Hírek

Új ASUS hozzáférési pontok cégeknek

Az *ASUS* bemutatta két olyan hozzáférési pontját (acces point) – *WL-320gE* és *WL-320gP* –, melyekkel nyílt terepen akár *850 méter hatósugár* is elérhető. Azonban leginkább a többszörös *SSID*, a *VLAN* és az *SNMP* támogatás emeli ki a többi termék közül, hiszen ezek nem sok konkurens termékben találhatóak meg. A termékek minden jelenleg használt titkosítást támogatnak.

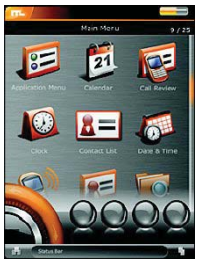
Az ipari felhasználást segíti továbbá a *WL-320gP* esetén az *Ethernet* kábelen keresztül történő táplálás (*PoE*), így az elhelyezésnél nem szükséges külön áramellátásról is gondoskodni.

Access point, többféle protokollal

A *Symbol Technologies* előreláthatóan 2007 első negyedévében dobja piacra *Linux*os vezetékmentes switch-ét, az *RFS7000*-et, mely átjárást biztosít majd a különböző technológiák, mint például az *RFID*, a *802.11a/b/g/n*, a *VoWifi*, *Wimax* között. Az ára egyelőre nem ismert.

➲ http://www.linuxdevices.com/news/NS3031866097.html

Linuxos mobil



Várhatóan 2007 első negyedévében kerül a piacra egy *Linux*os, *GPS* vevővel ellátott négysávos mobiltelefon, a *Neo1973*, mely a nyílt forrású *OpenMoKo* platformot fogja használni. A bejelentés szerint *apt-get*-szerű alkalmazás segíti majd a szoftverek telepítését. A telefon lelke egy *ARM9*-es processzor 128 megabájt memóriával és 64 megabájt flash tárhellyel, mely *microSD* kártyával – akár *1 gigabájtig* – bővíthető tovább. A *2,8 hüvelykes* érintésérzékeny kijelzőre sem lehet panasz, hiszen *480x640 képpontos* felbontást fog nyújtani. Üröm az örömben: a telefon első szériája előreláthatólag nem fog tartalmazni se *Bluetooth*-t, se *WIFI*-t. Ennek ismeretében a 350 dolláros ár nem mondható túl barátinak.

➲ http://www.linuxdevices.com/news/NS2986976174.html

Linux az E-Ten G500-as PDA-n

Bebizonyosodott, hogy a *GPS*-es *PDA* telefonon, melyen gyárilag *Windows Mobile* van, *Linux* is futtatható. A készüléken *GPE (GPE Palmtop Environment)* fut. Természetesen nem 100%-os a támogatás, hiszen pár hardver eszköz támogatása még hiányzik.

➲ http://www.linuxdevices.com/news/NS9228125055.html

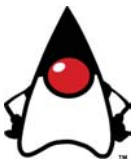
Windowshoz Linuxos tűzfal

Az izraeli székhelyű *Yoggie Security Systems* már *180 dollárért*, illetve *220 dollárért* kínál beágyazott *Linux*os *tűzfalat* (*Gatekeeper*) *Windowsos munkaállomásokhoz*. A rendszer lelke egy *Intel PXA270*-es processzor, mely kiépítéstől függően *416* vagy *624 MHz*-es, illetve memóriából és flash tárhelyből is egyaránt *64-64,* illetve *128-128* megabájttal gazdálkodhat a rendszer, azonban *SD kártyával bővíthető*. A természetesen mindkét eszköz támogatja a *VPN*-t, és tartalmaz betörés védelmet, azonban *SMTP*, *POP3 proxy*t, illetve vírusszűrést és spamszűrést csak a drágábbik modell tartalmaz. Az eszközt természetesen weben keresztül konfigurálhatjuk.

➲ http://www.linuxdevices.com/news/NS2860172381.html

Nyílt forrású Java

A *Sun* 2007-re teljesen nyílt forrásúvá teszi a *Java*-t. Eddig csupán a szerződött partnereknek volt joga felhasználni, azonban ezután bárki szabadon felhasználhatja, módosíthatja.



A nyílt forrás ellenére a *Sun* nem száll ki az üzletből, továbbra is kínál *Java*-s megoldásokat azoknak, akiknek a nyílt forrás nem megfelelő.

A *Sun* nemcsak a *Java*-t tette nyílt forrásúvá, de *Duke*-ot is, a *Java* kabaláját.

➲ http://www.pcmag.com/article2/0,1895,2058281,00.asp
➲ http://www.linuxdevices.com/news/NS6857451192.html

Papírvékony LCD

A *Samsung* bemutatta az eddig gyártott legvékonyabb *LCD*-t, amely mindössze ***0.82mm vastag***. A *Samsung* méretcsökkenés érdekében teljesen áttervezte az *LCD*-t. 2007 második negyedévétől találkozhatunk ezekkel az *LCD*-kkel mobiltelefonokban és *mp3* lejátszókban. A vastagság csökkenése ellenére megmaradt a ***300-as fényerő*** és az ***500-as kontraszt***. (Körülbelül ilyen paraméterekkel bír egy hétköznapi *LCD* monitor is.)

➲ http://www.pcmag.com/article2/0,1895,2062859,00.asp

Linux már a digitális TV-kben is

A *Toshiba* bemutatta legújabb processzorait, melyeket elsősorban alsó és középkategóriás digitális televíziókba szánnak. Az egychipes megvalósítás és a *Linux* támogatás olcsóbbá teszi a gyártást. A chip lelke egy ***162 MHz-es 64 bites RISC*** processzor, melyet a célalkalmazásnak megfelelően elsősorban videófeldolgozásra készítettek fel.

➲ http://www.linuxdevices.com/news/NS4072801160.html

75% a felső 500-ban

Idén is elkészült a szuperszámítógépek 500-as toplistája. A dolog érdekessége, hogy a *Linux*ok aránya ***75% fölötti***, míg a *Windows*t futtató szuperszámítógépek lecsúsztak a listáról. Az első helyezett gép (*IBM BlueGene/L*) *Linux*ot futtat, a számításokat 65536 darab 700 MHz-es *PPC* processzor végzi. A rendszer teljes memóriakapacitása 32 terabájt, míg a háttértár 806 terabájt.

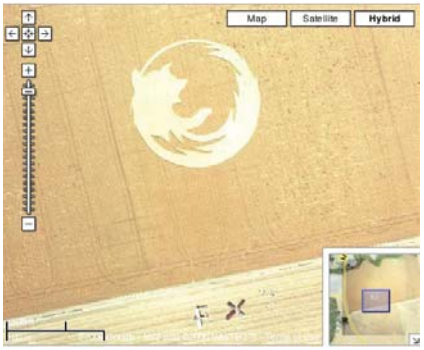
➲ http://www.llnl.gov/asc/computing_resources/bluegenel/bluegene_home.html
➲ http://top500.org/lists/2006/11/

Mono 1.2 .NET kiegészítésekkel

A *Microsoft* és a *Novell* együttműködésének gyümölcseként a *Novell* bemutatta a ***Mono 1.2-es*** verzióját, mely lehetővé teszi a *Microsoft .NET* kódjainak használatát az alkalmazásokban. Az új verzió számos újítással szolgál virtuális gépek, *Java* támogatás, memóriahasználat és stabilitás szempontjából. A friss *Mono* keretrendszer a projekt honlapjáról tölthető le.

➲ http://www.mono-project.com/downloads
➲ http://www.linuxdevices.com/news/NS4510949127.html

A Firefox logó az űrből is látszik



Az októberi hírek között adtunk számot egy gabonakörről, mely egy *Firefox logót* mintázott. Ez a *Google Maps*-en (és a *Google Earth*-en) is megtalálható. A műholdkép a gabonakör elkészítése után nemsokkal készült, hiszen a logótól nemsokkal délebbre látszik a repülőgép, amelyből fotózták, valamint öt autó is, melyek F és X betűket formáznak.

➲ http://maps.google.com/?ie=UTF8&z=18&ll=45.123437,-123.113694&t=h

SD kártya – üres területkijelzéssel

2007-től az *A-Data* cég kínál olyan *SD kártyát* is, amelyről leolvasható az üres terület. Ez például olyankor jöhet jól, ha több *SD kártyát* is használ az ember. A működési elve egyszerű: minden esetben, amikor áramot kap a kártya (*PDA*, fényképezőgép, kártyaolvasó, stb.) frissül az érték, amely kihúzás után leolvasható a kijelzőről, hiszen *csak a módosításkor fogyaszt áramot*, megjelenéskor nem.

➲ http://www.adata.com.tw/adata_en/adata_newscenter.php?news_id=178

A Zune is megadja magát?

Egykoron a *BSD*-re mondták, hogy akár a kenyérpirítón is elfut. Lelkes barkácsolók szerint a *Microsoft* médialejátszójára (*Zune*) is lehet *Linux*ot telepíteni. Az *Ipod*okra például már van *Linux*, a *Zune* miért maradnak ki? A projekt mellett szól, hogy teljesen hétköznapi processzor van benne – egy *Freescale i.MX31*, amit például a *Mobilinux* támogat –, és csábító a *WIFI*, valamint a *30 gigabájtos merevlemez* is. Egyedül talán a *Zune* 300 dolláros ára, ami visszatarthatja egy darabig a bátor barkácsolókat.

➲ http://www.networkworld.com/newsletters/linux/2006/1127linux1.html

Csokit kér a gép is

Elképzelhető, hogy nemsokára, ha merülő félben van a *laptop*unk vagy a *PDA*-nk, akkor nem kell kikapcsolnunk. Elég lesz, ha a munka közben elnassolni vágyott *csokoládé egy részét odaadjuk* a gépnek, így hosszabbítva meg az üzemidőt.
Angliában a *Birminghami Egyetemen* ugyanis sikerült bizonyos baktériumok segítségével kinyerni hidrogént édesipari hulladékból. A keletkezett hidrogént üzemanyagcellába vezetve elektromos energiát nyertek. A reakció melléktermékе csupán víz, így környezetkímélő. A kezdeti kísérletek után bizakodóak a kutatók.

➲ http://environment.about.com/od/renewableenergy/a/chocolatefuel.htm

Zenwalk 4.0

Megjelent a *4.0*-ás *Zenwalk GNU/Linux* disztribúció, melyet talán a régebbi gépek tulajdonosai fogadnak örömmel, hiszen *486-osra* van optimalizálva és *Xfce*-t használ ablakkezelőnek, így a rendszerigénye manapság minimálisnak mondható. A disztribúció *Slackware* alapú.

➲ http://www.desktoplinux.com/news/NS5275232485.html

Egy újabb Knoppix, ezúttal clusterekhez

ParallelKnoppix néven letölthető egy módosított *Knoppix live CD*, melyet *clusterek*hez ajánlanak. A *CD* személyre szabható, illetve képes a merevlemezre is lementeni a beállításokat, így nem kell minden indításkor végrehajtanunk kézzel. A *ParallelKnoppix*-hoz nem szükséges két (vagy több) számítógép, megfelelő memória és számítási kapacitás mellett *Qemu* vagy *Vmware* alatt is kipróbálható. 64 bites verzió is várható pár hónapon belül.

➲ http://www.linux.com/article.pl?sid=06/11/09/1931219

➲ http://parallelknoppix.cebacad.net/

Kábeltévés internet 100 megabit felett Dél Koreában

*Dél Koreá*ban már akár *100 megabit fölötti* sebességgel is internetezhetünk, amennyiben az *ARRIS* cég szolgáltatja a hálózatot. Noha még nem jelent meg hivatalosan a *DOCSIS 3.0* szabvány, amely akár *160 megabites letöltést* és *120 megabites feltöltést* adna, az *ARRIS* már kínál *100 megabites* csomagot. A megnövekedett sávszélesség nagy felbontású videók (*IPTV*) továbbítását is megkönnyíti. A *Föld* többi részén egyelőre kivárnak az új szabványt illetően.

➲ http://arstechnica.com/news.ars/post/20061110-8195.html

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.


Linuxvilág
www.linuxvilag.hu

Mi újság a rendszermag fejlesztése körül

Alan Cox, *Jeff Garzik* és néhány más fejlesztő egy olyan tervvel állt elő, amelynek célja az *IDE* alrendszerrel való „végleges leszámolás”. Nevezettet teljes egészében a *libata* váltaná fel. Bár ekkorát nem lehet egyszerre lépni, úgy tűnik, hogy maga az ötlet általános támogatottságot élvez. Még az *IDE* alrendszer szülőatyja, *Mark Lord* is úgy gondolja, hogy ez a jövő útja. Ezzel együtt pillanatnyilag csak annyi történik az üggyel kapcsolatban, hogy az *Andrew Morton* által fönntartott *-mm* fából több kód kerül át a hivatalos kernelforrásba, a fejlesztők figyelmét pedig felhívják arra, hogy a különböző hardvereszközökkel kapcsolatos kódrészletekben támaszkodjanak inkább az új, nagyobb tudású alrendszerre. Ami az *IDE* alrendszer végleges eltávolítását illeti, az egyelőre még évekig várat majd magára. *Alan* bejelentése tehát összességében csak az első megtett lépés egy hosszú úton.

Az *ext3* fájlrendszerből kiindulva fejlesztett *ext4* immár valóság. Egyszer volt, hol nem volt, volt egyszer egy *ext2* fájlrendszer, amihez egyesek új szolgáltatásokat, például naplózást kívántak írni. Történt azonban, hogy ezt a művi beavatkozást az eredeti fejlesztők annyira veszélyesnek vélték, hogy a haladó gondolkodásúak kénytelenek voltak egy új fejlesztési vonalat kezdeni *ext3* néven. Telt múlt az idő, és újabb bátor emberek jelentek meg, akik az immár kipróbált *ext3*-ba olyan szörnyűséges dolgokat akartak beleírni, mint az *extent*-ek meg a nagy lemezblokkok (*large block sizes*) kezelése. Így aztán nekik is pont úgy kellett eljárniuk, mint egykoron elődeiknek.

Linus Torvaldsnak az volt a határozott véleménye, hogy egy operációs rendszerben a legáltalánosabban használt fájlrendszer nem lehet a fejlesztők játékszere, annak meg kell maradnia a stabil állapotában, hogy továbbra is bizton támaszkodni lehessen rá. Ezzel együtt az *ext4* már jó úton halad a hivatalos kernelbe való bebocsáttatás felé. Aztán hogy eléri-e valaha a népszerűségnek azt a fokát, amit az *ext3*-nak sikerült, azt egyelőre nem lehet tudni. Mindeközben egyesek, például *Hans Reiser* úgy vélik, hogy az *ext4* túlságosan is könnyen került be a hivatalos rendszermagba, ami ismét csak azt bizonyítja, hogy a fejlesztői közösség mennyi részrehajló tud lenni. *Hans* azonban a jelek szerint nem ért valamit: az intelligencia és a programozói tudás a kernelfejlesztők kultúrájának csak egy része. Elvégre az igazán nagy dolog, amire *Linus* rájött az volt, hogy szinte bárki képes valami hasznosat hozzátenni a műhöz, vagyis a kernelfejlesztőknek nem muszáj mind „überhackereknek” lenniük. Sőt, még csak szépfiúknak se kell lenniük, amint arra *Alexander Viro* és néhányan mások olyan büszkén rámutattak. Ugyanakkor alapkövetelmény velük szemben, hogy értelmesen válaszoljanak a visszajelzésekre, a munkájukat pedig valamiféle többé-kevésbé „hagyományos módon” tárják a nagyközönség elé.

A játék ugyanis valahogy úgy működik, hogy minél inkább bízik a közösség abban, hogy valaki jól dolgozik és jó dolgokat csinál, annál könnyebb dolga van az illetőnek, amikor a munkája felvételét kéri a rendszermagba.

Új, stabil kernelként *Adrian Bunk* fogja fönntartani a 2.6.16-os rendszermag forrásfáját. Ezzel gyakorlatilag visszatért a régi stabilitásra utaló számozási rendszer, hiszen míg *Adrian* a 2.6-os fát kezeli, addig *Linus* 2.7-es kódszámmal intenzíven fejleszteni kezdett egy új magot. Az egyetlen lényeges eltérés a fák elnevezése, illetve az, hogy a korábbiaktól eltérően maga *Linus* egyáltalán nem vesz majd részt a 2.6-os fa „stabilizálási folyamatában”. *Adrian* munkája várhatóan megold majd néhány a felhasználók által a *Greg Kroah-Hartman* és *Chris Wright* által fönntartott *w.x.y.z* stabil fával kapcsolatban jelzett problémát. Ez az utóbbi fa ugyanis amellett, hogy fönntartói a folyamatos stabilitásra törekedtek azt nem is próbálta meggátolni, hogy a 2.6-os mag különböző változatai között megváltozzanak bizonyos programozási felületek. Az *Adrian* által fönntartott fánál ez már alapkövetelmény lesz, ami önmagában csökkentheti a fölmerülő problémák számát.

Pavel Machek elkészítette a *ThinkPad*-okon található ujjlenyomat-olvasó meghajtóját. Az első felhasználók viszonylagos sikerekről számoltak be vele kapcsolatban, bár egyelőre vannak bizonyos elég könnyen reprodukálható hibái. *Pavel* számára a legnagyobb kérdés pillanatnyilag az, hogy hagyja meg a meghajtót a felhasználói térben, vagy helyezze át a magtérbe. Ez egy érdekes kérdés, hiszen az alapelvek szerint mindent, amit egyáltalán kívül lehet hagyni a kernel saját terén, az kívül is kell. Ugyanakkor az is teljesen általános, hogy a hardvereszközöket támogató alrendszerek a mag részét képezik, eltekintve persze néhány különleges kivételtől. Egyszóval *Pavel* kódjának alakulása egyben azt is eldöntheti, hogy mi lesz a jövőben a sorsa az egyéb meghajtóknak.

