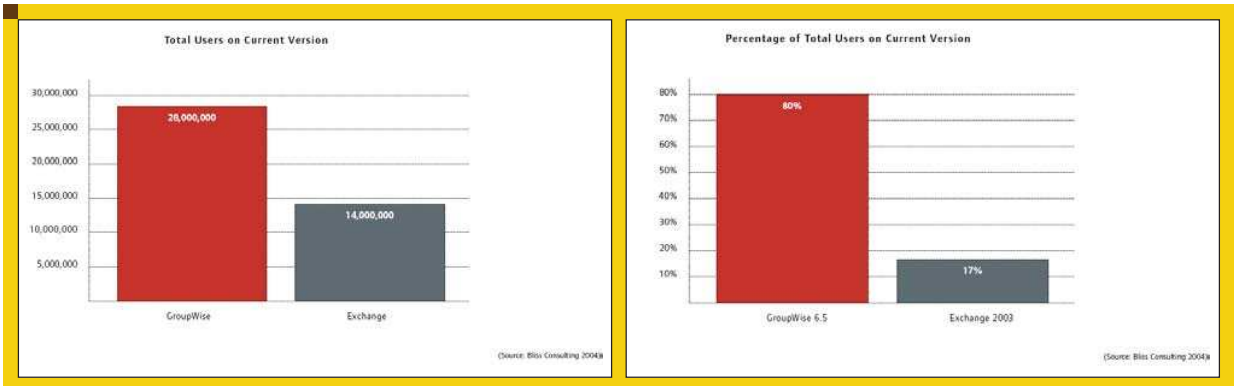
A Novell GroupWise az egyik legelterjedtebb üzenetkezelő rendszer

Az utóbbi néhány évben a Novell GroupWise folyamatosan támadások kereszttüzében és erőteljes versenyben állt. A szkeptikus és lojális felhasználók egyaránt figyelemmel kísérik a rendszer sorsát, azon tűnődve, vajon milyen szerepet fog játszani a Novell Linux történetében és az üzenetkezelési piacon.

15 éven át a *Novell GroupWise* arra az elképzelésre épült, hogy a biztonság és az alacsony költség teszi egyedivé ezt a rendszert.
A *GroupWise* három fő jellegzetessége mindig is a biztonság, a stabilitás és az alacsony költség volt. Ezeknek a jellemzőknek köszönhetően a *GroupWise* népszerűsége egyenletesen nőtt a jogi cégek, a szövetségi és állami kormányok, valamint az egészségügyi intézmények és pénzintézetek körében. A legnagyobb piacra azonban az oktatási területen talált, a felsőoktatásban és a középiskolák területén egyaránt.
A felhasználók úgy tapasztalták, hogy a *GroupWise* jelentősen lefaragta az üzemeltetési és karbantartási költségeiket anélkül, hogy a hőn áhított biztonságról le kellett volna mondaniuk.
A *Microsoft* megszüntette az *NT* és az *Exchange 5.5* támogatását, így a felhasználók tízmillióit kényszerítette választásra.
Gyakran egy rendkívül összetett, egymástól távoli részekből álló, távoli felhasználókat kiszolgáló többplatformos *GroupWise* rendszert is egyetlen rendszergazda tartott karban. Több 7-10000 főt foglalkoztató vállalat egész e-mail rendszerét egyetlen személy üzemeltette.
Sok szervezet, főleg a kis informatikai költségvetéssel rendelkezők, kezdtek el átállni a Novell *GroupWise* használatára. Légitársaságok, bankok, rendőrségek, tűzoltóságok és egyéb, biztonságra és takarékosságra törekvő szervezetek kezdték a *GroupWise* levelezőrendszert telepíteni és használni.



A népszerűség növekedése során a *Novell* még egy fontos lépést tett, amellyel megszilárdította első helyét az *Exchange* alternatívák között.



1. ábra A Novell GroupWise felhasználóinak (28 millió) körülbelül a 80 százaléka használja a termék legújabb változatát, míg az Exchange esetében (17 millió felhasználó) ez az arány csak 17 százalék.

A ***Novellnek*** egy ***Outlookhoz*** hasonló kliensre, valamint egy operációs rendszerre volt szüksége, ezért felvásárolt két vállalatot, amelyek segítségével ezután egyidejűleg két különböző piacra is összpontosíthatott. A ***Ximian*** felvásárlása révén jutott hozzá az ***Evolution*** klienshez, amelyre a vállalatok és a nagyközönség miatt volt szüksége. A ***SUSE LINUX*** akvizíciójával pedig a vállalati környezetben való hatékony működéshez szükséges operációs rendszert szerezte meg. Minden összeállt ahhoz, hogy a ***Novell GroupWise*** tovább erősítse szerepét, mint a piac egyik domináns szereplője.

15 éven át a ***Novell GroupWise*** arra az elképzelésre épült, hogy a biztonság és az alacsony költség az, ami egyedivé teszi ezt a rendszert. Jelenleg számos vállalat keres alternatívát a ***Microsoft Exchange/Active Directory/Windows*** rendszerről való áttérésre. Csakúgy, mint 1984-ben az ***IBM*** esetében: a vállalatokat eltántorítja az olyan egyetlen monolitikus infrastruktúra létrehozásának bonyolultsága és költsége, amely nem kínál rugalmasságot és minden alkalmazása költségvonzattal jár. A ***GroupWise*** figyelemre méltó növekedést és lendületet mutatott az elmúlt években, mint az üzenetkezelési piac erős versenyzője és a ***Microsoft Exchange/Active Directory/Windows*** rendszerek alternatívája.

Tekintsük át a ***Novell GroupWise*** növekedését alátámasztó adatokat. A ***Microsoft Exchange*** körülbelül 103 millió felhasználóval rendelkezik világszerte, míg a ***Novell GroupWise*** 35 millióval. A 103 millió ***Exchange*** felhasználó 40 százaléka azonban az 1997-ben kiadott ***Exchange 5.5*** rendszert használja. Az ***Exchange 5.5*** ráadásul ***NT*** rendszeren fut, amely megint csak kifutóban van. Így hát a 103 millió ***Exchange*** felhasználó közül 40 millió olyan terméket használ, amely közel 8 éves, és 2005-ben megszűnik. A 103 millió felhasználó további 46 százaléka jelenleg ***Exchange 2000***-t használ, amely öt éve jelent meg. Marad tehát a jelenlegi ***Exchange*** tábor körülbelül 17 százaléka, avagy 17 millió felhasználó, amely a kétéves ***Exchange 2003*** terméket használja.

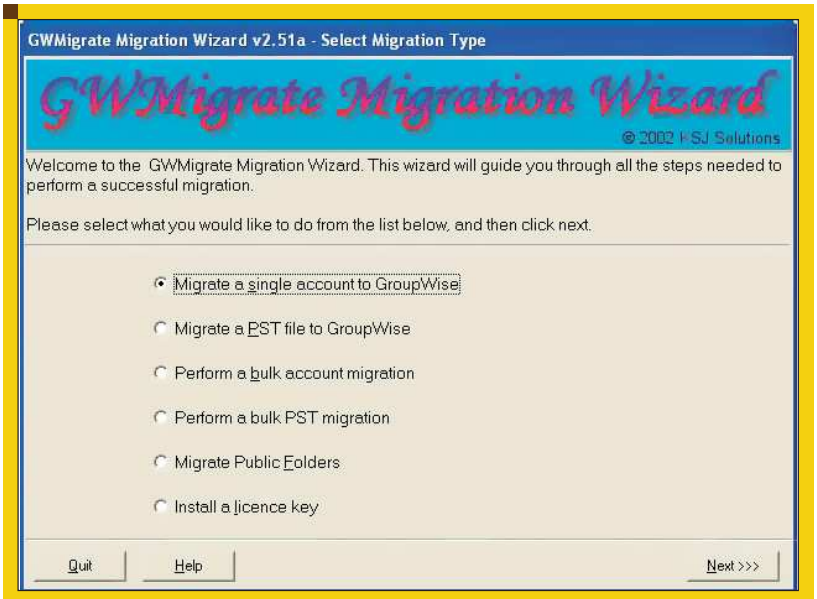
Ezzel ellentétben a 35 millió ***Novell GroupWise*** felhasználó több mint 80 százaléka már átállt a ***GroupWise 6.5***-re, amely a ***Microsoft Exchange 2003***-mal körülbelül egyidőben jelent meg. Ez hozzávetőlegesen 28 millió ***Novell GroupWise 6.5*** felhasználót jelent a 17 millió ***Exchange 2003*** felhasználóval szemben, ami az új verziók esetén jelentős többséget jelent a ***Novell*** javára (Lásd az *1. ábrát*) .

A GroupWise és a jövő

Az utóbbi 20 évben az e-mail iparág megismételte önmagát. A piac egyik domináns szereplője mindenkit egyetlen üzenetküldési megoldásra kényszerít, a piac megváltozik, a technológia továbbhalad, új igények keletkeznek. A domináns szereplő nem képes alkalmazkodni a változásokhoz, így egy másik megoldás kerül előtérbe. Ez a séma ment végbe 20 évvel ezelőtt is, amikor az ***IBM PROFS*** elvesztette fölényét. A ***PROFS***-ot nem egy másik mainframe/monolitikus alkalmazás taszította le a trónról, hanem a ***cc:Mail***, egy olcsó, rugalmas alkalmazás, amely sokak számára tette lehetővé az e-mail telepítését helyi hálózaton keresztül. A ***cc:Mailt*** nem egy másik ***LAN*** megoldás szorította ki, hanem az ***Outlook***, a mindenki által a kis mérete, olcsósága és rugalmassága miatt átvett kliens, amely révén a felhasználók szinte bárhol küldhettek e-mailt. Nyilvánvaló, hogy a jövőben változásra lesz szükség. A kérdés csak az: ki vagy mi lesz képes megoldásokat kínálni a holnap problémáira. A következő üzenetkezelési megoldás, amely átveheti a piac vezetését, bizonyos jellemzőkkel kell, hogy rendelkezzen:

- Rugalmas háttérrendszer, amely egyben biztonságos és olcsón telepíthető.
- Rugalmas kliens, amellyel többféle platform nyújthatja ugyanazt az élményt.
- Telepíthető címtár, amely lehetővé teszi az egész rendszer felügyeletét különféle platformokon és alkalmazáson.
- Olyan operációs rendszer, amelyre rugalmasan lehet szolgáltatásokat építeni.

A ***Novellre*** a stratégiai felvásárlásai és belső fejlesztései révén komoly szerep vár a jövőben. A ***Novell GroupWise*** eddigi két fő jellegzetessége a biztonság, valamint az üzemeltetés és a karbantartás alacsony költsége volt. A ***Ximian Evolution*** kliens és a ***SUSE LINUX*** most először



2. ábra A GWMigrate egyszerre több migrációs munkaállomást is használhat, így az Exchange kiszolgálón tárol fiókok, PST fájlok és nyilvános mappák tetszőleges kombinációját megadhatjuk neki forrásként. Természetesen akár egyetlen Exchange fiókot is átvihetünk vele GroupWise alá.

teszi lehetővé a valódi függetlenséget a felhasználók számára. Az ***Evolution*** e-mail kliens használatával a felhasználó teljesen függetlennek érezheti magát az informatikai felügyelet háttérrendszerétől. A ***SUSE LINUX*** használatának lehetőségével a felhasználók immár nincsenek egyetlen platformhoz kötve. A ***GroupWise 6.5*** rendszert a ***Novell*** portolta ***Linuxra*** , így már három platformon kínál biztonságot és rugalmas háttérrendszert több konfiguráció támogatására egy szervezeten belül. Az ***eDirectory*** évek óta piacvezető a címtár és metacímtár rendszerek között, az utóbbi időkben a ***SUSE LINUX*** pedig az alacsony költségű és rugalmas asztali és szerver operációs rendszereket kereső szervezetek igényeit elégíti ki.

Asztal nélküli dolgozók

Az üzenetküldési infrastruktúra sok különböző részből áll. Egy új, gyorsan változó igény az asztal nélküli dolgozó szerepe. Az asztal nélküli dolgozó egy olyan szakember, akinek a munkaköre nem igényli, vagy nem teszi lehetővé egy asztali számítógép mindennapos használatát. Az ilyen munkakörre jó példa lehet egy légitársaság pilótája, utaskísérői, tűzoltók, rendőrtisztek és a városi szolgáltatások munkatársai. Az e-mail hozzáférést igénylő szakemberek listája csak növekszik. Az ***Egyesült Államok*** munkaerejének 60 százaléka az asztal nélküli dolgozó kategóriába esik. Ezek a dolgozók hagyományosan képtelenek voltak a hatékony kommunikációra. A ***GroupWise*** fejlesztői felismerték az asztal nélküli dolgozók előtti kommunikációs korlátokat, így a rendszer különböző eszközök sorát kínálja számukra, hogy munka közben tarthassák a kapcsolatot a szervezet többi részével. A ***Novell WebAccess*** hagyományosan az elsődleges eszköz a mobil dolgozók számára, akik így elektronikus leveleiket a munkaasztaltól távol is elérhetik. Utazó légitársasági alkalmazottak például elolvashatják elektronikus leveleiket nyilvános fülkékben vagy számítógépeken, így folyamatosan kapcsolatban maradhatnak. A ***WebAccess*** azonban nem rendelkezik közvetlen, azonnali kommunikációs lehetőségekkel. A ***GroupWise*** a vezeték nélküli eszközök szállítóival együttműködve lehetővé teszi a szervezetek számára, hogy ***PDA***-kat, például ***Blackberry*** eszközöket vezessenek be a szervezetüknél. Így egy tűzoltó állandóan magával hordhat egy ***Blackberry*** készüléket, amelyhez ráadásul egy névre szóló e-mail cím is jár. Az alkalmazottak így képesek az eszközzel e-mailt küldeni és fogadni, és nincs szükségük személyi számítógépre a levelezéshez.
Ez a kezdeményezés világszerte sok szervezet számára tette lehetővé egy költséghatékony, működőképes megoldás bevezetését.
Ez csak egy példa a sok közül, amely bizonyítja, hogy a ***Novell GroupWise*** képes a változó környezethez alkalmazkodva könnyen telepíthető, a helyzetnek megfelelő megoldást biztosítani.

GroupWise, az első számú alternatíva

A ***Microsoft Exchange*** felhasználóbázisának közel fele használja az ***Exchange 5.5*** verziót. Ők azok, akik ellenálltak a ***Microsoft*** által kínált frissítési módoknak. A felhasználó bázis fele csak 2004-ben tért át az ***Exchange 5.5***-ről. A frissítések legnagyobb akadálya a költség volt. Az ***Exchange 5.5*** viszonylag könnyen karbantartható és alacsonyabb költségű. Ez a két érték felhasználók tízmillióit tartotta vissza attól, hogy egy drágább, kevésbé stabil és kevésbé biztonságos platformra térjen át.
A ***Novell GroupWise*** további sikerrel és növekedéssel kecsegtet az e-mail piacon, mivel ez az ***Exchange*** első számú alternatívája.
A ***Microsoft*** azonban 2005 januárjában, illetve decemberében megszünteti az ***NT*** és az ***Exchange 5.5*** még meglévő incidens alapú támogatását (a hagyományos javítások már 2004 év elejével megszűntek), így a felhasználók tízmillióit kényszeríti választásra. Sokuk nem fogja a ***Microsoft*** által kínált frissítési módokat követni, hanem inkább a ***GroupWise***-ra tér át.
A szervezetek úgy tekintenek a ***Novell GroupWise***-ra, mint a biztonságukat garantáló szükséges eszközre. Ráadásul az alacsony költséget további előnyként értékelik. Kétségtelen, hogy mivel az ***Exchange*** felhasználók máshol is keresnek megoldást, a ***Novell GroupWise*** további sikereket és növekedést fog elérni az e-mail piacon, az ***Exchange*** első számú alternatívájaként.

Egyről a kettőre: Exchange-ről GroupWise-ra – esettanulmány

2004 nyarán az ***Egyesült Államok Észak-Dakota*** államában működő ***MeritCare Health Systems*** készen

állt arra, hogy 7000 ***Exchange 5.5*** felhasználóját ***GroupWise***-ra állítsa át. Az átállás alapvető eszköze a ***GWMigrate Exchange to GroupWise*** eszköz volt. A ***GWMigrate*** segítségével könnyedén lehet felhasználók tömegeit ***Exchange***-ről ***GroupWise***-ra átállítani, valamint a segítségével megoldható a ***PST*** fájlok tömeges migrációja is.
A ***GWMigrate*** konfigurálható úgy, hogy több átállási munkaállomás használatával végezze el az ***Exchange*** kiszolgáló postafiókok, ***PST*** fájlok vagy nyilvános mappák átállítását. Akár egyetlen ***Exchange*** postafiók is átállítható ***GroupWise***-ra (lásd a ***2. ábrát***). (A ***GWMigrate*** eszközről további információt a ***www.caledonia.net*** lapon talál.)
Egy olyan nagy rendszer, mint a ***MeritCare*** nagyon érdekes statisztikát és útmutatást szolgáltat bármilyen léptékű ***Exchange-GroupWise*** áttéréshez. Ez az esettanulmány néhány előkészületi lépést vizsgál meg, amelyek felgyorsíthatják az áttérést, és némi információval szolgál a ***MeritCare Health Systems*** tényleges áttéréséről.

Az Exchange rendszer előkészítése

A ***MeritCare*** rendszergazdáinak egyik legjobb lépése az volt, hogy előkészítették az ***Exchange*** rendszerüket az áttérés gyorsabb és hatékonyabb elvégzésére. A rendszergazdák még az áttérés előtt megtanították az e-mail felhasználóknak az alapvető postafiók karbantartási feladatokat. A következő lépések a ***MeritCare*** szerint kifejezetten hasznosnak bizonyultak az áttérés felgyorsításához.

- ***A rendszer kitisztítása*** – A felhasználók gyakran mindent megtartanak a postafiókjukban, egyszerűen azért, mert megtehetik. Az áttérés remek alkalom arra, hogy a felhasználók kitisztítsák postafiókjukat és a rendszer többi részét is. Ajánlott csak bizonyos mennyiségű vagy egy adott időtartamra vonatkozó adatot átvinni. Ilyen irányelvek megkövetelésével jelentős mennyiségű időt lehet megtakarítani az áttérés során.

- ***Régi felhasználók eltávolítása*** – Ha vannak a rendszerben a szervezethez már nem tartozó felhasználók, akkor érdemes őket a migráció előtt törölni a rendszerből.

- ***Törölt elemek megtartásának kikapcsolása*** – Ne vigyünk át szemetet.

- ***Töredezettségmentesítés***

A ***GroupWise*** áttérés révén a szervezetek lecsökkenthetik költségeiket és megnövelhetik termelékenységüket a biztonság védőhálójában.

A GroupWise előkészítése

Az ***Exchange*** előkészítésén felül a ***MeritCare*** előkészítette a ***GroupWise*** rendszert is az áttérés fogadására.
A ***GroupWise*** importáló-exportáló eszköze ezt az előkészítést könnyen és gyorsan elvégezte. A ***GroupWise*** rendszer előkészítéséhez a ***MeritCare*** módszere alapján a következő lépéseket kell elvégezni:

- Tartományok meghatározása
- Postafiókok meghatározása
- Felhasználók létrehozása

Mit mozgassunk át?

Egészségügyi szolgáltatóként szó szerint életbevágó volt, hogy a ***MeritCare*** gyors áttérést hajtson végre, minimális üzemszünettel. Az ***Exchange*** rendszer számos eleme elhagyható, nem szükséges a ***GroupWise***-ba való átvitelük
Ha kevés a tárhely vagy az idő, vagy csak egyszerűen egy tisztább ***GroupWise*** rendszert szeretne, a ***MeritCare*** szerint a következőket érdemes átvizsgálni:

- csatolmányok
- elküldött elemek
- törölt elemek
- ismétlődő találkozók, amelyek pár hónapnál előrébb nyúlnak a jövőben

Ha a szervezet megteheti, hogy a fentiek bármelyikét elhagyja, akkor nem csak az áttérés idejét csökkentheti le, hanem az átvitt összevisszaság mértékét is.

Az áttérés

Tekintse át az alábbi statisztikákat az áttérés megtervezése előtt.
A ***MeritCare*** több számítógépen egy időben futtatta a felhasználók különböző részhalmazainak áttérését, így növelve a folyamat teljesítményét.

Felhasznált eszközök

Minden egyes ***GWMigrate*** áttérést futtató PC 3,0 GHz-es processzorral és 1 GB RAM-mal rendelkezett. Mindegyik munkaállomás 100 MBit sebességű hálózati kártyával volt ellátva. Ha a szervezetben rendelkezésre áll GBit sebességű hálózat a munkaállomások számára, a teljesítmény tovább növelhető. Bár a gyorsabb processzorok gyakran hasznosak, jelen esetben a memória mérete tűnik az áttérés kulcselemének.
Az ***Exchange*** szerverek 2,8 GHz-es számítógépek voltak. Az áttérés során az ***Exchange*** szervereket minden időpillanatban 20-36 ***GWMigrate*** folyamat terhelte. Bár ez a tevékenység lelassította az ***Exchange*** szervereket, lépést tudtak tartani a migrációs folyamattal. Az áttérési munkaállomások számának meghatározásához számos tényezőt kell alapul venni, például a rendelkezésre álló eszközöket, és hogy az ***Exchange*** szerverek tudják-e kezelni a terhelést.
Az átállás közben a ***MeritCare*** majdnem kizárólagos hozzáféréssel rendelkezett a hálózathoz (munkaidőn kívül végezték, kevés aktív felhasználóval). A következő statisztika jó képet adhat arról, hogy hogyan is kell egy áttérésnek lezajlania.
A legutolsó 90 nap adatának átvitele az ***Elküldött elemek*** mappa tartalma nélkül:

- 1,2 GB/óra
- 102 000 elem/óra

Az utolsó 90 nap ***Elküldött elemekkel*** együtt:

- 759 MB/óra
- 62 000 elem/óra

Utóbbi 1 év ***Elküldött elemek*** mappa nélkül:

- 753 MB/óra
- 61 000 elem/óra

Utóbbi 10 év ***Elküldött elemek*** nélkül:

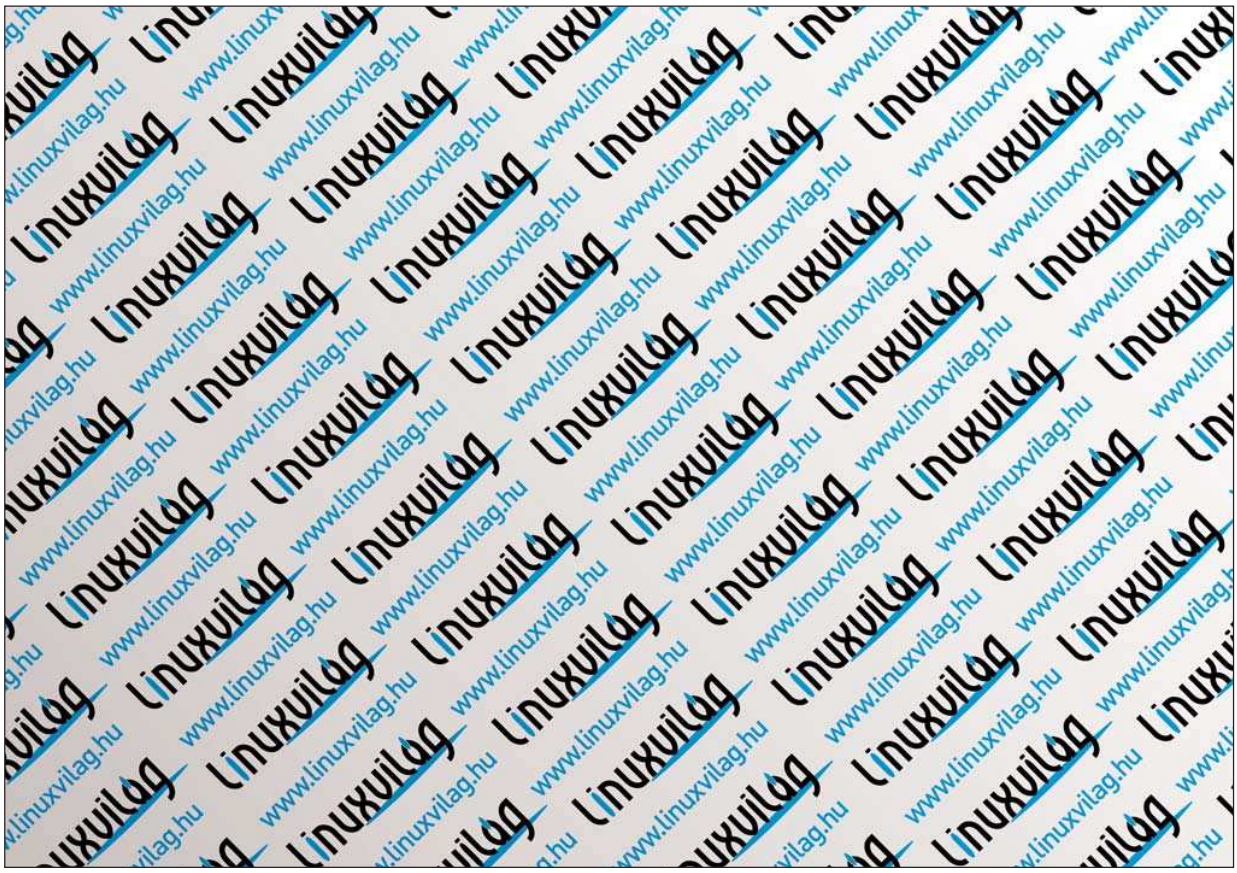
- 715 MB/óra
- 58 000 elem/óra

Ahogy a fentiekből is kitűnik, minél nagyobb az átvinni kívánt adathalmaz, annál lassabb a folyamat. Ezen felül az ***Elküldött elemek*** mappa nagyobb feldolgozási teljesítményt és ugyanazt az áteresztőképességet igényli, mint a beérkező üzenetek. Ha a szervezet a lehető leggyorsabb átállást igényli, ezeket a tényezőket megéri számításba venni. Ha mindent át akarunk vinni, akkor hatalmas teljesítményre van szükség.
A ***MeritCare*** az egész áttérést egy hétvége alatt végre tudta hajtani úgy, hogy mire az alkalmazottak hétfő reggel dolgozni jöttek, az új rendszer már üzemelt.

Összegzés

A ***GroupWise*** felhasználói tábora folyamatosan bővül. Néhány szervezet a vezetők nyomása miatt kénytelen számításba venni a ***GroupWise***-ról történő átállást, azonban a költségek, a bonyolultság és a biztonsági kockázatok fényében általában újragondolják a döntést. Ha a ***Microsoft Exchange*** felhasználók számát elemzés nélkül figyeljük, akkor látszólag ez a rendszer uralja a piacot. Előfordul, hogy a felsővezetés ennek tükrében úgy dönt, követnie kell a rosszul értelmezett piaci mozgást, és elhatározza a ***Microsoft*** rendszerre való átállást. De amikor szembesülnek az ezzel járó pénzügyi és biztonsági kockázattal, valamint az alkalmazottak termelékenységnövekedésének hiányával, a legtöbben újragondolják a lépést.
Nemrég egy több mint 8500 fős, 260 postafiókkal és 21 MTA-val (levéltovábbító alkalmazás) rendelkező szervezetnek a felsővezetés miatt kellett fontolóra vennie az ***Exchange***-re való áttérést. A jelenlegi ***GroupWise*** rendszerük költsége kevesebb, mint 200.000 USD. Az ***Exchange*** rendszerre való áttérés becsült költsége egymillió dolláros többletet jelentene – ráadásul a termelékenység növekedése nélkül. Az ***Exchange***-ről a ***GroupWise***-ra való áttérés gyakran alacsonyabb költséggel hajtható végre, mint egy ***Exchange***-ről ***Exchange***-re való áttérés. Mi több, a ***GroupWise***-ra történő áttérés révén a szervezetek csökkenthetik költségeiket és megnövelhetik termelékenységüket a biztonság védőhálójában.
A ***MeritCare*** az áttérést sikeresnek találta, és jelenleg is élvezi az új rendszer alacsony költségű üzemeltetését és karbantartását, valamint a ***GroupWise*** nyújtotta biztonságot.
Minden szervezetnek szüksége van alkalmazásokra az üzletmenetéhez. De a sok között egyetlen olyan alkalmazás van, amelyhez minden egyes vállalat igényel hozzáférést, és ez nem egy szövegszerkesztő, nem egy táblázatkezelő, hanem az e-mail. A számviteltől a műszaki tervezésen át a gyártásig, a vezetőktől a kiszolgáló személyzetig minden egyénnek szüksége van a többiekkel való kapcsolattartásra a szervezet megfelelő működéséhez. Ezért vált az üzenetküldés ennyire kulcsfontosságúvá minden szervezet számára, és ezért játszik a ***GroupWise*** központi szerepet a jövőben.

Richard Bliss



3D ábrázolás PoVRay (4. rész)

A PoVRay világában létrehozott testek és objektumok általában az origó környékén jöttek létre, illetve egy rájuk jellemző irányban és méretben jönnek létre. Ezen a problémán segít az affin transzformációnak nevezett matematikai eljárás, illetve ennek a barátságosabb megvalósítása.

A transzformációk három komponensből tevődnek össze: elmozgatás, átméretezés, elforgatás. Ez a három komponens egy közös affin mátrix alapján végezhető el *3D* testeken egy lépésben, amely egy összetett matematikai eljárás, s ezt is megoldották helyettünk a *PoVRay* programozói.