Az *Intelnél* dolgozó *Keith Packard* olyan nyílt forrású meghajtókat jelentett be, amelyekkel az *Intel 965 Express* lapkakészlettel szerelt alaplapokon található grafikus chip működtethető. Ez a lépés egyébként része annak a munkának, amely az *Intel Open Source Technology Center* nevű laboratóriumában folyik. Az *Intel* úgy tűnik a legmegfelelőbb magatartást tanúsítja, vagyis belátta, hogy minden hatékony kódnak szüksége van tesztelésre, illetve hogy a felhasználói visszajelzések értékesek. Éppen ezért várja a kernelfejlesztők véleményét, javaslatait és egyéb hozzájárulásait. Van azért egy olyan érdekesség, amire azonnal felfigyelt a kernellel kapcsolatos levelezési lista közönsége is. A kód helyenként egy nyilvánosan nem hozzáférhető *intel_hal.so* nevű binárisra támaszkodik. *Keith* persze azonnal megindokolta a dolgot: ebben a fájlban vannak azok a kódrészletek, amelyek forrását az *Intel* nem adhatja ki. Ilyen például a *Macrovision* regiszterekkel kapcsolatos anyag, illetve néhány egyéb ipari titok. Ezeknek a használata ugyanakkor opcionális, vagyis ha valaki nem szeretne olyan binárist látni a rendszerében, amelynek nem rendelkezik a forrásával, vagy amit az *Intel* ügynökei írtak, az megteheti. *Keith* szerint ettől a meghajtó még teljesen működőképes marad és a funkciókészlete sem csorbul az előző kiadásokhoz képest.

Linux Journal 2006., 152. szám

Zack Brown

Újabb levél Bill Gates-nek

Kedves Bill!

Remélem immár a nyugdíjas éveidet élvezed. Ami engem illet, én – természetesen – továbbra is elfoglalt vagyok, hiszen tolnom kell a nyílt forrás szekerét.

Nemrég azt olvastam a *Wall Street Journalban*, hogy te is ráéreztél az információ szabadságában rejlő lehetőségekre. A cikk szerint megköveteled, hogy azok a kutatók, akik támogatást kapnak az általad létrehozott alaptól, másokkal is megosszák mindazt, amit létrehoznak. Király! Lefogadom, most meg vagy róla győződve, hogy ezt is te fedezted fel, de el kell hogy keserítselek. Az a helyzet, hogy amit kitaláltál, az valójában a *Szabad Szoftver Mozgalom* legfőbb alapelve, minek következtében mi már vagy 35 éve így csináljuk a dolgainkat.

Emlékszem, még valamikor 1969-ben, amikor hallgató volt a *Drexel Egyetemen*, a főépület alagsorában találtam pár számítógépet, amiket szoftver nélkül szállítottak. Ahhoz, hogy valamire is használni tudjam őket, vagy meg kellett írnom rájuk a megfelelő programokat, vagy meg kellett vennem őket.

Abban az időben egyetlen fordítóprogram körülbelül 100.000 dollárba került, és az még az az idő volt, amikor százezer nem kis pénznek számított. Ezek után talán nem meglepő, hogy a kevéske ösztöndíjamból inkább ennivalót meg sört vettem, mára úgyse lett volna elég. De akkor jött a meglepetés: a *Digital Equipment* felhasználói közösségében volt egy csomó olyan ember, akik programokat írtak, majd átadták azt a közösség könyvtárának, hogy mások is szabadon hozzáférhessenek. Számomra ezek a szoftverek tették lehetővé, hogy informatikát tanuljak. És ez az, amit soha nem felejtettem el. Veled gondolom egész más volt a helyzet, nem volt efféle felemelő élményed a dologgal kapcsolatban. Te eleve a *Harvardra* mentél tanulni, a fordítóprogramot meg egyszerűen megengedhetted magadnak. Persze azt is el tudom képzelni, hogy egyszerűen csak mások gépeit és szoftvereit használtad ahhoz, hogy elvégezhesd a munkádat.

Akárhogy is volt, ami engem illet, miután elhagytam az iskolát, a való életben kötöttem ki, de azt már pontosan tudtam, hogy csapatban dolgozni sokkal jobb, mint egyedül. Így aztán magam is folytattam a nyilvános kódok fejlesztését. Sokszor csak egy-egy részletet tettem közkinccsé, de néha egész programokat. És biztos vagyok benne, hogy ezzel hozzájárultam ennek az iparágnak a fejlődéséhez. Végezetül támadt néhány nagy ötletem. Íme:

1. Minden olyan szoftver, ami a támogatásoddal készül, legyen szabad szoftver.

2. A saját munkádhoz is használj kizárólag szabad szoftvereket.

3. Csak olyan orvosi műszerekre bízd az életedet, amit szabad szoftver vezérel.

Biztos vagyok benne, hogy te is pillanatok alatt átlátod majd, miként illeszkednek ezek az elvek ahhoz a kalandhoz, amibe épp most kezdtél bele.

A legjobbakat!
Jon „maddog” Hall

Mi újság a rendszermag fejlesztése körül

Alan Cox, *Jeff Garzik* és néhány más fejlesztő egy olyan tervvel állt elő, amelynek célja az *IDE* alrendszerrel való „végleges leszámolás”. Nevezettet teljes egészében a *libata* váltaná fel. Bár ekkorát nem lehet egyszerre lépni, úgy tűnik, hogy maga az ötlet általános támogatottságot élvez. Még az *IDE* alrendszer szülőatyja, *Mark Lord* is úgy gondolja, hogy ez a jövő útja. Ezzel együtt pillanatnyilag csak annyi történik az üggyel kapcsolatban, hogy az *Andrew Morton* által fönntartott *-mm* fából több kód kerül át a hivatalos kernelforrásba, a fejlesztők figyelmét pedig felhívják arra, hogy a különböző hardvereszközökkel kapcsolatos kódrészletekben támaszkodjanak inkább az új, nagyobb tudású alrendszerre. Ami az *IDE* alrendszer végleges eltávolítását illeti, az egyelőre még évekig várat majd magára. *Alan* bejelentése tehát összességében csak az első megtett lépés egy hosszú úton.

Az *ext3* fájlrendszerből kiindulva fejlesztett *ext4* immár valóság. Egyszer volt, hol nem volt, volt egyszer egy *ext2* fájlrendszer, amihez egyesek új szolgáltatásokat, például naplózást kívántak írni. Történt azonban, hogy ezt a művi beavatkozást az eredeti fejlesztők annyira veszélyesnek vélték, hogy a haladó gondolkodásúak kénytelenek voltak egy új fejlesztési vonalat kezdeni *ext3* néven. Telt múlt az idő, és újabb bátor emberek jelentek meg, akik az immár kipróbált *ext3*-ba olyan szörnyűséges dolgokat akartak beleírni, mint az *extent*-ek meg a nagy lemezblokkok (*large block sizes*) kezelése. Így aztán nekik is pont úgy kellett eljárniuk, mint egykoron elődeiknek.

Linus Torvaldsnak az volt a határozott véleménye, hogy egy operációs rendszerben a legáltalánosabban használt fájlrendszer nem lehet a fejlesztők játékszere, annak meg kell maradnia a stabil állapotában, hogy továbbra is bizton támaszkodni lehessen rá. Ezzel együtt az *ext4* már jó úton halad a hivatalos kernelbe való bebocsáttatás felé. Aztán hogy eléri-e valaha a népszerűségnek azt a fokát, amit az *ext3*-nak sikerült, azt egyelőre nem lehet tudni. Mindeközben egyesek, például *Hans Reiser* úgy vélik, hogy az *ext4* túlságosan is könnyen került be a hivatalos rendszermagba, ami ismét csak azt bizonyítja, hogy a fejlesztői közösség mennyi részrehajló tud lenni. *Hans* azonban a jelek szerint nem ért valamit: az intelligencia és a programozói tudás a kernelfejlesztők kultúrájának csak egy része. Elvégre az igazán nagy dolog, amire *Linus* rájött az volt, hogy szinte bárki képes valami hasznosat hozzátenni a műhöz, vagyis a kernelfejlesztőknek nem muszáj mind „überhackereknek” lenniük. Sőt, még csak szépfiúknak se kell lenniük, amint arra *Alexander Viro* és néhányan mások olyan büszkén rámutattak. Ugyanakkor alapkövetelmény velük szemben, hogy értelmesen válaszoljanak a visszajelzésekre, a munkájukat pedig valamiféle többé-kevésbé „hagyományos módon” tárják a nagyközönség elé.

A játék ugyanis valahogy úgy működik, hogy minél inkább bízik a közösség abban, hogy valaki jól dolgozik és jó dolgokat csinál, annál könnyebb dolga van az illetőnek, amikor a munkája felvételét kéri a rendszermagba.

Új, stabil kernelként *Adrian Bunk* fogja fönntartani a 2.6.16-os rendszermag forrásfáját. Ezzel gyakorlatilag visszatért a régi stabilitásra utaló számozási rendszer, hiszen míg *Adrian* a 2.6-os fát kezeli, addig *Linus* 2.7-es kódszámmal intenzíven fejleszteni kezdett egy új magot. Az egyetlen lényeges eltérés a fák elnevezése, illetve az, hogy a korábbiaktól eltérően maga *Linus* egyáltalán nem vesz majd részt a 2.6-os fa „stabilizálási folyamatában”. *Adrian* munkája várhatóan megold majd néhány a felhasználók által a *Greg Kroah-Hartman* és *Chris Wright* által fönntartott *w.x.y.z* stabil fával kapcsolatban jelzett problémát. Ez az utóbbi fa ugyanis amellett, hogy fönntartói a folyamatos stabilitásra törekedtek azt nem is próbálta meggátolni, hogy a 2.6-os mag különböző változatai között megváltozzanak bizonyos programozási felületek. Az *Adrian* által fönntartott fánál ez már alapkövetelmény lesz, ami önmagában csökkentheti a fölmerülő problémák számát.

Pavel Machek elkészítette a *ThinkPad*-okon található ujjlenyomat-olvasó meghajtóját. Az első felhasználók viszonylagos sikerekről számoltak be vele kapcsolatban, bár egyelőre vannak bizonyos elég könnyen reprodukálható hibái. *Pavel* számára a legnagyobb kérdés pillanatnyilag az, hogy hagyja meg a meghajtót a felhasználói térben, vagy helyezze át a magtérbe. Ez egy érdekes kérdés, hiszen az alapelvek szerint mindent, amit egyáltalán kívül lehet hagyni a kernel saját terén, az kívül is kell. Ugyanakkor az is teljesen általános, hogy a hardvereszközöket támogató alrendszerek a mag részét képezik, eltekintve persze néhány különleges kivételtől. Egyszóval *Pavel* kódjának alakulása egyben azt is eldöntheti, hogy mi lesz a jövőben a sorsa az egyéb meghajtóknak.

Az *Intelnél* dolgozó *Keith Packard* olyan nyílt forrású meghajtókat jelentett be, amelyekkel az *Intel 965 Express* lapkakészlettel szerelt alaplapokon található grafikus chip működtethető. Ez a lépés egyébként része annak a munkának, amely az *Intel Open Source Technology Center* nevű laboratóriumában folyik. Az *Intel* úgy tűnik a legmegfelelőbb magatartást tanúsítja, vagyis belátta, hogy minden hatékony kódnak szüksége van tesztelésre, illetve hogy a felhasználói visszajelzések értékesek. Éppen ezért várja a kernelfejlesztők véleményét, javaslatait és egyéb hozzájárulásait. Van azért egy olyan érdekesség, amire azonnal felfigyelt a kernellel kapcsolatos levelezési lista közönsége is. A kód helyenként egy nyilvánosan nem hozzáférhető *intel_hal.so* nevű binárisra támaszkodik. *Keith* persze azonnal megindokolta a dolgot: ebben a fájlban vannak azok a kódrészletek, amelyek forrását az *Intel* nem adhatja ki. Ilyen például a *Macrovision* regiszterekkel kapcsolatos anyag, illetve néhány egyéb ipari titok. Ezeknek a használata ugyanakkor opcionális, vagyis ha valaki nem szeretne olyan binárist látni a rendszerében, amelynek nem rendelkezik a forrásával, vagy amit az *Intel* ügynökei írtak, az megteheti. *Keith* szerint ettől a meghajtó még teljesen működőképes marad és a funkciókészlete sem csorbul az előző kiadásokhoz képest.

Linux Journal 2006., 152. szám

Zack Brown

Újabb levél Bill Gates-nek

Kedves Bill!

Remélem immár a nyugdíjas éveidet élvezed. Ami engem illet, én – természetesen – továbbra is elfoglalt vagyok, hiszen tolnom kell a nyílt forrás szekerét.

Nemrég azt olvastam a *Wall Street Journalban*, hogy te is ráéreztél az információ szabadságában rejlő lehetőségekre. A cikk szerint megköveteled, hogy azok a kutatók, akik támogatást kapnak az általad létrehozott alaptól, másokkal is megosszák mindazt, amit létrehoznak. Király! Lefogadom, most meg vagy róla győződve, hogy ezt is te fedezted fel, de el kell hogy keserítselek. Az a helyzet, hogy amit kitaláltál, az valójában a *Szabad Szoftver Mozgalom* legfőbb alapelve, minek következtében mi már vagy 35 éve így csináljuk a dolgainkat.

Emlékszem, még valamikor 1969-ben, amikor hallgató volt a *Drexel Egyetemen*, a főépület alagsorában találtam pár számítógépet, amiket szoftver nélkül szállítottak. Ahhoz, hogy valamire is használni tudjam őket, vagy meg kellett írnom rájuk a megfelelő programokat, vagy meg kellett vennem őket.

Abban az időben egyetlen fordítóprogram körülbelül 100.000 dollárba került, és az még az az idő volt, amikor százezer nem kis pénznek számított. Ezek után talán nem meglepő, hogy a kevéske ösztöndíjamból inkább ennivalót meg sört vettem, mára úgyse lett volna elég. De akkor jött a meglepetés: a *Digital Equipment* felhasználói közösségében volt egy csomó olyan ember, akik programokat írtak, majd átadták azt a közösség könyvtárának, hogy mások is szabadon hozzáférhessenek. Számomra ezek a szoftverek tették lehetővé, hogy informatikát tanuljak. És ez az, amit soha nem felejtettem el. Veled gondolom egész más volt a helyzet, nem volt efféle felemelő élményed a dologgal kapcsolatban. Te eleve a *Harvardra* mentél tanulni, a fordítóprogramot meg egyszerűen megengedhetted magadnak. Persze azt is el tudom képzelni, hogy egyszerűen csak mások gépeit és szoftvereit használtad ahhoz, hogy elvégezhesd a munkádat.

Akárhogy is volt, ami engem illet, miután elhagytam az iskolát, a való életben kötöttem ki, de azt már pontosan tudtam, hogy csapatban dolgozni sokkal jobb, mint egyedül. Így aztán magam is folytattam a nyilvános kódok fejlesztését. Sokszor csak egy-egy részletet tettem közkinccsé, de néha egész programokat. És biztos vagyok benne, hogy ezzel hozzájárultam ennek az iparágnak a fejlődéséhez. Végezetül támadt néhány nagy ötletem. Íme:

1. Minden olyan szoftver, ami a támogatásoddal készül, legyen szabad szoftver.

2. A saját munkádhoz is használj kizárólag szabad szoftvereket.

3. Csak olyan orvosi műszerekre bízd az életedet, amit szabad szoftver vezérel.

Biztos vagyok benne, hogy te is pillanatok alatt átlátod majd, miként illeszkednek ezek az elvek ahhoz a kalandhoz, amibe épp most kezdtél bele.

A legjobbakat!
Jon „maddog” Hall

B.U.É.K.!

Ismét öregebbek lettünk egy évvel. Minden ruhánk tavalyi, minden számlánk új, s tudjuk, a holnap sohasem jön el – de a másnap igen.

Újévi kedves szokás szerint homlokokat tartunk csészék felett, majd mindenféle szépeket és jókat kívánunk szeretteinknek. E kívánságok közvetítésére a postagalambot már régen felváltotta az e-mail és az *SMS*, de legújabban ismét egy madár segítségét kérhetjük: üzenjünk pingvinnel!

A ➲ ***www.paperboy.nl*** oldalon egy jeges hegycsúcs tetejéről indul őrült útjára az üzenetközvetítő madár; hason száguldva lefelé a mélybe ide-oda cikáz, hogy behasalva és felszántva a tájat, az út végén kirajzolódhasson a hóba írt szövegecske. Tán praktikus és spórolós eme szolgáltatást igénybe venni a gázártámogatási kérelemnél is, felhívva figyelmét a köz szolgálójának, hogy befagyhat a hátsónk. (Persze az is igaz, hogy amikor seggbe rúgnak, mégiscsak előbbre jutunk egy lépéssel… Mert amint megmondjuk, mire van szükségünk, rögtön megfelelik, hogyan lehetünk meg nélküle.)

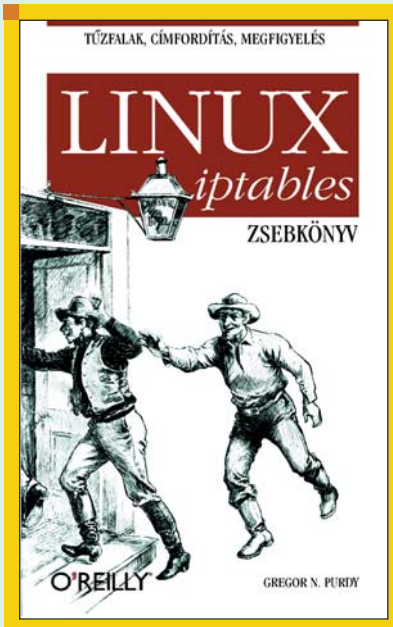
De az új esztendő hozzon pozitív gondolkodást! Ha a padlóra kerülünk, legalább szedjünk fel onnét valamit. Így tett ***Lala*** is, a japán pingvin, ki a civilizációba csöppenve nem bízza vacsoráját avatatlan kezekre. Stílszerűen pingvines hátizsákját felkapkodva nekiindul a macskakőnek, és még ha nem is tartja be a kötelező haladási irányt, céltudatosan és magabiztosan haladva elbattyog a sarki boltba. Bizonyára nem mindennapos egy élelmiszerüzletben, hogy egy pingvin jelenik meg a mirelitárunál, így megesik, hogy az eladó még ellenárat is elfelejt kérni az elemózsiáért.

Jól végezvén dolgát ***Lala*** a visszaútra tér, majd családi körbe érkezve lemosásk-slagozzák róla az út porát.

➲ ***http://subba.blog.hu/2006/11/13/szupercukisag_a_maximumon_a_bevasarlo_pi***

Halusz Léna

Linux iptables zsebkönyv – kicsi a bors, de tömény



Cím: Linux iptables zsebkönyv – Tűzfalak, címfordítás, forgalomszámlálás
Szerző: Gregor N. Purdy
Kiadó: Kiskapu (O'Reilly)
Oldalszám: 121
Ár: 1800 Ft

Manapság egyre nagyobb hangsúlyt kap az internetes biztonság, így ha valaki linuxos rendszereket üzemeltet, vagy egyszerűen csak az asztali gépét szeretné biztonságban tudni, nem mehet el mellette. Ez a könyv elsősorban a linuxos rendszergazdák életét hivatott megkönnyíteni.
A ***Linux*** (is) rendszerszinten biztosít tűzfalfunkciókat (ezt a 2.2-es kernelig ***ipchains-nek***, a 2.4-től azonban már ***iptables***-nek nevezik), melyekkel – feltételezve persze a megfelelő hozzáértést – szinte bármilyen bonyolultságú tűzfalat építhetünk. A legnagyobb problémát talán az jelenti, hogy a beállításokat, tűzfalszabályokat nem egy grafikus felületen kattintgatva adhatjuk meg. Ugyanakkor ennek a módszernek is megvannak a maga előnyei, sőt egyesek szerint még szép is.
Kétféle rendszergazda létezik. Az egyik típushoz azok tartoznak, akik fejből tudják az összes ***iptables*** kapcsolót, portszámokat és időnként gondolatátvitellel állítják be a tűzfalat. A másik kategóriát azok alkotják, aki folyton a súgóoldalakat (***man***) bújják, a ***Google***-on keresgélnek, és így próbálnak célt érni.
Nos, ez a zsebkönyv inkább a második csoportnak szól, hiszen már az elején látványos diagramokkal szemlélteti a ***INPUT***, ***FORWARD***, ***OUTPUT***, ***PREROUTING***, ***POSTROUTING*** kapcsolódási pontokat. A „vizuális típusok" így könnyebben átlátják ezek lényegét, és az adatfeldolgozási sorban elfoglalt helyüket. A könyvben jó pár hasznos táblázat is található, amelyek gyakran könnyítik majd meg az életet. Ilyen „kincs" például egy-egy rövidke összefoglalás a gyakori ***portszámokról***, az ***ICMP*** kódokról, és az ehhez hasonlókról.
A könyv alapvetően három részből áll össze.
Az első harminc oldalon a különböző megoldások találhatóak:

- Forgalomszámlálás
- Címfordítás
- Terheléselosztás
- Hasznos segédprogramok

A második – mintegy 80 oldalas – részben találja meg az olvasó az ***iptables*** modulokat és kapcsolókat, illetve ezek leírásait.
A harmadik, egyben utolsó részben az ***iptables-restore*** és ***iptables-save*** kerül terítékre, melyekkel a tűzfalszabályok lementhetőek illetve – akár egy másik gépen – visszatölthetőek. Az így lementett tűzfalszabályok amúgy akár ***CVS***-ben, vagy más verziókövető rendszerben is tárolhatók.
Nem lenne a könyv teljes egy jól használható tartalomjegyzék és a tárgymutató nélkül. Az utóbbi a könyv terjedelméhez képest igencsak részletes, hét oldalt foglal el, de efféle összefoglalásnál ez nemhogy természetes, hanem a keresgélés során kifejezetten hasznos is.
A könyvre összességében inkább gyors referenciaként kell tekinteni, vagyis ebből nem biztos, hogy meg lehet tanulni a tűzfalak alapjait. Alapozás céljából inkább a ***Linuxvilág*** archívumot ajánlanám vagy esetleg a megfelelő ***HOGYAN***-t. Viszont olyankor jól jöhet, amikor a ***Google*** nem elérhető – mondjuk épp egy hibás tűzfal beállítás miatt – vagy nincs idő a ***man*** oldalon bogarászni a dokumentációt vagy esetleg az egyik kollégának kell távsegítséget adni.
Amire viszont tökéletes: a már megszerzett tudás finomítása. A méretéből fakadóan szinte bármikor elővehetjük. Nincs az a kabátzseb vagy laptoptáska, ahova be ne férne. Persze nem azt mondom, hogy szórakoztatóbb, mint egy ***Garfield*** zsebkönyv, de kétségkívül hasznosabb.

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.

Informatika érettségi egy szabadabb környezetben

Immáron hivatalosan is választhatóak magyar fejlesztésű Linux disztribúciók a 2007. május-júniusi vizsgaidőszak kétszintű informatika érettségi vizsgáihoz, köszönhetően az Oktatási és Kulturális Minisztérium valamint az OKÉV együttműködési megállapodásának. Az érintett UHU-Linux és SuliX rendszereket a Magyar Linux Alapítvány illetve az ULX Kft. térítésmentesen bocsátja az iskolák, tanárok, tanulók és vizsgázók rendelkezésére.

Egylemezes változatok lévén alapkiszerelésben ugyan nem sok alternatívát kínálnak, de tartalmazzák az informatika érettségi vizsgán használatos szoftvereket, mint például az *OpenOffice.org* illetve a *Gimp is*. Így legalább ezeket nem szükséges külön beszereznünk és a telepítésükkel bajlódnunk. Természetesen amennyiben később kedvet kapunk hozzá mindkét rendszer tetszőlegesen bővíthető további csomagokkal.
Míg az ***UHU*** leginkább a mindennapok ***Linux*** rendszerének mondható, addig a ***SuliX*** több szempontból is speciálisan oktatási célokra lett fejlesztve. Talán tanulásra ez lehet a legoptimálisabb választás, bár kétség kívül régebbi verziós alkalmazásokat tartalmaz, mint a most megjelent ***UHU-Linux 2.0***. A CD-képfájlok letölthetők az ***OKÉV*** weboldaláról egy-egy rövid tájékoztatóval együtt, amik segítenek az első lépésekben.

1. ábra UHU-Linux első látásra

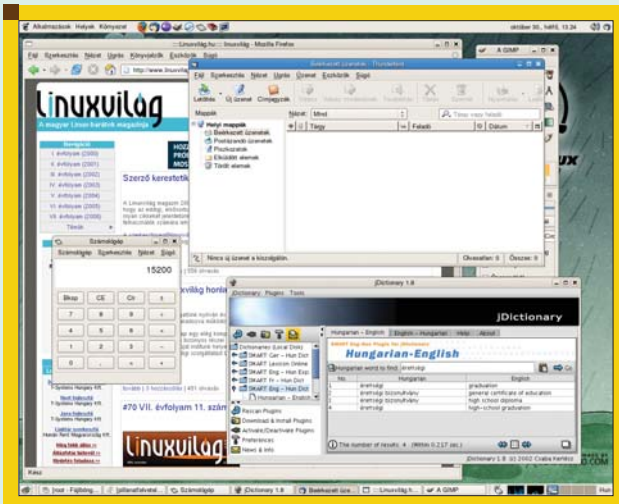
Próba cseresznye

Aki nem biztos a dolgában, de mégis szívesen kipróbálná valamelyik rendszert, annak az úgynevezett ***Live*** kiadásokat ajánlom közelebbi ismerkedésre. A ***SuliX*** ebben is remekel. Ezeket egyetlen ***CD*** lemezről bootolva telepítés nélkül futtathatjuk, régi rendszerünk pedig érintetlen marad. Így mindenki különösebb kockázat nélkül bizonyosodhat meg róla, hogy számára melyik a legmegfelelőbb. Amennyiben lehetőségünk van rá több rendszert is érdemes kipróbálni, hogy szélesebb körben megismerjük a különböző ***Linux*** disztribúciók bizonyos előnyeit vagy hátrányait.

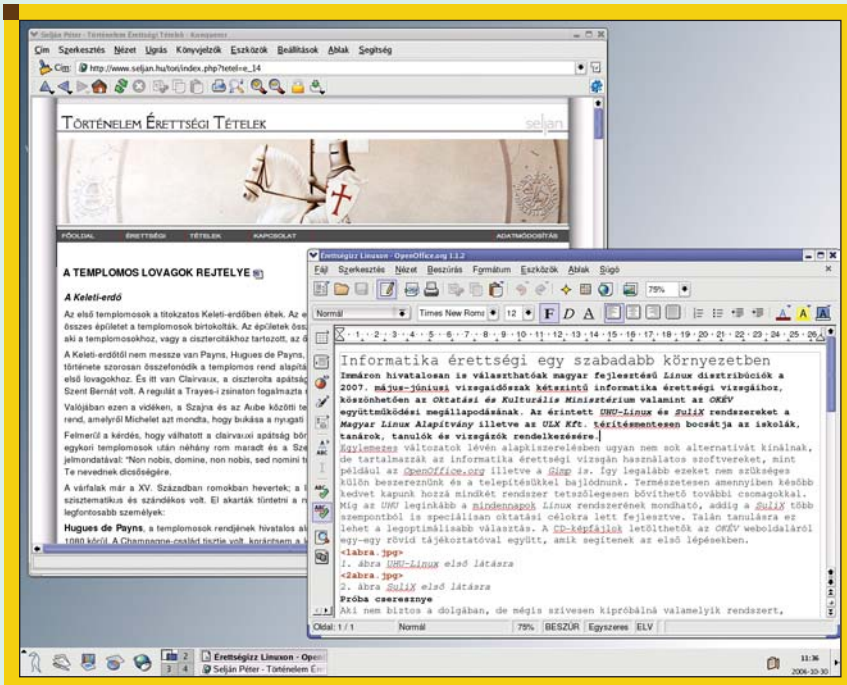
Keleten a helyzet

Sajnos hazánkban egyelőre még mindig csekély a ***Linux*** rendszert vagy egyéb szabad szoftvert használó intézmények száma. Reméljük ez a szám a jövőben csak emelkedni fog, és egyre többen fogják kihasználni a szabad szoftver nyújtotta megbízhatóságot, stabilitást és nem utolsó sorban ingyenességet, ami jelentős anyagi megtakarításokat eredményezhet az intézményeknek bármely szférában. Ugyan eddig is volt lehetőség szabad szoftveres környezetben vizsgázni informatikából, most azonban

2. ábra SuliX első látásra



3. ábra UHU-Linux a mindennapokban



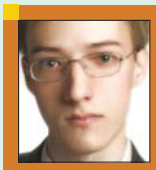
4. ábra SuliX első sorban oktatási célokra

konkrétan az érettségi vizsgára szánt, magyar fejlesztésű rendszereket is választhatunk. Az érettségi vizsgákat szervező intézményeknek és informatika-oktatóiknak is fel kell készülniük a megfelelő szoftver környezet biztosítására és a diákok számára való segítségnyújtásra.
Ez várhatóan tovább hajtja majd a szabad szoftver szekerét. Valamelyest oldja a *Linux* rendszerekkel szemben kialakult „ismeretlentől való félelmet" a diákokban esetleg a tanárokban, és idővel komolyabb szerepet kaphatnak az ilyen alapú szoftverek az oktatás más területein is.

Különvélemény

A szabad szoftver létjogosultságát erősítendő, hogy az informatika, mint tantárgy oktatása nem konkrét szoftverek használatának tanítását jelenti. Legalábbis nem ezt kéne jelentenie. A szövegszerkesztéshez, táblázatkezeléshez szükséges készségek az *OpenOffice.org* segítségével épp olyan könnyedén és érthetően elsajátíthatók, mint a *Microsoft Word* és *Excel* vagy éppen más alkalmazás használatával. Márpedig akkor a legértékesebb a megszerzett tudás, ha később más környezetben is képesek a diákok hasznosítani azt.

A rendszerek közti különbségeket pedig mindenki a maga előnyére fordíthatja.
A tavaszi vizsgákig még rengeteg idő van a nekünk megfelelő szoftver kiválasztására és a felkészülésre. Remélem az iskolapadokban ülve minél többen kedvet kapnak ahhoz, hogy valami régit, de mégis újat tanuljanak. Sikeres felkészülést kívánok mindenkinek!



Selján Gábor
(gabor@seljan.hu)

Szabadúszó Flash webfejlesztő és a szabadidejében hobbi Linux felhasználó.

KAPCSOLÓDÓ CÍMEK

UHU-Linux letöltések

Office 2.0
➲ http://www.uhulinux.hu/office/letoltes

Érettségi 2.0
➲ http://www.om.hu/main.php?folderID=266

Live 2.4
➲ http://ftp.uhulinux.hu/uhu/live/2.4/

SuliX letöltések

Professional 4.0
➲ http://www.om.hu/main.php?folderID=266

Live 2.0
➲ http://ftp.fsn.hu/pub/CDROM-Images/sulix/2.0/

Teljes körű kép vállalata IT vagyonáról Novell ZENworks Asset Management – nem csak a szoftvergazdálkodást valósítja meg, de egyéb költségmegtakarítást is eredményezhet

A mai gazdálkodási környezetben elengedhetetlen, hogy a rendelkezésünkre álló informatikai erőforrásokat jogszerűen és hatékonyan használjuk fel. Ebben segít az üzemeltetett IT infrastruktúra pontos felmérését és használatkövetését megvalósító megoldás, a hazai BSA által is elfogadott vagyongazdálkodási szoftver, a Novell ZENworks Asset Management.

A termék vagyonleltárt, szoftverhasználati információkat és licencegyeztetést biztosít, ezzel teljes és pontos képet ad a rendelkezésre álló telepített szoftverekről és licencekről. Használatával – a szoftverkiadások szabályozása érdekében – a vállalat egészében nyomon követhetik a rendszerinformációkat és az eszközöket, valamint biztosíthatják a szoftverlicencek törvényi szabályozásoknak való megfelelését, így elkerülhető az illegális szoftverhasználat veszélye.

A hatékony és jogszerű licencgazdálkodás a vagyon pontos felmérésével kezdődik

A hardver, szoftver és vásárlási adatok kombinált lekérdezése lehetővé teszi, hogy a cég teljes körű képet kapjon *IT* vagyonáról. Az ***Asset Inventory*** részét képező szabadalmazott ***ZENworks Recognition Technology*** (***ZENworks*** felismerési technológia) – amely az utóbbi 14 évben több mint 10 millió munkaállomáson bizonyította, hogy megbízható és pontos – automatikusan felismeri és leltározza a szoftver- és hardverelemeket, valamint részletes információkat közöl a vállalat munkaállomásairól, szervereiről és hálózatairól. Rendelkezésére állnak a vállalat licenc megfelelőség biztosításához szükséges adatai, megtakarításokat érhet el a szoftverlicencelés és támogatás terén, és más kritikus fontosságú *IT* projektekre koncentrálhat, többek között a verzióváltásokra, a haszonbérletek kezelésére, a felhasználó támogatásra, a költségvetésre, tervezésre és a vagyontárgyak újraelosztására.

Licenckezelés

A termék szoftvermegfelelőséget biztosító összetevője a telepített

A szoftverkészlet pontos felmérése nem csak a szoftvergazdálkodást valósítja meg, de egyéb költségmegtakarítást is eredményezhet

A vállalatok és intézmények számára fontos kérdés, hogy az informatikai rendszerükben használt szoftverek jogtiszták és legálisak legyenek. Ennek ellenőrzését hazánkban a ***BSA (Business Software Alliance)*** végzi, legújabb ***„NagyVizIT”*** kampányuk során több száz kis-és középvállalatot, valamint önkormányzatot látogatnak meg, hogy ellenőrizzék az általuk használt szoftverek legalitását. A Novell a kampányban nem vesz részt, vagyongazdálkodási megoldásával azonban segítséget nyújt a felhasználóknak abban, hogy minél kisebb ráfordítással képesek legyenek a megfelelőség bizonyítására. A ***Novell ZENworks Asset Management*** a ***BSA*** által is elfogadott, a piacon egyedülálló informatikai vagyonkezelési szoftver, amely a teljes körű hardver/szoftver eszközleltár, a valós online szoftverhasználati információk, valamint a licencek egységes kezelését valósítja meg. A ***Novell ZENworks Asset Managementtel*** csökkentheti licenckköltségeit és nyugodtan várhatja az esetleges ellenőrzéseket. A ***ZENworks Asset Management*** egy 100 felhasználós tipikus környezetben egy nap alatt telepíthető, és windowsos környezetekben is kiválóan futtatható. A rendszer gyors és hatékony használatba vételéhez a ***Novell*** tanácsadó részlege ingyenes elektronikus bevezetési támogatást is biztosít. A ***Novell ZENworks Asset Management*** segítségével könnyedén, naprakészen számon tartható a szoftverleltár.

szoftvereket rendeli össze a licencjogosultságokkal egy kölcsönösen egyértelmű kapcsolatba, így valódi „licenc nézetet” biztosít, tehát a szoftvereket a licencek alapján is felmérhetjük. Továbbá a program a licencbejegyzések megtekintéséhez web alapú nézetet biztosít, így mindig nyomon követhetők, az aktuális megfelelőségi állapotok.

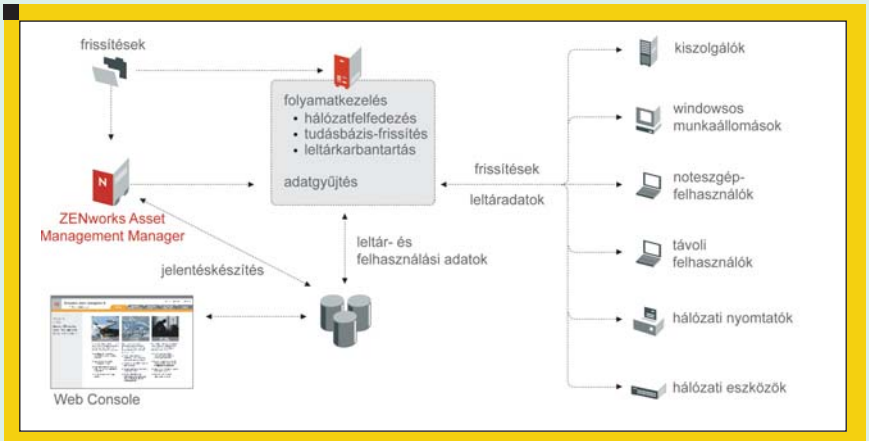
A szoftverek használati trendjei

A *Novell ZENworks Asset Management* lehetőséget biztosít a munkaállomásokon található alkalmazások használati trendjeinek és részleteinek megtekintésére. A jelentések jelzik, hogy mely termékeket és milyen típusú szoftvereket kik használnak, és ami még fontosabb, melyek azok az elemek, melyek használaton kívül vannak. Biztosíthatja a megfelelőséget úgy is a vállalat, hogy csak olyan licenceket kell megvásárolnia és támogatnia, melyekre valóban szüksége van. A használati jelentések emellett segítenek megalapozni és fenntartani a vállalati szabványokat, és kiszűrhetővé teszik a nem megfelelő alkalmazásokat, például a hacker eszközöket, a peer-to-peer szoftvereket és így tovább.