Elmozgatás

A legegyszerűbb transzformáció az elmozgatás, amely során a vektorban meghatározott irányba mozdítjuk el a testet, mégpedig a megadott mértékkel. Ez egyszerűen egy `translate` kulcsszót kell megadnunk a meghatározott vektorral. Az *1. ábrán* látott két gömb teljesen azonos méretű és azonos pozíciójú, egyetlen különbség közöttük, hogy a vörös színű testet egy egységgel eltoltuk az `<1,1,1>` vektor mentén.

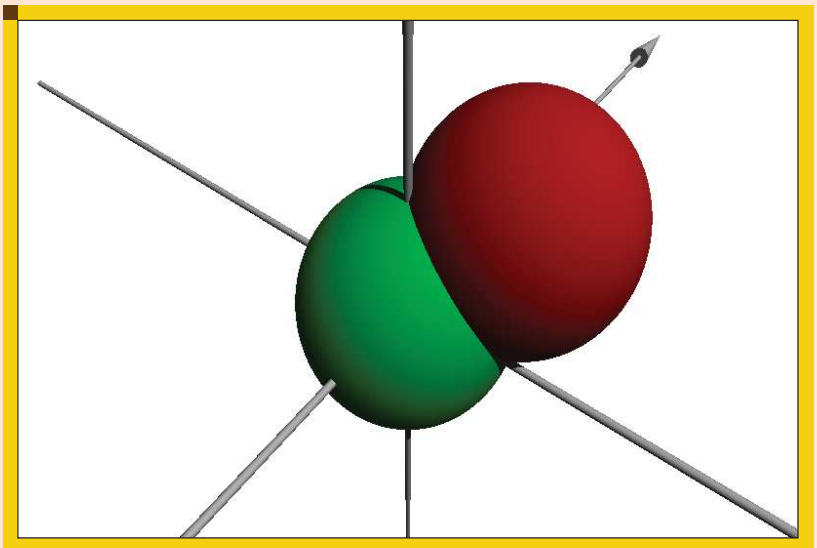
```
sphere{
 <0,0,0>,1.5
 texture{
  pigment{
   color Red}}
 translate <1,1,1>}

sphere{
 <0,0,0>,1.5
 texture{
  pigment{
   color Green}}}
```

Bár a példában a vörös gömb új pozíciója egyezik a megadott eltolással, az eltolás nem abszolút pozíciót jelent, hanem relatív pozíciót. Ennek megfelelően az alább meghatározott gömb pozíciója azonos lesz a fent meghatározott vörös gömb pozíciójával:

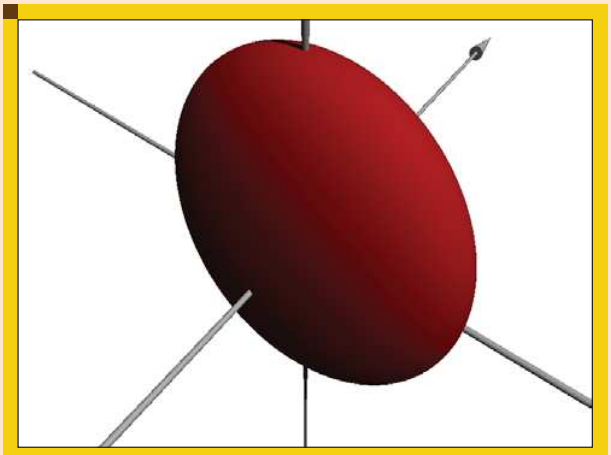
```
sphere{
 <2,2,2>,1.5
 texture{
  pigment{
   color Red}}
 translate <-1,-1,-1>}
```

Ha csak egy irányba kell elmozgatni egy testet, akkor tudjuk használni az előre definiált vektorokat is, tehát az alábbi részlet alapján a gömb új pozíciója a `<5,2,2>`:

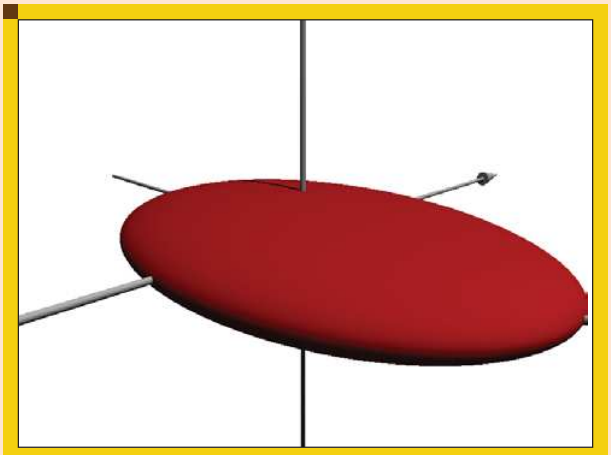
```
sphere{
 <2,2,2>,1.5
 texture{
  pigment{
   color Red}}
 translate 3*x}
```



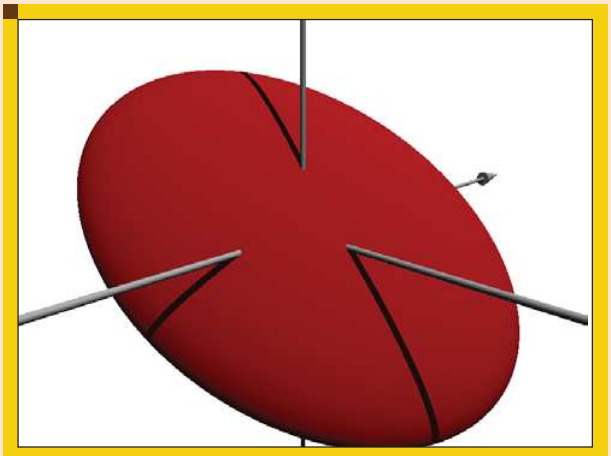
1. ábra Két gömb eltolással



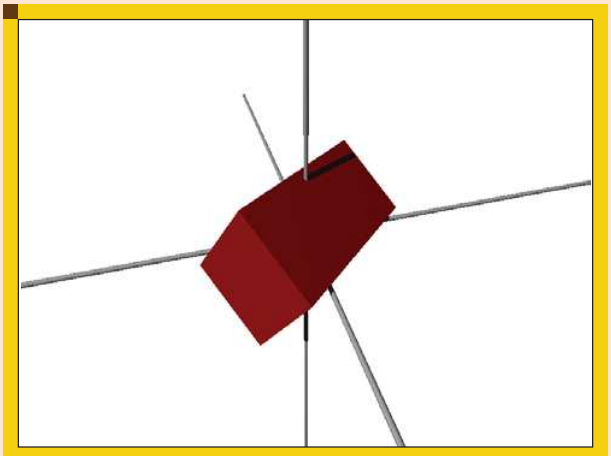
2. ábra Nyújtott gömb = ellipszoid



3. ábra Három tengely mentén átméretezett gömb



4. ábra Egy test elforgatása 30 fokkal



5. ábra Elforgatás X-Y-Z sorrenddel

Átméretezés

Az előző rész megismerése során biztos felmerült a kérdés, hogy miért nincs ovális keresztmetszetű test, más néven ellipszoid, amely egy térben megforgatott ellipszis alapján jön létre. Ennek az az egyszerű oka, hogy az ellipszoid a gömb megnyújtása az egyik tengely mentén, példánkban ez az X tengely (*2. ábra*):

```
sphere{
 <0,0,0>,1.5
 texture{
  pigment{
   color Red}}
 scale <2,1,1>}
```

Ha a megadott vektorban egynél nagyobb számot adunk meg, akkor megnyúlik a test, ha egynél kisebb számot, akkor összenyomódik. A megadott vektor abszolút értéke számít, nyugodtan megadhatunk akár negatív számot is.

```
sphere{
 <0,0,0>,1.5
 texture{
  pigment{
   color Red}}
 scale <3,0.5,2>}
```

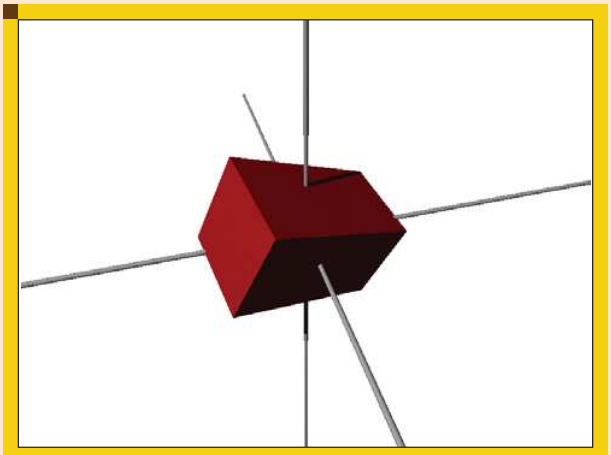
A *3. ábrán* egy gömbből kiindulva egy lapos, kissé ovális korongot készítettünk, egyszerűen az X tengely mentén 3 egységgel megnagyítottuk, az Y tengely mentén felére zsugorítottuk, s a Z tengely mentén 2 egységgel széthúztuk.

Elforgatás

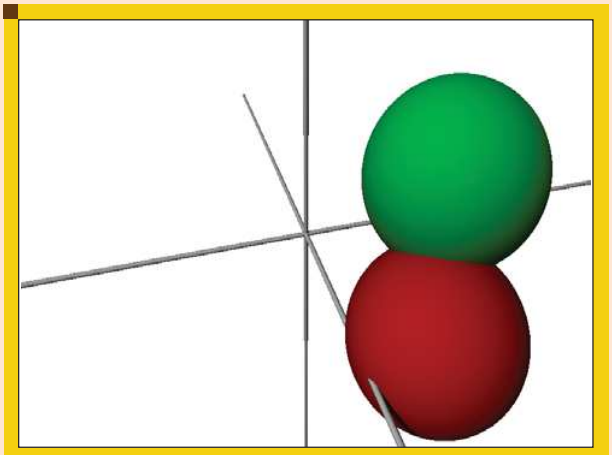
A három transzformáció közül a leginkább problémás, s így a legtöbb gondolkodást igénylő művelet az elforgatás. Például a *3. ábrán* látható testet szeretnénk elforgatni 30 fokkal minden tengely mentén, akkor a következőt kell használnunk:

```
sphere{
 <0,0,0>,1.5
 texture{
  pigment{
   color Red}}
 scale <3,0.5,2>
 rotate <-30,-30,-30>}
```

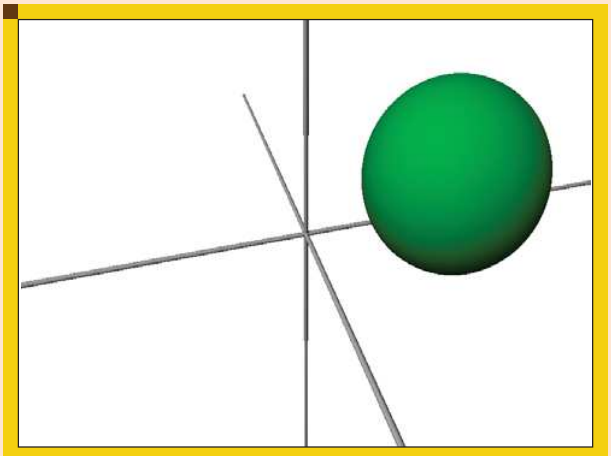
A forgatást megtehetjük több lépésben is, mivel a *PoVRay* az egy vektorban megadott értékek szerint először az X, aztán az Y, s végül a Z tengely körül forgatja meg a testet. Ez forgásszimmetrikus testek esetén nem nagy probléma, de más sorrenddel más-más eredményt kaphatunk (*5.* és *6. ábra*).



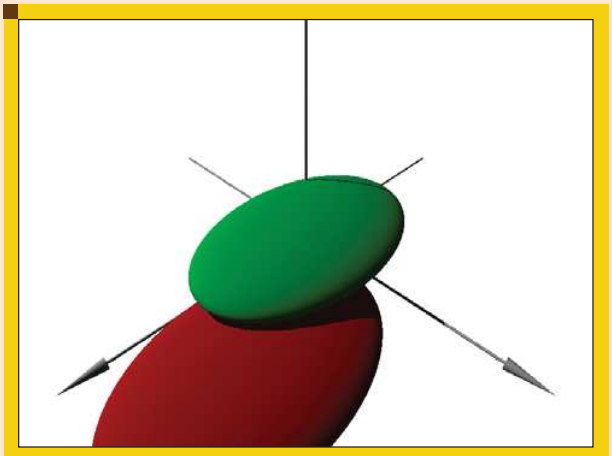
6. ábra Elforgatás Y-X-Z sorrenddel



7. ábra Gömb elforgatása eltolás nélkül



8. ábra Gömb elforgatása eltolással



9. ábra Gömb átméretezése

Ha a PoVRay programra hagyjuk a sorrendet:

```
box{
 <-1,-2,-1>,<1,2,1>
 texture{
  pigment{
   color Red}}
 rotate <-60,-30,-45>}
```

Ezzel szemben saját sorrend esetén a megfelelő sorokat megismételjük, és nullát írunk azon helyekre, amely tengelyek körül nem akarunk forgatni, a sorrend a példában Y-X-Z tengely:

```
box{
 <-1,-2,-1>,<1,2,1>
 texture{
  pigment{
   color Red}}
 rotate <0,-30,0>
 rotate <-60,0,0>
 rotate <0,0,-45>}
```

A különbség – ha nem is látványos – avatatlan szemnek is észrevehető, s az elforgatott tárgy újbóli elforgatásából adódik.

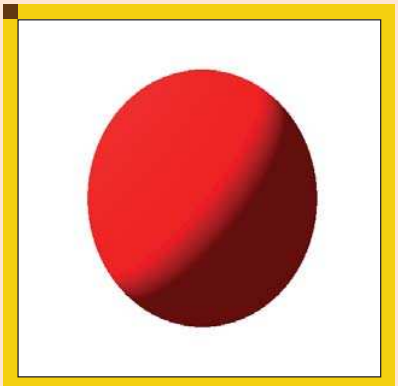
Fontos tudnivalók

Arra kell figyelnünk, hogy az elforgatás nem a test középpontja körül történik, hanem az origó körül. Ez a tulajdonság az *7. ábrán* látható igazán, ahol definiáltunk egy zöld és egy vörös gömböt, amelyek közül a vöröset elforgattuk minden tengely körül 60 fokkal. Mivel a gömb középpontja nem az origóban volt, így a két gömb nem esik egybe

```
sphere{
 <2,2,2>,1.5
 texture{
  pigment{
   color Red}}
 rotate <-60,-60,-60>}
sphere{
 <2,2,2>,1.5
 texture{
  pigment{
   color Green}}}
```

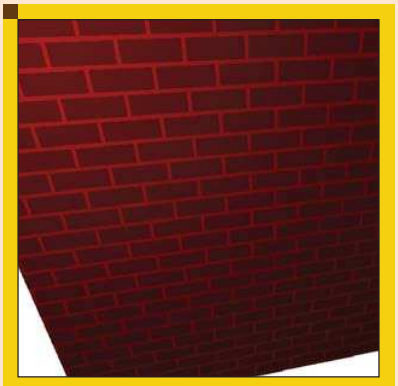
Ennek a problémának a kiküszöbölése egyszerűen annyi, hogy a gömböt az origóban hozzuk létre, majd az elforgatás után toljuk el a végleges helyére (*8. ábra*):

```
sphere{
 <0,0,0>,1.5
 texture{
  pigment{
   color Red}}
 rotate <-60,-60,-60>
 translate <2,2,2>}
```

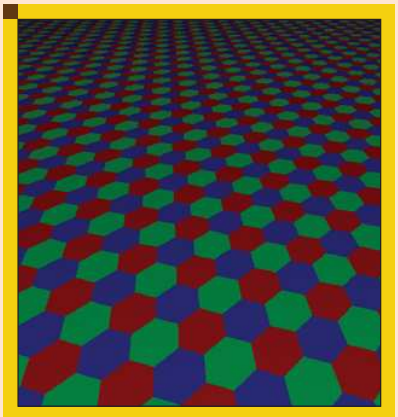


10. ábra Világító vörös szín

11. ábra „Sakktábla” felszín



12. ábra A „téglafal”



13. ábra Hatszög alakú csempék a padlón

A képen csak egy gömböt láthatunk, mivel a gömböt bármerre is forgatjuk: ugyan azt a kinézetű testet láthatjuk, mivel jelenleg csak színeket használtunk, textúrát nem. Célszerű minden testet az origóban létrehozni, majd forgatás és átméretezés után elmozgatni a végleges helyére. Ezzel sok munkától mentesülünk, ha később egy-egy testet el kell forgatnunk.
A fentihez hasonlóan az átméretezés sem a test középpontjához képest történik, hanem az origóhoz képest:

```
sphere{
 <2,2,2>,1.5
 texture{
  pigment{
   color Red}}
 scale <3,0.5,2>}


sphere{
 <0,0,0>,1.5
 texture{
  pigment{
   color Green}}
 scale <3,0.5,2>
 translate <2,2,2>}
```

Bár a *9. ábrán* látható két gömböt azonos pozícióba mozgattuk, az átméretezés okán már mérettel rendelkeztek.
A fentiek alapján a testeket mindig az origón hozzuk létre, végezzük el rajtuk az elforgatást és a méretezést, s csak a végén mozgassuk őket a végső helyükre. Ettől eltérő sorrend esetén nem biztos, hogy a kigondolt helyen és méretben találjuk meg a létrehozott testet.

Egyszerű textúrák

Eddig minden test egyszerű színekkel rendelkezett, amelyek – valljuk be – nem túl realisztikusak. De ne szaladjuk előre, nézzük meg az egyszerűbb színek és textúrák megadási lehetőségeit.
Mint azt már megtapasztaltuk, a színek is a textúra egyfajta megadása, mégpedig a pigment kulcsszó vezeti be őket, majd egy három elemű vektor követi, amely a szín vörös, zöld, illetve kék összetevőjét jelenti:

```
texture{
 pigment{
  color <1.0,0,0>}}
```

Normál esetben a színkomponensek 0.0 és 1.0 közötti értéket vesznek fel, de a ***PoVRay*** lehetőséged ad negatív illetve egynél nagyobb számok megadására is. Negatív szám esetén az adott szín ***nagyon*** sötét lesz, így erősen megvilágítva sem mutat az adott színre vajmi hajlandóságot. Egynél nagyobb színkomponens esetén a test az adott színnel szinte ***világítani*** fog, bár ez a saját fény más tárgyakra nem vetül rá (*10. ábra*):

```
texture{
 pigment{
  color <4.0,0.2,0.2>}}
```

A szín megadásakor egy apró trükkel könnyedén tudunk készíteni sakktáblához hasonló felszínt (*11. ábra*):

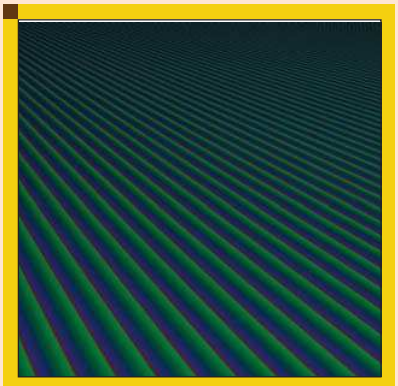
```
texture{
 pigment{
  checker color Red, color
Blue}}}
```

Mivel a gömb felszínére nem feszíthető olyan „sakktábla”, amely mindenhol szépen illeszkedik, kicsit összetörve látjuk viszont az erőfeszítésünket.
Tudunk készíteni „téglafalat” is, ha a `brick` kulcsszót használjuk a színek megadásakor. Ekkor a program a megadott színnel rajzolja a fugákat, s ennél kisebb fényerővel ábrázolja téglákat (*12. ábra*):

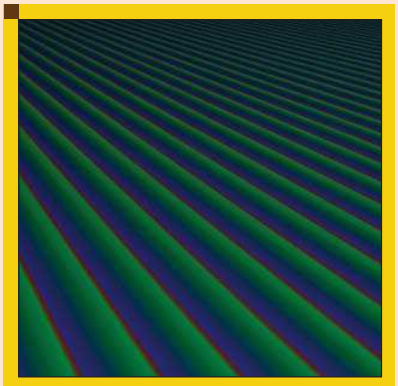
```
texture{
 pigment{
  brick color <1.0,0.0,0.0>}}
```

Megtölthetjük egy test felületét akár hatszögekkel is, amelyeket a program három színnel ábrázol, mégpedig úgy, hogy azonos szín nem lehet szomszédos hatszög (*13. ábra*):

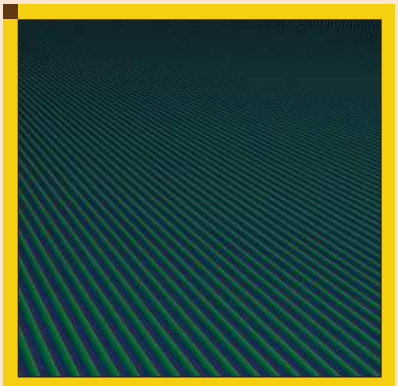
```
texture{
 pigment{
  hexagon color Blue, color
Red, color Green}}
```



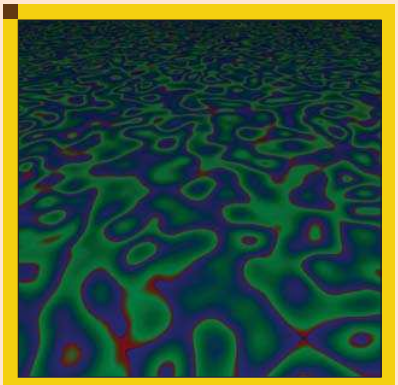
14. ábra Színátmenetek



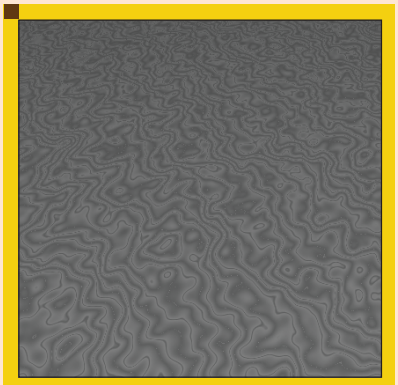
15. ábra Színátmenetek transzformálva



16. ábra Színátmenetek sűrítve



17. ábra „Leopárd-bőr" textúra



18. ábra „Márvány" textúra

19. ábra „Agate" textúra

Kicsit variálva a színekkel képesek leszünk színátmenetekre (`color_map`) is, amelyek egymás követik a gradiensnek megadott irányban (*14. ábra*):

```
texture{
 pigment{
  gradient x
  color_map{
   [0.00 color Red]
   [0.11 color Green]
   [0.99 color Blue]
   [1.00 color Red]}}}
```

A színek előtt megadott szám nulla és egy között lehet, s ezen az egy egységen belül határozza meg az átmenet méretét. A cikk első felében megismert transzformációk alkalmazhatók a textúrákra is, így képesek leszünk megváltoztatni a testek felszínét (*15. ábra*):

```
texture{
 pigment{
  gradient x
  color_map{
   [0.00 color Red]
   [0.11 color Green]
   [0.99 color Blue]
   [1.00 color Red]}
  rotate <45,0,0>
  scale 2.0}}
```

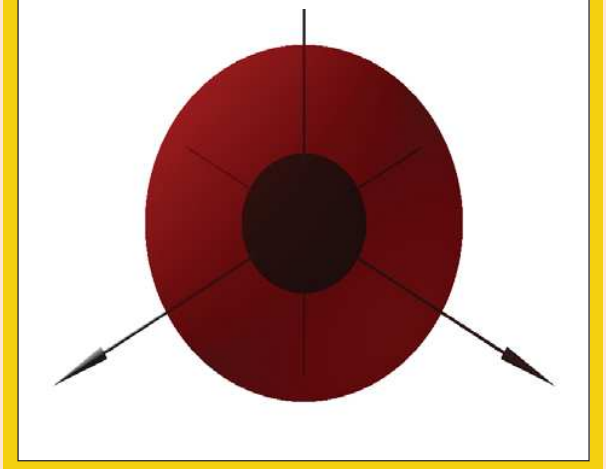
Van néhány módosító kulcsszó a textúráknál, amelyek érdekesebbé vagy mássá varázsolják a felületet. A legelső ilyen a `frequency` is, amelynek hatására a színátmenet sűrűbb vagy ritkább lesz. Ez a módosító ugyanis azt határozza meg, hogy egységnyi távolságon mennyi alkalommal ismétlődjön a megadott minta (*16. ábra*):

```
texture{
 pigment{
  gradient x
  frequency 4
  color_map{
   [0.00 color Red]
   [0.11 color Green]
   [0.99 color Blue]
   [1.00 color Red]}
  rotate <45,0,0>
  scale 2.0}}
```

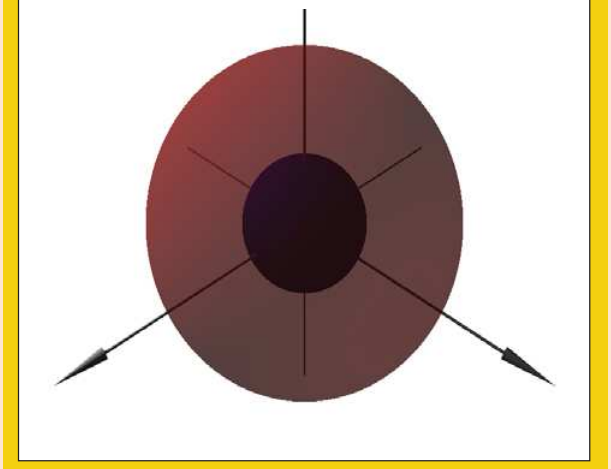
Ha a gradiens helyett kicsit összekuszáljuk a textúrát (`bozo`), akkor egyfajta leopárdmintát fogunk eredményül kapni (kiválóan alkalmazható vízfelszínhez is):

```
texture{
 pigment{
  bozo
  frequency 4
  color_map{
   [0.00 color Red]
   [0.11 color Green]
   [0.99 color Blue]
   [1.00 color Red]}
  rotate <45,0,0>
  scale 2.0}}
```

Lehetőségünk van „márványosítani" a színátmenetet, amely jól jön a faerezet és a kőmintázatok elkészítésekor (*18. ábra*). Az `omega`, a lambda valamint az `octaves` kulcsszavak



20. ábra Szűrő átlátszóság



21. ábra Áteresztő átlátszóság

a turbulencia paraméterezésére szolgálnak, más értékekkel másképp alakul a mintázat:

```
texture{
 pigment{
  marble
  frequency 4
  turbulence 0.5
  omega 0.8
  octaves 5.0
  lambda 1.5
  color_map{
   [0.00 color Gray]
   [0.11 color White]
   [0.99 color Gray]
   [1.00 color White]}
  rotate <45,0,0>
  scale 2.0}}
```

Valahol a `bozo` és `marble` között helyezkedik el az `agate`, amely egyfajta keveréke a kettőnek:

```
texture{
 pigment{
  agate
  frequency 4
  turbulence 0.5
  omega 0.8
  octaves 5.0
  lambda 1.5
  color_map{
   [0.00 color Gray]
   [0.11 color White]
   [0.99 color Gray]
   [1.00 color White]}
  rotate <45,0,0>
  scale 2.0}}
```

Átlátszóság

A testek lehetnek áttetszőek vagy átlátszóak, mint a víz vagy az üveg. Ez a tulajdonság nem okoz azonban automatikusan fénytörést, csak úgy tesz a tárgy, mintha áteresztené a mögötte lévő másik tárgyak fényét. A ***PoVRay*** megkülönböztet áteresztő (***transmittance***) és szűrő (***filtering transparency***) átlátszóságot. A kettő közötti különbséget képeken látni a legjobban (*20-21. ábra*).
A szűrő átlátszóságot az `rgbf` kulcsszó vezeti be, illetve egy négy paraméterből álló vektor, amelynek az első három paramétere a szín három komponense, az utolsó paramétere pedig az átlátszóság (*20. ábra*):

```
sphere{
 <4,4,4>,4
 texture{
  pigment{
   color rgbf <1,0,0,0.5>}}}
```

A szűrés a test saját színére vonatkozik, így a vörös gömb teljes mértékben kiszűri a nem vörös színeket, így a mögötte lévő kék gömb fekete színűnek látszik. Ez igaz a fehér háttérre is, amelyből így csak a vörös komponens marad meg.
Az áteresztő átlátszóságot az előzőhöz hasonlóan az `rgbt` kulcsszóval tudjuk meghatározni.

```
sphere{
 <4,4,4>,4
 texture{
  pigment{
   color rgbt <1,0,0,0.5>}}}
```

Ebben az esetben a tárgy színe nem jelent szűrést, egyszerűen az átlátszóság függvényében átlátszóvá válik, így a mögötte lévő kék gömb színe is átlátszik.
Egészen eddig csak fényeket, színeket és formákat hoztunk létre, ideje a különféle anyagokra jellemző tulajdonságokat is modellezni: a fémek csillanjanak meg, a fának legyen szép erezete, az üvegben törjön meg a fény. Lássuk el textúrával a tárgyakat, de erről már a következő részben!