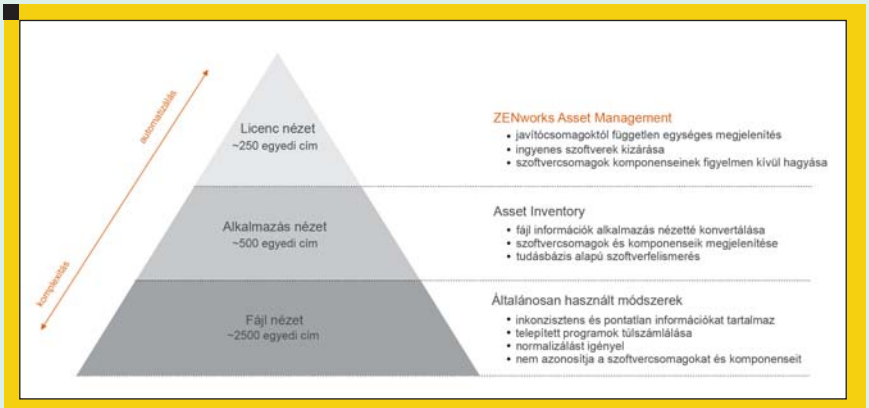
A Novell megoldása alkalmazkodik a környezethez

A *Novell ZENworks Asset Management* teljesítménye alkalmazkodik a környezethez függetlenül attól, hogy a vállalat egy helyszínen rendelkezik 100, vagy világszerte akár 100 000 munkaállomással.
A *ZENworks Asset Management* díjnyertes vagyonkövetési és felderítő eszközei páratlan pontosságot biztosítanak a hardver- és szoftverelemek könyvelése során.
A *ZENworks Asset Management* az informatikai eszközök teljes skáláján – a szerverektől és routerektől kezdve az asztali gépekig és laptopokig – elvégzi a jelentéskészítést, valamint a rajtuk futó szoftvereket is megjeleníti. A szoftverhasználat és a licencek követésével a vállalatok csökkenthetik a végfelhasználói támogatás költségeit, valamint kisebb jogi kockázatot kell vállalniuk a szoftverlicencek megfelelőségének leegyszerűsített kezelése által.

„A szoftverhasználati szabályok be nem tartása óriási problémát jelenthet egy vállalat számára, mivel a büntetések tíz-, vagy százmilliókra is rúghatnak. Az eszközkezelő, használatelemző és szoftverelosztó termékek segítségével az ügyfelek megfigyelhetik a használati trendeket és nyomon követhetik a jogosulatlan alkalmazások használatát, így pontosabban használhatják a szoftverelosztást a vállalati szabványok megadása és betartatása érdekében.” – mondta ***Patricia Adams***, a ***Gartner*** kutatási igazgatója.



1. ábra A Novell ZENworks Asset Management felépítése



2. ábra A Novell ZENworks Asset Management licenc nézet

A Novell ZENworks Asset Management legfontosabb szolgáltatásai

- Helyben telepített és szerver alapú alkalmazásokra vonatkozó jelentések
- Futásidejű alkalmazáskövetés (az előtérben és a háttérben futó alkalmazásoké is)
- A nem használt, ritkán használt és gyakran használt alkalmazások azonosítása
- Felhasználói és eszköz információk a többfelhasználós eszközök esetében is, melyek hozzárendelésre kerülnek az alkalmazások használatára vonatkozó információkhoz
- A használati szintek csoportosított nézete részlegek és helyszínek között
- Idő alapú jelentések az alkalmazáshasználatról és a nem használt elemekről
- Részletesen visszakövethető lekérdezési lehetőségek
- Használatkövetés – még akkor is, amikor az eszközök nincsenek csatlakoztatva a hálózatra
- Felhasználók számára észrevétlen: kis erőforrásigény és csendes működés

Linux, mint stratégiai platform – Linux támogatás az Oracle-től

Az Oracle 1998-ban adta ki adatbázis-kezelőjének első Linuxos verzióját (ez az Oracle 8.0.5 volt), mely az első kereskedelmi adatbázis-kezelő volt Linuxon. Sokáig azonban ez önmagában nem hozta meg az áttörést. Felismerve a problémát az Oracle nem csak portolja termékeit Linux platformra, hanem 2002 óta egyedülálló módon kód szintű terméktámogatást biztosít a Linux operációs rendszerre is Red Hat, SUSE Linux és Asianux Linux disztribúciók esetében.

Teszi mindezt oly módon, hogy egy *Linux* kernel fejlesztői csapatot működtet, mely közreműködésével a professzionális *Linux* verziók már eleve az adatbázis kezelők által támasztott nagy teljesítmény, megbízhatósági és skálázhatósági elvárásoknak megfelelnek. Az októberi *Oracle Open World* konferencián aztán mindezt megfejelte a világcég azzal, hogy bejelentette az *Oracle Unbreakable Linux* támogatást, mely keretében ugyanezt a szolgáltatást már nem csak a *Linux* disztribútornál meglévő support szerződéshez köti, hanem az *Oracle*-től is megvásárolható jóval olcsóbban, ezzel is jelezve, hogy a *Linux* már érett a vállalati felhasználásra (További információk az *Oracle Unbreakable Linux* bejelentésről szóló decemberi cikkünkben).

Nyílt forráskódú megoldások

A terméktámogatás mellett természetesen az *Oracle* kernel csapat a fejlesztőmunkában is aktívan részt vesz. Az eddigi legjelentősebb fejlesztés az *Oracle Cluster File System (OCFS)* amely egy olyan nyílt forráskódú állományrendszer, amely segítségével egy fürtözött rendszeren minden node konkurens módon látja ugyanazon állományokat és adatokat. Nincs szükség a bonyolult *raw device* menedzselésre, ezt nagyszerűen helyettesíti

az *OCFS*. Az *OCFS* sikerét igazolja, hogy ma már több Linux disztribúciónak is szerves része. Emellett például a *PHP* nyelvet is komolyan támogatja az *Oracle*, amit igazol a nemrég megjelent *Zend Core for Oracle*. Ez egy teljeskörűen tesztelt és támogatott *PHP 5* disztribúció, melyet integráltak az *Oracle Database* kliens könyvtárakba. Segítségével percek alatt beüzemelhető egy *PHP*-s környezet *Oracle* adatbázison. Az *Oracle* egyéb nyílt forráskódú projektjeiről a ➲ *http://oss.oracle.com* oldalon lehet bővebb információt találni.

Fejlesztés Linuxon

Az *Oracle* ma már nem csak azt mondhatja el, hogy minden terméke elérhető *Linux* platformon. Egyrészt belső ügyvitelének jelentős részét áttette Linuxra, másrészt 2003 október 6-án 5000 *Oracle E-Business Suite* fejlesztő átállt Linux fejlesztési platformra! Ez csak az első lépés volt, azóta már a technológiai termékek (*Oracle Database*, *Oracle Fusion Middleware*, stb) több mint 4000 fejlesztője is *Linuxon* készíti az új verziókat, és onnan portolják át más platformokra a kódot.

Csúcsteljesítmény Linuxon

Különböző teljesítmény mérések során az *Oracle*/*Linux* kombináció gyakran felülmúlja versenytársait. Az eddigi legjobb *TPC benchmark* eredmény fürtözött környezetekben például *Red Hat Enterprise Linux AS 3* szerveren született *1.184.893* percenkénti tranzakciószámmal. Az *Oracle Grid Computing* koncepció felhasználásával olcsó, *„commodity”* hardverekkel valósíthatunk meg üzletileg kritikus nagyvállalati rendszereket *Linux* platformon. Kevesebb erőforrással nagyobb teljesítményt érhetünk el, ha *GRID*-megoldással a feladatokat megfelelő módon szétosztjuk az erőforrások között, illetve a számító kapacitásokat igény szerint allokáljuk.

Bevizsgált Linux konfigurációk

Gyakran okoznak problémát komplex rendszerek esetén a különböző komponensek kompatibilitási problémái. Az *Oracle* idén ősszel elkészítette az *Oracle Validated Configurations* listáját, mely előre tesztelt, validált architektúrákat tartalmaz dokumentált

telepítési, konfigurálási útmutatókkal, beleértve nem csak az operációs rendszert és az adatbázis-kezelőt, de a hardvert, diszk alrendszert és hálózati komponenseket is. Az *Oracle Validated Configurations* lista megtekinthető a következő linken:
➲ *http://www.oracle.com/technology/tech/linux/validated-configurations/index.html*

Oracle Database 10g Express Edition

Az *Oracle* nevéről legtöbbeknek az adatbázis-kezelője jut eszébe, holott ma már üzleti alkalmazásokat és köztesszoftvereket is készít. Ugyanakkor az adatbázis-kezelő esetén is vannak újdonságok, melyek az adatbázis fejlesztők munkáját is könnyíti. Mindenekelőtt idén év elején jelent meg az *Oracle Database Express Edition (XE)*, mely az *Oracle Database 10g* alapfunkciókat tartalmazó ingyenes változata, mely *Debian*, *Mandriva*, *Novell*, *Red Hat* és *Ubuntu Linux* disztribúciókra telepíthető. Az *Oracle Database XE* ugyanarra a forráskódra épül, mint az *Oracle Database 10g* második változata, és az összes integrált alkalmazás programozói interfészt tartalmazza, így ideális eszköz a *PHP* vagy *Java* alapú és hasonló adatbázisra épülő alkalmazások fejlesztéséhez. Saját egyéni fejlesztőkörnyezetek ugyanúgy könnyen létrehozhatók vele, mint több fejlesztő által közösen használt adatbázisszerverek.
Az *Oracle Database XE* tetszőleges méretű gépre telepíthető, azonban összesen 4 GB-nyi felhasználói adatot kezel, és egyetlen processzoron fut maximum 1 GB operatív memória használatával. Az adatbázis-adminisztráció böngészős felületről történik, és a fejlesztett alkalmazások szabványos interfészekkel – *SQL*, *JDBC*, *ODP.NET* – vagy az *Oracle* egyedülálló *Application Express* funkcionalitásával illeszthetők az adatbázishoz.
Az ingyenes *Oracle* adatbázis-kezelő ideális azoknak is, akik még sosem foglalkoztak *Oracle* technológiával korábban. Az *Oracle Database 10g Express Edition* egyszerűen használható webes felhasználói interfésszel rendelkezik a közismert parancs soros *SQL*Plus* mellett, így a *DBA* ismeretekkel nem rendelkező a fejlesztők is könnyedén használatba tudják venni az adatbázis-kezelőt.

A fejlesztők egy közel teljes funkcionalitással rendelkező *Oracle Database 10g*-t tudnak futtatni akár a laptopjukon, így nem lesznek kötöttek egy nagy, központi adatbázis-szerverhez a fejlesztés és tesztelés ideje alatt. Bár az *XE* változathoz nem jár *Oracle support* szolgáltatás, a felhasználók a telepítését követően regisztrálhatnak az *Oracle Database XE* online fórumára, ahol nem más, mint *Tom Kyte*, *Oracle* alelnök, technológiai szakértő ad ingyenes segítséget a bajba jutott felhasználók számára.

Oracle Application Express

Az *Oracle Database XE* integráltan tartalmaz egy fejlesztő eszközt, mely 2004 februárjában jelent meg *Oracle HTML DB* néven, majd idén átnevezték *Oracle Application Express*-re. Ezzel a fejlesztő eszközzel *Oracle* adatbázisra rendkívül gyorsan lehet webes alkalmazásokat készíteni. Ráadásul ha bármilyen adatforrásból szeretnénk adatainkat egy relációs adatbázis-kezelőbe tölteni és másokkal megosztani, akkor ezzel az eszközzel ezt egyszerűen megtehetjük mindössze egy web böngésző használatával. Természetesen az *Oracle Application Express* örökli az *Oracle Database* összes funkcionalitását, előnyét, mint például a biztonságot, skálázhatóságot, teljesítményt és megbízhatóságot. Az *Oracle Application Express* felhasználásának egyik legjobb és legismertebb példája az *Oracle online support* rendszere, a ➲ *http://metalink.oracle.com*.

Oracle SQL Developer

Bár a fejlesztők legtöbbször a köztes rétegen, vagy a megjelenítési rétegen dolgoznak, néha mégis le kell nyúlniuk a rendszer gyökeréig, így például az adatbázisban lévő tárolt eljárásokon kell dolgozniuk, vagy az adatbázis sémán bármilyen módosítást kell végrehajtaniuk. Ekkor jön képbe az *Oracle SQL Developer*, amely bár nem nyílt forráskódú, de ugyancsak ingyenes termék az *Oracle*-től.
A vastag kliens megoldással készített grafikus felhasználói interfésszel ellátott eszköz fejlesztők számára kényelmes tárolt eljárás, trigger írást, módosítást, nyomkövetést tesz lehetővé. Eddig hasonló eszköz csak más gyártóktól volt elérhető *Oracle* adatbázis-kezelőhöz, ezért mindenképp hiánypótló termékről van szó. Nem igényel *Oracle* klienst, így bárhonnan futtatható, letölthető az *Oracle Technolgy Network* oldaláról (➲ *http://otn.oracle.com*). ■

Múzeum a város szélén...

Manapság minden nagyobb városban van ilyen vagy olyan múzeum, ahol állandó vagy idényjellegű kiállítások várják a látogatókat, legyen az akár művészeti, akár ipari múzeum. De járt-e már a kedves Olvasó informatikával kapcsolatos múzeumban? Mert ugye nehéz elképzelni egy ENIAC elektroncsöves számológépet mai fejjel...

*Szeged*en található magyar ***Informatikatörténeti Múzeum*** 11 ezer részegységnek ad otthont, tehát nem kell külföldre utazni ahhoz, hogy régi monstrumokat lássunk testközelből. A több tonnányi (jelenleg körülbelül 240 tonna az össztömeg) muzeális számítógép ***2002-től*** a ***volt szovjet laktanya*** egyik épületében található, mely ***előre egyeztetett időpontban látogatható***.

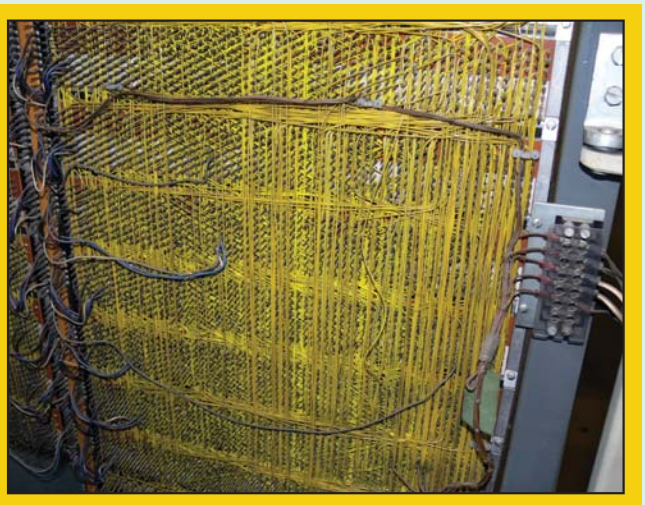
A múzeum ötlete a hetvenes években merült fel ***dr. Kovács Győző*** és ***dr. Muszka Dániel*** részéről, hiszen számos olyan gép működött ***Magyarország***on, melyek egyedi, magyar fejlesztésűek voltak, vagy a nyugati eszközök másolatai. Ne felejtsük el, a hidegháború alatt a két pólus között nem igazán volt ilyen tekintetben áruforgalom.

A gyűjtemény ***1992-től*** működik ***alapítványi*** keretek között, az alapító tagok között olyan neves szervezetek voltak, mint az ***Országos Műszaki Múzeum***, a ***Neumann János Számítógéptudományi Társaság*** és az ***Állami Számítógépes Szolgálat***. Társult tagként pedig a ***Szegedi Tudományegyetem*** és ***Szeged Város Önkormányzata*** képviselteti magát.

Az eltelt időszak költözésekben gazdag volt, de remélhetőleg a jelenlegi helyről már nem kell tovább költöznie a páratlan gyűjteménynek. A gépeket mozgatni se könnyű, lévén hogy gyakran több száz kilósak, illetve még a leggondosabb költöztetés is károsíthatja a készülékeket.

A gyűjtemény nem lenne teljes távközlési berendezések nélkül. Láthatunk például különböző generációs telefonközpontokat is. A ma is működő ***lyukszalagos telex*** pedig eddig nagy sikert aratott a látogatók körében.

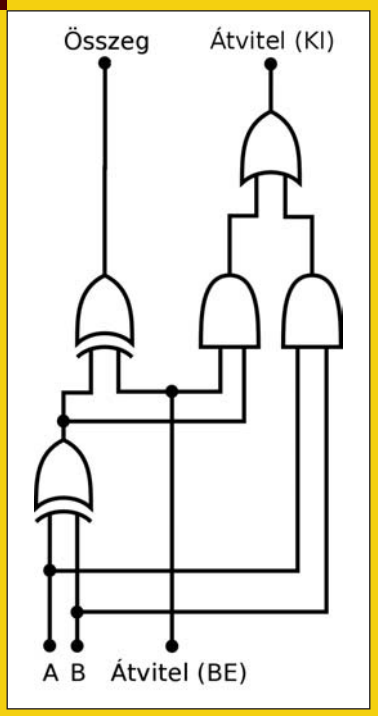
A gyűjtemény jelenleg a rendelkezésre álló épületből két és fél szintet foglal el, azonban folyamatosan gyarapszik. Földszinten találhatóak a „szekrények", hiszen lift nélkül felvinni az emeletre embert próbáló vállalkozás lenne. A szekrény szót itt szó szerint kell érteni, hiszen van pár olyan, ami nálam is magasabb. Mai ésszel felfogni se nagyon lehet, hogy annak idején például mire lehetett ezeket a monstrumokat használni, amelyek gyakran csupán ***64 kbyte***-nyi memóriával rendelkeztek, működési ***sebességük*** pedig a mai ***zsebszámológépekét se mindig érte*** el. Programozás szempontjából se volt könnyű dolguk



1. ábra Huzalos programozás

A teljes összeadó áramkör

Több teljes összeadó áramkör alkalmazásával tetszőlegesen hosszú bináris számokat adhatunk össze. Tehát például ***két 32 bites szám összeadásához 32 darab ilyen logikai áramkör*** szükséges. Az áramkörök A és ***B*** bemeneteire a két szám megfelelő helyi értékű bitjét kell küldenünk, míg a kisebb helyi értékű áramkör kimenő átvitelét össze kell kötnünk az eggyel nagyobb helyi értékű áramkör átvitel bemenetére. Az egész leegyszerűsítve úgy működik, ahogy mint ahogy az ember papíron is végzi tízes számrendszerben az összeadást. Ha a legnagyobb helyi értékű áramkör kimeneti átvitelén egyes jelentkezik, akkor beszélünk túlcsordulásról.



2. ábra A teljes összeadó logikai kapcsolása

3. ábra Logikai kapuk magyarázata

1. táblázat ***A teljes összeadó igazságtáblája***

Bemenet		Kimenet		
A	B	Átvitel BE	Összeg	Átvitel KI
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

4. ábra Az URAL-2-es egyik modulja (a kétezer csőből, itt csupán 200 van)

5. ábra RAZDAN-3

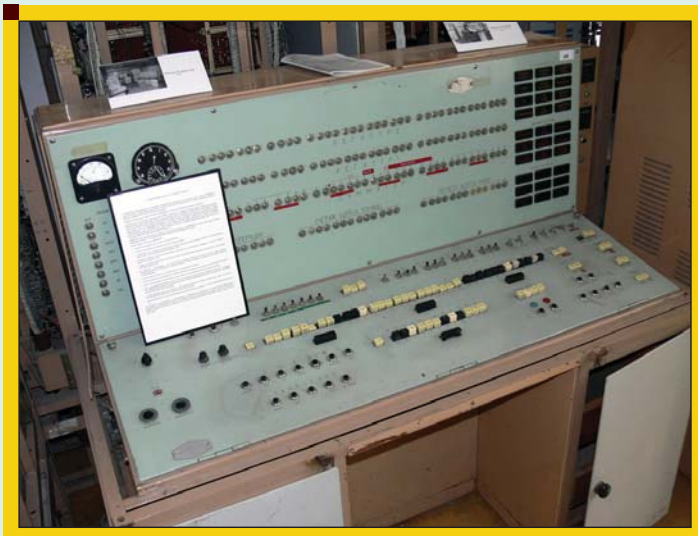
6. ábra A RAZDAN-3 szalagos egysége

a szakembereknek, hiszen fordító programok és szerkesztők helyett ***kezdetben huzalokkal kellett programozni*** a gépet olyan szinten, ahol az ***ÉS***, ***NEM***, ***VAGY***, ***kizáró VAGY*** kapuk dolgoznak.

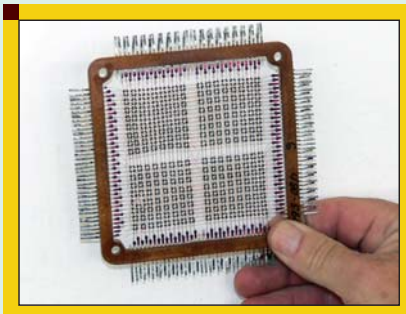
A ***huzalos*** számítógépek ***kiváltására elektroncsöves*** számítógépeket építettek. A számítási kapacitás nőtt és sebesség is javult ugyan, de cserébe ***tetemes hőtermelés*** jelentkezett, amely gyakran az alkatrészeket is megviselte. Nemegyszer a gépek többet álltak, mint üzemeltek. Az ***1946-os ENIAC***-ban mintegy ***17 ezer*** ilyen ***elektroncső*** volt, míg a múzeumban megtalálható ***1963-ból*** származó ***orosz URAL-2***-ben „csupán” ***2 ezer***. ***Charles Simonyi*** – magyar származású informatikus – is egy ilyen gépen tanulta meg az alapokat. Később a ***Xerox***nál részt vett a ***Bravo*** fejlesztésében (ez volt az első olyan szövegszerkesztő volt, amelyen már úgy látható az anyag, ahogy nyomtatásba is kerül), később pedig a ***Microsoft***nál a ***Word*** és az ***Ex-cel*** projekt vezetője lett. Sok helyen hivatkoznak rá úgy, mint az az ember, aki a ***Microsoft***ot naggyá tette. Végül, de nem utolsósorban ő fejlesztette ki a ***magyar jelölés***t (***hungarian notation***).

A számítógépek ***elfogadható méretű***re csökkentésében ***nagy szerepet játszottak a tranzisztorok***. Az ***1948-ban szabadalmaztatott alkatrész*** nagymértékben meghatározta az informatika fejlődési útját. Egy tranzisztor mind

7. ábra MINSZK-22



8. ábra A MINSZK-22 mérnöki pultja



9. ábra Egy kisebb méretű ferritmagos memória, leginkább gyöngyfűzésre hasonlít

10. ábra TPA 1001

11. ábra TPA 11/440

Unix és a PDP-11

A ***DEC*** hivatalosan az ***Ultrix***-ot támogatta, de nem hivatalosan elérhető volt például a ***2BSD*** és a ***Version 7 Unix*** is. A vasfüggöny mögött az elvtársak ***1982-ben*** elkezdték fejleszteni saját ***Unix***ukat ***DEMOS néven*** (Dialogovaja Edinaja Mobilnaja Operatsionnaja Szisztyema rövidítése, mely röviden interaktív hordozható operációs rendszert jelölt. A hordozhatóság itt természetesen a forráskódra értendő). Eredetileg ***UNAS lett volna*** a neve, mely oroszul (u nas) azt jelenti „miénk". A mottót az adta, hogy a ***UNIX*** szó kissé oroszosítva (u nich) „övék"-et jelent. Erről az elnevezésről azonban a vezetőség még időben letett.

fogyasztásban, mind méretben, mind megbízhatóságban jobb mutatókkal bírt az elektroncsöveknél.
Az ***1966-ban*** bemutatott, ***Jerevá***nban (***Örményország***) gyártott ***Razdan-3*** – melynek tárkapacitása ***32K szó*** volt, szavanként ***48biten*** – már tranzisztorokkal dolgozott. Háttértárként ***cserélhető mágnesszalagot*** használtak, mely gyakran látható 70-es, 80-as évekből származó filmekben is.
A gépet ***Algol 60*** programnyelv segítségével lehetett ***programozni***. Fontos megemlíteni, hogy talán a világon egyedül a szegedi múzeumban van teljes példány kiállítva.
A másik érdekes masina ebből a korszakból (***1965-68*** között gyártották) az orosz gyártmányú ***MINSZK-22,*** melynek tárolókapacitása ***kétszer 4096 szó*** volt (szavanként ***37 bittel***). Az adatok bevitelére ennél az eszköznél 8 csatornás lyukszalagot, lyukkártyákat (80 pozícióval) vagy úgynevezett ***mérnöki pultot*** használtak. Ha a kedves Olvasó

Látogatás és támogatás

Mint minden non-profit vállalkozás, így az ***Informatikatörténeti Múzeum Alapítványa*** is szívesen fogadja a támogatásokat, felajánlásokat, legyen az munka vagy anyagi jellegű. Anyagi jellegű támogatást az ***Alapítvány számlájára*** utalhatunk (11670009-07803500-70000002), végül pedig befizetett személyi jövedelemadónk (***SZJA***) ***1%-nak felajánlásával*** is támogathatjuk a alapítványt. Ebben az esetben a rendelkező nyilatkozatra ezt a számot írjuk: 18036170-1-01
Segítség felajánlást és látogatási igény bejelentését a múzeum honlapján ➲ ***http://www.infmuz.hu*** található elérhetőségeken várja ***dr. Bohus Mihály, Csorba Béla*** és ***dr. Muszka Dániel***.

12. ábra TPA bővítőkártya

megtekinti a gépegyüttest élőben, talán látni fogja, hogy akkoriban megállt a mondás, miszerint akkor terjed el az orosz mikroelektronika, ha kifér a gyárkapun.

A *vasfüggöny* mögött gyakran előfordult, hogy egy-egy nyugati gépet – minthogy lehetetlen lett volna behozni – a mérnökök újra „feltaláltak". Ilyen a *KFKI TPA 1001*-ese is, mely a népszerű *DEC PDP-8* másolata volt. (A *PDP* a *Programmed Data Processor*, míg a *TPA* a *Tárolt Programú Analizátor* rövidítése.)

Az alapgép *4K szónyi* (*12 bites szó*) *ferritmagos memóriát* használt, akárcsak a *PDP-8*. Ez egészen *32K szóig* volt bővíthető. (Még mindig 64 Kbyte memória alatt járunk.) A gépet *assembly*-ben lehetett programozni. Sebessége: *50 ezer művelet másodpercenként*.

A programozás iránt érdeklődők a *PDP-8*-ra írt *assembly* nyelvű *Hello-World*-öt a *Wikipedián* megtalálhatják. Szintén egy magyar klónra bukkanhatunk a *KFKI*-tól *TPA11/440* néven. Ez nem más, mint egy *PDP-11* kompatibilis eszköz. A *PDP-11*-re írt programok módosítás nélkül futtathatóak voltak *TPA 11/440*-en, így a különböző *Unix* variánsok is. Ennek köszönhetően fordulhatott elő az, hogy a kutatóknak, akik korábban *TPA* gépeken dolgoztak, nem jelentett újdonságot a vasfüggöny eltűnésekor a *Unix*.

Természetesen a *PDP* gépek engedély nélküli klónozása a *KGST* irányításával más vasfüggöny mögötti országban is folyt, így *Bulgária*ban, az *NDK*-ban, *Lengyelország*ban és a *Szovjetunió*ban is.

Ne feledkezzünk meg a háttértárak fejlődéséről sem. A legelső merevlemezt 1956. szeptember 13-án dobta piacra az *IBM RAMAC* (*Random Access Method of Accounting and Control*) néven, melyben 50 darab 24 hüvelykes (60 cm) lemez forgott, összesen *5 megabájtos* kapacitással. Az eszközt nem volt túl kellemes használni a zaj és a nagy fogyasztás miatt (*12 Kilowatt*) A múzeumban ez az eszköz sajnos nem látható élőben, de a kontraszt kedvéért négy merevlemez van egymás fölött 1967, 77, 87 és 97-ből, rendre 5 kilobájt, 20 megabájt, 850 megabájt és 6 gigabájt kapacitásokkal.

Az 1967-es eszköz mérete vetekszik egy kisebb hűtőével (a Razdan-3 része volt), míg a 97-es eszköz egy laptop merevlemez, amely már egy farzsebben is elfér. Már majdnem eltelt ismét 10 év: jelenleg 750 gigabájtos a legnagyobb merevlemez, ami kapható.

A cikk folytatásaként tovább kalandozunk régi idők feledésbe vesző eszközei között.

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

A múzeum honlapja
➲ http://www.infmuz.hu/

TPA-król átfogó információk
➲ http://hampage.hu/tpa/index.html

Mindent tudó Wikipedia
➲ http://en.wikipedia.org/

Áramtalanítva

Vajon mennyi pénzt fizetünk ki az áramszolgáltatóknak feleslegesen?

Munkahelyemen arra kaptam megbízást, hogy vizsgáljam meg, milyen eszközökkel lehetne az informatikai berendezések villamos energia felhasználását csökkenteni. Azonnal szóba kerültek az energiatakarékos ***TFT***-monitorok, hordozható számítógépek használata, asztali számítógépek energiatakarékos üzemmódjai, hibernálás lehetősége, hiszek ezen eszközökkel akár több tízezer forintot is meg lehetne spórolni évente. Mindezek mellett el kezdtem foglalkozni a szoftveresen kikapcsolt, mégis minimális áramot felvevő számítógépekkel, illetve a használaton kívüli, villogó lámpájú monitorokkal. Megrázó sorok következnek, gyengébb idegzetű olvasók lapozzanak...

Céghelyzet – elcsorgó kilowattórák

Két csoportba osztottam munkáltatóm több mint 140 számítógépét. Első csoportba az irodai gépek tartoznak, a másodikba pedig a termelésben elhelyezkedő számítógépek. Utóbbiakra jellemző, hogy üzem idejük „0-24” (általában csak vasárnap illetve ünnepnapokon vannak kikapcsolva). A 140 számítógépből körülbelül 100 darabra mondható, hogy adminisztratív jellegű, legalábbis abból a szempontból, hogy munkanapokon 8 órát vannak bekapcsolva, és hétvégén használaton kívül vannak. Ebből következik, hogy kikapcsolt (nem áramtalanított) állapotban egy átlagos héten a hétköznapi készenléti fogyasztásának költsége a következőképpen számítható:

„hétköznapi inaktív idő” x *„hétköznapok száma”* x *„átlagos készenléti fogyasztás”* x *„kWh egységár”*

Itt a hétköznapi inaktív idő körülbelül 16 óra, a hétköznapok száma értelemszerűen 5 nap, az átlagos készenléti fogyasztás (monitor + számítógép) pedig 5 W + 15 W. Az elektromos energia egységára jelenleg 20,45 HUF/kWh (nettó, nagy fogyasztóknak).

A végösszeg tehát:

16 h/nap x ***5 nap*** x ***0,02 kWh/h*** x ***20,45 Ft/kWh*** = 32,72 ***Ft***

Cég esetén ehhez még hozzáadódik a hétvége is, amikor az inaktív idő gyakorlatilag 24 óra:

24 h/nap x ***2 nap*** x 0,02 ***kWh/h*** x ***20,45 Ft/kWh*** = 19,63 Ft

Összesen tehát egy héten egy irodai gép 32,72 Ft + 19,63 Ft = 52,35 Ft értékű elektromos energiát használ el feleslegesen. Ez egy 100 gépet üzemeltető cég számára éves szinten 272.230 Ft teljesen fölösleges kiadást jelent. Gyártási folyamatokban használt számítógépeknél ez az érték valakivel kevesebb, mivel egyes területeken 2 illetve 3 műszakban is dolgoznak hétköznapokon, illetve a hétvégei munkavégzés is gyakoribb.

Ne hidd, hogy otthon jobb

Felbuzdulva az eredményeken, körbe néztem otthonomban is. ***TV***, ***DVD*** lejátszó, digitális beltéri, ***router***, ***switch***, számítógép... ezek mind-mind *„**stand-by**”* módban díszelegnek éjszaka, mint egy karácsonyfa. Gyorsan összeadogattam a 2-5 W közötti készenléti fogyasztásokat (összesen körülbelül 25 W), gyors számolás, és több, mint 5000 forint „többletfogyasztás” az eredmény (a 12. havi villanyszámlámat feleslegesen fizetem).

Amit tenni lehet

Egy lehetséges megoldásnak tűnik, ha hosszabb inaktív időben az informatikai berendezéseket áramtalanítjuk. Ennek legegyszerűbb módja, ha minden számítógép mellé könnyen elérhető és könnyen kezelhető megszakító kapcsolóval felszerelt elektromos elosztókat telepítünk. Egy ilyen elosztó kiválasztásánál figyelembe kell venni, hogy könnyen kezelhető legyen (egy mozdulattal lekapcsolható), asztallapra/falra felszerelhető legyen. Egy ilyen eszköz nettó kiskereskedelmi ára 2.000 forint körül mozog. 100 darab számítógép ellátása ilyen elektromos elosztóval 200.000 forintba kerülne, így körülbelül 70.000 forintot sikerülne megtakarítani.
Természetesen a megoldás nem csak ennyi, hiszen hiába telepítünk megfelelő eszközöket, ha a felhasználók nem hajlandók/képesek kihasználni ezeket a lehetőségeket. Egy rövid oktatásra, figyelem felkeltésre is szükség lenne, ahol megfelelő számadatokkal próbálnánk mindenkit meggyőzni a takarékoskodás eme egyszerű módjáról.

Gráma Tibor
(tibor.grama@hoya.lmh.hu)

1997 óta „Linuxozik”, UHU hívő. Szabadidejében gyermekeivel és vizsla kutyáival játszik, ha éppen nem kertészkedik vagy horgászik.

Programozzunk Pythonban (1. rész)

A szolid óriáskígyó

Elegáns, hatékony és könnyen tanulható programozási nyelv a Python. Egyaránt megállja a helyét kis feladatokban és nagy csapatmunkában, több platform alatt, akár ragasztónyelvként használva. Egy kezdőnek is lehet vele gyorsan sikerélménye, a profik kezében erőteljes szoftverfejlesztő eszköz.

Miért kezdjünk (ki) egy óriáskígyóval?

A programozási nyelvek száma szerencsére megdöbbentően nagy, ám ezekből egy átlagos felhasználó vagy fejlesztő általában csak néhány nevével találkozott már. Népszerűbb nyelvek többnyire azért válnak ismertté, mert adott feladatokra vagy adott környezetben jól használhatóak, vagy épp ellenkezőleg: általános célokra is, és többféle platformon is megfelelően működnek. A *Python* utóbbi szempontok alapján válhatna igazán elterjedtté, hiszen változatos futtatókörnyezetekben is megállja a helyét, legyen szó akár egy *Debian Linux*-disztribúciót használó munkaállomásról, *Symbian* operációs rendszert futtató mobiltelefonról vagy Solaris alapú webszerverről.

Mégsincs könnyű dolga, ha a fejlesztőket szeretné meghódítani, hiszen az informatikai világban nem is oly rég végbement változások következtében ma már roppant népszerűségnek örvend általános célú nyelvként a *Java*, a webes szkriptnyelvek között a *PHP* az egyik kedvenc, bővülni és megújulni látszik a *Perl*, a *JavaScript*, erősödik és függetlenedik a *C#*, töretlen adatbázismanipulációkban az *SQL* dominanciája, újra reflektorfénybe kerültek a héjprogramok, és még néhány, területspecifikus nyelv is tolong az elsőbbségért. Mit tud hát felmutatni a már nem is oly fiatal *Python*, amiért érdemes egyáltalán megpróbálnunk megszeretni, s reménykednünk abban, hogy a befektetett tanulás örömteli élményekkel, s talán komolyabb tudással is gazdagít?

Szerzője *Guido van Rossum*, aki a *Google* csapatát gazdagítja egy ideje, s jelenleg is foglalkozik alkotásával és használja is azt. Nevét nem az óriáskígyóról, hanem a *Monthy Python repülő cirkusza* című angol humorban gazdag alkotásról adta, ennek ellenére sokan előszeretettel használnak vidám zöld óriáskígyót különböző piktogramokon.