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

A PovRay projekt honlapja
➲ http://www.povray.org
A cikkben említett fájlok
➲ http://user.enaplo.hu/~auth.gabor/pov/

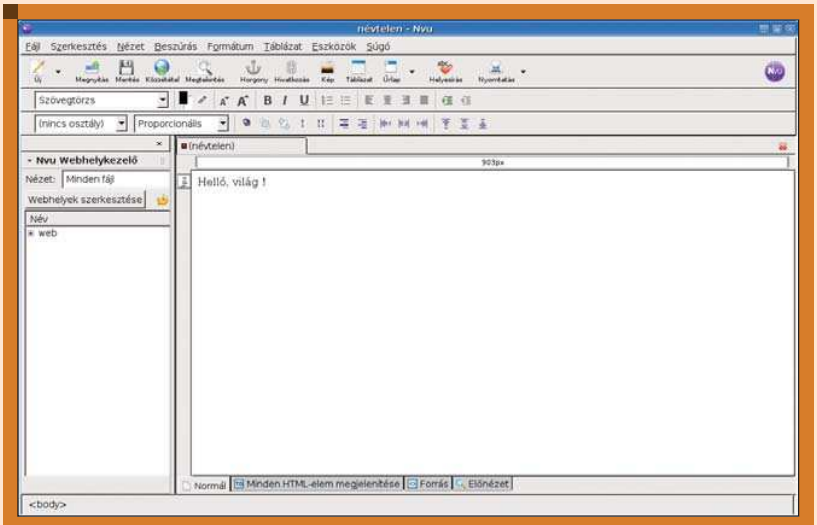
Bevezetés az NVU weblapszerkesztő használatába

Az NVU a Netscape, majd a belőle kialakult Mozilla böngésző kiegészítését képező Composer újjászületése. Amikor néhány évvel ezelőtt a Mozilla készítőiben megfogalmazódott a gondolat, hogy az addig egy csomagot képező funkciókat (böngésző, szerkesztő, levelező kliens, naptár alkalmazás, chat kliens) különálló alkalmazásokként fejlesztik tovább, a Composer alapjain megszületett az NVU.

Angolul kiejtve az *„n-view”* (*„new view”*) új nézetet, új nézőpontot jelent. A fejlesztést a *Linspire* – akkori nevén *Lindows* – cég támogatta, vezető fejlesztőnek pedig a *Composer* vezető fejlesztőjét, *Daniel Glazmant* bízták meg. Az *NVU* számos verziót élt meg első megjelenése óta, ebben az írásban a nemrég megjelent, eddigi legjobb, 1.0 változat legfontosabb funkcióit mutatjuk be. Az *NVU* ingyenesen hozzáférhető és felhasználható, *MPL/LGPL/GPL* licenszű, többplatformos – *Windows*, *Linux*, *MacOS* – alkalmazás, számos nyelvi lokalizációja és forráskódja is elérhető.

Ismerkedés

Első pillantásra az *NVU* egy, a *Frontpage*-re vagy *Dreamweaverre* hasonlító *„azt kapod amit látsz”* (*„what you see is what you get”*) szerkesztő alkalmazás képét mutatja, ám, amint az a továbbiakban kiderül majd, ennél jóval többet kínál.
Az *NVU* telepítőjét a honlapjáról lehet elsődlegesen elérni, minden platform alá, *Windows* esetén szabvány windowsos öntelepítő formátumban, *Linuxok* esetén a legnépszerűbb disztribúciók csomag-formátumaiban illetve csomagolt binárisok formájában. Természetesen a forráskód is elérhető. A telepítés menete minden esetben egyszerű, windowsos változat esetén grafikus felületű és alig egy-két kattintást igényel. Itt jegyezzük meg, hogy a magyar nyelvi kiegészítést a Magyar *Mozilla Projekt* honlapjáról lehet letölteni. Ehhez az íráshoz az *NVU 1.0* linuxos változatát használtuk, *Debian GNU/Linux SID* operációs rendszer alatt, *KDE* környezetben.

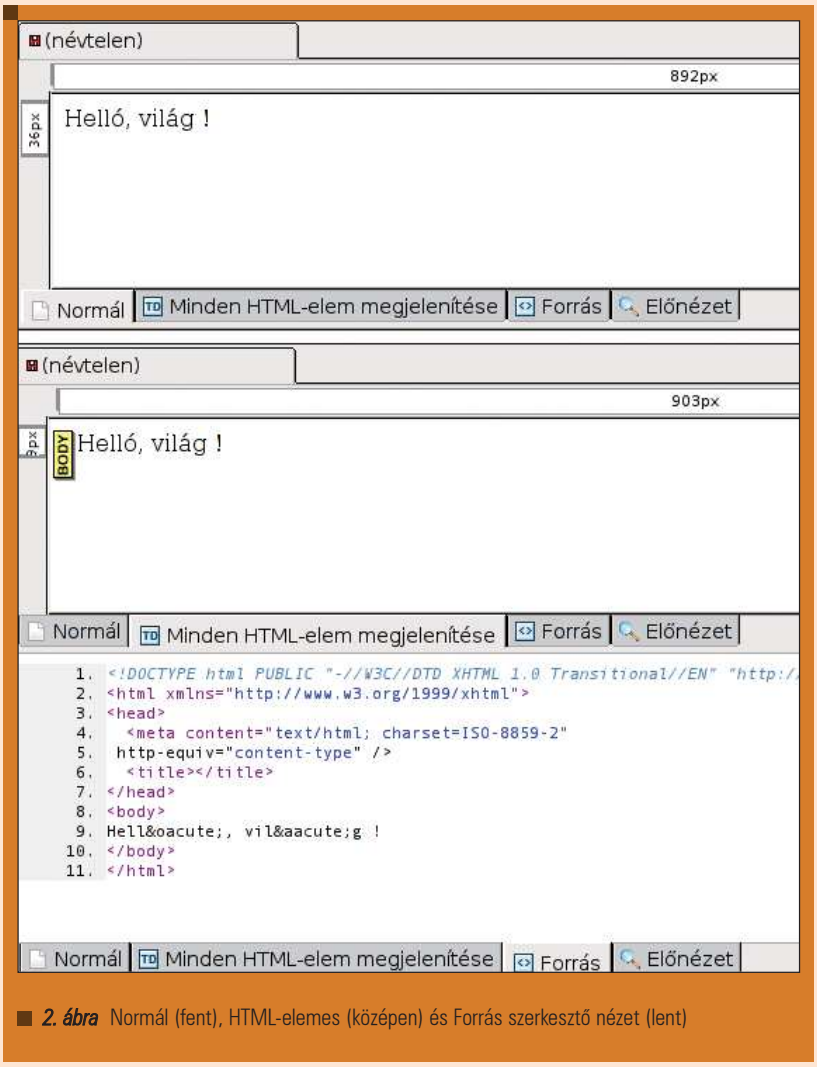


1. ábra Az NVU felülete

Az *NVU* felülete a *Mozillaban* is használt *Gecko* megjelenítési könyvtárra épül, jól illeszkedve bármilyen platform bármilyen felhasználói környezetébe. Az alkalmazás alap-felülete az *1. ábrán* látható.
Az alkalmazás felhasználói felülete nem ad túlzottan bonyolult összképet, ami egy fontos jellemző, tekintve, hogy első sorban olyan felhasználóknak szánták, akik nem professzionális weblapszerkesztők. A látható eszköztárak szövegszerkesztőkből megszokott alap-eszközöket tartalmaznak, amelyeket az új felhasználók többsége is nagy valószínűséggel már ismer. A későbbiekben ezek használatát is bemutatjuk.
Mielőtt az *NVU*-s weblapszerkesztés rejtelmeibe mélyülnénk, bemutatunk olyan beállítási lehetőségeket, amelyek fontosak a későbbi munka szempontjából.

Első lépések

Mielőtt új dokumentumot hoznánk létre és megkezdenénk a munkát, érdemes az *Eszközök->Beállítások* menüpont alatt kicsit elidőznünk. Az egyik fontos beállítás, amit érdemes első lépésként beállítani, a *Általános beállításoknál* szerepel, *„Képek és más csatolt fájlok mentése az oldalak mentésekor”*. Ezt azért fontos bejelölni,

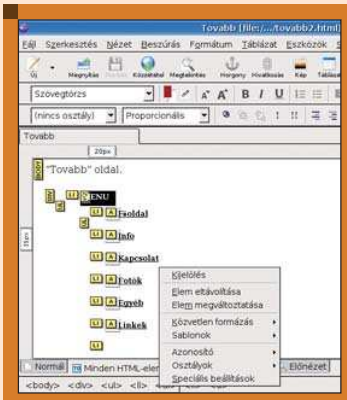


2. ábra Normál (fent), HTML-elemes (középen) és Forrás szerkesztő nézet (lent)

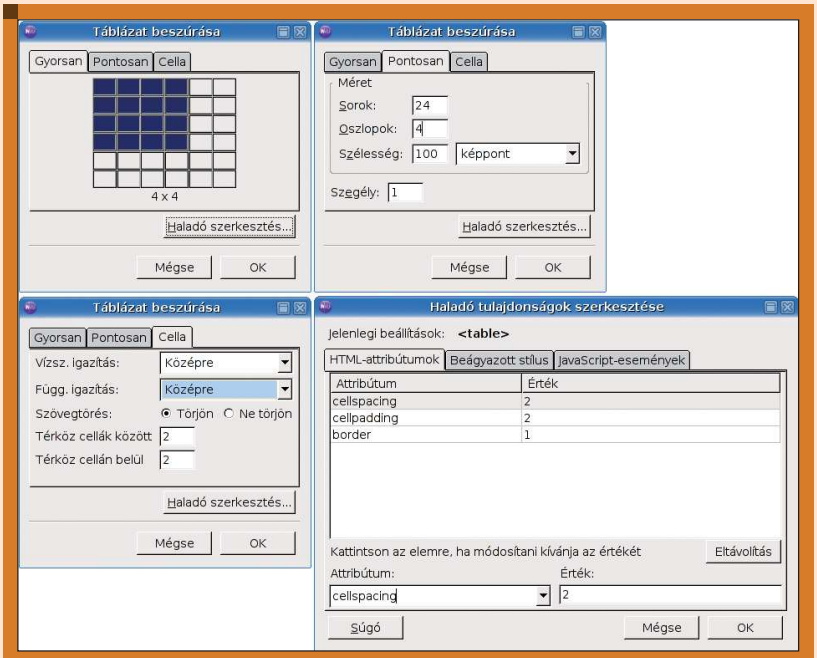
hogy amikor a készülő dokumentumunkba különféle külső elemeket szúrunk be (pl. képek), akkor ezek mentéskor az oldal forrásával együtt kerülnek mentésre. Az ***Új oldal*** paramétereinél fontos lehet előre megadni az oldal preferált nyelvét és a használni kívánt karakterkészletet – magyar szöveg esetén ez praktikusan ***ISO-8859-2*** különben és minden más esetben javasolt az ***UTF-8*** használata. A Haladó beállításoknál a készülő oldal kód-forrásának nyelvét lehet kiválasztani, ***HTML4*** és ***XHTML1*** közül választva. Kezdő és haladó felhasználóknak egyaránt célszerű az ***„XHTML1 Transitional”*** séma használata.
A ***HTML4*** és ***XHTML1*** közötti különbségekre itt most nem térünk ki.
Ezek után létrehozhatjuk az első dokumentumunkat, a ***Fájl->Új*** menüpont segítségével. Létrehozhatunk üres dokumentumot új munka kezdetekor, de létrehozhatunk új lapot sablonok alapján is, amelyeket mi is elkészíthetünk, de használhatunk például ***Frontpage*** sablonokat is. A mi esetünkben üres lapot fogunk létrehozni, amelyen a későbbiekben bemutatjuk az ***NVU*** képességeit.
Az ***NVU*** beépítetten tartalmaz egy weblap kezelő és közzététel eszközt is, amit ***NVU*** webhelykezelőnek hívnak, és ami az ***NVU*** elindításakor a felület bal-alsó részén található, ahogy az az ***1. ábrán*** is látható. Ha van saját webszerverünk vagy egy a szolgáltatónál előfizetett webtárhelyünk, ***FTP***-s hozzáféréssel, akkor itt beállíthatjuk ennek paramétereit. Ez annyiban könnyíti meg a munkánkat, hogy a későbbiekben, amikor már készül az oldal, a ***Fájl->Közzététel*** menüponttal illetve az eszköztár ***Közzététel*** eszközével egy kattintással a teljes megírt weboldalunkat és a hozzá tartozó kiegészítő állományokat (képek, további ***HTML*** oldalak, stb.) feltölthetjük a webszerverre, ahol ez azonnal publikusan elérhető lesz. Erről a funkcióról a későbbiekben még szót ejtünk.
Eljutottunk addig a pontig, ahol elkezdhetjük a létrehozott üres weblapunk szerkesztését. Fordítsuk most figyelmünket a szerkesztőablakra és az ***NVU*** egyik erősségére, a különféle nézet-ablakokra.

A szerkesztő

Az ***NVU*** felületét megvizsgálva könnyen észrevehetőek a fő szerkesztőablak alatt található fülecskék. Ezek a fülecskék szolgálnak a különféle nézetek közötti váltásra. Ugyanezek a funkciók elérhetők a Nézet menüben is. Ezek a nézetek azt a praktikus célt szolgálják, hogy szerkesztési lehetőséget nyújtsanak teljesen kezdő szintű felhasználóktól a profi felhasználókig. Vegyük most sorra ezeket a nézeteket, amelyekre egy rövid példa látható a ***2. ábrán***.
A ***Normál*** nézet olyan szerkesztési felületet ad a felhasználó keze alá, amellyel szinte mindenki boldogulni tud, akinek van némi tapasztalta grafikus felületű szövegszerkesztőkkel. Egyszerű oldalak elkészítéséhez ebben a nézetben egyáltalán nem kell – és nem is lehet – ***HTML*** kódot használni. Mivel az ***NVU*** egyik erőssége, hogy meglehetősen könnyen olvasható és tiszta kódot generál az elmentésre kerülő oldal ebben a nézetben is megfelelő lehet. A szöveg stílusát (font típusa, mérete, iránya, pozíciója, színe, stb.) az eszköztár segítségével ill. a ***Formátum*** menü pontjainak segítségével könnyen személyre lehet szabni. Képeket, hivatkozásokat, különleges karaktereket, táblázatokat, űrlapokat, stb. a ***Beszúrás*** menü pontjai tesznek könnyen elérhetővé és szerkeszthetővé, amelyek egy részére a későbbiekben részletesebben kitérünk.
A ***HTML*** nézet – teljes elnevezése ***„Minden HTML elem megjelenítése”*** – leginkább a ***Windowsra*** készült ***Adobe GoLive***-ból lehet egyesek számára ismert. Ez a nézet gyakorlatilag egy átmenet a ***Normál*** nézet és a ***Forrás*** nézet között. Az oldal egyes elemei (szöveg, kép, hivatkozás, stb.) között grafikus elemek jelzik az egyes részek határait – ahogy az a ***2. ábra*** középső képén is látható.



■ *3. ábra* Helyi menü az állapotsorban



■ *4. ábra* Táblázat beállításai

A *Forrás* nézet – például a *2. ábra* alsó része – egy olyan szerkesztési nézetet szolgáltat, amely leginkább a profi felhasználók számára otthonos terület. Ebben a nézetben a készülő lapunk teljes forráskódját láthatjuk és szerkeszthetjük. A szerkesztést az *NVU* ebben a nézetben a *HTML* szintaxis-elemek kiemelésével segíti.

Bármely nézetben nagy segítség lehet a szerkesztés során a *Szerkesztés* menü *Helyesírás-ellenőrzés* menüpontja, amivel könnyen elkerülhetők a gyakran nagyon zavaró helyesírási hibák.

Fontos, a gyors munka szempontjából egyik kimagasló eszköz az, hogy az oldal egy adott részén állva a kurzorral az *NVU* státusz-sorában láthatjuk az oda vezető utat az oldal *HTML*-elemeinek hierarchiájában. Ami ennél is fontosabb, hogy adott hierarchia-elemre jobb egérgombbal kattintva egy lokális módosító menü lép elő, amelynek segítségével gyorsan tudunk lokális paramétereket változtatni (*3. ábra*).

Az *Előnézet* módban az *NVU* lehetőséget ad arra, hogy külső alkalmazás indítása nélkül megtekinthessük a készülő oldalunkat.

Sajnos egyelőre az *NVU* nem tartalmaz úgynevezett *automatikus kiegészítés* (*autocomplete*) funkciót, vagyis az éppen beírt *HTML* parancsok beírás közbeni felismerésére és automatikus kipótlására, de remélhetőleg ez a funkció és beépítésre kerül a jövőben, hiszen nagyban megkönnyítené a *Forrás* nézetben végzett munkát.

A továbbiakban néhány fontos elem beillesztési lehetőségeit mutatjuk be. Ennek a bemutatónak nem célja a *HTML* nyelv bemutatása, így a konkrét *HTML* parancsokra csak akkor térünk ki, ha feltétlenül szükséges.

Táblázatok

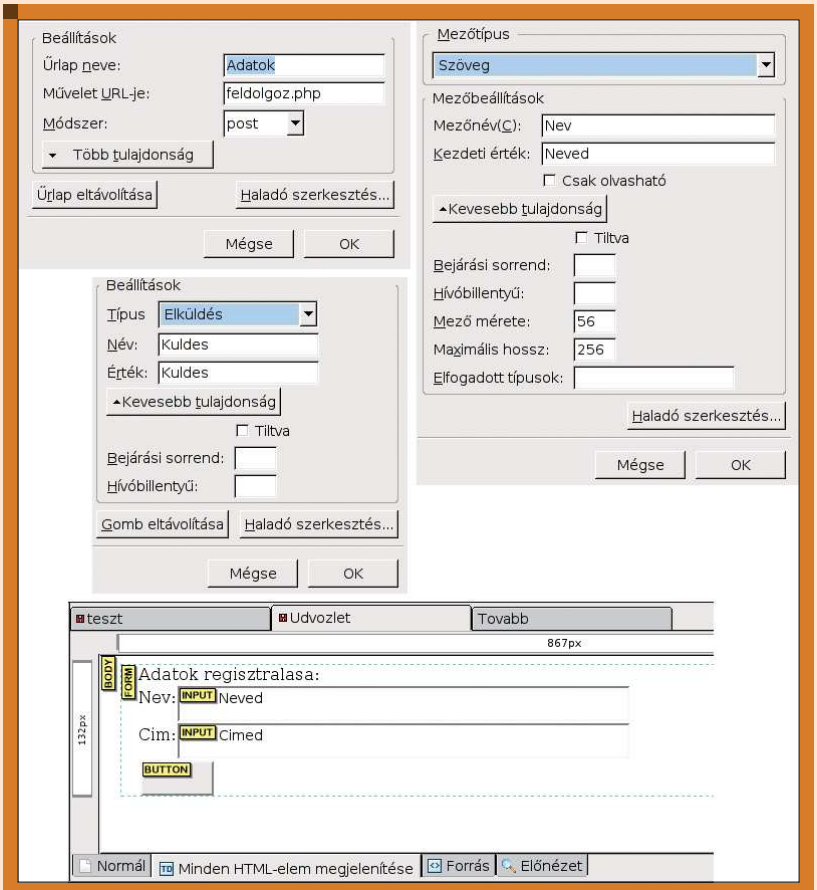
Weboldalak gyakori elemei a táblázatok. Nagyon sokrétűen használhatók, az egyszerű adat-reprezentációtól teljes oldalak szerkezetének felépítéséig. Az *NVU* könnyen kezelhető táblázat-beszúrásra ad lehetőséget.

Táblázatokat a *Táblázat->Beszúrás* menüpontban tudunk beilleszteni a dokumentumunkba. Beszúrás előtt és után is lehetőségünk van a táblázat paramétereit tetszés szerint beállítani, ahogy azt a *4. ábrán* láthatjuk. *„Haladó szerkesztés”*-t választva haladó felhasználóknak lehetősége van arra, hogy minden lehetséges paramétert kézzel beállítson. Ezt természetesen a Forrás nézetben is megtehető, de ez a módszer egy áttekinthető, könnyen kezelhető felületet biztosít ehhez a feladathoz, megkönnyítve a haladó felhasználók munkáját is.

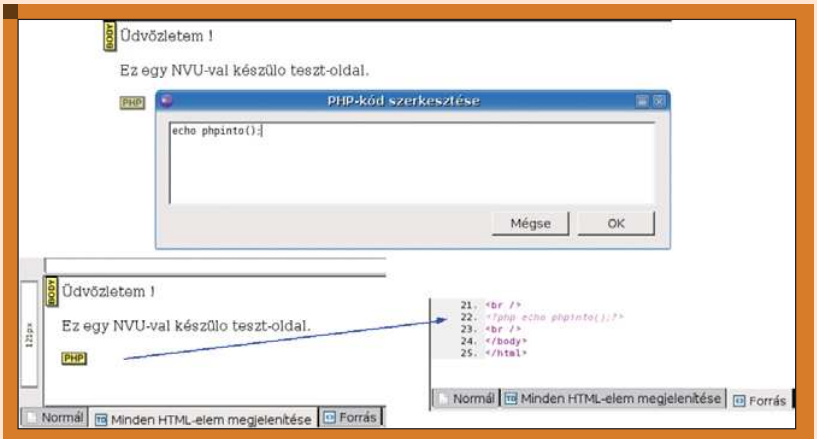
Űrlapok

Űrlapok (form) használatára is gyakran szükség lehet. Az űrlapok lehetőséget biztosítanak arra, hogy a weblapot látogatók adatokat közölhessenek, küldhessenek el, amelyeket a weboldal tulajdonosa felhasználhat, például felhasználók regisztrálása, vendégkönyvek, stb. esetén. Az NVU-ban űrlapok kezelésére is van lehetőségünk, a táblázatok kezeléséhez hasonló könnyedséggel. Erre a *Beszúrás->Űrlap* menü, ill. az eszköztár *Űrlap* eszköze biztosít lehetőséget. Az űrlapkezelést egy egyszerű példán keresztül mutatjuk be. Váltsunk a példa kedvéért a *HTML* elemes nézetre. Az *Űrlap->Űrlap-készítés*-t kiválasztva hozzunk létre egy „Adatok” nevű űrlapot, műveleti *URL*-nek adjunk meg például *feldolgoz.php*-t – ez lenne egy saját szkript, ami később feldolgozná a beérkező adatokat – , módszernek *post*-ot választva. Az *Űrlap->Űrlapmező*-t választva szúrjunk be két bekérési mezőt, egyet a felhasználó nevének, egyet pedig a címének. Mezőtípusnak válasszunk „szöveg”-et, *Mezőnév*-nek adjunk *„Nev”* es *„Cim”* neveket az első ill. a második esetben, *„Maximális hossz”*-nak pedig állítsunk be 256 karaktert, ebbe valószínűleg minden belefér. Ezek után szúrjunk be egy gombot, amire kattintva az űrlap tartalma elküldésre kerül, az *Űrlap->Gomb* beszúrása menüpont segítségével, a gomb *„Típus”*-ának „elküldés”-t választva. Az aktuális helyzetet a *5. ábra* mutatja.

A fenti példában egy külső *PHP* szkriptre hivatkoztunk azzal, hogy az



5. ábra Űrlap létrehozása: űrlap (bal felső), szöveges beviteli mező (jobb felső), gomb, és az aktuális állapot HTML elemes nézetben (alsó)



6. ábra PHP kód beágyazása a készülő dokumentumba. Fent: Beszúrás->PHP kód, alsó jobb: eredmény HTML elemes és (alsó bal) Forrás nézetben

adatok feldolgozását majd az végzi. Az *NVU* nem ad támogatást *PHP* szerkesztéshez, így ezt a szkriptet magunknak kell létrehoznunk – ennek megírására ebben az írásban nem térünk ki. Azonban az *NVU* lehetőséget biztosít *PHP* kód beillesztésére a készülő dokumentumba, a *Beszúrás->PHP kód* menüpont segítségével. Egy egyszerű *PHP* függvényhívás beágyazására mutat példát a *6. ábra*. A továbbiakban más szkriptek ill. egyéb külső elemek beágyazására adunk útmutatást.

Egyéb elemek beillesztése

Az előzőekben még nem ejtettünk szót képek beszúrásáról. Képek beillesztésére a *Beszúrás->Kép* menüpont ad lehetőséget. Ezt kiválasztva egy olyan dialógus-ablakot kapunk, amelyben minden kép-tulajdonságot könnyen beállíthatunk. A dialógus ablak négy fülecskéje (*„Hely”*, *„Méretek”*, *„Megjelenés”* és *„Hivatkozás”*) tartalmazza csoportosítva a képi paramétereket. A *„Hely”* fülecskével megadhatjuk a beilleszteni kívánt képet, a *„Méretek”*-ben a kép megjelenítési méreteit, a *„Megjelenés”*-ben megadhatjuk a kép körüli szöveg viselkedését – például a kép körüli térközök, hogyan fusson a szöveg a kép körül –, a *„Hivatkozás”* lapon beállíthatjuk, hogy a képünk legyen-e hivatkozás is egyben és ha igen, akkor hova mutasson ez a hivatkozás. Természetesen *„Haladó szerkesztés”*-t választva itt is szabad kezet kapunk minden lehetséges paraméter beállítására.

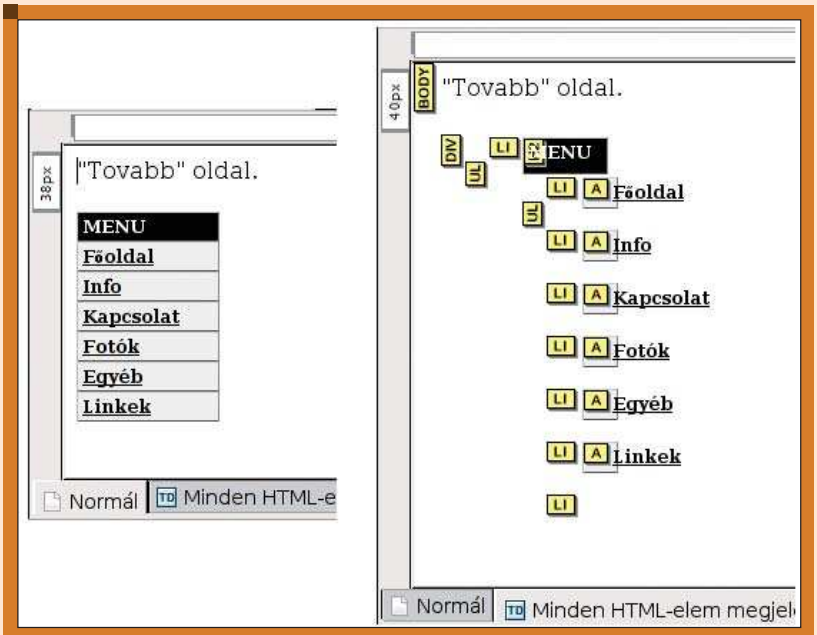
Ahogy már az előzőekben említettük, lehetőségünk van *PHP* kód beillesztésre is, a *Beszúrás->PHP* kód menüpontban. Hasonló módon külső forrásból származó *HTML* kódot is beilleszthetünk a dokumentumba a *Beszúrás->HTML* menüpont segítségével. Ezeken kívül különleges karaktereket és szimbólumokat is beilleszthetünk.

A fentiek mellett az *NVU* lehetőséget ad *JavaScriptben* írt úgynevezett *„Okos felületi elemek”* (a *„Beszúrás”* menüben) beillesztésére is. Példaként az *NVU* tartalmaz egy naptár elemet, és egy *„Made with NVU”* - *„NVU-val készítve”* – gombot, amit beilleszthetünk.

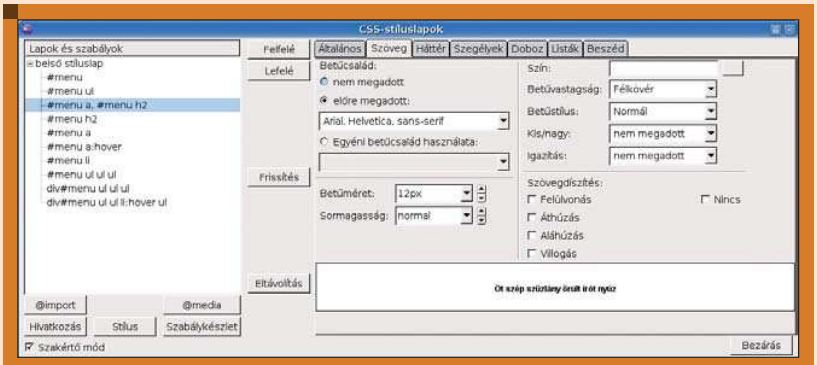
CSS

A *CSS* a *„Cascaded Style Sheet”* rövidítése, ami egy stílus-leíró nyelv. A *CSS* lehetővé teszi olyan weboldalak létrehozását, amelyekben a tartalom és a megjelenítés különválik, így könnyebbé téve a fejlesztést és a későbbiekben a külalak megváltoztatását a tartalom változtatása nélkül. Manapság már majdnem minden böngésző támogatja a *CSS*-t valamilyen szinten, az eltérések az egyes *CSS* elemek mennyiségének támogatásában illetve a támogatott *CSS* verzió miatt észlelhetők. Ma már a webfejlesztők többsége belátja a *CSS* stíluslapok használatának előnyeit, így fontos tulajdonsá-

ga az *NVU*-nak az, hogy nem amellett, hogy lehetőséget ad *CSS* elemek beépítésére a dokumentumba, egy *CSS* szerkesztő dialógust is tartalmaz, amellyel a kezdő felhasználók – és sokszor a haladók is – egy olyan eszközhöz jutnak. Ez nagyon sok esetben megkönnyítheti és felgyorsíthatja a munkát.
A *CSS* szerkesztő az *Eszközök->CSS szerkesztő* menüpont alatt érhető el.
A *CSS* előnyeinek bemutatásához rövid példaként egy *CSS* alapú menüt mutatunk be. Ennek alapja az, hogy az egyszerű felsorolás *HTML* elemeinek (*ul*, *li*) stílusait megváltoztatjuk úgy, hogy egyszerű egymásba ágyazott felsorolás helyett menüként jelenjenek meg. A példa menünk *Normál* és *HTML* nézetben a *7. ábrán* látható.
A stíluselemeket egyaránt szerkeszthetjük a *CSS* szerkesztő segítségével (*8. ábra*), vagy kézzel a *Forrás* nézetben (*9. ábra*).