A nyelvhez elérhető – általában hobbiból vagy egy-egy projekt kapcsán született, s később kanonizálódott – kiegészítőkön, fejlesztői könyvtárakon és modulokon kívül is találhatunk önálló keretrendszerré fejlődött, *Python* alapú vagy azt hasznosítható eszközöket, legismertebbek talán a komplett tartalomkezelő *Zope* és *Plone*, vagy a webes *Qweb*, *Cherrypy* és a *TurboGears* (utóbbi ráadásul felhasználja előbbit), a *BitTorrent* fájlcserélő, és a *Trac* verziókövető rendszer. Némelyikükkel már a *Linuxvilág* hasábjain is találkozhattunk, például *Juhász Attila* több keretrendszert összehasonlító remek kedvcsináló áprilisi cikkében.

1. ábra Egy jó Python kis helyen is elfér – akár mobilon is

➲ *http://www.linuxvilag.hu/system/files/cikk_63_16_18.pdf*

Fantasztikus történeteket olvashatunk e nyelv tanulhatóságáról, tömörségéről és hatékonyságáról az interneten. Mindez leginkább két dolognak köszönhető: szkriptnyelv volta ellenére objektum-orientált, és nagyon magas szintű nyelvnek nevezhető. Előbbivel többször fogunk majd találkozni, és részletezzük ezen tulajdonság előnyeit, utóbbi pedig leginkább azt jelenti, hogy rengeteg előre megírt objektum és felhasználható elem könnyíti életünket. Nézzünk egy könnyű példát,

vessük össze a *C* nyelvvel két változó értékének felcserélését, előbb *C* nyelven:

```
void main()
{
char x="kutya";
char y="macska";
char z; /*ideiglenes változó,
➥csak a csere idejére kell */
z=x;
x=y;
y=z;
}
```

Ugyanez *Python*ban:

```
x="kutya"
y="macska"
x,y = y,x
```

Még nyilvánvalóbb lenne a példa, ha a nyelv alapvető eszköztárának elemeit vetnénk össze, leginkább az ún. absztrakt adatszerkezeteket, melyek összetettségükkel hasznos és gyors segítőinkké válhatnak – ilyenek például a listák, szótárak, szekvenciák, asszociatív tömbök – s melyek megalkotása sok más nyelvben vagy a programozó feladata, vagy a nyelvnek nem szerves részét képező kiegészítő modulokban találhatók.
Gyakorta hasonlítják össze a sokszínű és szövegmanipulációban kiváló *Perl* nyelvvel, ilyen rövid cikk például – a magyar *Python* oldalról is elérhető egy hivatkozás segítségével – leginkább A *katedrális és a bazár* című művével hazánkban is népszerű *Eric S. Raymond* írása, melyben a *Python* forráskódok könnyű karbantarthatóságát, nagyobb projektekhez is jól használható tulajdonságait emeli ki. Ne törjünk pálcát egyetlen programozási nyelv felett sem, mert elsődlegesen a programozó feladata megoldani az adott problémát, s a nyelv ebben segítheti vagy gátolhatja őt, de gondolkodni helyette bizonyosan nem képes.

Milyen is egy Python?

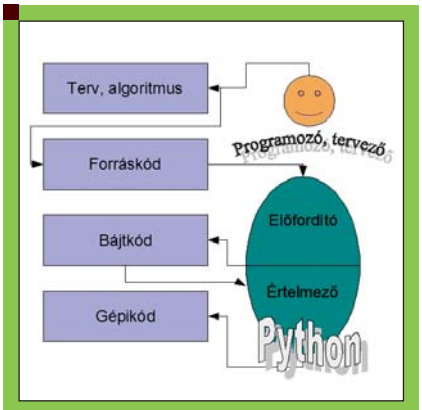
Kiválóan használható programozási nyelv, jó alapot biztosíthat a továbblépésre komplex és elterjedt objektumorientált programozási nyelvek irányába, mint amilyen a *C++* és a *Java* (feltéve hogy nem szeretünk bele túlságosan is kényelmes és gyors fejlesztést lehetővé tevő nyelvezetébe, és emiatt nem is akarunk továbblépni). Néhány fontosabb tulajdonsága:

- objektum-orientált szemléletű programozást támogat
- nem kell a memóriakezelés terhétől roskadozni, mert átvállalja helyettünk
- szkriptszerű, azonnal láthatjuk a változtatások eredményét
- rengeteg előre megírt kiegészítő és csatolófelület található hozzá
- jól áttekinthető, egyszerű, tömör
- szigorú: kötelező behúzásokat alkalmazni a tagoláshoz
- támogatja és elvárja a szabványos kivételkezelést
- kitűnő ragasztónyelv, jól illeszkedik más eszközökhöz (*C* nyelv, adatbázisok stb.)

Alapvetően szkriptnyelvnek tekinthetjük, bár kétségkívül kissé kilóg a sorból, csakúgy, mint például a *Java* a fordított nyelvekéből. Itt is úgynevezett bájtkóddal dolgozik az értelmező, azaz az általunk megírt forráskódból előbb egy átmenet – az értelmező számára emészthető és könnyen gépi nyelvre alakítható közteskód – s nem egyből gépikód keletkezik.
Két hátrányát mindjárt meg is említhetjük: elméletileg *Python* verziónként eltérhetne az általa készített közteskód értelmezése, s a dinamikus típusátadás miatt elég nehéz jó gyorsító-optimalizáló megoldást találni hozzá. Előbbit a forráskód mellékelésével, utóbbit a standard nyelv mások által átalakított, statikus típusátadást utánzó verziójával vagy modulokkal, például a *Pyrexszel* lehetne orvosolni – bár igazán csak akkor lehet ennek létjogosultsága, ha erősen számolásigényes feladatot próbálnánk meg kizárólag *Python* segítségével megoldani.
Létezik még egy viszonylag elterjedt, *Java* nyelven újraírt értelmező is, a *Jpython*, mely *Java* által értelmezhető közteskódot generál.

Kezdődjék a (repülő) cirkusz!

Mire lesz szükségünk? Természetesen magára a Python nyelvre, mely a legtöbb disztribúció telepítőlemezén megtalálható, jó eséllyel az olvasó is rátalálhat saját rendszerében, s kiirathatja verziószámát parancssorból:



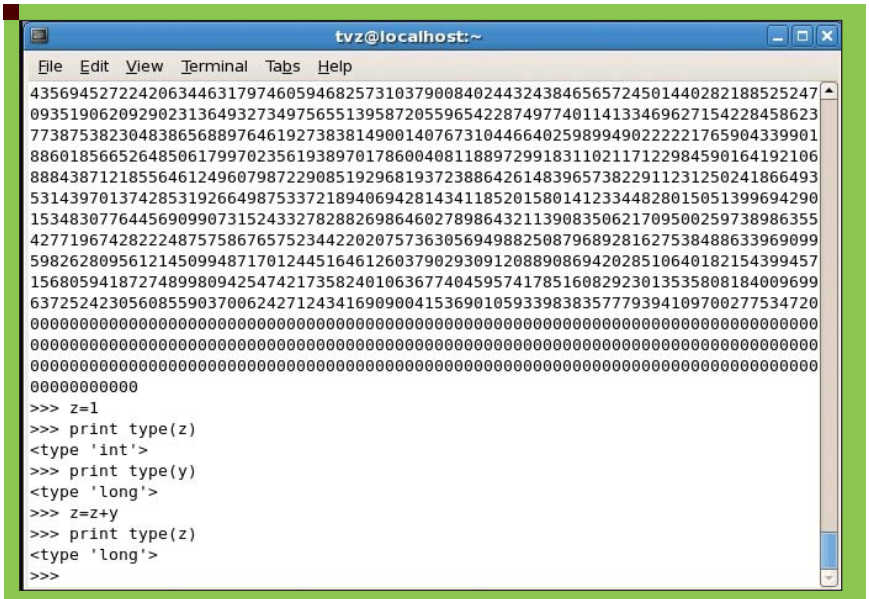
2. ábra A köztes bájtkód szerepe

```
python -V
```

Ha mégsem így lenne, a használt Linux változat hivatalos honlapján érdemes körülnézni először az elérhető csomagok között. Egy Debian rendszerben például aktív internetkapcsolat esetén általában elég kiadni megfelelő jogosultságokkal:

```
apt-get install python
```

Szükség van továbbá egy szövegszerkesztőre is, melynek segítségével a forráskódot írjuk, gyakorlatilag értelmezhető fájlokat készítünk.
S már csak a legalapvetőbb programozási ismeretek kellenek a sikeres kezdéshez – ezzel feltételezhetően mindenki rendelkezik, aki tanulmányai során egy csepp informatikával találkozott, vagy akár önszorgalomból *Linux*ot használva próbálkozott már egyszerűbb szkripteket írni. Oktatóanyagból és speciális feladatokra optimalizált leírásokból is jócskán találhatunk elektronikus és nyomtatott formában dokumentumokat, s talán a legkézenfekvőbb mű a nyelv alkotójának elektronikus formátumokban szabadon hozzáférhető írása.
Különböző trükkökkel elérhető akár programunk „közvetlen futtathatósága” is, például a *Google* nyílt forrású fejlesztéseiért felelős *Greg Stein* blogjában ➲ *http://www.lyra.org/greg/python/#dev* még a jóval korábbi verziókhoz írt egy kétsoros *Bash-szkript*et, mely által a webszerver */cgi-bin* könyvtárába helyezve a megírt *py* kiterjesztésű programunkat azonnal rávehetjük a rendszert annak értelmezésére. Hasonló hatást kelt,



3. ábra Típuskonverzió automatikusan – az óriáskígyó esete az óriásszámmal

ha héjprogramokban (***shellszkript***) is használható módon „futtathatóvá" tesszük programunkat, azaz először megadjuk egy direktívában a ***Python*** elérési útvonalát – egyszerű megtudnunk, például egy terminálablakban a `which python` paranccsal kiírathatjuk – majd megírjuk a programot, elmentjük, végül futtathatóvá tesszük. Példa ***vi*** szövegszerkesztő segítségével:

```
vi programunk.py
```

Ezzel megnyitottunk egy egyelőre még üres, de már névvel rendelkező fájlt, ez fogja tartalmazni a forráskódot, majd ***ESC*** billentyűt leütve tudatjuk a programmal, hogy beszúrni szeretnénk szöveget az üres sorba, erre szolgál a ***:i*** és utána az ***Enter*** billentyű. Jöjjenek a programsorok, természetesen ügyelve a kisbetű/nagybetű használatára:

```
#! /usr/bin/python
kiirando="Fut a program!"
print kiirando
```

ESC billentyű, majd `:w` és ***Enter*** (azaz mentjük), majd `:q` és ***Enter*** (azaz kilépünk).
A kész fájlt futtathatóvá a `chmod +x` paranccsal tehetjük, s nincs is más dolgunk, mint elindítani:

```
./programunk.py
```

Még egy sort érdemes beszúrni a forráskód elejére:

```
# -*- coding:Utf-8 -*-
```

Ezzel tudatjuk az értelmezővel, hogy mi a magyar ékezeteket is helyesen kezelő ***UTF-8*** karakterkódolást szeretnénk programunkban használni – bár tapasztalataim szerint enélkül is általában ékezethelyesen működik egy jól beállított, magyar nyelvű linuxon, mobiltelefonnal már nem voltam ilyen szerencsés.
A ***vi*** szövegszerkesztő használata ízlés kérdése, léteznek sokkal egyszerűbb és látványosabb programok is e célra (például ***Van Rossum*** által írt ***IDLE*** nevezetű fejlesztőkörnyezet), vitathatalan előnye viszont, hogy a legtöbb ***Unix/Linux*** disztribúcióban megtalálható valamilyen formában.
Használhatjuk a ***Python*** értelmezőjét úgynevezett interaktív módban is, ekkor egy hagyományos ***Linux*** héjhoz hasonlóan fog működni, azaz beírjuk a parancsokat, s ő végrehajtja vagy hibaüzenttel jelez vissza. Indítása roppant egyszerű:

```
python
```

Elszámolunk

Majdnem minden népszerű leírás megemlíti, hogy a ***Python*** remekül használható interaktív módban számológépként, ha erre kíváncsiak vagyunk, írjunk be egy egyszerű műveletet a készenléti jel (>>>) után:

```
20/6
```

Az eredmény 3 lesz, amit úgy kell érteni, mintha az eredménynek csak az egész számú részét írta volna ki az értelmező. Ha még nem programoztunk más nyelven korábban, valószínűleg újdonságot jelent, hogy van külön olyan művelet, mely képes az osztás maradékának kijelzésére is, ekkor a % jelet kell használnunk:

```
12%5
```

és megkapjuk a 2-t. E két művelettel például könnyedén meg tudjuk mondani egy számról, hogy páros-e:

```
a=15
if a%2==0 :
        print "Páros"
else :
        print "Páratlan"
```

Egyúttal több dolgot is megtanulhatunk e rövid példából. A legfontosabb: a ***Python*** nagyon ügyel a jó tagolásra, konkrétan a sorvégejeleket és a behúzásokat figyeli. Legyünk következetesek: e célból ne keverjük a szóköz és a tabulátor használatát (hacsak nem olyan szerkesztőprogrammal dolgozunk, ahol beállítható, hogy a tabulátorjeleket automatikusan szóközökre cserélje), mert a fordító különböző szintű tagolásnak fogja őket tekinteni.
Az értékadás (`a=15`) utáni sorban egy feltételes vizsgálat következett (`if a%2==0`), megnéztük, hogy az a változó értékét 2-vel osztva 0-t kapunk-e maradékul. A dupla egyenlőségjel (==) már nem értékadást, hanem összevetést jelent, azaz a bal és jobb oldalán szereplő értékeket próbáljuk meg összehasonlítani segítségével. Egy kettőspont zárta ezt a sort, mivel még nincs vége a kiértékelésnek, ez csak a művelet feje volt, és a több sorban leírt tartalmat így tesszük az értelmező számára egyértelművé. Ha teljesült a műveletfejben megfogalmazott feltétel, akkor a beljebb húzással jelölt utasítás (`print "Páros"`) hajtódik végre, míg ha nem igaz, akkor az `else` ága kerül

kiértékelésre. Itt már nem volt szükség újabb feltételt vizsgálni, de ez is egy műveletfejnek minősül, így itt is találunk kettőspontot. Az utolsó sor tartalmazza azt a kiírandó szöveget, mellyel akkor találkozunk, ha nem volt igaz az első összehasonlítás.
Itt beljebb húzott tartalom tudatja velünk, hogy logikailag alá van rendelve a feltételvizsgálatnak (ezúttal az `else` ágnak).
Nyomát sem láthattuk a példában a más nyelveken gyakori `;` , `{}` vagy egyéb blokkhatároló jeleknek, s mégis jól látható, meddig tart egy logikai egység. Bármennyire is kellemetlennek tűnik eltérő logika alapján szerveződő programozási nyelv használata után megbarátkozni a kötelező behúzáshasználattal, egy idő után rájövünk, valóban a mi érdekünket is szolgálja, hiszen átlátható kódot eredményez. Ez pedig elsőrendű szemponttá válhat egy nagy projekt esetén, ha csapatmunkáról van szó, vagy ha egyszerűen csak saját programunkat szeretnénk karbantartani hosszú idő elteltével. Ehhez kapcsolódva érdemes még két programozói szokást magunkévá tennünk: a # jellel bevezetett megjegyzések használatát (ezt az értelmező figyelmen kívül hagyja, nekünk viszont roppant hasznos lehet később), és a hibák – valamint kivételek kezelését.

Dinamikus átalakulások

Bonyolódjunk bele még egy kicsit a számokba, nézzünk egy egyszerűnek tűnő példát: a faktoriálist. Az általános képlete szerint n! = n * (n-1) * (n-2) * … * 2 * 1. Ennek kiszámítására több megoldás is lehetséges, a rekurzív függvényhívás egyik klasszikus példájaként is hivatkozhatnánk rá, most nézzünk inkább egy egyszerűbb megoldást:

```
x=4 #ciklusváltozó és a képlet
➥része is egyben
y=x #tárolja a részleges
➥eredményeket és a végleges n!
➥-t
while x>1 :
        x=x-1
        y=y*x
print y #kiíratjuk n!-t
```

Az x-et úgynevezett ciklusváltozóként használtuk, értékét folyamatosan csökkentjük, s a while utasítás segítségével mindig megvizsgáljuk, vajon teljesül-e még az a feltétel, hogy nagyobb 1-nél. Ha igaz, akkor minden ami a kettőspont után van végrehajtásra kerül, ha hamis, akkor kilép a ciklusból, s esetünkben be is fejezi futását a program.
A példában egy kis számot, a négyet néztük meg, de érdemes tudni, hogy a faktoriálisok roppant gyorsan növekszenek, 13! már egy tízjegyű számot jelent. Próbáljuk ki a programot egy nagyobb számra, például 1000-re (`x=1000`). Azonnal észrevehető, hogy a gép sokáig „gondolkodik”, hiszen hatalmas számokkal kell dolgoznia. Ami kevésbé látható: a dinamikus változókezelés következtében észrevétlenül átváltott az addig használt ***integer*** típusú egész számokról ***long*** típusú hosszú egészekre, mivel előbbi csak körülbelül kétmilliárdig képes a számokat fogadni, míg utóbbi kis közelítéssel a fizikai memória határáig. Ha nem hisszük, próbáljuk ki, előbb egy új változóval:

```
z=1
print type(z)
```

Válaszul azt kapjuk, hogy a z változó típusa ***int*** , azaz ***integer***. Kérdezzük le az 1000! értékét tartalmazó y változó típusát is:

```
print type(y)
```

Valóban ***long*** típusú a változó. Mi történik akkor, ha e két típus keveredik, azaz például egy összeadás során mindkettő szerepel?

```
z=z+y
print type(z)
```

Az eredménynek olyan típusúnak kell lennie, amelyben elfér a nagyon nagy szám és a kisebb összege, azaz logikusan ***long*** típusú lesz. Ehhez hasonló ún. implicit típuskonverziót a *C* nyelv is használ.
Az eddigi példákban használt egész számokat vegyíthetjük lebegőpontos számokkal (***float***) is, s ott is hasonló jelenséget tapasztalunk, azaz például egy ***int*** típusú és egy ***float*** típusú házasságából ismét csak ***float*** kategóriába tartozó szám születik.

```
a=1
        print type(a)
b=2.5
        print type(b)
c=a+b
        print type(c)
```

Az ***int***, ***long***, ***float***, ***complex*** (eddig még nem említettük, a komplex számokat jelenti) kulcsszavak segítségével rákényszeríthetjük egy változóra az adott típus használatát (*C* és hasonló nyelveken ez az ún. explicit típuskonverzió). Ennél talán még érdekesebb a következő típusváltás:

```
az_elet_ertelme="42"
szam = 42
eredmeny=int(az_elet_ertelme)
➥+szam
print "Az élet értelme
➥bizonyosan nem " , eredmeny,
➥"!"
```

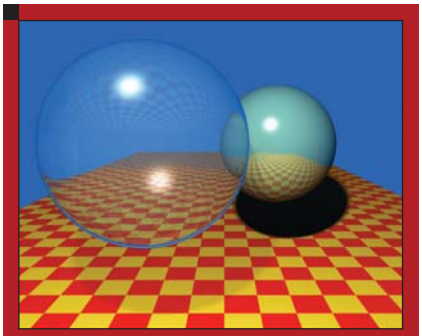
Az első változó egy egyszerű szöveg típusú változó volt, noha szemmel láthatólag csak egy számnak látszó értéket tárolt. Az `eredmeny` változóban ezt ki is használtuk, s ***int*** típusú számmá konvertálva már össze tudtuk adni egy igazi számmal.
A dokumentációt olvasva sok érdekes átalakítással és hasonló „trükkel” találkozhatunk, melyeket jól használva saját munkánkat könnyíthetjük meg. Kisebb példaprogramok írásával már ennyi ismeret birtokában is tudunk saját szerzeményünkkel hasznot hajtani. A különböző információforrások (levelezőlisták, hivatalos honlap, wiki stb.) jó alapot nyújthatnak, és a továbblépést is megkönnyítik ha elakadnánk valahol. A számok után a szöveg típusú változókkal, műveleti sorrenddel érdemes foglalkozni, s mindeközben észrevétlenül barátkozhatunk az objektum-orientált szemlélettel is.

Tóth Virgil Zoltán
(m_v@c2.hu)
Szoftverfejlesztő informatikus és rendszergazda, kedvence a Debian disztribúció. Szabadidejét legszívesebben felesége és szépirodalmi regények társaságában tölti. Lenyűgözőnek tartja a Linux rugalmasságát, és a vele dolgozók aktivitását.

Sugárkövetési algoritmusok (2. rész)

Ismét jelentkezik a sugarak szerelmeseinek szóló cikkünk, melyben tovább folytatjuk a fények birodalmában megkezdett utazásunkat. A fénysugarak rekurzív követésével olyan jelenségek valósághű szintézise válik lehetővé, amelyek más eljárásokkal csak igen nehézkesen állíthatóak elő. Részben ennek is köszönhető a több mint 26 éves múltra visszatekintő algoritmus aktualitása napjainkban is.

Sugárkövetés a 16. században
(Albrecht Dürer – Underweysung der Messung mit dem Zirkel und Richtscheyt)



1. ábra A rekurzív sugárkövetés lehetővé teszi az árnyékok, a tükröződés és a fénytörés modellezését

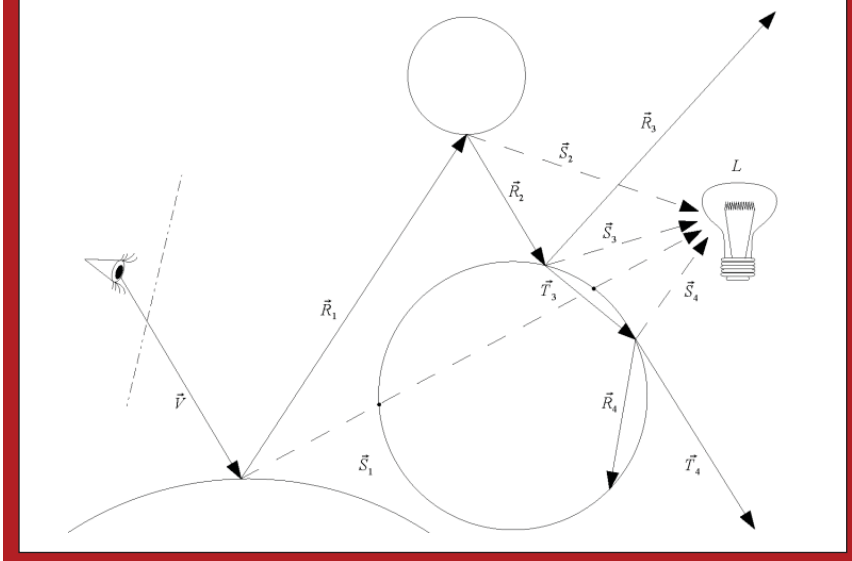
Az előző cikkben beszéltünk a sugárkövetés alapvető elemeiről és fogalmairól, valamint a ***sugárvetés*** (***ray casting***) megvalósításáról. Ebben a cikkben szó szerint tovább lépünk a fények útján, azaz a tükrözött és tört sugarak követését is megkíséreljük. De lesznek még itt további finomságok is, megismerkedünk két árnyalási modellel, további másodfokú ***implicit*** felületekkel, valamint egy procedurális objektummal. Meginvitálom hát a tisztelt olvasót, tartson velem e csodálatos utazáson. Útravalóként nem árt, ha egy kis matematikai tudást is csomagolunk, szükség lehet rá...

Rekurzív sugárkövetés

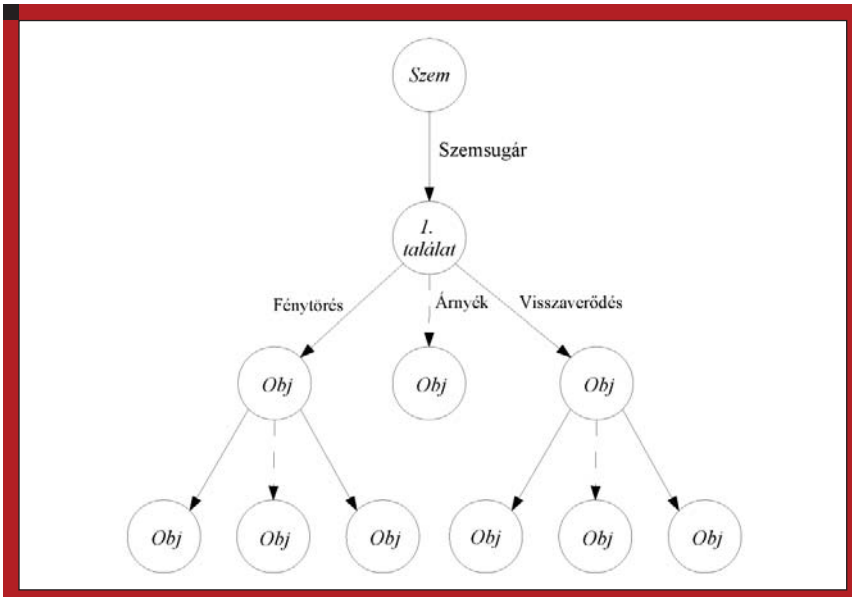
Az előző cikkben ismertetett ***sugárvetés*** (***ray casting***) a térben elhelyezett objektumok látható felületeinek meghatározására hivatott, azaz a szemből kilőtt sugárhoz megkeresi azt a felületi pontot, ahová az becsapódik. A ***rekurzív sugárkövetés*** ennél tovább merészkedik: a fény útját ennél tovább követi, így lehetővé válik olyan jelenségek szintézise, amelyeket más technikákkal bonyolult vagy talán nem is lehetséges előállítani. A szóban forgó jelenségek közül három olyan van, amely minden ***sugárkövető*** programban megtalálható: tükröződés (***reflexió***), fénytörés (***refrakció***), valamint a felületek által vetett árnyékok.

A most ismertetésre kerülő ***rekurzív sugárkövetés*** algoritmus ***Turner Whitted*** 1980-ban publikált munkáján alapszik. Hogy milyen módon követhetjük tovább a sugár útját? A megoldás a ***rekurzióban*** keresendő. A ***rekurzió*** egy olyan módszer, melynek során úgy számítunk ki egy végeredményt, hogy ugyanazt a műveletet ismételjük egymás után egy adott ***operanduson***, minden művelet egy részeredménnyel zárul, amelyet a sorban következő műveletkor (következő ***rekurzió***) felhasznál. ***Végtelen rekurzió*** az olyan ***rekurzió***, amely soha nem ér véget, ez ugye a legtöbb esetben használhatatlan. Nekünk inkább olyan ***rekurzió*** kellene, amely egy bizonyos számú végrehajtás után kilép (egy jó példa a ***végtelen rekurzióra*** a Linux világában járatos olvasóknak minden bizonnyal ismerős „***GNU***”, amely a „***GNU is Not Unix***” rövidítése).

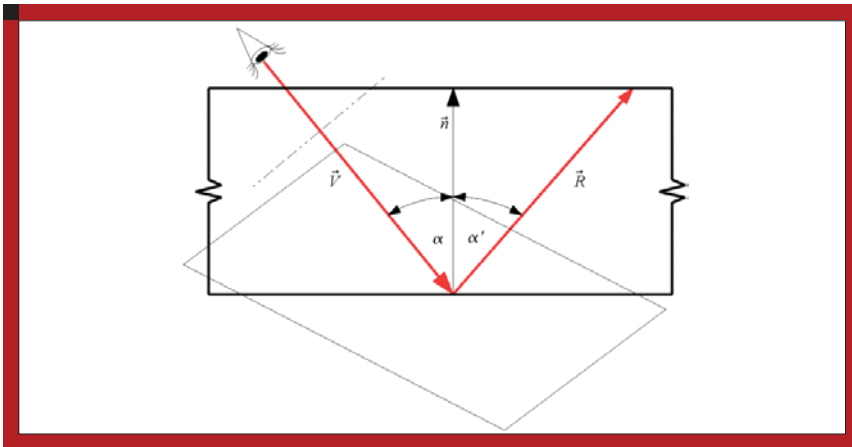
A ***rekurzív sugárkövetés*** – a ***sugárvetéshez*** hasonlóan – a ***szemsugár*** kilövésével kezdődik, majd az eltalált felület megállapítása után a folyamat megismétlődik: a kapott beesési pontból indulva újabb sugarakat indítunk útjukra, ám ezúttal a szem helyett a beesési pontból kiindulva a tükröződés, fénytörés, illetve a fényforrások irányába (***reflexiós***, ***refrakciós*** és ***árnyéksugarak***)



2. ábra Fényutak a térben (S – árnyéksugarak, R – reflexiós sugarak, T – refrakciós sugarak)



3. ábra A sugárfa



4. ábra Ideális visszaverődés (reflexió)

azért, hogy megtudjuk mely felületek tükröződnek, mely felületek takarják a fényforrásokat (mennyi közvetlen, azaz ***direkt megvilágítás*** érkezik a fényforrásokból), illetve mely felületet látjuk egy átlátszó objektumon keresztül. Ezeket a fénysugarakat ***másodlagos sugaraknak*** (***secondary rays***), míg a szemből indított sugarakat ***szemsugaraknak (eye rays)*** vagy ***elsődleges sugaraknak*** (***primary rays***) nevezzük.

Ha a ***másodlagos sugarak*** által eltalált felület is tükröző vagy áttetsző, újabb ***másodlagos sugarakat*** bocsátunk útjukra, azaz minden tört és visszavert sugár elindíthat újabb tört, visszavert és ***árnyéksugarakat.*** Az ismétlődő ***rekurziókkal*** (***másodlagos sugarak*** kilövésével) egy ún. ***sugárfa*** (***ray tree***) épül fel, ahol a ***szemsugár*** a fa gyökere.

A ***rekurziót*** addig folytatjuk, amíg az aktuális rekurzió a kép aktuális pixeléhez megfelelő mértékben járul hozzá, egyébként a ***fényút*** követését nincs értelme tovább folytatni.

Ezt legegyszerűbb módon úgy oldhatjuk meg, hogy egy konstanssal definiáljuk a ***sugárfa*** maximális mélységét, azaz ha a rekurziós szám elérte ezt az értéket, az aktuális eredményt tekintjük végleges eredménynek.

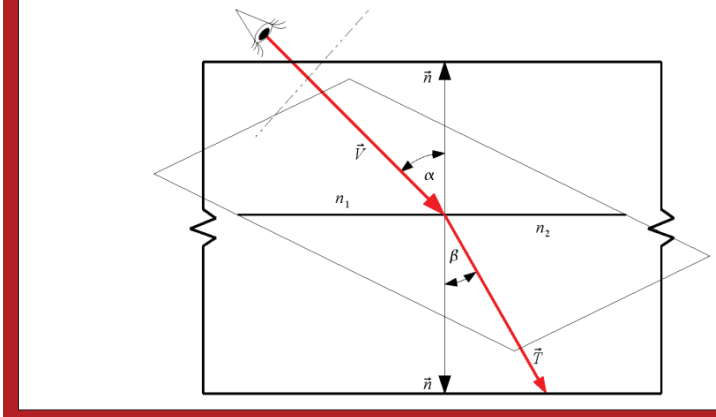
Az előző részben ismertetett ***sugárvetés*** algoritmus alapjait felhasználva játszi könnyedséggel készíthetünk ***rekurzív sugárkövetőt***, hiszen a ***rekurzív ray tracing*** működését tekintve nem más, mint ***rekurzív sugárvetés***.

A következő három fejezetben a már említett jelenségekkel fogunk kicsit részletesebben foglalkozni.

A tükröződés (reflexió)

Ha a fény valamely tükröző felület határára érkezik, akkor a beérkező fény egy része onnan visszaverődik, másik része pedig behatol a közegbe és ott elnyelődik azaz hővé alakul. Sima felületeknél a visszaverődés tükrös, durva felületeknél szórt, azaz diffúz jellegű. Mi a tökéletesen sima felületek ideális tükrözésével foglalkozunk, a lehetséges felületi durvaságot elhanyagoljuk, illetve a ***beesési szögtől*** függő tükrözött fényintenzítással nem foglalkozunk.

Először tisztázzunk néhány alapfogalmat. A 4. ábrán látható ***beeső fénysugár*** becsapódási pontjába állított ***felületi normálist beesési merőlegesnek***, a ***beeső***



5. ábra Ideális fénytörés (refrakció)

és a ***visszavert fénysugárnak*** a ***beesési merőlegessel*** bezárt szögét ***beesési*** illetve ***visszaverődési szögnek*** hívjuk.
Ahhoz, hogy a tükrözött fény útját tovább tudjuk követni, szükségünk van a beesési pontból indított, tükrözött fénysugár irányvektorára, melynek kiszámításához az ideális fényvisszaverődés törvényét hívjuk segítségül, mely szerint a beeső fénysugár, a beesési merőleges és a visszavert fénysugár egy síkban vannak, valamint a beesési szög megegyezik a visszaverődési szöggel. Az utóbbi szabály fordítva is érvényes, tehát a fénysugarak visszirányú követésénél is teljesül.
Ennek ismeretében a tükrözött sugár ***R*** irányvektora a következőképpen számítható:

$$\vec{R} = \vec{V} - 2 \cdot \left(\vec{n} \cdot \left(\vec{n} \cdot \vec{V}\right)\right)$$

Fénytörés (refrakció)

Ha a fény két optikailag átlátszó közeg határára ér, akkor vagy visszaverődik, vagy belép az új közegbe. A fény terjedési útja mindkét esetben (általában) megváltozik, de a fénytörés jelenségéről akkor beszélünk, ha a fény be is lép az új közegbe és a megtört irányban folytatja útját. Ennek az irányváltozásnak az az oka, hogy a fény különböző sebességgel terjed a két különböző anyagban.
A tört fény irányának meghatározásához segítségül hívjuk a ***Snellius-Descartes*** törvényt, azaz az ***ideális fénytörés törvényét***, mely kimondja, hogy a ***beeső fénysugár***, a ***megtört fénysugár*** és a ***beesési merőleges*** egy síkban vannak, illetve a ***beesési szög szinusza*** egyenesen arányos a ***törési szög szinuszával***, az arányossági tényező pedig a második közeg elsőre vonatkoztatott relatív törésmutatója.

$$n_{2,1} = \frac{c_1}{c_2} = \frac{\sin\alpha}{\sin\beta} = \frac{n_2}{n_1}$$

A különböző anyagokat a sugárkövetés során az ún. ***abszolút törésmutatóval*** (***index of refraction***) jellemezzük, amely a fény vákuumbeli és az adott anyagban mért terjedési sebességének a hányadosa (másképp megfogalmazva az anyag vákuumhoz viszonyított ***relatív törésmutatója***), tehát 1-nél mindig nagyobb szám.
Egy átlátszó anyag abszolút törésmutatóját felhasználva a fény vákuumból anyagba lépésekor fennáll a következő összefüggés

$$\sin\alpha = n \cdot \sin\beta$$

Könnyen belátható, hogy a fenti egyenletnek nem lesz megoldása, ha a beesési szög meghalad egy határértéket, nevezetesen a ***teljes visszaverődés határszögét.***

$$\Theta = \arcsin\frac{n_1}{n_2}$$

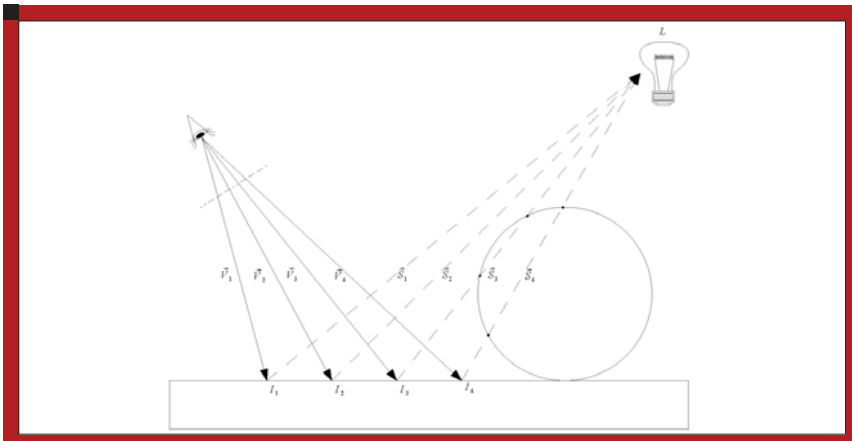
Ha a beesési szög meghaladja a teljes visszaverődés szögét, a fény nem tud behatolni az anyagba, így a beesési pontban a fent ismertetett reflexió jelensége jön létre, a felület tükrözőként viselkedik.
A fénytörés esetében ki kell számítanunk a tört fény irányát, hisz a sugarat ebben az irányban kell tovább követnünk, ennek kiszámítása a következő formulák alapján történik:

$$\cos\alpha = \vec{V} \cdot \vec{n} \quad \cos\beta = \sqrt{1 - \left(\frac{n_1}{n_2}\right)^2 \cdot \left(1 - (\cos\alpha)^2\right)} \quad \vec{T} = \left(\frac{n_1}{n_2}\right) \cdot \vec{V} + \left(\cos\beta - \frac{n_1}{n_2} \cdot \cos\alpha\right) \cdot \vec{n}$$

A fénytörési mutató a valóságban a fény hullámhosszától is függ (gondoljunk csak a prizmajelenségre), ettől mi eltekintünk, azt minden hullámhosszon konstansként értelmezzük.
A sugárkövetés során figyelnünk kell arra hogy a fény a becsapódási pontban belép az objektumba vagy éppen kilépni készül onnan. Az objektumból kilépő fény számításakor a felületből kifelé mutató normál vektor inverzét kell használnunk (azaz a felületi ***normális*** befelé mutat).
Néhány anyag törésmutatója a teljesség igénye nélkül: vákuum=1, levegő=1.000292, műanyag=1.11, üveg=1.22, víz=1.33, alkohol=1.362, gyémánt=2.417.