7. ábra A CSS-alapú példa-menünk Normál (bal) és HTML (jobb) nézetben



8. ábra A CSS-alapú példa-menünk paramétereinek szerkesztése a beépített CSS szerkesztővel

Kiterjesztések

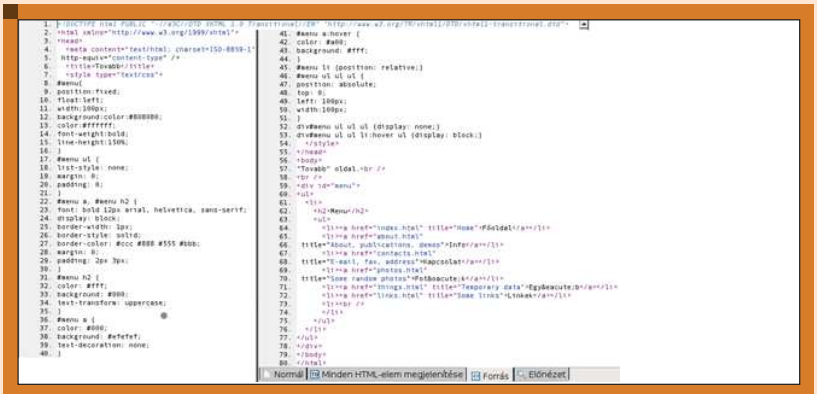
Az *NVU*-hoz léteznek úgynevezett *kiterjesztések* (*extension*). Ezek olyan kiegészítések, amelyeket bárki megírhat, majd beilleszthető az *NVU*-ba. Ennek a módszernek a mintája a Mozilla illetve a *Firefox* böngészők kiegészítéskezelése, lévén, hogy az *NVU* az ezek által használt *Gecko* könyvtárakra épül. *NVU*-ban kiterjesztéseket az *Eszközök->Kiterjesztések* menüponttal tudunk telepíteni.
Itt két kiegészítésről ejtünk szót.
Az egyik a lokalizációs kiegészítések, például a magyar nyelvi lokalizáció is egy ilyen kiegészítéssel telepíthető.
A másik kiegészítés, amelynek itt helyet adunk lehetőséget ad *NVU* felhasználók számára, hogy a méltán elhíresült *Mambo* (illetve új nevén *Joomla*) tartalom-kezelő portálrendszerhez *sablonokat* (*template*) illetve tartalmat készítsen. Ennek neve *Mambo template builder*. Telepítés után az *NVU* eszköztára számos új eszközzel bővül (10. ábra), amelyek segítik a *Mamboval* a *template*-ek készítését.

Publikálás, fájlkezelés

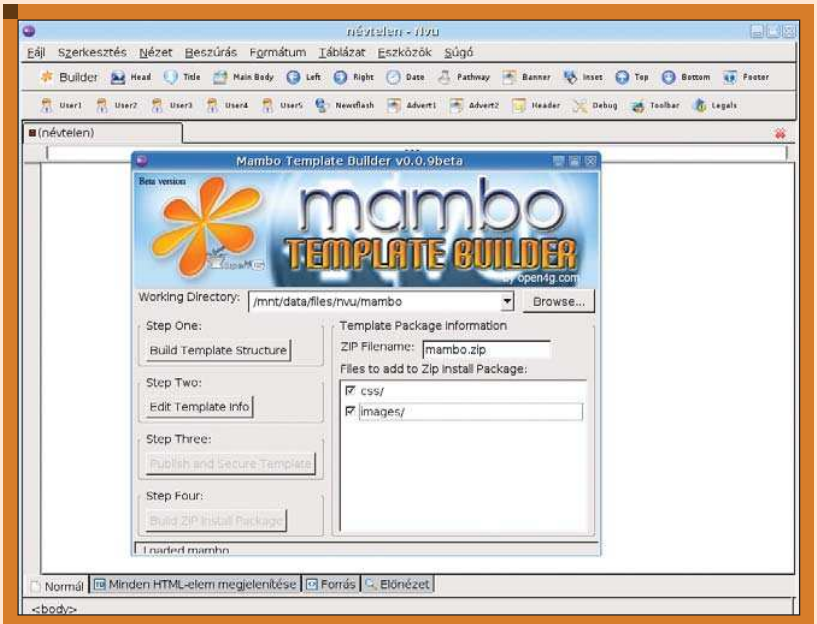
Ahogy azt az előbbiekben már említettük, az *NVU*-ban rendelkezésre áll egy publikáló/közzététel eszköz, amely megkönnyíti az oldalak menedzselését, és lehetővé teszi, hogy egy kattintással (*„Közzététel”* eszköz) a létrehozott és/vagy módosított oldalakat feltöltsük a webszerverre, ahonnan majd publikusan elérhető lesz.
A webhelyek szerkesztése gombra kattintva a *11. ábrán* látható dialógus ablak lép elő. Itt beállíthatjuk a webszerver elérési adatait, címet, belépést. Később, amikor a *„Közzététel”* eszközt használva feltöltjük az elkészített oldalakat, az itt beállított elérést használva történik a feltöltés (*12. ábra*).
Ahhoz, hogy ezt az eszközt használni tudjuk *FTP* szintű elérésre van szükség a webszerverhez, vagy a webszolgáltatónkhoz. Sajnos az *NVU* jelenleg nem támogat titkosított fájl-átviteli protokollokat (*scp*, *sftp*), remélhetőleg a jövőben változni fog.

Végszó

Ebben a meglehetősen rövid gyorstalpalóban megpróbáltunk betekintést adni az *NVU* lehetőségeinek és eszközeinek világába. Talán sikerült egy olyan ingyenes és nyílt forráskódú eszközt megismertetnünk az olvasóval, amely ezáltal még szélesebb körben elterjed. Természetesen az elterjedésnél fontosabb, hogy hasznos eszközévé válhasson a webszerkesztő társadalomnak, és reméljük, hogy ez a rövid ismertetés elég bepillantást nyújtott ahhoz, hogy esetleges hasznosságáról az olvasó legalábbis elgondolkodjék.
Végszóként röviden azon időznénk el egy keveset, hogy milyen jó és rossz tulajdonságokat tudunk az *NVU*



9. ábra A CSS-alapú példa-menünk forrása Forrás nézetben



10. ábra A Mambo template builder eszköztárai

mellett és ellen felsorakoztatni. Kezdjük a rosszakkal. Számos webszerkesztő hiányosságnak ítélheti azt, hogy keretek és keretcsoportok (frame, frame-set) kezelése hiányzik az *NVU*-ból. Viszont ezt a tulajdonságot a szerző éppen ellenkezőleg, előnynek tekinti. Ennek oka az, hogy az új, elterjedő *HTML* szabványok (*XHTML 1.x strict/transitional*) már nem támogatják a keretcsoportok használatát. A frame-set-ek külalakját és viselkedését teljes egészében reprodukálni tudjuk *CSS* stíluslapokkal és a szabványok követése érdekében ezt is javasolt tennünk. A hátrányok ill. hibák közé kell sorolnunk azt, hogy az *NVU* hajlamos arra, hogy esetenként figyelmen kívül hagyja a beállításoknál tett azon kívánságunkat, hogy ne formátumozza újra a *(X)HTML* forrást mentéskor, aminek esetenként az az eredménye, hogy újratördeli a forrásokat. Ez általában nem probléma, de mivel sok fejlesztőnek fontos a jól olvasható *(X)HTML* forrás, ez mindenképp hátrány. A *HTML* parancsok szín-kiemelése is lehetne sokkal jobb, ami segítené a forrás-szerkesztés hatékonyságát. A rossz dolgok után térjünk rá a pozitív tulajdonságokra. Mindenképpen pozitívum az, hogy dokumentum-létrehozáskor a *HTML*, *XHTML, Strict/Transitional DTD* választási lehetőségek széleskörű szabad kezet adnak. Nagyon fontos pozitívum a beépített *CSS* szerkesztő, mint ahogy a webes közzététel eszköz is. A többféle nézetes szerkesztő felület szintén a jó tulajdonságokat gazdagítja. Ezek mellett nagyon fontos pozitívum, hogy egyrészt ingyenes és nyílt forrású az alkalmazás, másrészt hogy *Windows*, *Linux* és *MacOS* operációs rendszerekre fordított változat is elérhető. A kedves olvasónak kellemes webszerkesztést kívánunk az *NVU*-val!

Kovács Levente
(leventek@gmail.com)
26 éves, informatikus- és villamosmérnök. Jelenleg doktorandusz a Veszprémi Egyetem Képfeldolgozás és Neuroszámítógépek Tanszékén. Hat éve használ különféle Linux disztribúciókat.

KAPCSOLÓDÓ CÍMEK

Az NVU honlapja:
➲ http://www.nvu.com

A Mozilla honlapja:
➲ http://www.mozilla.org

A Linspire honlapja:
➲ http://www.linspire.com

A Macromedia honlapja:
➲ http://www.macromedia.com

GNU GPL:
➲ http://www.gnu.org/copyleft/gpl.html

Magyar Mozilla Projekt NVU oldala (magyar nyelvi fájlok):
➲ http://www.fsf.hu/nvu.html

HTML4:
➲ http://www.htmlhelp.com/reference/html40/

XHTML1:
➲ http://www.w3.org/TR/xhtml1/

PHP:
➲ http://www-php.net

JavaScript:
➲ http://www.w3schools.com/js/default.asp

CSS:
➲ http://www.w3schools.com/css/default.asp

Mambo:
➲ http://www.mamboserver.com

NVU Mambo sablonkészítő:
➲ http://www.open4g.com

VLC – médialejátszó Linuxra

A VLC nevezetű multimédia-lejátszó egy új fejlesztésű alkalmazás, mely méltó „ellenfele” lehet a közismert, régóta fejlesztett MPlayer-nek és Xine-nak. Kis mérete és viszonylag puritán kinézete ellenére egy sokat tudó, kompakt, felhasználóbarát programmal van dolgunk, melynek nevét még sokat fogjuk emlegetni a jövőben.

Manapság a ***Linux*** operációs rendszer már mindenre használható. Régebben ez közel sem volt így.
Az újdonságnak számító, ***Minix*** kernelből létrejövő rendszer a 90-es évek elején csaknem kizárólag szerver-funkciókat látott el.
Az idő előrehaladtával és az internet egyre szélesebb körű elterjedésével viszont mára már csaknem mindegyik ***Linux*** disztribúció alkalmas otthoni, úgynevezett desktop használatra is.
Grafikus felületek és programok garmadája áll rendelkezésünkre, melyek minél kényelmesebb és egyszerűbb használatot biztosítanak számunkra.
Az otthoni számítógép-használatot több „szórakoztató alkalmazás” varázsolja sokszínűvé.
Gondoljunk csak a 80-as évek elejétől kezdve töretlen népszerűséggel bíró számítógépes játékokra, melyek úgymond a ***Commodore*** piacra törése óta jelen vannak, jelen korunkban pedig már a hardvergyártó cégek alkalmazkodnak hozzájuk.
A másik, szintén a desktop felhasználók kikapcsolódását elősegítő alkalmazáscsalád a médialejátszók széles skálája.
A zenék és filmek lejátszására alkalmas médialejátszóknak 3 fő fajtáját különböztetjük meg:

- Csak zenefájlok (például ***mp3, wav, ogg***) lejátszására alkalmas programok (például ***xmms***)
- Csak videófájlok (például ***mpg, avi***) lejátszására alkalmas programok (például ***aviplay***)

- Mindkét formátumot ismerő programok (például ***MPLayer, Xine, VLC***)

Ebben a cikkben a mindkét formátumot támogató (multimédia) lejátszók csoportjának egyik tagjáról a ***VLC***-ről lesz szó.

A VideoLAN Client

A VLC (VideoLAN Client) egy multiplatformos multimédia-lejátszó program. Hivatalos weboldala a ➲ ***http://www.videolan.org/vlc***, ahonnan az alkalmazás letölthető, illetve a ➲ ***http://www.videolan.org/doc*** oldalon egy nagyon részletes dokumentációt találunk több nyelven.
A támogatott platformok (operációs rendszerek) a következők: ***Windows95/98/ME/2000/XP/CE, Linux, MacOSX, BeOS, FreeBSD, OpenBSD, NetBSD, Solaris, QNX.***
Előregyártott bináris csomagok (precompiled binaries) léteznek ***Windows, MacOSX, BeOS, Debian GNU/Linux, Mandriva Linux, Fedora Core, Familiar Linux, YOPY/Linupy, Zaurus, SuSE Linux, Red Hat Linux, WinCE/PocketPC*** rendszerekhez.
Például ***Debian GNU/Linux*** alatt ha egyszerűen kiadjuk az

```
apt-get install vlc
```

parancsot, akkor a ***VLC*** automatikusan letöltődik és feltelepül gépünkre.

Rendszerek, amelyekhez nincs előregyártott csomag, de támogatottak: ***FreeBSD, OpenBSD, NetBSD, Solaris, QNX, Gentoo Linux, Crux.***
Az alkalmazás jelenlegi verziószáma 0.8.4-test1 beta. Amennyiben rendszerünkhöz nem áll rendelkezésre a bináris csomag, a ➲ ***http://www.videolan.org/vlc/download-sources.html*** oldalról letölthetjük a forráskódot.
A fordításhoz szükségünk lesz az alapvető ***gcc*** és ***make*** programokra, valamint rendelkeznünk kell a következő függőségekkel: ***slang***, ***gtk*** és egyéb függőségei, ***wxgtk*** és egyéb függőségei.
Miután letöltöttük a forráskódot, majd kicsomagoltuk és beléptünk a vlc könyvtárba, a

```
./configure
make
make install
```

parancshármassal fordíthatjuk le/telepíthetük a kódot a gépünkre.
Ennyi bevezető után lássuk, hogy miket is tud a ***VLC***.

Bemeneti források

A VLC a mai kor elvárásainak megfelelően rengeteg bemeneti forrást támogat (***UDP, RTP, HTTP, FTP, MMS, audio/vidao fájl, EVE, VCD, SVCD, DVD, Audio CD, V4L***-kompatibilis eszközök.). A média állományokon kívül lehetőségünk van ***Video CD*** (***CD***-n tárolt filmek), ***Super Video CD***, ***DVD***, ***Audio CD*** (zenei ***CD***) lejátszására is.
Adást sugározhatunk az ***UTP/RTP*** támogatásnak köszönhetően. ***Broadcast*** adásokat (például ***fix.tv Linuxportál*** című műsor) kísérhetünk figyelemmel, ezen felül a Web-, és FTP helyeken tárolt médiafájlok direkt elérése is biztosított, mely szélessávú internetkapcsolat mellett felettébb hasznos/kellemes lehet.
A webkamerával rendelkező felhasználók külön értékelni fogják a ***V4L*** (***Video for Linux***) támogatást.

Bemeneti formátumok

Ez a lejátszó mindent támogat! – gondolhatja mindenki, ha a támogatott formátumok impozáns listájára tekint (***.mpg, .mpeg, .mp3, .avi, .asf, .wmv, .wma, .ogg, .ogm, .mp4, .mov, .3gp, .wav, .dv, .aac, .ac3, .a52, .flac***).
Bizony, a Windows saját médiaformátumaitól kezdve (***wmv*** = ***Windows Media Video***, ***wma*** = ***Windows Media Audio***, ***asf***), a ***Quicktime***-on (***mov***); a telefonok által készített videófájlokon (***3gp***) és a különböző audió formátumokon át (***ogg***, ***flac***, ***mp3***, ***wav***, ***ac3***, stb) a szabványos videóformátumokig (***mpg***, ***avi***) szinte minden van itt, ami egyáltalán előfordul a gyakorlatban.
Néhol akadnak apróbb gondok (én például már láttam olyan ***wmv***-t, ahol csak hangot játszott le kép nélkül), de összességében nézve minden formátumot korrekt módon kezel.

Video codec-ek

A lista itt is impozáns: ***MPEG-1, MPEG-2, DivX 1 / 2 / 3, MPEG-4, DivX 5, XviD, 3ivX D4, Sorenson, DV, Cinepak, Theora, H263/H263i, MJPEG, WMV 1/2, IV32***.
Mindegyik kodeket eleve tartalmazza a program, vagyis nem kell azokat külön letölteni és feltelepíteni.
Ezek biztosítják azt, hogy a lejátszó minél több médiaformátumot tudjon kezelni. A ***DivX*** fájlok, illetve a ***DVD-k*** lejátszásához már egy erősebb gépre van szükségünk. A mostanság kapható gépeken bármilyen ablakkezelővel akadásmentesen futnak.
Ha régebbi (***p2-p3***) géppel rendelkezünk, akkor érdemes minimum ***256 MB*** memóriát használnunk és kis erőforrásigényű grafikus felületet (***Blackbox, Fluxbox, Enlightenment***, stb.) választanunk.

Feliratok

A feliratok kezelése még erősen fejlesztési stádiumban áll, de a támogatott feliratformátumok (***DVD, SVCD, CVD, DVB, OGM, Matroska, MicroDVD, Vobsub, SubRIP, SubViewer (v1 & v2), SSA1-4, SAMI, Vplayer)*** előre sejtetik, hogy mit hoz a jövő.
Ha egy felirat nem töltődik be rendesen, az azért van, mert az alkalmazás fejlesztése még gyerekcipőben jár.
A következő verzióban sok mostani hiba javítva lesz.

Videószerkesztés

Lehetőségünk nyílik az összes létező szerkesztési trükköt (***Összefűzés, Darabolás, Képfal (image wall), Fényerő-Telítettség-Kontraszt, Forgatás, Fejjel lefelé, Logo elhelyezés)*** véghezvinni a videófájljainkon.
Céges, vagy egyéb logókat helyezhetünk el, a túl nagy állományokat feldarabolhatjuk kisebbekre, vagy éppen egy ***DVD***-rip-et vághatunk fejezetekre.
Természetesen fordítva is eljárhatunk: sok kis videót fűzhetünk egybe, akár fejjel lefelé is.

Videókimenetek

Rengeteg kimeneti típus van támogatva: ***natív, X11, Xvideo, SDL, Framebuffer, ASCII Art, MGA, GGI***.
Filmjeinket pixelek helyett karakterek kirajzolása segítségével is nézhetjük az ***ASCII Art*** funkciónak köszönhetően (ez amúgy egy érdekesség, amit mindenkinek ajánlok, legalább egy-

szer). Bármilyen grafikus felület alatt használhatjuk a VLC-t, az *X11* és az *SDL* támogatás miatt nem fogunk akadályba ütközni.
A framebuffer támogatás lehetővé teszi, hogy *X* környezet nélkül, konzolon is élvezzük filmjeinket.

Audio codec-ek

Ha van egy olyan audió állományunk, amelyet eddig még semmivel sem sikerült lejátszanunk, akkor a *VLC* nagy eséllyel tudni fogja. Én legalábbis nem nagyon ismerek olyan audió formátumot, amit az audio kodekek listája (*MPEG Layer 1 / Layer 2, MP3, AC3/A52, DTS, LPCM, AAC, Vorbis, WMA 1 / 2, ADPCM, DV Audio, FLAC, MACE, Speex)* nem tartalmazna. Nagy előnye az alkalmazásnak, hogy kompakt, ezért nem kell hetvenhét különböző funkciójú programot használnunk minden egyes dologhoz. A *VLC* mindet egyszerre tudja.

Audio kimenetek

Ahogy már ezt a *VLC* esetében megszokhattuk, a támogatottsági lista (*natív, S/PDIF, multi-csatorna (OSS/ALSA), SDL, ESD, aRts, fájl)* itt sem szűkölködik összetevőiben.
Aki használt már *Licq*-t, az tapasztalhatta, hogy ha filmnézés közben írtak neki a partnerei üzenetet, akkor azok nem adtak hangot annak ellenére, hogy a hangfunkciók nem voltak letiltva.
Mivel addig a film „fogta le” a hangeszközt, ezert az *Licq* hangjai csak a film lejátszásának végeztével egyszerre szólalhattak meg.
Az *OSS/ALSA* rendszer (➲ *http://www.alsa-project.org/*) a *2.6-os Linux kernel* (➲ *http://www.kernel.org*) része, ráadásul a legújabb *ALSA* már támogatja a sokcsatornás megszólaltatást. Tehát, mint annak idején *Windows-ban*, most *Linux* alatt is megtehetjük azt, hogy filmnézés közben például zenét hallgassunk.

Felületek

A grafikus felületek *Linux* alatt különböző motorral működnek. A *KDE Qt*-t, míg a *Gnome GTK*-t használ (hogy csak a két legnagyobbat említsem).
Meg kell jegyeznem, hogy *Qt*-t csak a *KDE*-s alkalmazások használnak, minden más ablakkezelő a *GTK*-t preferálja egységesen.
A *VLC* illeszkedik mindkét nagy családba megjelenésében (*natív, GTK+, Gnome, QT, KDE, WxWidgets, HTTP/Weblap)*, illetve akár weblapként is megjeleníthetjük.

Egyéb jellemzők

Hihetetlen mennyiségű egyéb szolgáltatást tartalmaz a program: *Parancssor támogatás, Forróbillentyű (hotkey) támogatás, RTSP kliens támogatás, Távirányító támogatás, Infrared támogatás, Mozilla Plugin, DVD-menük támogatása, SVCD-menük támogatása, Audio vizualációs effektek, Lokalizáció, ID3-tagek támogatása, CD-TEXT támogatás, CDDB adatbázis lekérdezési lehetőség, IPv6 támogatás, IGMPv3 támogatás*.
Nézzünk néhányat részletesen.
Nem szükséges felhasználói felületet használnunk, parancssorból is kitűnően kezelhető alkalmazásunk. Ezekhez segítségünkre lehetnek a

```
man vlc
```

illetve a

```
vlc -help
```

parancsok.
Egeret csak végső esetben kell használnunk, mivel a teljes vezérlést billentyűkkel is végezhetjük (ezeket természetesen szerkeszthetjük is).
A *Mozilla Plugin* lehetővé teszi, hogy böngészés közben élő kamerák által sugárzott adást nézzünk (gondoljunk csak a valóságshowkra).
A *DVD* és *SVCD*- menük teljesen élethű kezelése is biztosított
Zenefájljainkat *ID3-tag*ekkel láthatjuk el (*előadó, albumcím, évszám*), illetve ha ezeket fejből nem tudnánk, akkor lehetőségünk van kapcsolódni egy *CD-adatbázishoz* az *Internet*en, ahonnan ezen információkat lekérdezhetjük.
A fenti nagyon bő és részletes listából láthatjuk, hogy egy igazán full-extrás programmal hozott minket össze a sors. Szinte mindent tud, amit egy multimédia lejátszónak tudnia kell, sőt, még annál is többet.
Alapjáraton egy nem túl tetszetős felhasználói felület (*skin*) fogad bennünket. Ettől nem kell megijedni, hiszen bármikor lecserélhetjük egy másikra (lásd a kapcsolódó címeket). Természetesen saját felületek készítésére is van lehetőségünk.
A *DVD*-menüket nagyon szépen kezeli, ezáltal *DVD* lejátszásra szerintem alkalmasabb a szintén kiváló és magyar fejlesztésű *Mplayernél*, ami köztudottan nem támogatja a *DVD*-menüket.
Broadcast adások vételére remekül használható, stabil, masszív, könnyen kezelhető, megbízható. Néhol a feliratokat nem kezeli tökéletesen.
Összességében mindenkinek csak ajánlani tudom ezt a programot. Kellemes filmnézést és zenehallgatást!

Apagyi György, (killall)
(user@killall.eof.hu,
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).

KAPCSOLÓDÓ CÍMEK

- ➲ http://www.videolan.org/vlc/download-skins.html
- ➲ http://www.videolan.org/vlc/download-skins2.html
- ➲ http://www.videolan.org/vlc/skins-create.html

Kiadványszerkesztés Linux alatt – Scribus (1. rész)
Bevezetés, alapok

A Linux alatti kiadványszerkesztés évekkel ezelőtt csak elképzelhető volt, de megvalósítható még nem, legalábbis nem egyszerűen. Egy, a Quark hiányát kiváltó szoftver megjelenése és erőteljes fejlesztése/fejlődése a képzeletből a valóság talajára vezette a DTP-vel Linux alatt foglalkozni szándékozók hadát.

Jelen írásomban a német származású ***Scribus*** (nyílt forrású ***GNU GPL***) telepítésén és egyszerűbb beállításán túl, egy könnyebb példán – egy oldalpár elkészítésén – keresztül mutatom be használatát, és a hozzá kapcsolódó funkciókat.
A szedés és a helyesírási szabályokra nem térek ki, akiket mélyebben érdekel a kiadványszerkesztés, azoknak a cikk végén található irodalomjegyzék és link gyűjtemény jó kiindulási alapként szolgálhat.
Egy kis információ az általam használt számítógépes konfigurációról. ***AMD 2600+ Barton*** magos processzor, ***512 MB RAM***, 19" monitor. A rajta futó operációs rendszer az ***UHU Linux 1.2***, amely a telepítőlemezeken szállítja a ***Scribus 1.2***-es változatát. Itt jegyezném meg, hogy a kiadványszerkesztés egyik legfontosabb hardvereszköze a monitor, ezért érdemes ebből egy jobb minőségű és nagy méretű példányt beszerezni. Meg fogja hálálni.

Telepítés

A ***Scribus*** a honlapjáról letöltött legutolsó forráscsomag (***1.3.0***) kibontása után az ismert

```
./configure
make
make install
```

csodatévő hármasparancs kiadásával telepíthető. Igaz, hogy az ***UHU*** tartalmazza az ***1.2***-est, de én mégis a legújabb bemutatását választottam (hibajavítások és új funkciók miatt).
Kényelmesebb megoldás a használt disztribúció csomaglelőhelyeinek átkutatása, amelyek szinte mindegyike tartalmazza a lefordított, függőségekkel ellátott és behálózott csomagot.
A forrásból történő telepítés némi időbe kerül, az én gépemmel úgy 30-35 perc volt. Ezt követően a

```
scribus
```

parancs kiadásával indítható is a program.

Beállítás

Mielőtt elkezdenénk a munkát, érdemes a szerkesztést megkönnyítő beállításokat elvégezni, átnézni. Ezeket tekinthetjük az elkövetkezendő munkák állandó/alapértelmezett értékeinek is. Ezt az ***Edit*** menü ***Preferences*** almenüjében tehetjük meg (*1. ábra*).
A fontosabb tulajdonság lapokat táblázatokba gyűjtöttem össze.

■ ***1. ábra*** Itt történik a Scribus finomhangolása

A General menü

Előfordulhat, hogy az indítást követően a ***Scribus*** kevert magyarsággal szólal meg, ez a nem teljes fordítás eredménye. Akit zavar a program kétnyelvűsége (magyar-angol vegyes használata), a ***Language*** pontnál átállíthatja angolra. Én ezt a beállítást fogom használni a szoftver bemutatása során.
ICC (International Color Consortium) Profiles, ez nem más mint a munkaállomás és a levilágító közötti közös hangot (vagyis színt) megteremtő színprofil. A ***Scribus*** honlapján megtalálható három profil, érdemes ezek mindegyikét felmásolni a megadott elérési útba. Fontos, hogy legalább egy ***RGB*** és egy ***CMYK*** profil fent legyen. A monitorok, nyomtatók telepítőlemezein is érdemes körülnézni, legtöbbjükön megtalálható a színprofilfájl, ami a még pontosabb szín visszaadást hivatott elvégezni.

Document menü

Ez a rész egy kicsit hasonlít a szövegszerkesztők beállításaira. Ami eltérést mutat, az a margók kezelésében található meg. Oldalpár készítése esetén a bal- és jobboldali margók jelentése és viselkedése megváltozik kötés és vágás felőli margóra. Ilyenkor egyszerre két egymást követő oldal szerkesztésére nyílik lehetőség (kinyitott újság vagy könyv látszatát kelti). Ebben az esetben felkínál egy plusz lehetőséget, hogy melyik oldalon

1. táblázat *A General menü tartalma*

GUI	Grafikus felhasználói felület
Language	A használni kívánt nyelv
Theme	Felhasználói felület megjelenése
Font Size	A felületen látható betűk mérete
Wheel Jump	Hány sort gördítsen az egérgörgő
Paths	**Elérési utak**
Documents	A dokumentumok helye
ICC Profiles	Színprofilok helye
Scripts Python	Szkriptek helye
Documents Templates	Ideiglenes tároló

2. táblázat *Document menü tartalma*

Page Size	A lap mérete
Size	Az oldal mérete
Orientation	Az oldal tájolása
Units	A használni kívánt mértékegység
Width és Height	A tetszőleges (Custom) oldal mérete
Facing Pages	Oldalpár készítése
Left Page First	Az első oldal baloldalt lesz
Margin	**Guides Margók**
Top	A felső margó
Bottom	Az alsó margó
Left/Inside	A baloldali vagy kötés felőli margó
Right/Outside	A jobboldali vagy vágás felőli margó
Autosave	Automatikus mentés és annak gyakorisága
Undo/Redo	Művelet visszavonás mélysége

(bal vagy jobb) lesz az első oldal.
A példában ezt a szerkesztési módot fogjuk áttekinteni.
Mértékegység, itt számos a nyomdászatban használatos egységet is megtalálunk ***Pica-pont (pt)***, ***pica (p)***, ***cicero (c)***. Használatuk előnye a könnyebb számolás, a méretek egyszerűbb meghatározása. Alapértelmezésben a ***pt*** egység áll (ez maradhat is), ***1pt=0,353mm=0,08p***, vagyis ***1p=4,233mm=12pt***.

A Guides menü (segédvonalak)

A szerkesztést megkönnyítő/segítő vonalak és hálók színeinek, egymástól való távolságának beállítását szolgálja ez a lap. Az alapvonal pontos beállítása kihat az egész dokumentum megjelenésére. Rossz értékek megadása esetén megkeseríthetí a ***DTP***-s életét.
Az alapvonalat úgy kell elképzelni, mint a vonalas füzet vonalait, amik az egyenes sortartásért (hasáboknál és oldalpár szerkesztésnél látszódik leginkább ennek előnye) – más néven soregyenért – felelnek.
Az alapvonalat érdemes láthatóvá tenni, ha szerkesztés közben mégis zavaró a jelenléte, akkor a ***View*** menüben ki lehet kapcsolni, de ez igaz számos más szerkesztési segédeszközre is.
Nézzük át a beállítását! Érdemes a vonalak közötti távolságot olyan mértékre állítani, ami a kenyérszöveg betűtörzs méretének maradék nélküli osztásának eredménye (általában a ***3pt*** jó szokott lenni). Egy másik módszer, amikor az alapvonalak közötti távolság a betűfokozat ***10-20%***-kal növelt értéke, például ***10pt***-os betűnél ***12pt***-os a sortávolság (ezt így jelölik: ***10/12***). Most állítsuk be ***12pt***-ra.
Az alapvonal eltolását állíthatjuk a felső margó értékére is (***40pt***), de ajánlatosabb a 0pt-on hagyása. Miért is? Minden új dokumentum elkezdésénél megadhatjuk külön a margók értékeit, mivel a ***10pt***-os betű és a ***12pt***-os sortávolság a legtöbb kiadvány kenyérszövegéhez megfelelő, ezért a margók és ezzel együtt a laptükör méretezésénél ezt figyelembe tudjuk venni. Vagyis a felső margót a ***12pt***-os sortávolság többszörösére állítjuk (***12***, ***24***, ***36*** stb.). És itt jön a képbe a fent említett ***3pt***-os sortávolság, ami még nagyobb szabad kezet ad a méretezésben. Inkább állítsuk ***3pt***-ra.

Typography – Tipográfia

Egyéb tipográfiai jellemzők, az indexbe kerülő karakterek eltolásának és méretváltoztatásának, karakter vonalazás eltolásának és vonalvastagságának állítása. Kiskapitális, kisbetűk amelyek kinézetre megegyeznek a nagybetűkkel, de azoknál – a megadott százaléknak megfelelően – kisebbek. Az automatikus sorköz értékének hatására a sorköz ennyivel lesz nagyobb a használt betűméretnél. Indexek eltolásához (fel vagy le) a ***33%***-ot, betűméret változtatáshoz ***75%***-ot javaslom. Az automatikus sorközhöz pedig a ***20%***-ot.

Új dokumentum

A munkát az elkészítendő oldalpár jellemző laptulajdonságainak megadásával, a ***File*** menü ***New*** pontjával kezdjük. Itt ugyanazokat a beállításokat találjuk meg, mint az előzőekben bemutatásra került ***Preferences*** menüben. Felajánlja a lehetőséget az esetleges módosításokra, ha a készülő dokumentumot más tulajdonságokkal szeretnénk felruházni. Újdonságként az oldalak száma (***Number of Pages***) és azok számozásának kezdete (***First Page Number***), valamint a szövegdoboz(ok) automatikus létrehozása, hasábozása (***Automatic Text Frames***)

tűnik fel. (Amire nem térek ki, azt nem kell átállítani, az a jelen feladathoz jó beállítással bír.)
Kezdjük a felső margóval a sortávolság (***3pt***) korábban beállított értékének tudatában ez legyen ***63pt***. Az alsó margó meghatározásánál már az alkalmazott betűhöz tartozó sortávolságot vegyük figyelembe (***10/12***). Itt egy kis számolgatást kell eszközölni, ennek eredménye ***46,89pt*** lesz. A vágás (belső) oldali margó ***48,19pt***. A külső pedig legyen ***31,18pt***. Ezek az értékek amúgy nem hasraütésszerűen alakultak ki, hanem a ***Linuxvilág*** margóméreteit vettem alapul.
Kattintsunk az ***OK***-ra. Ha nagyítóval megnézzük az alsó- és felső margókat, látható, hogy szépen rajtaülnek az alapvonalakon (*3. ábra*). Ez azért jó, mert így tölti ki a szöveg normálisan, teljesen, különböző optikai torzítások igénybevétele nélkül a szövegtükröt. Azt hiszem megér ez egy kis számolgatást, mert utána nem kell küszködni az oldal tördelésénél.
A jelen feladathoz szinte elegendő is lesz az egér jobb gombjának, kattintásra előugró helyi menüjében található szerkesztési eszközök használata (*4. ábra*).

Szöveg betöltése/Get text

Külső forrásból illesztjük be a szöveget (a helyesírás-ellenőrzést érdemes már előtte elvégezni, akárcsak az egyéb javítgatásokat, a szövegkorrektúrázást), formátuma lehet: ***OpenOffice.org***, ***Text*** fájl, ***HTML*** fájl. Formázott dokumentumok esetén figyelembe veszi azok beállításait és lehetőséget ad azok átvételére vagy figyelmen kívül hagyására.
Az ***Overwrite Paragraph Styles*** a meglévő stílusokat átírja a betöltött dokumentummal hozottakra.
A ***Merge Paragraph Styles*** kibővíti a meglévő stílusokat a betöltött dokumentummal hozottakkal, míg a ***Use document name as...*** a betöltött dokumentum nevét előtagként hozzáadja a behozott stílusok nevéhez, a könnyebb megkülönböztethetőség miatt.
Ahogy a képen is látszódik semmit sem jelöltem be, mert ki tudja milyen beállításokat hoz magával a szöveg, valamint a stílusok tárgyalására most nem kerül sor.
A sikeres betöltés után valami hasonlót kell látnunk a monitoron (*6. ábra*).

3. táblázat *A Guides menü tartalma*

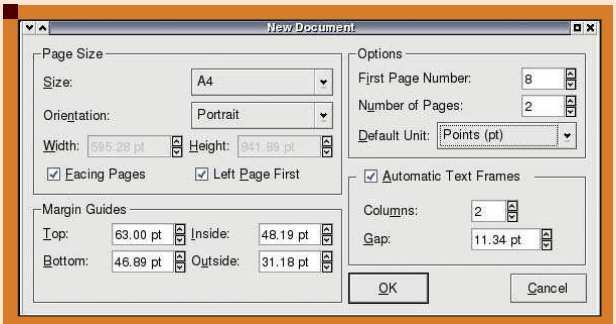
Common Settings	Általános beállítások
Placing in Documents	A dokumentum előterében vagy hátterében helyezkedjenek el
Snapping	„Mágnesesség"
Show Guides	Segédvonalak mutatása
Show Margins	Margók mutatása
Show Page Grid	**Lapháló**
Major Grid	A nagy háló
Minor Grid	A kis háló
Show Baseline Grid	Alapvonal mutatása
Baseline Settings	**Alapvonal beállítása**
Baseline Grid	Az alapvonalak közötti távolság
Baseline Offset	Az első alapvonal távolsága a lap tetejétől

4. táblázat *A Typography menü tartalma*

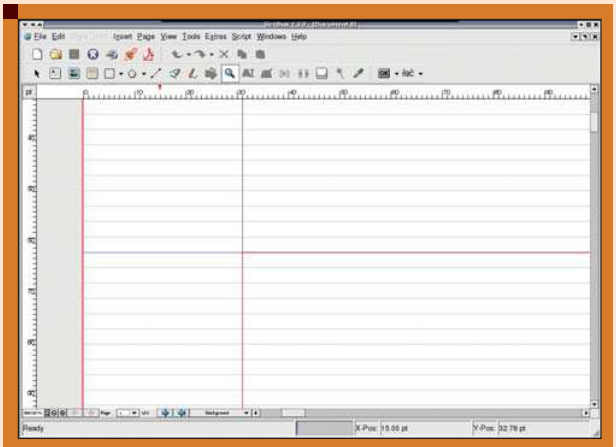
Subscript	Alsó index
Superscript	Felső index
Underline	Aláhúzott
Strikethru	Áthúzott
Small Caps	Kiskapitális
Automatic Line Spacing	Automatikus sorköz

5. táblázat *A New Document menü tartalma*

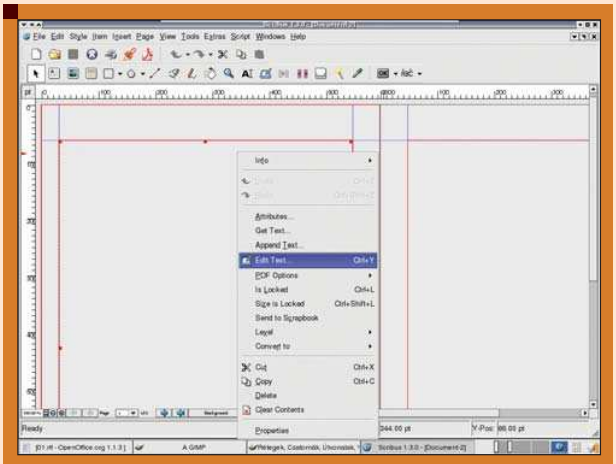
Number of Pages	Ennyi oldalból fog állni a dokumentum.
First Page Number	Az első oldal oldalszáma. Előfordulhat, hogy egy dokumentumot két vagy több részből állítanak össze – megosztva a munkát vagy bármi más okból kifolyólag –, amit a szerkesztés után fűznek össze egy állományba (létezik egy nagyon jó PDF összefűző program: PDF toolkit).
Automatic Text Frames	Ezt érdemes bejelölni és a beállított margóknak megfelelően elkészíti a szövegdobozt vagy – dobozokat.
Columns	A hasábok száma. A példában 2 hasábos tördelést alkalmazunk.
Gap	A hasábok közötti távolság, általában 11,34pt (4mm).
Margin Giudes	A margók segítségével határozhatjuk meg a szedéstükör méretét. A tükör tartalmazza a szöveget, képeket, ábrákat, címeket stb., de az oldal alján és tetején, esetleg szélén található kiegészítéseket már nem. Ilyen kiegészítés lehet: oldalszámozás, indexfül, díszítő csíkozás, feliratozás.



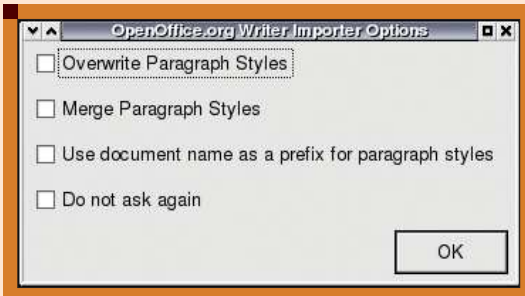
2. ábra Új dokumentum létrehozása



3. ábra Margó és az alapvonal találkozása



4. ábra Szerkesztési eszközök menüje



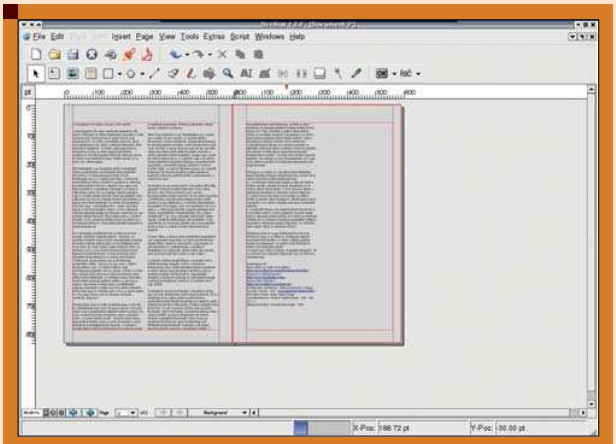
5. ábra OpenOffice.org dokumentum importálható tulajdonságai

A szövegdobozok kijelölését (***CTRL + A***) követően módunk nyílik a betűk egységes típusra, méretre, sortávolságra állítására és még számos más egyébre is (*7. ábra*). Ezt az ablakot a jobb egérgombra felbukkanó helyi menü ***Properties*** pontját választva csalhatjuk elő. Innen most minket a ***Text*** fül mögött megbújó tartalom érdekel. Kiválaszthatjuk a betűtípust, változatot, méretét, sortávolságát, szélességének és magasságának százalékos arányát, alapvonaltól való eltolásának mértékét, színét, és speciális alkalmazásait (többek között jobbról-balra írás, körvonal, kiskapitális stb.). Szintén itt adható meg a sorzárás (balra, középre, jobbra, sorkizárt és teljesen sorkizárt), a stílus szerkesztőben előzőleg elkészített stílus, valamint a szóelválasztó készlet nyelvezete. Ha mindent beállítottunk a megfelelő értékre (sorkizárt, ***Nimbus Roman No9 L Regular***, ***10/12***), akkor a szövegben előforduló „nem odaillő” karaktereket ki kell cserélni „odaillőkre”.
Ez jól látszik a mellékelt képen (*8. ábra*), ahol is vörös négyzetek bíztatják a szerkesztőt a mielőbbi cserére.

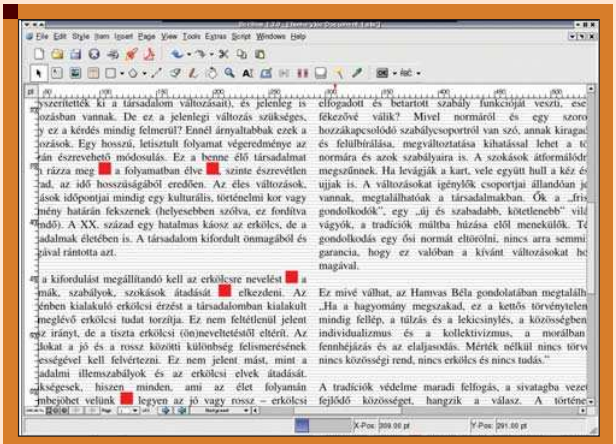
Esetünkben a gondolatjelek hiányoznak, amik jelben megegyeznek a nagykötőjellel (***en dash***), de írásban másként alkalmazzák (nem tapad az előtte és utána lévő szavakhoz). Ilyen speciális jeleket az ***Insert*** menüben találunk (***Character, Quota, Space***). Állítsuk a kurzort az egyik cserélendő elem elé és szúrjunk be egy gondolatjelet, majd jelöljük ki mind a kettőt és másoljuk a vágólapra (***CTRL + C***). A kijelölést megszüntetve töröljük a hibás karaktert. Az ***Edit*** menü ***Search/Replace*** pontjával lehet keresni/cserélni karaktereket, betű típusokat és más formázási elemeket. A ***Text*** szövegkeretekbe kell beilleszteni (***CTRL + V***) a kimásolt két karakter egyikét és másikát, majd a ***Search*** és azt követően a ***Replace/Replace All*** gombok használatával elvégezhetjük a cseréket (*9.ábra*).

Címsor

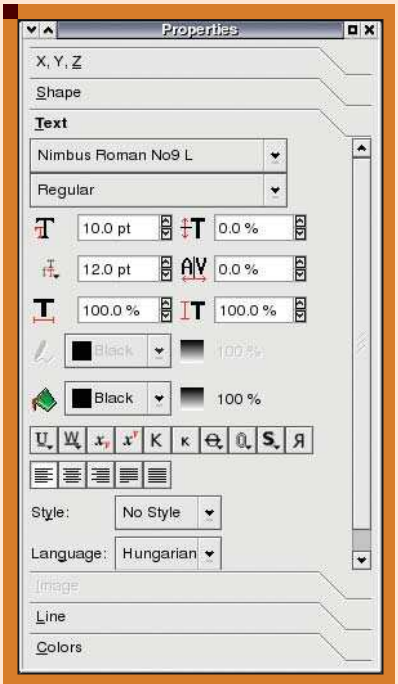
A címsort egy külön szövegdobozba helyezzük, ami átöleli a két hasábot, mérete legyen ***101,98pt***. Az ***Insert*** menü ***Text Frame (T)*** pontját választva rajzolhatjuk meg a dobozt, a szélességét és elhelyezkedését igazítsuk a nagy szövegdobozhoz, a magasságát pedig a ***Properties*** ablak ***X,Y,Z*** fülénél írjuk be (itt módosíthatjuk a méreteket, pozíciókat). A szöveg befolyt a doboz alá! Ezt a ***Properties*** ablak ***Shape*** fülének ***Text Flows Around Frame*** bejelölésével szüntethetjük meg. Másoljuk be a címet a dobozba és a ***Text*** fülnél a szöveg formátumát állítsuk balra zártra, ***Nimbus Sans L Bold***, ***22/26,4***-re, az alcímnek maradhat ez a típus, csak a többi tulajdonságát változtassuk meg: ***Regular***, ***14/26,4pt*** méretű.
Térjünk vissza a ***Text Flows***-hoz! Itt található további két pont, ami a doboztól való külső szövegeltartást hivatott állítani.
A ***Use Boundig Box*** egy második keret segítségével állítja be az eltartást, míg a ***Use Contour Line*** egy második vonal segítségével állítja be az alapvonalat. A *10. ábrán* jól látszódik az eltartásért felelős kék, vékony vonalú keret, és a ***Nodes*** ablak (amit az ***Edit Shape***-pel hozhatunk elő). Kezdésként jelöljük



6. ábra Az oldalpár szövegtükrei a betöltött szöveggel



8. ábra Karakterek, amik nem „illenek oda"



7. ábra Tulajdonságok menü

be a ***Use Contour Line*** sort, majd nyissuk meg a ***Nodes*** ablakot. Itt válasszuk az ***Edit Contour Line***-t, az egérrel menjünk az egyik kék körre és jelöljük ki. Ekkor az ablakban állíthatóvá válnak az X és Y pozíciók. Állítsuk be a megfelelő értékekre majd menjünk az oldal másik felére és ott is játsszuk el ugyanezt, majd ***End Editing***. Ha jó értékeket állítottunk be (arra kell figyelni, hogy az első sor betűtalpai egy a beállított sorközre eső alapvonalra üljenek, például ***180pt***), akkor a lap alján így kell kinéznie a szövegnek és a két oldal közötti soregyen is rendben lesz (*11. ábra*).

A szövegdoboz belső eltartását a ***Distance of Text*** alatti mezőben állíthatjuk. Fentről húzzuk lejjebb egy sorral (***12pt***), a többi maradhat.

Iniciálé

Sajnos a ***Scribus***-ban még nincs külön iniciálé készítő, de ez nem is okoz különösebb gondot. Vegyünk elő egy szövegdobozt és rajzoljuk meg a kívánt helyre, az első bekezdés elejére. Magassága három sort fogjon át (***36pt***), a szélessége majd a szerkesztés közben kialakul. Írjuk bele a kezdő betűt – esetünkben az ***A***-t –, és állítsuk be a méretét (***26/31,2pt***), típusát (***Nimbus Roman No9 L Medium***). Az eddig még nem használt betű szerkesztő eszközöket most kipróbálhatjuk. Az alapvonaltól való eltolás ***-53%***-os, a vízszintes és függőleges méret növelés ***170%***-os legyen. Ha a doboz kellően szűk volt, akkor most semmi nem látszódik a betűből, csak egy kis négyzetes kereszt jelenik meg a jobb alsó sarokban. Ez azt jelenti, hogy van szöveg a dobozban, csak a megjelenítéshez túl kicsi a doboz. Növeljük meg a szélességét annyira, hogy látszódjon (34pt). Ja, és nem árt a körülfolyást is beállítani! (*12. ábra*)

Képek beillesztése

A legismertebb, legelterjedtebb formátumú (***TIFF***, ***JPEG***, ***PNG***, ***GIF***, ***XPM***, ***BMP***, ***PDF***, ***EPS***) képeket könnyedén illeszthetjük be a dokumentumba. Most két képet helyezünk be, egy egy hasábot és egy két hasábot átfogó képet. Ugyanúgy kell eljárni, mint a szövegdobozoknál, csak itt képek kerülnek a dobozba szöveg helyett. ***Insert*** menü ***Image Frame (I)*** pontját választva rajzolhatjuk is a képdobozt. Képdobozoknál a dobozok körül a hasábköznek megfelelő eltartást kell beállítani (***11,34pt***), ez esetünkben majdnem megfelel egy sornak (***12pt***). A dobozban látható hatalmas kereszt, a kép hiányát vagy kikapcsolt állapotát jelenti. Képet betölteni a jobb egérgombra előbukkanó helyi menü ***Get Image*** pontjával lehet, láthatóvá vagy láthatatlanná tenni pedig az ***Image Visible*** pont kapcsolgatásával. Állítható a felbontása is (alacsony, normál és magas), de ezek csak a megjelenítésre vonatkoznak. Az ***Edit Image*** hatására betölti a ***Gimp*** képszerkesztőbe és már alakítható is a kép. A mentést követően, hogy a képen történt módosítások biztosan bekerüljenek a dokumentumba az ***Update Image***-et választva a friss kép kerül betöltésre.
Az első képdoboz a dokumentum bal alsó részébe került (mérete ***252,28pt*** x ***180pt***) (*13. ábra*). A ***Properties*** ablak ***Image*** fülén a képre vonatkozó méretezések (nagyítás, kicsinyítés, képdobozhoz igazítás), valamint a színprofilok is megtalálhatók. A második képdobozt a jobb oldal közepére helyeztem (mérete ***515,91pt*** x ***252pt***). Az eltartásokhoz itt is alkalmazható ugyanaz az eljárás, mint a szövegdobozoknál.

Képfeliratok vagyis aláiratok

Ezt a műveletet szövegdobozokkal lehet elvégezni, a képek alá kell elhelyezni, megfelelő mértékű eltartásokkal. A dobozok mérete a szokásos sorköz többszöröse (***36pt***), betűből használhatunk valamilyen másik típust (***Eras Normal 10/12***), a doboz belső-felső eltartásának ***5pt***-ot állítottam be.

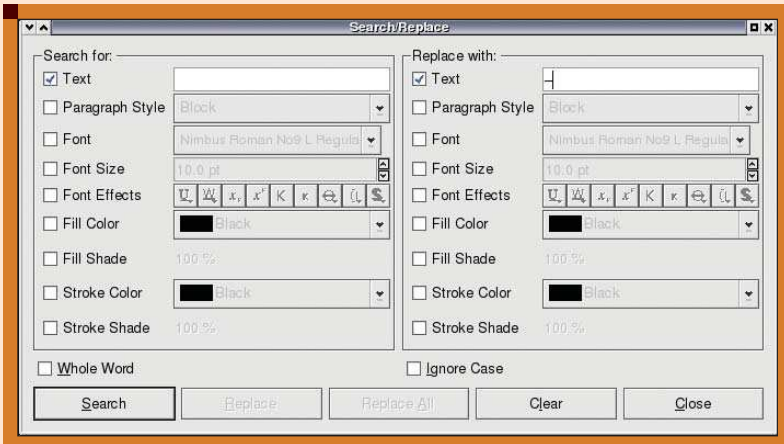
Elválasztás

Erről még nem volt szó, pedig az egyik legfontosabb része a tördelésnek, ha nem a legfontosabb. A ***Scribus*** rendelkezik beépített elválasztás kezelő és elválasztás megszüntetővel is. Ezeket az ***Extras*** menü alatt találjuk meg (***Hyphenate Text*** és ***Dehyphenate Text***). A dokumentum betöltése után, de akár szerkesztés közben is ráuszíthatjuk a szövegre az elválasztás kezelőt. Ennél a példánál én a képdobozok behelyezése után használtam. Mire jó az elválasztás megszüntető? Tördelés közben keletkezhetnek fattyú- és/vagy árvasorok, ezek nem kedvelt résztvevői egy kiadványnak (*15. ábra*). Ezek megszüntetésére lehet használni a ***Dehyphenate Text*** menüpontot, csak ki kell jelölni a kívánt részt, bekezdést és ha mindenki úgy akarja, akkor átcsúszik még egy-két sor és már jó is (*16. ábra*). Egy apró megjegyzés egy hibára, csak azon részt szabad így elválasztásmentesíteni, ami tartalmaz elválasztást, mert különben az egész szöveget mentesíti.

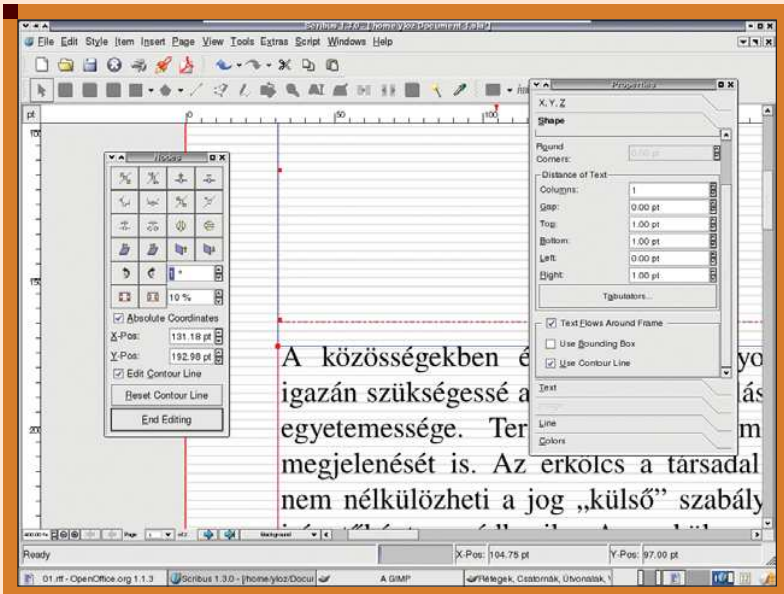
Oldalszámozás

Ismét előveszünk egy szövegdobozt és beállítjuk a méreteit (515,91pt x 36pt). Azért, hogy az oldalszámozás valóban oldalszámozás legyen, egy speciális karakter fogunk használni, ez megtalálható az ***Insert*** menü ***Character*** pontjában ***Page Number*** néven (***CTRL + ALT + SHIFT + P***). Azért érdemes ezt használni, mert állandóan követi az oldalszámváltozásokat (oldal beszúrás, kivágás stb.). Formázni ugyanúgy lehet, mint bármely más elemet. A másik oldalra a szövegdoboz átmásolásával és behelyezésével hozhatjuk létre a legegyszerűbben az oldalszámozást (csak a zárást kell átállítani).

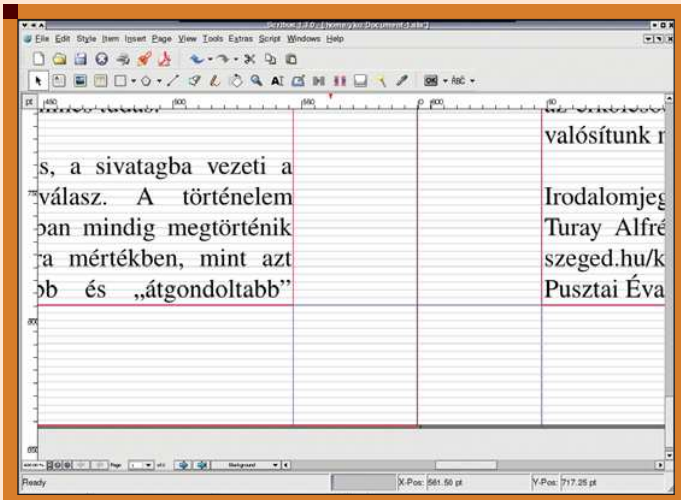
Ezzel el is készült kétoldalas kiadványunk. A dokumentumot elmenthetjük a ***Scribus*** saját formátumába (***SLA***), de exportálható más egyéb formátumokba is (***PDF***, ***EPS***, ***SVG***). Nyomtatni a ***File*** menü ***Print*** pontjában lehet. A kiválasztott nyomtató és annak tetszőlegesen beállított tulajdonságain kívül választható a több szín nyomtatás funkció is (***Print Separations***). Ennek lényege, hogy a négy szín (***CMYK***) külön lapokra kerül kinyomtatásra. Ez megfelelő gépen (lézernyomtató, legalább 600dpi)



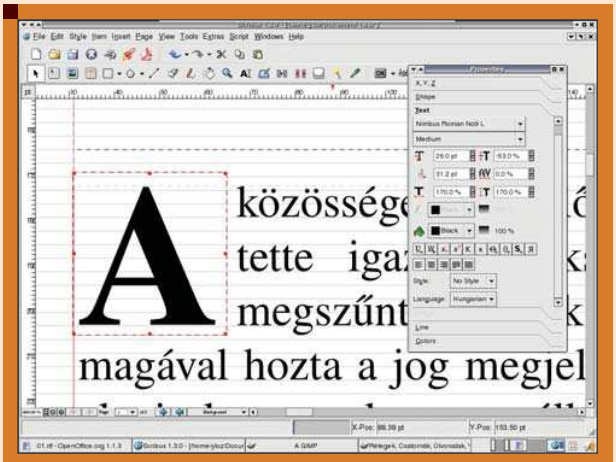
9. ábra Keresés és Csere menü



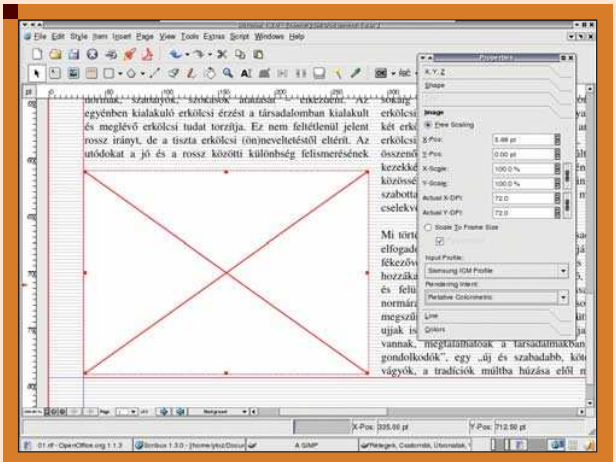
10. ábra Szöveg eltartás a szövegdoboztól



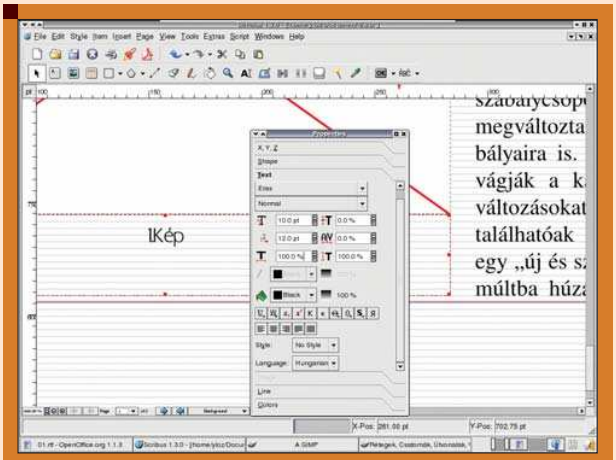
11. ábra Az a bizonyos oldalak közötti soregyen



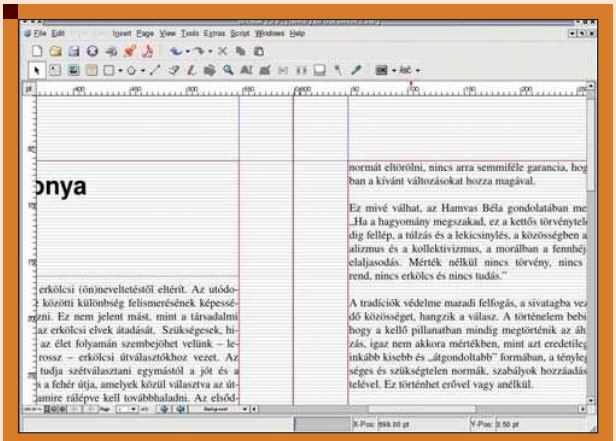
■ *12. ábra* Az elkészült iniciálé



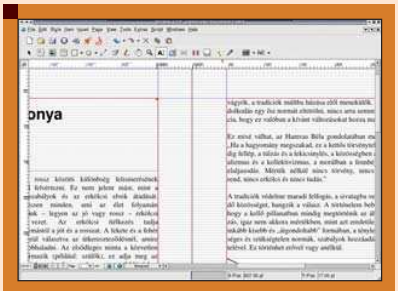
■ *13. ábra* Egy gyakran használt objektum: a képdoboz



■ *14. ábra* Írjunk aláiratot



■ *15. ábra* Dehyphenate Text előtt



■ *16. ábra* Dehyphenate Text után

és megfelelő hordozóra (pausz) nyomtatva az esetek többségében (szövegnél és vonalas ábráknál, de fényképeknél és árnyalat gazdag képeknél már nem) kiváltja a levilágítást, és csökkenti a költségeket.
És ha már pausz nyomtatásra adjuk a fejünket érdemes az ***Advanced Options***-ban a ***Mirror Page(s)*** ***Horizontal***-t bejelölni (vízszintes tükrözés). De figyelem! Ez csak pausz nyomtatásakor érvényes, ***PS*** írásakor nem szabad tükrözni, azt majd a levilágításkor elvégzik.
Úgy vélem, hogy a funkciók gazdagságát (sok minden nem került még bemutatásra: ***PDF*** kezelés, stílusok, stb.) tekintve egy igen jól fejlődő és használható kiadványszerkesztő programot tudhat magáénak a linuxos közösség. Azért, hogy a többiek se szomorkodjanak, a ***Scribus*** fejlesztői dolgoznak a mac-es és a windowsos átiratokon is. Ezekkel a különböző platformok közötti hordozhatóságot is biztosítanak.
A következő részben a ***Preferences*** menü további lehetőségeit valamint a ***Master Pages*** készítését, használatát nézzük át.

Lelovics Zoltán
(ylozmarr@freestart.hu)
Mindig is foglalkoztatott, hogyan lehet szemmel, kézzel vagy füllel „fogható” szépet alkotni számítógép segítségével, hogyan lehet mindezt összekötni és milyen lehetőségek kínálkoznak e cél eléréséhez.

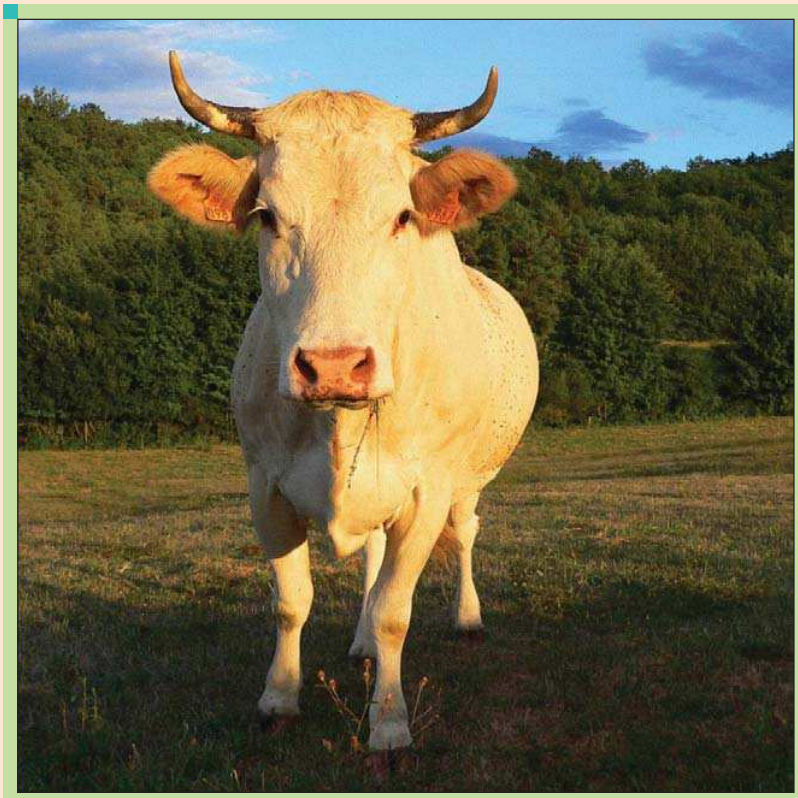
KAPCSOLÓDÓ CÍMEK

Scribus: ➲ http://www.scribus.org.uk/
Linkgyűjtemény: ➲ http://dtp.lap.hu/
Tipográl.hu: ➲ http://www.tipogral.hu/
PDF toolkit:
➲ http://www.accesspdf.com/pdftk/

Gimp bővítmények (1. rész)
A művészi szűrők

A Gimp, a legkedvesebb rajzprogramunk ma már rengeteget tud. Ebben a szoftverkategóriában természetesnek tűnik az is, hogy tudása kiterjeszthető, illetve testre szabható. A program alapját képező eszközöket, mint például a kijelölő és festő eszközöket számos másikkal egészíthetjük ki! Alapvetően két út áll előttünk ha újabb tudással szeretnénk felvértezni programunkat: vagy bővítményeket (plugineket) vagy szkripteket adhatunk hozzá.

Óriási különbség nincs a két megoldás között, azt azonban nem árt megjegyeznünk hogy míg a bővítmény (***plugin***) egy önálló futtatható program, ami bármire képes amire egy program képes lehet, addig egy szkript csak a ***Gimp*** által értelmezett parancsgyűjtemény, tehát valamelyest biztonságosabb. Ezért mindig figyeljünk arra, hogy kiterjesztéseket csak megbízható forrásból töltsünk le! Ilyen például a ***Gimp*** bővítmény adatbázisa (➲ ***http://registry.gimp.org/***).
Szerencsére a ***Gimppel*** együtt számos plugint, szkriptet kapunk, amiket azonnal használhatunk is! Gyakorlatilag minden, ami a ***Szűrők (Filters)*** menüpont alatt található, plugin, a szkripteket pedig a ***Python-Fu*** illetve a ***Script-Fu*** menüpont alatt találhatjuk. Az egyszerűség kedvéért összefoglaló néven kiterjesztésnek fogom hívni őket.
Vannak nagyon egyszerű és egyértelmű kiterjesztések, mint például a az elmosás szűrő, amivel lággyá, életlenné tehetjük a képet, de számos másik rengeteg beállítási lehetőséget tartalmaz. Nem árt ezért vetnünk rájuk egy pillantást, hiszen a nem túl részletes magyarázatok alapján gyakran nehéz megérteni mit és mire lehet valójában használni!
A továbbiakban egy-egy ***szűrő (filter)*** csoportot fogunk szemügyre venni.

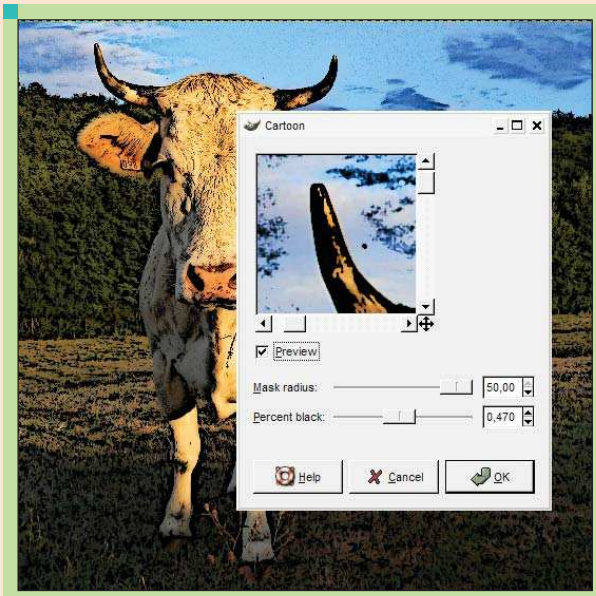


1. ábra A Vászon alkalmazása szűrő hatása

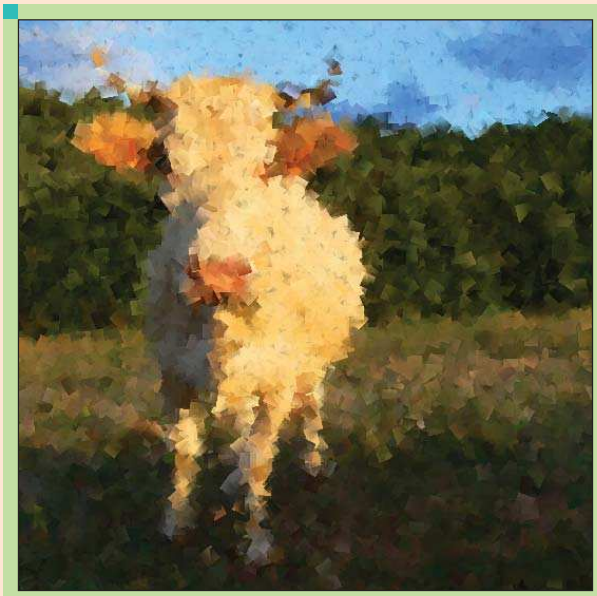
Elsőként vegyük szemügyre a művészi hatásúakat, melyek a ***Szűrők->Művészi*** menüpont alatt találhatóak! Angol nyelvű program esetén a ***Filters->Artistic*** alatt találjuk őket.

Vászon alkalmazása (Apply canvas)

Az első amit érdemes szemügyre vennünk, az a vászon alkalmazása szűrő. Sokszor szeretnénk utánozni a festett képek hatását, erre jó ez az ötletes eszköz, amely a képünket



2. ábra A képen jól látható, hogy csupán pár beállítással is érdekes hatást érhetünk el a Karikatúra szűrő segítségével



3. ábra A Kubizmus szűrő

úgy alakítja át mintha egy vásznon lenne. Láthatjuk a mintázatát, sőt beállíthatjuk, annak mélységét is. Az elölnézeti képen jól látszik, hogy minél nagyobb mélység értéket adunk meg, annál nagyobb kontrasztja lesz a vászon szálainak. Bár ezt az értéket 1 és 50 között állíthatjuk, 15 egység felett már túl intenzív a látvány ahhoz, hogy túl sokat alkalmazzuk. Az irányt is beállíthatjuk, amely meghatározza, hogy melyik sarokból kezdje el a program a vászon textúráját a képre helyezni. A lehetőségek között azonban nincs jelentős különbség.

Karikatúra (Cartoon)

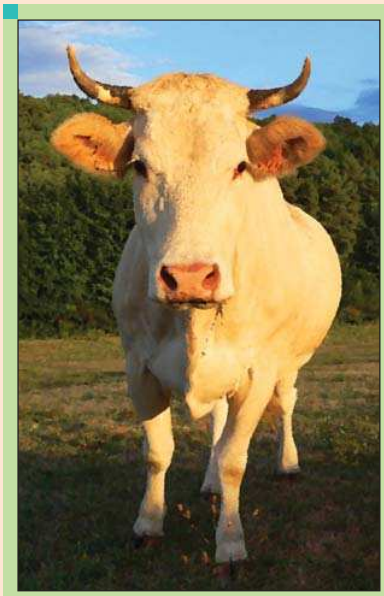
Vessünk egy pillantást a karikatúra szűrőre! Úgy találtam, hogy képregényszerűvé teszi a képünket ez a szűrő, mivel az éleket és a sötét körvonalakat kiemeli és feketévé festi át. Eredményül nem fekete-fehér képet kapunk, hanem megmaradnak a színek, de a kép elnagyolt, kontrasztos lesz, kiemelt körvonalakkal. Minél nagyobb maszk sugarat használunk, annál vastagabb, kontrasztosabb és feketébb éleket kapunk. A második csúszka azt szabályozza, hogy hány százaléka legyen feketévé alakítva a képnek. Minél nagyobb százalékot adunk meg, annál kevesebb részlet marad meg és annál stílusosabbá válik a képünk.

Kubizmus (Cubism)

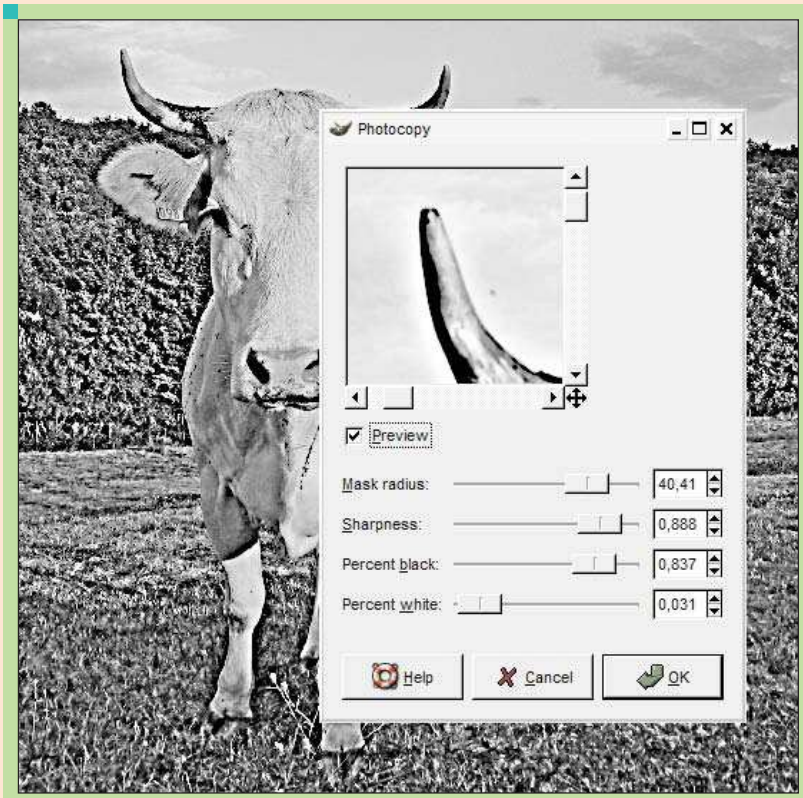
El is érkeztünk az első igazán művészi hatást mutató, kubizmus szűrőhöz. Az 1908 és 1912 között kialakult kubista művészet remekeivel ugyan nem fogják összekeverni a képeinket, de a stílus és a képi hatás elbűvölhet mindenkit akinek megmutatjuk az alkotást. Nézzük tehát milyen eszközt ad a kezünkbe a Gimp, ha Picasso nyomdokaiba szeretnénk lépni! A szűrő kicsiny csempékből rakja össze a képünket, úgy hogy minden csempedarab valamelyest átlátszó és a csempedarabkák egymással is átfedésben vannak. A csempék színe visszatükrözi az eredeti kép színeit. Mintha mozaikkockákból próbálnánk előállítani a képet, de a mozaikkockák félig átlátszóak és nagyobbak mint a rendelkezésükre álló hely, ezért aztán egymás elé, alá csúsznak és persze rendezetlenül elfordulnak a vízszinteshez képest. Beállíthatjuk a csempedarabkák méretét és színük telítettségét. A legtöbbször szinte lehetetlen felismernünk az eredeti képet a kubizmus szűrő alkalmazása után, viszont nagyon is tetszetős eredményt hoz létre.

Olajfestmény (Oilify)

Tradicionálisan művészi eszköz az olajfesték. Vajon miért ne próbálnánk ki mi is? Erre ad lehetőséget az olajfestés szűrő. Nincs sok beállítási lehetőség, vagyis pár pillanat alatt meg is tekinthetjük az eredményt. Ahogy a valóságban is, választhatunk a különböző nagyságú ecsetek közül. Szerencsére a *Gimp* megkönnyíti életünket, mivel nem kell több tíz különböző ecsettel rendelkeznünk, mindössze egy beállítást kell megváltoztatnunk. A beállítás a ***maszk méretének (mask size)*** beállítása jeligére hallgat. Nagy érték esetén kevésbé részlet gazdag eredményt kapunk, ugyanúgy mintha nagyobb ecsetet használtunk



4. ábra Valós festmények hatását utánozhatjuk az Olajfestmény szűrővel



5. ábra Fénymásoljunk virtuálisan a Fénymásolás szűrővel

volna. Választhatunk továbbá, hogy intenzitásos módszert szeretnénk-e használni a képünk feldolgozásához. Az intenzitásos módszer legfőbb előnye, hogy segít megőrizni a részleteket és a színgazdagságot.

Fénymásolás (Photocopy)

Az fenti szűrők segítségével mutatós képeket alkothatunk. Sok esetben azonban nem áll módunkban színesen sokszorosítani az alkotásainkat, hanem a fénymásolás mellett döntünk.
De vajon hogy fog kinézni az alkotásunk fénymásolva? Tegyünk egy próbát a fénymásolás szűrő segítségével!
A beállítási lehetőségek bőségesek és könnyű őket használni. Beállíthatjuk ismét a maszk méretét, mely a részletgazdagságot szabályozza. Ugyancsak meghatározhatjuk az élességet, aminek kis értéket adva gyönyörűen visszaadhatjuk az ősöreg fénymásolókon készült életlen fénymásolatok hangulatát! Fontos beállítás a fekete és a fehér aránya. Ezzel a kettővel játszva alakíthatjuk ki a nekünk megfelelő kontraszt és festék arányt. Ha tényleg fénymásolni szeretnénk alkotásunkat, célszerű kevesebb festéket használnunk a költségek leszorítása céljából, tehát hagyjuk alacsonyan a fekete arányát és vigyük fel a fehérét. Persze vigyázzunk, hiszen ha nagyon magas a fehér szín aránya a lapon, az azt is jelenti, hogy könnyen lehet hogy élvezhetetlenül kevés részlet fog látszani a végeredményen!

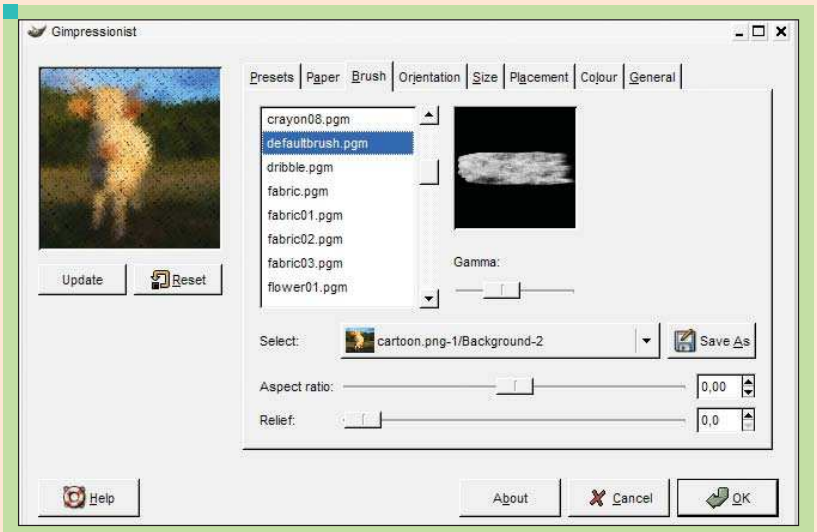
Finom ragyogás (Softglow)

Mindennapi fotókhoz elegendő eszköz áll rendelkezésünkre, de mi történik, ha egy angyal van a képen? Földöntúli ragyogást varázsolhatunk rá a finom ragyogás szűrő segítségével. A ragyogás a kép világos részeiből indul ki, tehát csaknem változatlanul hagyja a sötétebb részeket, hacsak nem egy világos rész mellett fekszenek, mivel akkor rájuk is kiterjed a fényesség. A ragyogás sugarát, a fényességét és az élességét változtathatjuk kedvünkre. A sugár megadja, hogy adott pontból mekkora távolságra terjedjen ki a ragyogás, a fényességgel pedig az fény intenzitását változtathatjuk. Nem érdemes azonban nagyon erős fényesség értéket választanunk, mert akkor biztosan elvesznek a részletek és csak a kontúros sötét részek maradnak meg. Persze elképzelhető, hogy éppen ezt szeretnénk.

GIMPresszionizmus (GIMPressionism)

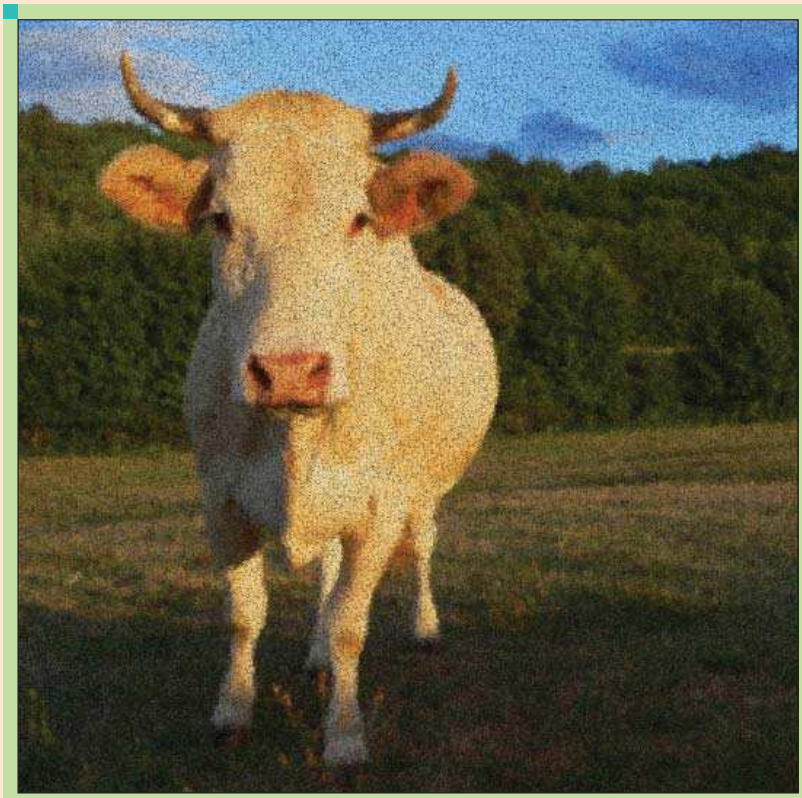
Az előzőek eszenciájaként vegyük szemügyre a menüpontból utolsónak a *GIMPresszionista* szűrőt! Ezzel a sokat tudó eszközzel biztosan kielégíthetjük művészi hajlamainkat. Mindent tud amit a vászonra helyezés és a kubizmus szűrő és még sokkal többet is! Az igazi festmények hatását érhetjük el képeinken a használatával! Kiválaszthatjuk a vászon fajtáját, az ecset alakját, lenyomatát és még az ecsetvonások irányát is! Természetesen a számtalan lehetőséghez, számtalan beállítás is tartozik, amik sűrűjében könnyen el lehet veszni, különösképpen, mivel ez a szűrő nincs túlságosan jól dokumentálva a *Gimp* súgójában. Annyira azért nem félelmetes a helyzet, nézzük tehát szép sorjában a lehetőségeket!
A szűrő ablakában fülek segítenek eligazodni a beállítások között.
A fülektől balra eső részt az ***előnézet (Preview)*** foglalja el. Érdemes sokat használni az előnézet funkciót, mivel csak így tudjuk hatásosan ellenőrizni, hogy tényleg úgy néz-e ki az eredmény ahogy mi elképzeltük. Fontos megjegyezni, hogy az előnézeti kép nem változik magától, hanem nekünk kell frissítenünk, amikor kíváncsiak vagyunk rá! Hasznos dolog, hogy a visszaállítás gomb segítségével visszatérhetünk az eredeti állapotba ha rájövünk, hogy rossz úton járunk. Mivel rengeteg beállítási lehetőségünk van, és ezek kombinációja pedig szinte végtelen számú különböző eredménnyel kecsegtet, fontos megismerkedni a mentés és a visszatöltés lehetőségével! Az első, a ***Profilok (Presets)*** névre hallgató fül alatt tehetjük meg mindezt. Választhatunk a már meglévő profilok közül, ha listában kijelöljük és aztán az alkalmaz gombra kattintunk. A frissítés gomb a listát frissíti, a törlés pedig letörli a kijelölt profilt. Természetesen lehetőségünk van saját beállításaink elmentésére is, amit később bármikor visszatölthetünk! A rendelkezésre álló listában olyan előre elkészített hatásokat találunk, mint a vászonra helyezés vagy a kubizmus. Ez is mutatja ennek a szűrőnek a nagyon is tág lehetőségeit. Ne feledjük, hogy amikor betöltünk egy profilt, akkor az felülírja az aktuális beállításokat!

Vigyázzunk, nehogy több tíz perc munkája menjen veszendőbe ez által! Vessünk egy pillantást a ***Papír fül (Paper)*** beállításaira! Válasszunk egy papír mintát, amelyre majd a képünket tesszük. A lehetőségek közt van papírlap, kő, vászon és még téglafal is! Invertálhatjuk is a mintát és megengedhetjük hogy átfedésben legyenek egymással a minta mozaikjának elemei. Fontos beállítás a méretezés, mellyel a minta az eredeti képhez viszonyított százalékos méretét változtathatjuk meg, tehát ismerkedjünk meg vele jól! Jobb hatást kaphatunk, ha a minta csak párszor ismétlődik a képünkön, mivel így nem vehető észre az ismétlődés. A kiemelkedéssel pedig a minta domborításának fokát adhatjuk meg.



7. ábra Az Ecsetek fül beállításai

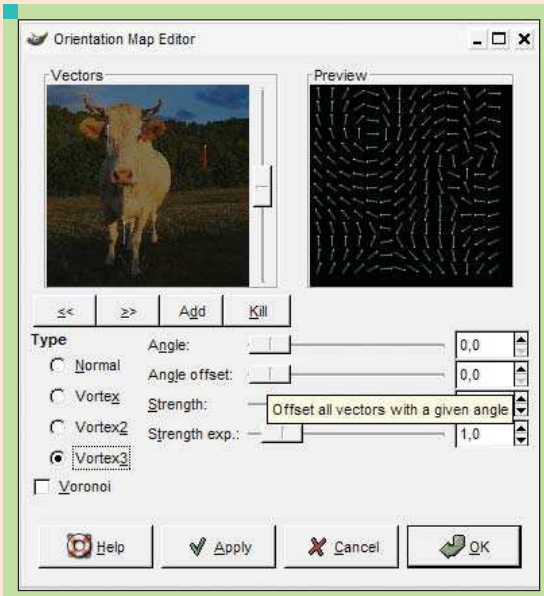
A következő fül alatt ***(Brush)*** választhatunk ***ecseteket***. Sok ecset közül választhatunk, nincs okunk panaszra! Érdekes dolog, hogy az a szűrő külön ecsetválasztékkal rendelkezik, tehát nem az előre beépítetteket használja. Az ecsetek formaválasztéka kielégít minden igényt, igényesen választotta ki őket a fejlesztő. A gamma érték módosításával az ecset fényességét is átállíthatjuk, melynek nagy értéket adva kontrasztosabb, de kevésbé részletgazdag ecsetmintát kapunk. Ha azonban ennél is nagyobb szabadságot szeretnénk, akkor lehetőségünk van saját ecset használatára! A kijelölés felirat melletti listából szabadon választhatunk képünk rétegei közül, hogy melyiket szeretnénk használni mint ecsetet! Nagyon ötletes megoldás! A kiválasztott ecsetet el is menthetjük. A többi beállítás már ismerősként köszön ránk, mivel a papírok beállításakor is azonosak voltak.



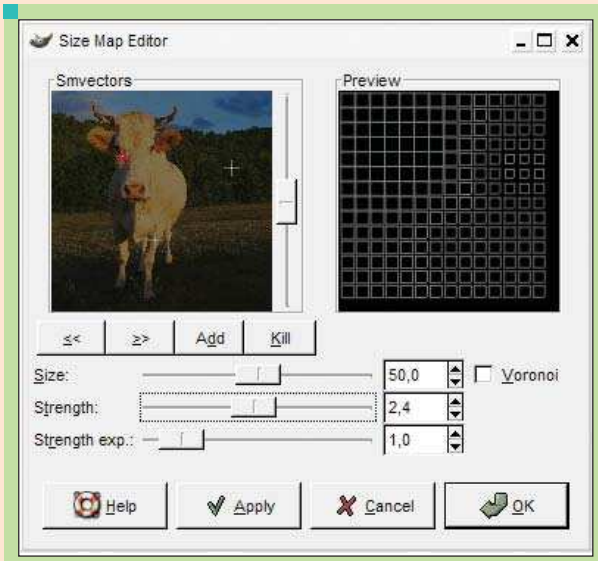
6. ábra Teljes művészi eszközkészletet kapunk a GIMPresszionizmus szűrővel

A ***Tájolás (Orientation)*** fül alatt fergeteges dolgokat találunk! Szinte hihetetlen hogy egy szűrő ilyen mélységű dolgokat valósítson meg! Ebben a menüpontban ugyanis az ecsetvonások irányával kapcsolatban állíthatunk be szó szerint mindent! Vegyük sorra a lehetőségeket fentről lefelé haladva! Elsőként beállíthatjuk mennyi irányt engedünk meg a szűrőnek összesen. Minél nagyobb ez az érték, annál jobban fog hasonlítani a képünk egy tényleges festményre, hiszen egy művész sok irányt használ! Megadhatjuk a kezdőszöget is és a szög kiterjedést is. Az érdekes dolgok az irányok felirat alatt következnek. A lehetőségeink szélesek, nézzük mi határozhatja meg az ecsetvonások irányát!

- ***Érték***: ekkor a tájolás mindig az adott terület fényességének a függvénye.
- ***Sugár***: a vonások irányát a középponttól mért távolság adja meg.
- ***Véletlen***: ezt úgy hiszem nem szükséges magyarázni. (***Linux*** guruk megkérdezhetnék, melyik véletlenszám generáló eszközt használja fel az irányok meghatározásához a szűrő... Kíváncsi lennék, hogy befolyásolja-e észrevehetően a végeredményt!)



8. ábra Az ecsetvonások irányának kézi megadása



9. ábra Az ecsetméretek minden igényt kielégítő beállítása

- ***Radiális***: a tájolás a középponttól mért irány függvénye.
- ***Folyamatos***: az ecsetvonások egy folyamatos mintát követnek.
- ***Árnyalat***: a terület árnyalata alapján számítja ki a szűrő az irányt.
- ***Adaptív***: az eredeti képhez legjobban illeszkedő megoldást használja, vagyis maximálisan alkalmazkodik az eredetihez.
- ***Kézi:*** Igen, ide is elérkeztünk! Akár magunk is megadhatunk egy vektorteret, melyet követ majd az ecsetvonások iránya.

Amennyiben a kézi iránymegadást választjuk, definiálnunk kell egy vektorteret. Ez elég szörnyen hangzik. Nem kell megijednünk, mivel a szerkesztés feliratú gombra kattintva máris egy könnyen használható szerkesztőbe jutunk. A szerkesztő felső részét két kép foglalja el. A bal oldaliban a képünket láthatjuk elsötétítve. Ez azért jó, mert a kép csak viszonyítási pont, a hangsúly a vektorokon van. Egerünkkel a képen kattintva áthelyezhetjük az éppen aktív, tehát pirossal jelölt vektort. A kép alatt lévő hozzáadás gombbal újabb vektort adhatunk a meglévőkhöz, és persze törölni is lehetőségünk van.
A jobbra és a balra mutató nyilakkal ellátott gombok segítségével lehet a vektoraink között váltani. Nézzük, mit is rejt a jobb oldali kép! Itt láthatjuk egy vázlatát annak, hogy a kép adott pontja hogy fognak reagálni a vektorok hatására. Vagyis, hogy milyen festési irányokat fog használni a program a kép különböző pontjain. Az előnézeti kép alatt az egyes vektorok paramétereit állíthatjuk. Beállíthatjuk a szögüket és az erősségüket. Csalóka megoldás tehát nem árt tudni, hogy az ***eltolási szög (Angle offset)*** és az ***erősség kitevő (Strenght exp.)*** megváltoztatása az összes vektorra hatással lesz, tehát nem csak a kijelöltre. Egyébként ez logikus is, ha megértjük az elnevezésük logikáját. Az erősség kitevő szorul mindössze egy kicsivel több magyarázatra. Mit is jelenthet ez a fogalom? Választunk egy erősség kitevőt, és a program a vektor erősségét az erősség kitevő és az erősség szorzataként számolja. Vagyis ha az erősség kitevőt megváltoztatjuk, akkor az összes vektor erőssége változni fog. Fontos kiválasztanunk még a vektor típusát. A normál beállítás esetén a vektor közelében lévő pontok azonos festési irányt fognak kapni mint maga vektor. Az örvény típust választva, a vektor körül örvényként rendeződnek az irányok. Örvény típusból több is választható, az előnézeti képen jól látható, hogy borzasztóan hasonlítanak bizonyos fizikai erőterekre. Az örvény3 típus gyakorlatilag egy mágneses erőtér hatását kelti.
Van még egy fontos beállítás, amiről szólnom kell. Alapesetben egy pontban a festési irányra minden vektor hatással van. Ha azonban ***voronoi módra*** váltunk, akkor csak a legközelebbi vektor lesz rá hatással.
Ugorjuk át a ***Méret (Size)*** fülre! Itt állíthatjuk be a használandó ecsetméretek számát a méretek nevű csúszkával, majd megadhatunk egy minimális és egy maximális méretet is. Alatta beállíthatjuk hogy az ecset mérete egy adott pontban mitől függjön. Az elgondolás hasonló mint a tájolás menüpontban, az ott leírtakkal teljesen megegyezik, kivéve a méretszerkesztőt, amely egy kicsit eltér a tájolásnál látott szerkesztőtől. A különbség azonban csak annyi, hogy pár lehetőségre ebben a szerkesztőben nincs szükség, tehát eltávolították őket. A használata tehát még egyszerűbb is lett.
Az ***Elhelyezés (Placement)*** fül alatt találjuk azokat a beállításokat, amelyek meghatározzák miként legyenek elosztva az ecsetvonások. Lehetnek véletlenszerűen és lehetnek egyenletesen elosztva a képen. A véletlenszerű elosztás természetesen jóval reálisabb végeredményt hoz létre. A relatív sűrűség feliratú csúszka segítségével beleszólhatunk mennyire álljon sok ecsetvonásból a festményünk. Kis értéket választva előfordulhat hogy a papír vagy a háttér kilátszik. Tehát ha tökéletes festményt szeretnénk, nem árt magasra állítanunk ezt az

értéket. Ha kijelöljük a középre feliratú kiválasztót, akkor az ecsetvonásokat a kép közepére fókuszáljuk.
A ***Szín (Colour)*** fül alatt nincs sok beállítási lehetőség. Mindössze azt adhatjuk meg, hogy miként döntse el, hogy milyen színnel fessen egy egy ecsetvonást. Választhatunk az eredeti kép ecsetvonás alatti átlaga és az ecset középpontja alatti képpont színe között. Reálisabb végeredményt kapunk, ha adunk valamennyi színzajt a képhez, vagyis minden színérték egy kicsit el fog térni a tökéletestől.
Már csak pár általános beállítást kell elvégeznünk, nézzük tehát az ***Általános beállítások (General)*** fület! Fontos kérdés a festményünk háttere. Ha például kevés ecsetvonást használunk, akkor előfordulhat, hogy kilátszik az háttér. Ez persze lehet szándékos hatás is, de nem minden esetben mutat jól. Ha a háttérnek az eredeti képet használjuk fel, alig fog feltűnni, ha marad pár befestetlen pont. Ha azonban a papírt választjuk, akkor megeshet, hogy néhol csak egyszerű papír minta lesz a festményünk helyén. Választhatunk egy egyenletes színt is, és ha a képünk nem a háttér réteg, vagyis van alatta másik réteg, akkor választhatjuk azt is hogy a festmény háttere átlátszó legyen. Ezzel megkímélhetjük magunkat a felesleges munkától ha például montázst készítünk. Hasznos beállítások tömkelege van még ebben a csoportban, például a ***szélek megrajzolása(Paint edges)*** kiválasztó. Amennyiben kijelöljük, abban az esetben faltól falig fog dolgozni a program. Amennyiben nem jelöljük ki, akkor, a széleken nem lesz tökéletesen befejezett a kép, amivel félkész akvarell képek hatását imitálhatjuk.
A csempézéshez használható kiválasztóval ***(Tileable)*** gyakorlatilag háttérképet kapunk a festményünkből. Egymás mellé rakva nem fog nagyon feltűnni a széle a képnek, hanem szép átmeneteket kapunk. Az árnyék kiválasztó pedig minden ecsetvonást árnyékkal lát el. Szerintem ez kevésbé eredményez szép képi hatást, de azért lehet vele kísérletezni. Vannak még csúszkáink is. A Szélek sötétítése csúszka segítségével meghatározhatjuk, hogy az ecsetvonások széleinek felé sötétedjenek-e a színek, és ha igen, mennyire. Ugyanígy megadhatjuk, hogy az árnyékokat kiemelje-e a szűrő, illetve, hogy mekkora mélységük legyen és mennyire legyenek sötétek. Ezeknek persze csak akkor van jelentőségük, ha előzőleg beállítottuk, hogy árnyékolja az ecsetvonásokat.
Azt hiszem bátran kijelenthetjük, hogy a ***Gimp*** igen sokrétű és jól használható eszközzel rendelkezik. A művészi szűrők csak egy kis részét alkotják a programhoz rendelkezésre álló tengernyi kiterjesztésből. Remélem mindenki sikerrel fogja használni őket ***Picassot*** megszégyenítő alkotásokhoz is!

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.

Térhatás – A Blender használata (4. rész)
Anyagok és textúrák

Az elmúlt hónapban számos kellemes meglepetés ért. Számos régi ismerősökkel találkoztam, gépjárművezetői engedélyt szereztem, és kipróbálhattam a Blender legújabb – bár egyelőre még csak alpha – verzióját. A továbbiakban erről, majd az anyagok és textúrák használatáról lesz szó.

Miből lesz a cserebogár

Sajnos a dolgok nem mindig olyan egyszerűek, mint amilyennek elsőre tűnnek. Szerencsére azonban nem is olyan bonyolultak. Általánosságban elmondható, hogy egy tárgy anyagának megfelelő megválasztása, illetve textúrázása legalább akkora munka, mint a modellezés. Nem elég ugyanis egy fényforrást letenni, egy színt beállítani, majd megnyomni a ***render*** gombot. Egy tárgy lehet átlátszó, visszaverheti, vagy elnyelheti a fény egy részét, csilloghat, stb... Nézzünk egy kicsit a dolgok mélyére.

Legyünk materialisták...

Ha kiválasztunk egy frissen kreált objektumot, majd megkeressük a ***Material*** panelt, az *2. ábrán* látható szegénység fogad minket. Az ***OB*** illetve ***ME*** gombok az objektumot, és a hozzá tartozó ***mesh***-t jelentik. Mindkettőhöz külön-külön rendelhetünk anyagot. Ez azért lehetséges, mert egy ***mesh***-t több objektum is használhat. Ha az New gombbal egy új anyagot hozunk létre, az rögtön hozzárendelődik az adott objektumhoz (vagy ***mesh***-hez), és a panelünkön további gombok jelennek meg (*2. ábra – jobb oldal*).

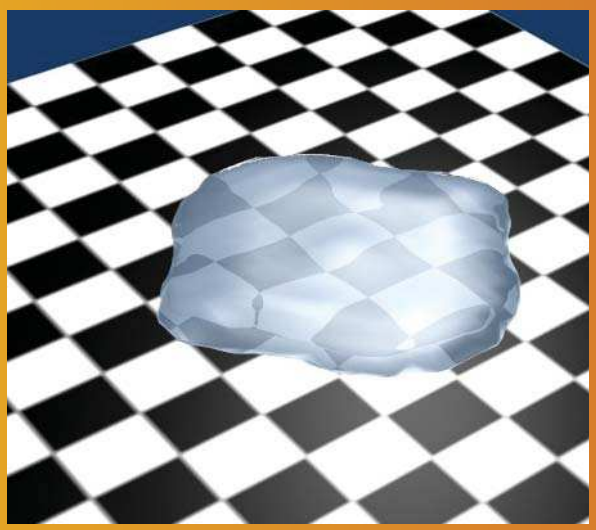
Állíthatjuk az anyag színét (***Col***), a visszaverődő fény (***Spe***) színét, és a tükröződés színét (***Mir***) is. Bár a ***Blender*** automatikusan elnevezi nekünk az új anyagokat, nem árt ha új nevet adunk nekik: Valljuk be, a ***„Material.003”*** nem túl beszédes, és nem is fantáziadús név.

Egy objektumhoz több anyag

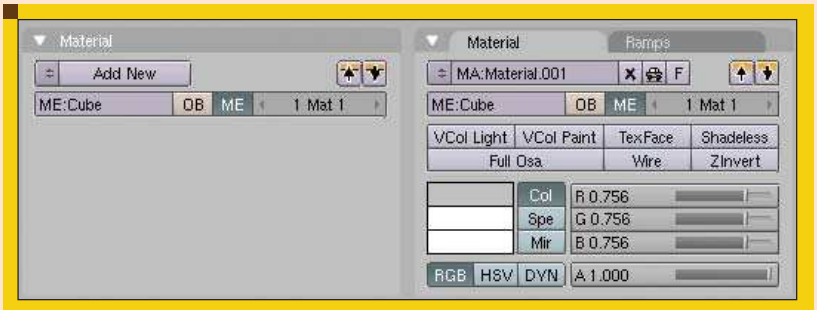
Elég szegényes lenne az élet, ha egy adott objektumhoz vagy ***mesh***-hez csak egyetlen anyagot rendelhetnénk hozzá. A megoldás az úgynevezett ***Material Groups***, amivel egy ***Mesh*** adott részeihez, különböző anyagot rendelhetünk.

Blender 2.40 alpha1

Bár az előző szám megjelenése előtt, de sajnos már lapzárta után tudtam meg, hogy letölthető a ***Blender*** legújabb verziójának teszt változata (➲ ***http://www.blender.org/cms/Blender.31.0.html***). Mivel ez nem a végleges változat, sok funkció nincs még implementálva, de a fejlesztők azt ígérik, hogy számos újdonságot fogunk találni, ha elég türelmesek vagyunk. Főleg a ***mesh*** modellezést fejlesztették: újraírtak néhány dolgot, állítólag egyszerű ***subdivide***-ból nyolcféle lesz, és a knife tool is gazdagodott. Egy teljesen új opció is belekerült a tesztverzióba, a ***Fluid simulation***. Némiképp a gombok elrendezése is változott, a panelek elrendezése kicsit logikusabb lett, de a gyakran használt funkciók (és a pótolhatatlan ***Suzanne***) szerencsére a helyükön maradtak. A végső verzióba számos hasznos bővítés, és állítólag nagy adag hibajavítás fog belekerülni. Ha szerencsénk lesz, és a a fejlesztők is úgy akarják, a következő hónapban már többet tudok írni róla. Addig is egy kép az említett ***fluid simulation***-ról.

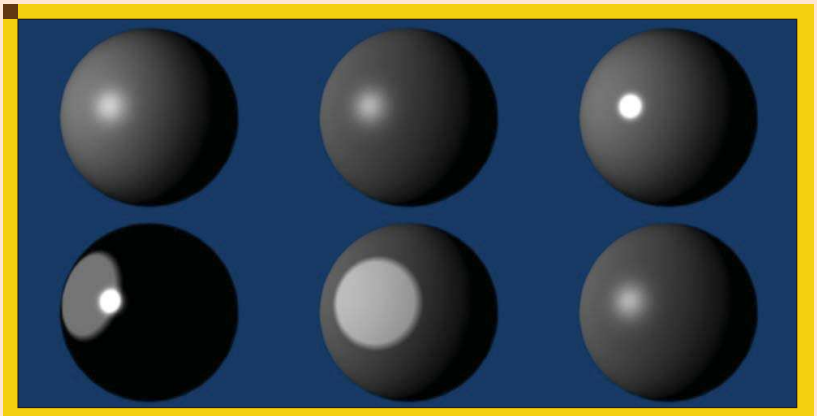


1. ábra A Blender 2.40 alpha, és a fluid simulation



2. ábra A Material panel anyag létrehozása előtt, és után

3. ábra Egy objektum, több anyag

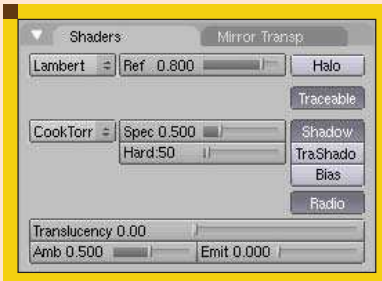


4. ábra Diffuse Shaders, Specular Shaders

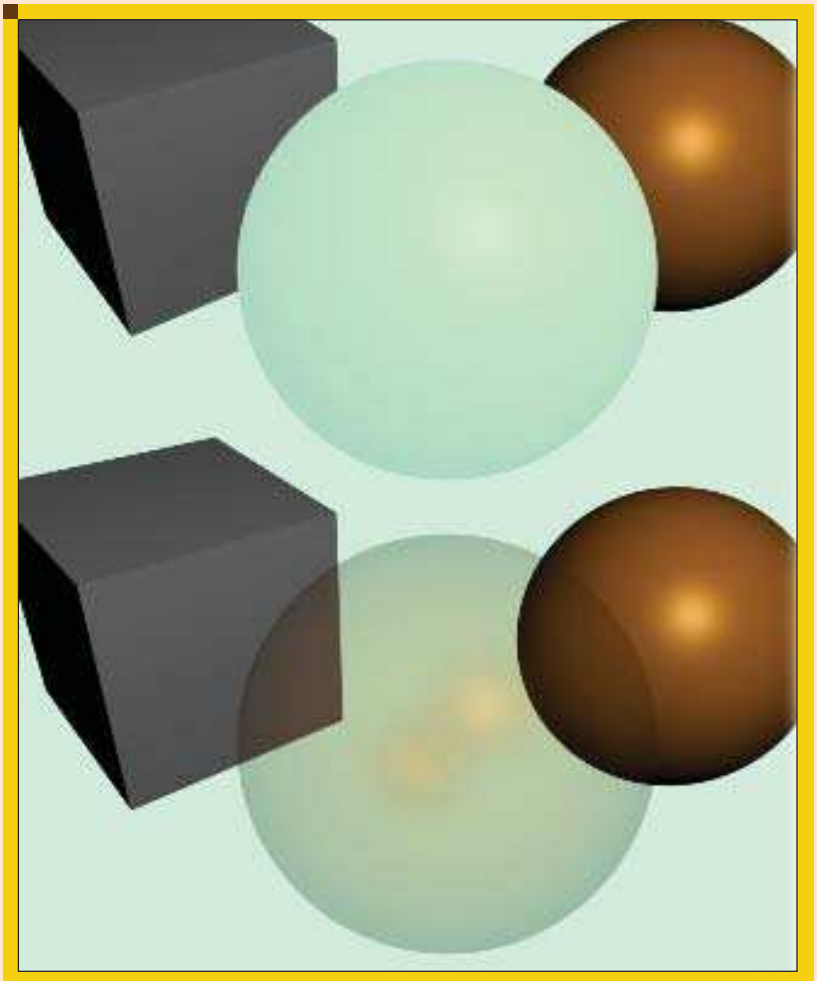
Nyissunk egy új fájlt, jelöljük ki a kockánkat, majd váltsunk ***edit mode***-ba, és keressük meg a gombok között a ***Links and Materials*** panelt. A ***Vertex Groups*** gombokról már volt szó, tőle jobbra található néhány hasonló funkciójú gomb. Az egyetlen különbség, hogy itt minden ***face*** pontosan egy csoporthoz tartozhat (tehát minden ***face*** tartozik valahová, de csak egy csoportba). Válasszuk ki kockánk egyik oldalát, hozzunk létre két új csoportot (nyomjuk meg kétszer ***New*** gombot), majd az egyikhez társítsuk hozzá kockánk kijelölt oldalát (***Assign***). Az másik csoport az összes többit tartalmazni fogja.
A ***Select*** és ***Deselect*** gombokkal ellenőrizni tudjuk munkánkat. A legjobb, ha ilyenkor a ***3D View*** fejlécén ***Face Select mode***-ra váltunk.
Ha végeztünk a csoportosítással visszatérhetünk a ***Material*** panelre. Az ***OB*** és ***ME*** gomboktól jobbra található gombbal válthatunk a ***Material Group***-ok között. Most mindkét csoporthoz ugyanazt az anyag van rendelve, de egy új anyag létrehozásával ezen könnyen segíthetünk. Valószínűleg már feltűnt, hogy bár szabadon hozhatunk létre és rendelhetünk objektumokhoz anyagokat, törölni nem tudjuk őket. (egyel több ok, hogy megfelelő nevet válasszunk nekik) Ha már nézegettük az ***Outliner*** ablakot, azt is megfigyelhettük hogy ***mesh***-t sem tudunk törölni: hiába töröljük magát az objektumot, maga a ***mesh*** az ***Outlinerben*** megmarad. Bevallom először programhibának véltem, de utánaolvastam és mint kiderült, ez nem véletlenül van így. A program bezárásáig a ***Blender*** megőrzi a nem linkelt textúrákat/anyagokat/mesh-eket, így egy nagyon hasznos visszavonási lehetőséget ad a kezünkbe. Ha véletlenül törlünk egy objektumot, egyszerűen létrehozunk egy újat, majd a régi ***mesh*** szülőjeként az új objektumunkat állítjuk be. (***Links and Materials*** panel) A program bezárásával azonban az ún. „nem linkelt” adatok elvesznek, és ezen a mentés sem segít.

Fényvisszaverődés

A tárgyakra érkező fény többféle módon verődik vissza. Mivel tökéletesen sima felület nem létezik, ezért a fény egy része szétszóródik. Ezt hívják ***diffúz visszaverődésnek (diffuse reflection)***. A másik az úgynevezett ***Specular reflection (tükröződő visszaverődés)***, a visszaverődő fénynek az a része, amely közvetlenül a kamerába jut. Ilyenkor a tárgy az adott helyen jóval világosabb, csillog, adott esetben elvakíthatja a kamerát (próbáljuk csak meg a tükröt lefényképezni bekapcsolt vakuval).
A ***Blenderben*** minden tárgynál lehetőség van mindkétféle visszaverődés finomhangolására. Jelenleg ötféle ***Specular Shader (Wardlso, Toon, Blinn, Phong, CookTorr)*** és négyféle ***Diffuse Shader (Minnaert, Toon, Oren-Nayar, Lambert)*** algoritmus közül választhatunk, amiket külön-külön is finomhangolhatunk. A *4. ábrán* ugyanazt a gömböt láthatjuk különböző ***Specular*** illetve ***Diffuse shaderekkel***. Az *5. ábrán* a ***Shaders*** panel látható. Itt választhatjuk ki melyik algoritmust szeretnénk használni a visszaverődések számolásakor. A jobb oldali gombok közül a ***Traceable*** gombbal állíthatjuk be, hogy a tárgy vethet-e árnyékot, illetve a ***Shadow*** gombbal, hogy más tárgy árnyéka megjelenhet-e a tárgy felületén. A ***Halo*** feliratú gomb egy érdekes effektet rejt, vele később még foglalkozunk.



5. ábra A Shaders panel



6. ábra Ugyanaz a kép Ztranp nélkül (felül) és vele (alul)

Tükröződés, és átlátszóság

A ***Shader*** paneltől jobbra (pontosabban a háta mögött) található a ***Mirror Transp*** névre hallgató panel. Az itt található beállítások egy részének csak akkor van jelentősége, ha rendereléskor bekapcsoljuk a sugárkövetést (raytracing). A ***Ray Mirror*** a szomszédos tárgyak tükröződését jelenti, a ***Ray Transparency*** az átlátszóságért felelős. A ***Ztransp*** gomb hatására egy átlátszó tárgy mögött nem a háttér, hanem a ténylegesen mögötte található tárgy lesz látható. (*6. ábra*) Mindkét beállítás finomhangolható, de használatukhoz a raytracing bekapcsolása szükséges, ami nagyságrendekkel megnövelheti a renderelés idejét. Megéri.

Mire jók még az anyagok

Két lehetőséget említenék még meg, ami szorosan az anyagokhoz tartozik. Az első a ***Material*** panelen található ***Wire*** gomb. Hatására mesh-ünkből csak egy drótvázat fogunk látni a renderelt képen.

A másik, ennél érdekesebb lehetőség a ***Shaders*** panelen található ***Halo*** gomb a következőképpen működik: Objektumunk a képen nem fog látszani, azonban minden vertex-e helyettesítődik egy apró fényforrásszerű ponttal. Ennek a pontnak a méretét, színét, intenzitását és egyéb tulajdonságait szabályozhatjuk, így változatos effekteket tudunk vele elérni. Érdemes kipróbálni a ***Halo*** gomb alatt (ha a ***Halo*** bekapcsolt állapotban van) található gombokat.

Textúrák

A ***Blender*** igazi gazdagsága a textúrázásban nyilvánul meg. Sokan azt hiszik, hogy a textúrázás annyit jelent, hogy egy képet ráhúzunk a testre, amitől az szép színes lesz. Azonban ennél jóval többről van szó. Egy textúra nem csupán a tárgy színét változtathatja meg, de beállíthatjuk úgy is, hogy csupán a beeső vagy visszavert fényre vagy a tükröződő objektumok képére legyen hatással. Egy jó textúrával és pár kattintással egy sima felületű gömbbel is csodákra lehetünk képesek.

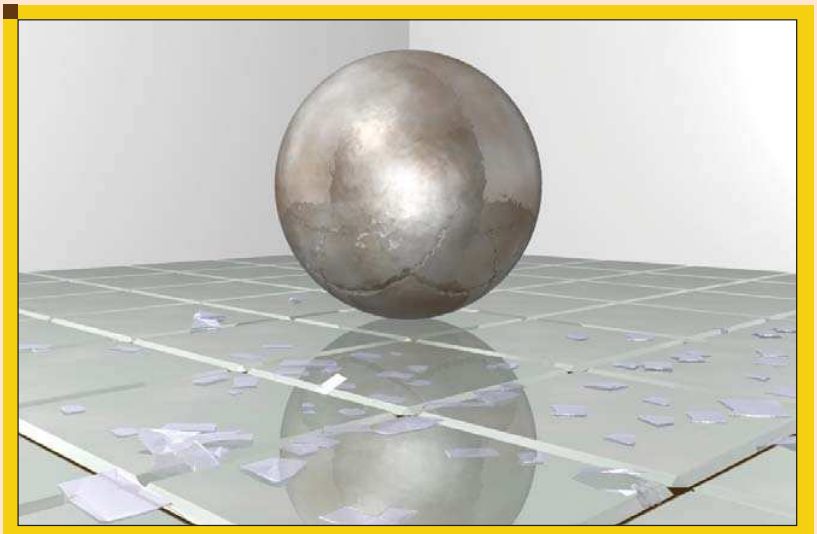
A ***Blenderben*** minden egyes objektumhoz tíz textúra csatorna létezik, amiket egyesével finomhangolhatunk, a tárgyhoz igazíthatunk. A renderelő ezeken sorba megy, míg végül meghatározza az adott pixel színét. Ebből az is következik, hogy egy egyáltalán nem átlátszó textúra teljesen felülbírálhatja egy alatta lévő textúra vagy az anyag beállításait.

Új textúra hozzáadása

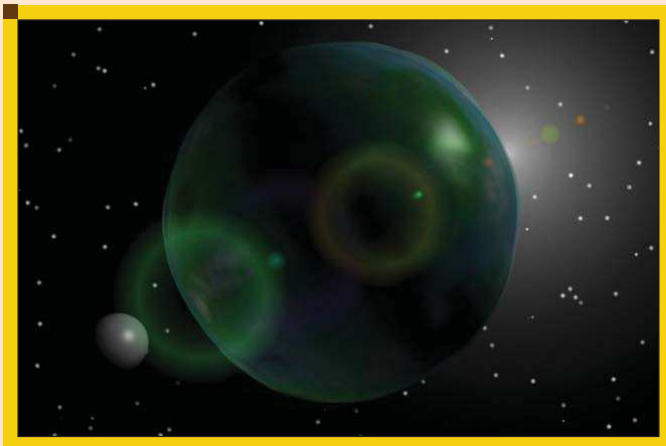
Keressük meg a ***Texture*** panelt. (Ebből rögtön kettőt is találhatunk. Egyet a ***texture buttons***, egyet pedig a ***material buttons*** gombok között) Válasszunk ki egy textúra csatornát, majd nyomjuk meg az ***Add New*** gombot. Hogy ezt megtehessük, az objektumunknak már rendelkeznie kell egy anyaggal. Ezzel létrehoztunk egy új textúrát, aminek típusát (***Texture Type***) a két ***Texture*** panel egyikén definiálhatjuk. Alapméretezésben ez ***„none”***, azaz semmi, azonban a legördülő menüből kiválasztva egy opciót, újabb panelek lesznek elérhetőek, attól függően, hogy mit választunk ki. A ***Colors*** panelen tetszőleges számú színből álló színátmenetet adhatunk meg textúránknak amihez az átlátszóság (***Alpha***) is dukál, így részben átlátszó textúrákat is készíthetünk. Az egyes textúra típusok leírása felesleges lenne, de az ***Image*** nevűre érdemes kitérni, vele ugyanis tetszőleges képet (sőt, videót) felhasználhatunk textúrának.

A ***Load Image*** gombbal többféle képformátumot tudunk betölteni.

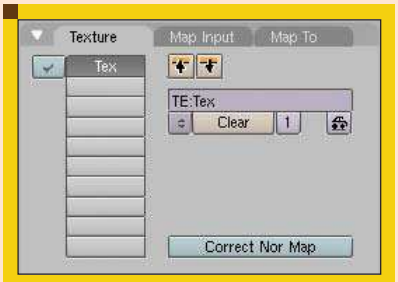
A ***MinX***, ***MinY***, stb. gombokkal beállíthatjuk, ha a képnek csak egy részét szeretnénk felhasználni. Egy ötletes lehetőség a ***Movie*** gomb, amit ha bekapcsolunk, animációt is betölthetünk textúrának. Természetesen beállíthatjuk hogy a videófájl melyik részét szeretnénk használni, az ismétlések számát, és még néhány dolgot. Egyetlen hátránya hogy elég finnyás a videótömörítésre, így nem mindegy hogy a használni kívánt videófájl milyen codec-el készült.



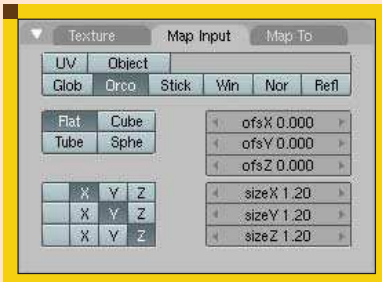
7. ábra Ray Mirroring



8. ábra Mire jó a Halo



9. ábra Texture panel (az egyik)



10. ábra A Map Input Panel

11. ábra A Map To Panel

Texture mapping

Mivel textúráink két dimenziósak, a programnak szüksége van adatokra arra vonatkozóan, hogy a kép *2D*-s koordinátáit hogyan képezze le *3D*-s koordinátákra. Ezt a folyamatot a szaknyelv ***Texture Mapping***-nek nevezi, paramétereit pedig a ***Map Input*** panelen állíthatjuk be.

A Map To panel

Mint már említettem a textúrázás nem csupán a tárgy színének megváltoztatását jelenti. Minden egyes textúrához beállíthatjuk, hogy mire legyen hatással, ezt a ***Map To*** panelen található gombok szabályozzák. Ezek közül egyszerre többet is használhatunk. A legfontosabbak:

- *Col* – Azaz *Color*. Az ilyen textúra a színt változtatja meg. Ez az alapméretezett
- *Nor* – A textúra a test normálvektorára van hatással az adott helyen, kiválóan alkalmas a ***Bump Mapping***-re.
- *Csp* – A ***Specular Reflection*** vagyis a visszaverődő fény erősségét szabályozza a textúra.
- *Cmir* – A textúra a ***Ray Mirroring***-ra van hatással
- *Disp* – A *Nor*-hoz hasonló a hatása, de nem csupán a normálvektort változtatja meg, hanem ténylegesen kiemeli vagy süllyeszti a test felszínét.

Ezek közül néhánynak az intenzitása is beállítható, de csak óvatosan vele. Nagyon könnyen áteshetünk a ló túloldalára, és az előnézet is sokszor hazudik. Végül egy jó tanács: ne adjuk fel, kísérletezzünk módszeresen! Az anyagok és textúrák használata nem olyan dolog, amit könnyen meg lehet tanulni. Tapasztalat és sok-sok próbálgatás kell hozzá. Bár helyhiány miatt az ***UV Mapping*** nem került bemutatásra egy későbbi számban még előfordulhat. Mindenesetre a következő két hónapban az animáció kerül sorra, addig is jó szórakozást.

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.

Az SVG világa (4. rész)
Képzeletünk határai

Most, hogy már megismertük, mire is képesek a szabad szoftverek SVG fronton, itt az ideje, hogy a horizontunkat kicsit kitágítsuk! Láttuk a fájlformátumot, a rá épülő alkalmazásokat is, de a lehetőségeknek itt korántsincs vége. Tulajdonképpen azt kezdünk az SVG-vel, amit akarunk! Számtalan programozási nyelv áll szolgálatunkra és némelyeknél már nem kell a világot újra felépítenünk magunknak.

Egyszóval elkényeztetnek minket *SVG* terén is. Persze a szabad szoftver itt is óriási plusz: a fekete doboz mentalitás helyett (vö.: bemenet-kimenet – közte a nagy homály) mindig a dolgok mélyére tudunk ásni, ami a programozás terén érdekes is és a hibakeresést is megkönnyíti. Kezdődjék az enumeratio, jöjjenek az objektumok!

A nyelvek kavalkádja és a megközelítés

SVG objektumokról szólj nékem *Google*! És szól is! Nem könnyű választani, hogy mely programnyelvekről legyen szó most, de a választás egyben szükséges és kényszerű. Szükséges, hiszen a nyelvek népszerűsége nyomán nem minden érdekli annyira a nyájas olvasót és kényszerű azért, hiszen egy Linuxvilágnyi terjedelmet is meg lehetne tölteni a különféle nyelvek *SVG*-vel kapcsolatos lehetőségeivel. Így aztán tervszerűen, előre megfontolt szándékkal a *PHP* és a *Perl* lesz a két alanya e cikknek a főszereplő *SVG* mellett. Nem magyarázkodásképpen, sokkal inkább magyarázatképpen jó tudni, hogy Magyarországon ebben az évben október 22-én először volt *Perl* konferencia, amin *Larry Wall*, a nyelv megalkotója is részt vett, valamint a *netcraft.com* szerverinformációs eszközével könnyen rájöhetünk, hogy hallatlan sok szerver van felszerelve a *PHP*-val. A *Perl* pedig hatalmas, szervezett struktúrában kínálja a nyelvhez a kiegészítő elemeket, ami nagy könnyebbség. Rutinosak a *CPAN*-ról már egyből tudják, mire is kell gondolni.

Alapvetően kétféle módon közelíthetjük meg az *SVG* formátum felhasználását programjainkban. Felhasználhatunk már meglévő *SVG* képeket, hogy azokat megjelenítsük vagy magunk állíthatunk elő *SVG* ábrákat. Az első esetnek nem sok érdekessége van, hiszen erre kész megoldások léteznek. *KDE*-re épülő alkalmazáshoz az *svg.kde.org* címen nézzünk szét, *Gnomehoz* pedig a *librsvg.sourceforge.net* címen. Sokkal érdekesebb helyzet, ha tartalmat akarunk létrehozni, ebben sokkal inkább van perspektíva!

Ma a legnépszerűbbek a szkriptnyelvek; és ahol ezeket a programokat futtatni szokás, az korántsem meglepő módon a web. Nem egészen véletlen a webre kerülő alkalmazások nagy népszerűsége, hiszen bárhol, bármikor elérhetőek és gyakorlatilag platformfüggetlenül használhatóak. A *Perl* kezdetekben uralta a webes programok felségterületét, aztán a *PHP* mára átvette ezt a stafétabotot. Gyakran hangoztatott érv, hogy a *Perl* túl lassú. De nem kell messzire menni sikeres *Perl* alapú portálokért: a népszerű *Slashdot* is ezt használja, ami kellő bizonyíték, hogy nagy terhelés mellett is lehet Perlt használni. A következőekben a *PHP*-s és *Perl*-es *SVG*-gyártásról lesz szó. A továbbiak megértéséhez az alapvető programozási fogalmak, mint változó, ciklus és függvény ismerete szükséges. Az objektumé nem feltétlen, bár azokkal fogunk dolgozni.

Az objektum, pontatlan megfogalmazásban, nem más, mint egy saját emlékezettel ellátott függvény. Az adatszerkezeteket egységbe foglalhatjuk a rajtuk műveleteket végző függvényekkel. Ez valójában egy végletesen leegyszerűsített magyarázat, de ahhoz elég, hogy az *SVG* objektumokkal dolgozni tudjunk.

A cikkben újfent alapvető dolgokról lesz szó, így az elején tekintsük át azért, hogy milyen óriási lehetőségek rejtőznek abban, ha saját programunkkal készítünk *SVG*-t! Összekapcsolhatjuk a lentieket adatbázisokkal, mobil eszközökkel(a WAP már elég elterjedt, csak éppen nincs rá meg az a húzóalkalmazás, amitől használnák is a tömegek) vagy akár a *Google API* (➲ *http://www.google.com/apis/*)-val is. Néhány ötlet az itt felsoroltakra: webáruház adatbázisának *SVG*-s vizualizációja, mobiltelefonra térkép vagy akár a *Google Fight* (➲ *www.googlefight.com*) animált *SVG*-s verziója is elkészíthető. A lehetőségeket tényleg a fantáziánk határolja be.

Perl

„Puszta kézzel" is lehetne *SVG*-ül sorokat kiírni a szabványos kimenetre, hiszen tulajdonképpen ennyiből áll a tennivaló, ezt sokkal elegánsabban

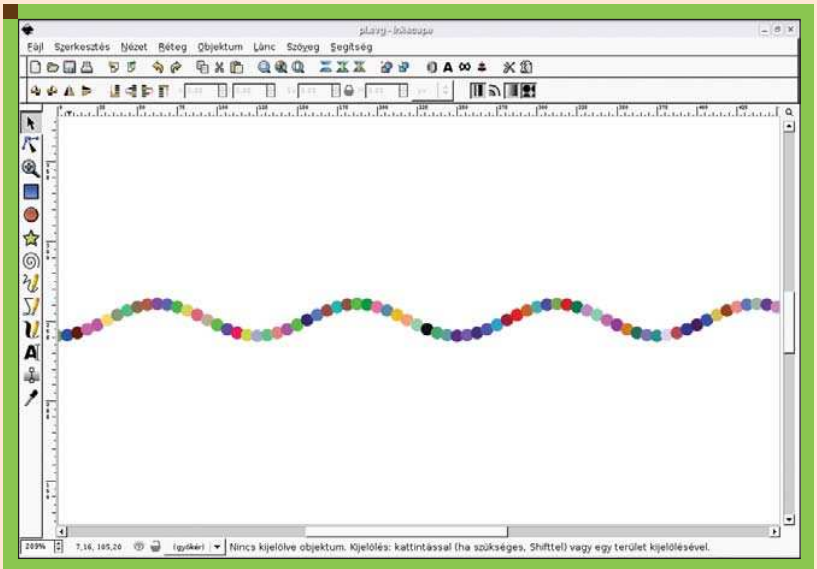
is megoldhatjuk, arról nem is beszélve, hogy ha az *SVG*-vel kellene bajlódni, az alapprobléma sikkadna el.
A már emlegetett *CPAN*-ról szóljunk egy kicsit! A *CPAN* betűszó feloldása a *Comprehensive Perl Archive Network*. Ebben a cikk írásának pillanatában 8849 modult lehetett találni. A *CPAN*-ban nem csak szoftverek, hanem dokumentációk is vannak. Ebből az óriási kavalkádból természetesen találunk *SVG*-hez kézreálló segítséget. A megtalált kiegészítést fel kell először telepíteni, amit a *CPAN* parancsértelmezővel tehetünk meg az alábbi módon:

```
perl -MCPAN -e shell;
```

A fenti parancsot root-ként adjuk ki, hiszen most programkönyvtárat fogunk telepíteni a rendszerre.
Ha először indítjuk el ebben az üzemmódban a Perlt, néhány triviális kérdésre válaszolnunk kell, hogy hogyan tudja elérni a *CPAN* archívumot, azután munkához láthatunk. A `cpan>` prompt után adjuk ki azt az `install SVG` parancsot, és ha minden rendben lezajlik, akkor minden előfeltétel adott ahhoz, hogy elkezdhessük a munkát.
Ha valakinek nincs internetkapcsolata, ez a módszer nem működőképes. Ez esetben letöltheti *tar.gz* formátumban a modult és telepítheti kézzel a *README*-ben leírtaknak megfelelően.
Ennyi bevezetés után igazán szeretnénk olyan kódot látni, ami már *SVG*-t generál. Rajzoljunk színes(véletlen alapon) köröket egy szinuszhullám mentén (*1. Lista*)!
A keletkező képet bármilyen *SVG*-képes nézegetővel/szerkesztővel meg tudjuk nyitni. A `dec2hex`, `szin`, `rgb` függvények magukért beszélnek. A `dec2hex` egy tízes számrendszerbeli számot tizenhatos számrendszerbelivé alakít, az `rgb` `00`-tól `FF`-ig ad vissza hexadecimálisban egy véletlenszámot, az rgb pedig három ilyen felhasználásával egy *SVG* értelmező számára is emészthető színkódot ad. A `dec2hex` függvényben a `? :` operátorra azért volt szükség, hiszen a 0-t `00` alakban kell megadni(és így tovább, mindig kétjegyűre bővítve), mert feltétlenül hat karakterből kell álljon az érvényes színkód.
A `$szelesseg`, `$magassag` változóban az elkészülő ábra méreteit tároljuk,

1. Lista szinusz.pl

```
use SVG;

sub dec2hex($) {
   my( $dec ) = shift;
   return length(sprintf("%lx", $dec ))== 2?sprintf("%lx",
   ➥$dec ):sprintf("0%lx", $dec);
}

sub szin {
   return dec2hex(int(rand 255));
}

sub rgb {
   return "#" . szin . szin . szin;
}

$szelesseg=500;
$magassag=500;
$kordarab=80;
$PI=3.14159265;
my $svg= SVG->new('width',$szelesseg,'height',$magassag);
for($i=0; $i<$kordarab; $i++)
{
   $svg->circle(id=>'kor' . $i,
   cx=>(($szelesseg/$kordarab)*$i),
   cy=>($magassag/2)+(cos($PI*$i*0.1)*10),
   r=>$kordarab/20,
   style=>{fill=> rgb }
   );
}
print $svg->xmlify;
```



1. ábra A szinusz.pl kimenete Inkscape-ben megnyitva

a `kordarab`-ban pedig azt, hogy hány kört szeretnénk elhelyezni a szinuszhullámon.
A `for` ciklus előtt már található érdekes rész: egy új *SVG* objektumot hozunk létre, amit a `$svg`-vel jelölünk.
A létrehozáskor megadjuk a kiterjedését. A ciklusban található az egyetlen rajzoló utasítás. Itt a létrejövő új elem *XML* azonosítójára, az X és Y koordinátákra és a kitöltés színére van szükségünk. Az utolsó előtti sorban azt

1. táblázat

Függvény neve	Geometriai alakzat	Függvény paraméterei
`circle`	kör	• `cx` – középpont x koordinátája • `cy` – középpont y koordinátája • `r` – sugár
`ellipse`	ellipszis	• `cx` – középpont x koordinátája • `cy` – középpont y koordinátája • `rx` – ellipszis magassága • `ry` – ellipszis szélessége
`rectangle`	téglalap	• `x` – x koordináta • `y` – y koordináta • `width` – szélesség • `height` – magasság • `rx` – saroklekerekítés x sugara • `ry` – saroklekerekítés y sugara
`image`	bitkép beszúrása	• `x` • `y` • `width` • `height` • `'-href'` – a bitkép elérési útja

mondjuk, hogy az eddigi tényke-déseinket a `$svg` objektumot alakítsa át *SVG* formátumba és ezt írja ki a képernyőre. Ebből a kis szkript-ből is látszik, hogy sokkal több időt töltöttünk el a valódi feladattal (színkódok és szinuszhullám), mint amit az *SVG*-re fordítottunk. Ez nagyobb terjedelmű munkák esetén csak fokozódik.
Szép és jó, hogy egy kört el tudunk helyezni valahol a síkon, de az *SVG* ennél azért többet tud.
A ➲ *http://search.cpan.org/~ronan/SVG-2.33/lib/SVG/Manual.pm* oldalon találjuk a teljes referenciát, de a leg-fontosabb tudnivalók itt is olvashatók. Ahogy az *SVG*-ben is megvoltak az alapvető építőelemek, éppen úgy itt is. Az *1. táblázat* ezeket foglalja össze. Ezeket a következő formában lehet használni:

```
$svg_objektum->Függvény
➥neve(Függvény paramétere=>
➥"valami", Függvény
➥paramétere=>"valami");
```

A szöveg használata még fontos lehet az elemi alkalmazásoknál is, ez azon-ban nem sorolható be a fentiek közé formai szempontból:

```
$svg->text(x=>10, y=>10,
➥-cdata=>'LinuxVilág');
```

Persze ha a kedvenc újságunk neve helyett mást akarunk leírni, annak sincs különösebb akadálya. A betűtí-pust a most következő style használa-tával tudjuk beállítani.
Ugyan az alapokon túl vagyunk, de formázás még hiányzik. A `style` pont erre való a következőképpen:

```
style => { CSS tulajdonság=>
➥'érték', CSS tulajdonság=>
➥'érték' }
```

Ezt lehet mindegyik elemhez használ-ni, hogy mi módon, az a példaprog-ramból kiderül. Ezekből az alapvető építőelemekből és azokból amiket itt nem részleteztünk, felépíthetőek az *SVG* fájlok a *Perl* segítségével.

PHP

A *PHP*-nál nem egy hivatalos hely-ről, hanem a ➲ *www.phpclasses.net* oldalról fogjuk beszerezni az *SVG* gyártáshoz elengedhetetlen eszközt. A *PHP* alapvetően webes eszköz, így készítsünk most el egy olyan szkriptet, ami egy weboldalon be-kér egy szöveget és azt utána külön-féle színű és méretű (véletlenszeru-en) betűkkel írja fel egy SVG doku-mentumban. A szükséges fájlokat a ➲ *http://www.phpclasses.org/browse/package/457.html* címről tudjuk letölte-ni. Ez hasonlóan a Perles példához, egy objektum lesz. ami elvégzi a pisz-kos munkát *SVG* fronton. A példa ki-próbálásához nincs is másra szükség, mint egy webszerverre, ahol működik a *PHP*. Hozzunk létre egy könyvtárat például *svg* néven és oda töltsük fel a *phpclasses.org*-ról leszedett fájlokat. Ha ez megvan, az itt közölt példa-program is életre kel (*2. Lista*).
Az `SVG_CLASS_BASE`-t kell csupán úgy átírni, hogy megtalálja az objektum saját magát és ki is lehet próbálni az alkotást. A fájl elején pontosan ugyan azt kellett elvégezni, amit *Perlben* is, egy véletlenszerű hexadecimális szín-kódot előállítani. Attól függően, hogy kapott-e `POST` módon `szoveg` paramé-tert a program, vagy kiírja az űrlapot vagy előállítja az *SVG* fájlt. Az űrlap kiírása minimális *PHP* és *HTML* isme-retekkel megérthető, de az `if` előbbi ága már nem ennyire triviális.
Be kell először is állítani a már említett `SVG_CLASS_BASE`-t, hogy az objektum tudja, merre vannak a saját részei. Ezután a `require_once("Svg.php");` sorral jelezzük, hogy erre a fájlra szükségünk van a kódban. A

```
$svg =& new SvgDocument
➥($betukoz *($hossz+1) ,
➥$magassag);
```

sor létrehoz egy új példányt az objek-tumból, amivel megkezdődik a tényle-ges *SVG*-zés. A `for` ciklus által betűn-ként rakjuk a dokumentumba a szöve-get, ráadásul a színen kívül még a be-tűméretet is véletlenszerűen válasz-tjuk két határ között. A `$svg->printElement();` pedig mindent, amit a $svg-nek kiadtunk, *SVG* formá-ban visszaadja. Ezen *SVG* objektum részletes ismertetése nem célja ennek a cikknek, maga a *PHP* kódja már do-kumentálja a működését. A *Perlben* felsorolt lehetőségek mindegyike léte-zik itt is, a szintaktika tér el csak kissé. Néhány fontosabb függvény a forrás-kódból:

```
function SvgText($x=0, $y=0,
➥$text= "", $style="",
➥$transform="");
function SvgRect($x=0, $y=0,
➥$width=0, $height=0,
➥$style="", $transform="");
function SvgLine($x1=0, $y1=0,
➥$x2=0, $y2=0, $style="",
➥$transform="");
```

2. Lista betu.php

```
<?

function szin()
{
   $veletlen = dechex(rand(0, 255));
   return strlen($veletlen) <2? "0" . $veletlen :
   ➥$veletlen;
}

function szinkod()
{
   return "#" . szin() . szin() . szin();
}

if($_POST["szoveg"])
{
   define("SVG_CLASS_BASE",
$_SERVER["DOCUMENT_ROOT"] . "/svg/");
   $betukoz = 8;
   $min_betumeret = 8;
   $max_betumeret = 18;
   $magassag = 30;
   require_once("Svg.php");
   $szoveg =$_POST["szoveg"];
   $hossz = strlen($szoveg);
   $svg =& new SvgDocument($betukoz *($hossz+1) ,
   ➥$magassag);
   for($i=0; $i<strlen($szoveg); $i++)
   {
      $svg->AddChild(new SvgText(($i+1)*$betukoz,
      ➥$magassag/2, substr($szoveg, $i, 1),
                        "font-size:".
➥rand($min_betumeret,$max_betumeret) .
➥";text-anchor:middle; fill:" . szinkod()));
   }
   $svg->printElement();
}
else {
   header("Content-type: text/html;
   ➥charset=iso8859-2");
   ?><!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
   ➥Transitional//EN">
   <html>
   <head>
     <meta
   content="text/html; charset=ISO-8859-2"
   http-equiv="content-type">
   <title>SVG</title>
   </head>
   <body>
   <form method="post"
action="<?$PHP_SELF?>"><input
   name="szoveg"><input
   type="submit"></form>
   </body>
   </html>
<?
}
?>
```

2. ábra A betu.php futási eredménye Konquerorban

```
function SvgEllipse($cx=0,
➥$cy=0, $rx=0, $ry=0,
➥$style="", $transform="");
function SvgCircle($cx=0,
➥$cy=0, $r=0, $style="",
➥$transform="");
```

Ezekhez a függvényekhez sem szükséges megadni az összes itt felsorolt paramétert. Ne feledjük, hogy az `AddChild`-ot a példaprogramhoz hasonlóan használni kell a létrejött grafikai elemekre. Figyelem! A *PHP*-s program a bemenetet semmilyen módon nem ellenőrzi. Ez egy valódi alkalmazásban megengedhetetlen. Tanulságos eset történt a cikk írása közben, kapcsolódik a bevezetőben említett nyílt forrás – könnyű hibakeresés témához. Szokásomhoz híven a *PHP* és a *HTML* kódot *UTF-8*-as karakterkészlettel készítettem, de a program egyáltalán nem akart helyesen működni ékezetes karakterekkel. Ekkor megnyitottam a rendelkezésemre álló *php* fájlt (az *SvgDocument.php*-t) és megnéztem az objektum belső kódolását. Akár egy mozdulattal át is cserélhettem volna *UTF-8*-ra, de azt szerettem volna, ha az objektum módosítás nélkül felhasználható legyen.

Merre tovább?

Ez a két egyszerű példaprogram és a magyarázatuk azt a célt szolgálja, hogy az olvasó ráérezzen arra, hogy a szerveroldali *SVG* gyártás miféle lehetőségekkel bír. Akik jártasak *PHP*-ben és *Perl*-ben, azok tudják, hogy mindkét nyelvnek nagy erőssége az adatbázis-elérés, adatfeldolgozás. Az adatbányászat egyik eleme lehet az adatbázis megjelenítése *SVG* segítségével. Időjárás-ikon készíthető úgy, hogy egy szkript feldolgoz egy időjárás *RSS*-csatornát (➲ *www.rssweather.com*) és még sorolhatnánk. Egyszerű eszközökkel látványos megoldásokhoz juthatunk. A sorozat ugyan ennél a pontnál véget ér, de az *SVG* lehetőségei természetesen nem.

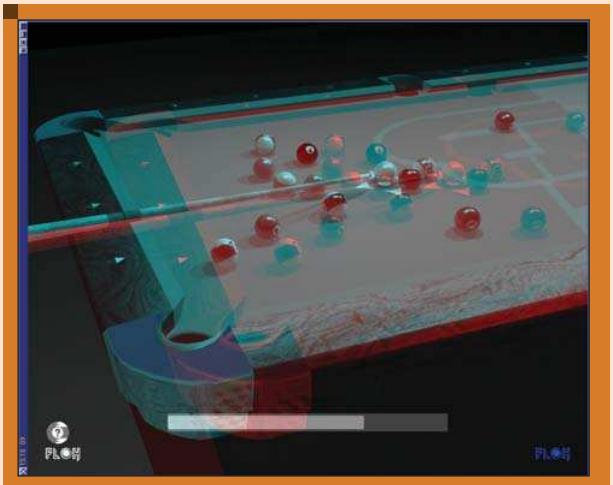
Novák Áron
(aaron@szentimre.hu)
BME-VIK-es gólya, műkedvelő rendszergazda. Jelenleg leginkább a NetBeans-szel és mindenféle hordozható eszközzel foglalkozik, legalábbis mindazokkal amelyeket meg lehet szólaltatni Linux alatt.

Kevés vagy a csocsóhoz? Biliárdozz!

Nincs annál jobb, mint egy átdolgozott nap után a haverokkal vagy a rendszergazda kollégákkal elmenni egy közeli pub-ba és egy-két sör kíséretében biliárdozni egy jót...

No de ne szaladjunk előre, érdemes pár szót vesztegetni eme nemes játék történelmére is. Én speciel sokáig azon gondolkodtam, vajon miért zöld a biliárdasztalon a szövet? Aztán kiderült. Sokáig a biliárd egyfajta krikettjáték volt, amit a füvön játszottak. ***XI. Lajos*** rendelte meg az első biliárdasztalt valamikor 1470 környékén – már ha hihetünk a feljegyzéseknek. 1868-ig elefántcsont golyókat használtak... Amúgy számos híresség és arisztokrata – például ***IX. Pius pápa*** és ***Viktória királynő*** – is biliárddal múlatta az idejét.

1. ábra Az asztal



2. ábra Van sztereó nézet is!

3. ábra Indulhat a játszma

Kevés vagy a csocsóhoz? Biliárdozz!

Mint a bevezetőben említettem, nincs jobb egy jó kis biliárdnál. A csocsó nem biztos, hogy sikerélményeket ad, ugyanis nem árt hozzá a jó reflex. Viszont a biliárd egész más... nem árt egy kis térlátás és gyakorlat. Egyedüli gyakorlásra pedig nincs is jobb partner a ***FooBillard***-nál. Nem helyettesíti az igazi biliárdasztalt, de lehet gyakorolni, hogy a való életben magabiztosabbak legyünk az asztal mellett.

Rendszerigények

Először is ajánlott egy ***OpenGL***-ben viszonylag erős videókártya. Nekem egy ***Nvidia Riva TNT2***-n szaggatás nélkül fut, igaz van alatta 900 Mhz-es processzor és 256 Mbyte memória. Lévén hogy a program alapértelmezett fordítás esetén erősen az ***OpenGL***-re alapozik, így elképzelhető, hogy mondjuk egy 4-500 Mhz-es masinán se szaggat. Amennyiben ***OpenGL***-hez van fordítva a ***foobillard***, úgy ***X*** alatt nem jó az ***nv*** driver, helyette az ***nvidia*** kell, már persze ha ***Nvidia*** kártyánk van. ***Ati*** kártyánál is érdemes a legfrissebb, ámde stabil meghajtóprogramot használni. Ajánlott egy hangkártya is, hogy a golyók koccanását halljuk, mert hát azért mégis...

Telepítés

Minthogy én ***Debian***t használok, így egyszerűen egy `apt-get install foobillard` után már megy is a játék. Amennyiben a ***disztribúció***nk nem tartalmazza eme remeket, úgy forrásból is telepíthető. Ha forráskódból telepítjük, akkor eldönthetjük, hogy az ***SDL***-t vagy ***OpenGL***-t részesítjük előnyben. Tapasztalataim szerint ***OpenGL*** esetén lényegesen gyengébb gép is elég.
A telepítés a megszokott módon történik, de ez röviden az ***INSTALL*** fájlban is le van írva.

Játék

Tegyük fel, hogy minden kedves Olvasó tisztában van a játékszabályokkal. Aki nem, az nézzen utána a neten. Ide nem merem berakni, mert a fél újságot megtöltené, és van egy olyan érzésem, hogy a szerkesztő nem értékelné a dolgot... Szóval a játék.
Miután a `foobillard` paranccsal elindítottuk a játékot máris lökdöshetünk. A ***fel***/***le*** gombokkal a lövés erősségét, míg a ***bal egérgomb*** nyomva tartásával és az egér húzogatásával válthatjuk az lökés szögét. A ***jobb egérgomb*** nyomva tartásával és az egér mozgatásával pedig rá tudunk zoomolni a fehér golyóra. A ***szóköz*** billentyűvel vagy a ***középső egérgombbal*** tudunk lökni, ha sikerült beállítani az előbbieket. Trükkös lökéshez – hogy be tudjuk állítani, hol szeretnénk megpöckölni a fehér golyót – használhatjuk a ***Shift + jobb egérgombot***.
A menü az ***ESC*** gombra jön be. Itt mindenféle finomság közül lehet választani. Például be lehet állítani, hogy milyen típusú biliárdot szeretnénk játszani, hogy milyen színű legyen az asztal és a végére hagytam a kedvencem. Ez pedig nem más, mint a ***sztereó nézet***. Vagyis ha van egy piros-kék szemüvegünk, akkor ál-3D-ben játszhatunk biliárdot, ugyanis a program képes ennek megfelelően megjeleníteni az asztalt. Mondjuk kicsit hülyén néz ki szemüveg nélkül, de spéci szemüveggel jó móka.

Összegfoglalás

A játék függőséget okozhat. Nálam például előfordul, hogy gyakran a TV nézés helyett biliárdozok magamban – ***Training mode*** – vagy a lakótársakkal, akik legelőször persze meglepődtek: *„**Jé, ilyen komoly játék is van Linuxra?**”*

Jó biliárdozást mindenkinek.

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

FooBillard
➲ http://foobillard.sunsite.dk/

Mindent a billiárdról
➲ http://billiard.lap.hu/