Árnyékok

Egy naív sugárkövető egy adott felületi pont fényességét a pontban vett felületi normális, a fényforrás illetve a kamera elhelyezkedését egy illuminációs modellben felhasználva számítja ki. Arról azonban elfeledkezik, hogy egy objektum el is takarhatja a fényforrást előle, azaz a pont árnyékba kerülhet.
Képzeljük magunkat egy felület valamely pontjába, s tekintsünk a fényforrás felé. Ha nem látjuk a fényforrást, biztosak lehetünk benne, hogy árnyékban vagyunk, mégpedig egy olyan felület által, mely a fényforrás és a szemünk között helyezkedik el.
Ahhoz, hogy meg tudjuk mondani, hogy a felületi pontból látjuk-e a fényforrást, egy másodlagos sugarat lövünk ki, amely a ***beesési pontból*** indulva a fényforrás felé mutat, majd ezen a sugáron megkeressük a legközelebbi pozitív ***t*** paraméterű metszéspontot. Ezután meg kell néznünk, hogy ez a metszéspont az ***árnyéksugár*** kiindulópontja és a fényforrás között van-e. Ha igen, akkor a vizsgált felületi pont árnyékban van, az árnyéksugár által eltalált objektum eltakarja a fényforrást a felületi pont elől.
Egy árnyéksugár kilövésének célja nem más, mint megállapítani, hogy egy adott pontba mennyi fény érkezik egy fényforrásból.
Áttetsző, fénytörő felületek esetében az eljárás kicsit bonyolódik.

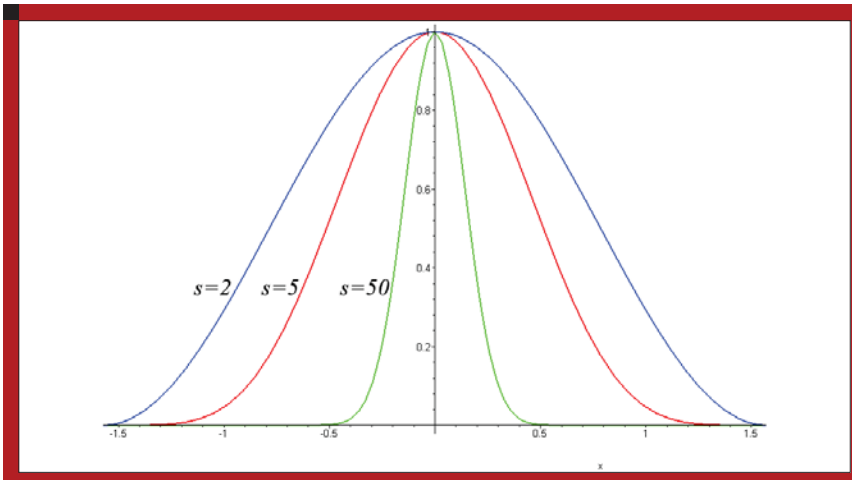


6. ábra Árnyéksugarak kilövése különböző beesési pontokból

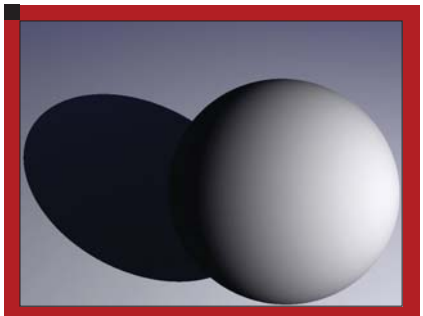
Képzeljünk el egy lencsét, amely a fókuszpontjába gyűjti a rajta áthaladó fénysugarakat. Ebben az esetben több sugár egy pontba gyűjtéséről beszélünk, s mivel a ***fényutat*** visszafelé követjük (***backward ray tracing***), sok-sok árnyéksugárra lenne szükségünk ahhoz, hogy a fókuszpontba gyűjtött fény nagyságát meg tudjuk becsülni (A kilőtt ***árnyéksugarak*** oroszlánrésze nem a fényforrásban érne véget).
Könnyen belátható, hogy az ***árnyéksugarakon*** értelmezett fénytörés és tükröződés erre a problémára nem nyújt hatékony megoldást. Az ilyen és ehhez hasonló ***indirekt illuminációs*** jelenségek modellezésére a ***globális illuminációs*** algoritmusok nyújtanak hatékony megoldást.
Az árnyéksugár kilövése folyamán az áttetsző anyagok fénytörését elhanyagoljuk és úgy számolunk velük, mintha a irányváltoztatás nélkül haladna át rajtuk. Meg kell keresnünk az összes olyan felületet, amely az árnyéksugáron a fényforrás és a vizsgált felületi pont között van, s ezen testek átlátszósági együtthatóival meg kell szűrnünk a fényforrás által kibocsátott fényt. Az átlátszósági együttható megmutatja, hogy a beérkező fényenergia mely részét nyeli el és mely része hatol be az anyagba.

Az Epsilon távolság

Az ***epsilon*** távolság használatának oka a ***számítási hiba***, mely a számábrázolás véges pontosságából adódik. Amikor megpróbáljuk megkeresni, hogy egy fénysugár melyik felületbe csapódik be, megkapjuk a fénysugárnak a legközelebbi felülettel vett metszéspontjának pozitív ***t*** pataméterét.
A számítási pontatlanság miatt azonban ezt a ***t*** paramétert visszahelyettesítve a sugár paraméteres egyenletrendszerébe, jó eséllyel egy olyan pontot kapunk, amely nem pontosan az eltalált felületen nyugszik, hanem a sugár mentén pozitív vagy negatív irányban el van tolva egy minimális távolsággal, ez a távolság a már említett ***számítási hiba***. Ha a kapott hibás pontot visszahelyettesítjük az eredeti felület egyenletébe, az az egyenletet nem fogja kielégíteni (mivel nem pontosan a felületen nyugszik).
Így tehát nem csoda, hogy ha a kapott hibás pontból indítjuk a további sugarakat és a hibás pont a felületen belülre került, azt tapasztaljuk, hogy a metszéspontkeresés ugyanazt a felületet hozza ki a legközelebbi felületnek, mint amely felszínéről indítottuk a ***másodlagos sugarat.*** Ennek a problémának a megoldására jól bevált megoldás egy olyan nagyon kicsi ***epsilon*** távolságot definiálni, amelynél kisebb távolságban levő metszéspontokat figyelmen kívül hagyunk.



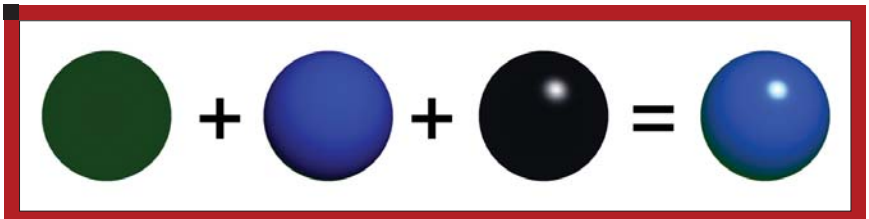
8. ábra A cos(x) függvényt különböző hatványokra emelve a Phong által megfigyelt jelenséget figyelhetjük meg



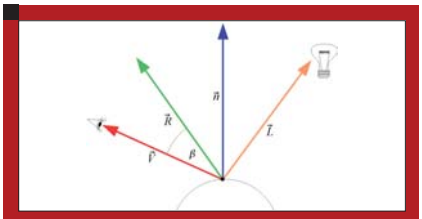
7. ábra Csendélet: diffúz gömb és árnyéka

Phong modellje

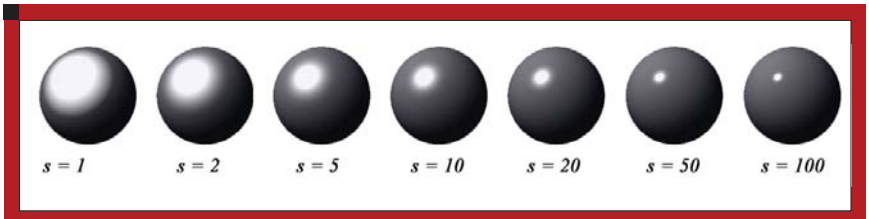
Az előző cikkben tárgyalt ***diffúz modellel*** nem lehet a fényes, „polírozott" felületű anyagok érzetét visszaadni, az csak durva felületek leegyszerűsített modellezésére alkalmas. A fényes felületekre az jellemző, hogy a beérkező fényenergia oroszlánrészét az ideális tükörirány környezetébe verik vissza, ennek köszönhető, hogy az ilyen felületeken egy ún.***spekuláris fényfolt*** jön létre.
A róla elnevezett eljárást ***Bui Tuong Phong*** amerikai kutató fejlesztette ki. ***Phong*** modellje ***empirikus***, azaz pusztán a megfigyelésen alapszik, nem a fény fizikai értelemben vett természetéből származtatott.
Megfigyelte, hogy fényes felületeken a ***spekuláris fényfolt*** nagysága ***a*** felületi anyag „polírozottságától", azaz fényességétől függ. Fényes felületű anyagoknál a fényfolt kicsi és ahogy távolodunk az ideális visszaverődési



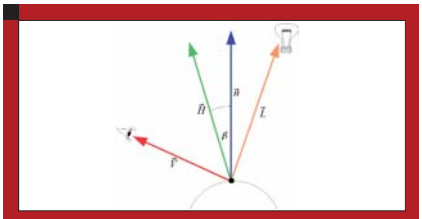
9. ábra A Phong modell tagjai balról jobbra: ambiens tag, diffúz tag és spekuláris tag. Ezek összegéből képződik a jobb oldalon látható kép.



10. ábra A Phong modell vektorkomponensei



11. ábra Phong modellel számított gömbök különböző s fényességi paraméterekkel. Jól megfigyelhető a spekuláris fényfolt átmérőjének változása



12. ábra A Blinn-Phong modell vektorkomponensei

irántól, egy bizonyos pont után meredeken zuhan az intenzitása. Matt felületeken a ***fényfolt*** nagy, az intenzitáscsökkenés pedig kisebb mértékű.
A megfigyelt viselkedés megvalósításához ***Phong*** a koszinusz függvényt vette alapul, az eredeti függvény hatványozásával modellezte a ***fényfolt***-jelenséget.
Phong modellje a következő ***árnyalási egyenlettel*** (***rendering equation***) írható fel:
L_p= ambiens tag + diffúz tag + spekuláris tag azaz

$$L_p = k_a i_a + \sum \left(k_d (\vec{L} \cdot \vec{n}) i_d + k_s (\vec{R} \cdot \vec{V})^s i_s \right)$$

ahol a $k_a i_a$ az ***ambiens*** tag, a $k_d (\vec{L} \cdot \vec{n}) i_d$ a ***diffúz tag, a*** $k_s (\vec{R} \cdot \vec{V})^s i_s$ pedig a ***spekuláris tag. A*** diffúz és ***spekuláris*** tagok számítását és összegzését minden fényforrásra el kell végezni, így kiszámolva a felületi pontban az ***eredő illuminációt***.
Az ambiens tag az indirekt megvilágítás hatását hanyag módon próbálja modellezni, intenzitása a felület minden pontjában azonos. A diffúz tagot már jól ismerhetjük az előző cikkből, a ***spekuláris*** tagról pedig már ejtettünk pár szót, ő a felelős a ***fényfoltok*** megjelenéséért.
A ***k*** tényezők az anyag az adott tagra vonatkozó színét (***ambiens***, ***diffúz*** és ***spekuláris*** színét) jelzik, az i_a a globális ambiens fényt, az ***i_d és i_s*** tényezők pedig a fényforrás megfelelő tulajdonságait, azaz külön definiálható a fényforrásból érkező ***diffúz*** és ***spekuláris*** intenzitás az anyagra nézve.
Nézzük meg, hogyan számoljuk ki a ***spekuláris*** tagot. A formulában szereplő ***V*** vektor a becsapódási pontból mutat a szembe, azaz az eredeti beeső sugár inverze. ***L*** vektor a becsapódási pontból a fényforrásba mutat, ***R*** vektor pedig a fényforrásból érkező fotonok ***ideális visszaverődési iránya***.
Ahogy a ***V*** vektor távolodik az ideális visszaverődési irány környezetétől, úgy nő a béta szög is, azaz az ***R*** és ***V*** vektorok skaláris szorzata a koszinusz függvény szerint egyre rohamosabban zuhanni kezd, ez pedig pontosan az a jelenség, amelyet ***Phong*** megfigyelései alapján leírt.
A ***spekuláris tagban*** található ***s*** hatványkitevővel az anyag felületi fényességét (***shininess***) szabályozhatjuk. Minél magasabb ez az érték, annál fényesebb anyag fényfoltját adja a ***spekuláris tag***.
A *11. ábrán* különböző ***s*** értékekkel ***renderelt*** gömböket láthatunk, ez kitűnően reprezentálja a ***spekuláris*** tag ***s*** paraméterének hatását.

A Blinn-Phong modell

A ***Phong*** modellhez szorosan kapcsolódik az ún. ***Blinn-Phong*** modell, mely mindössze annyiban tér el az eredeti ***Phong***-modelltől, hogy a spekuláris tag esetében a visszavert fény költséges kalkulációja helyett egy olyan ***H*** vektorral számolunk, amely a szemvektor és a beérkező fény között félúton helyezkedik el (***halfway vector***) és a következőképpen számítható:

$$\vec{H} = \vec{V} + \vec{L}$$

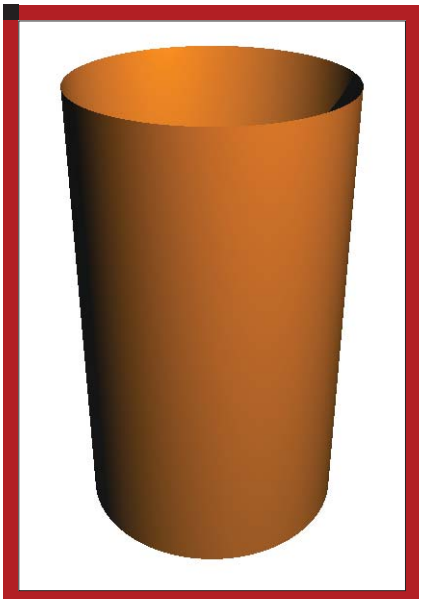
Ezt a vektort visszahelyettesítve a ***Phong modell R*** vektorába az

$$L_b = k_a i_a + \sum \left(k_d (\vec{L} \cdot \vec{n}) i_d + k_s (\vec{H} \cdot \vec{n})^s i_s \right)$$

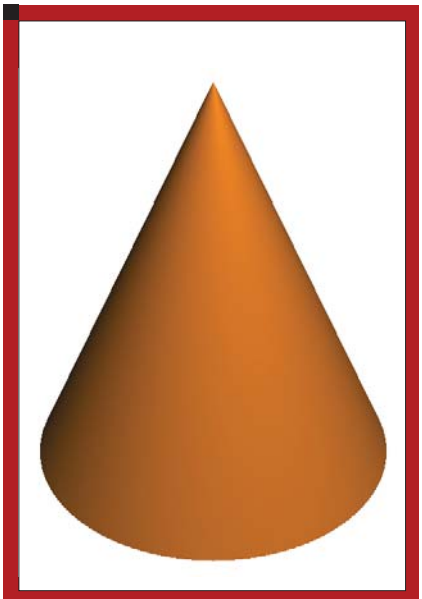
összefüggéshez jutunk, amely nem más, mint a ***Blinn-Phong megvilágítási modell*** egyenlete.
A ***Blinn-Phong*** modell figyelembe veszi a ***spekularitás*** változását annak függvényében, hogy milyen szögből nézzük a felületet, így az a ***Phong modellel*** szemben fizikailag helytállóbbnak (***plauzabilisebbnek***) nevezhető.
Az érdekesség kedvéért említem meg, hogy az ***OpenGL*** is a ***Blinn-Phong*** árnyalással számolja ki a háromszögek csúcsainak színét, majd ezeket interpolálja a felületen (***Gouraud shading***).

Az egyszerűsített (lokális) árnyalási egyenlet

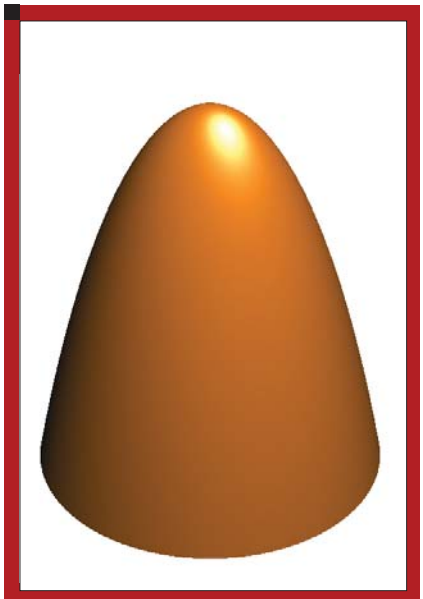
Eljutottunk oda, hogy van sok-sok komponensünk, ezekből össze kellene állítanunk egy általános árnyalási egyenletet, amely tartalmazza az eddig megismert komponenseket és bár elnagyoltan, de leírható vele a felületet adott irányba elhagyó fényenergia.
Azért elnagyoltan, mert ha visszaemlékszünk az eddig tanultakra, az indirekt megvilágítás költséges számolgatásától az egyszerű ***rekurzív sugárkövetés*** eltekint, azokat teljesen elhanyagolja.



13. ábra A hengerpalást



14. ábra A kúp



15. ábra A paraboloid

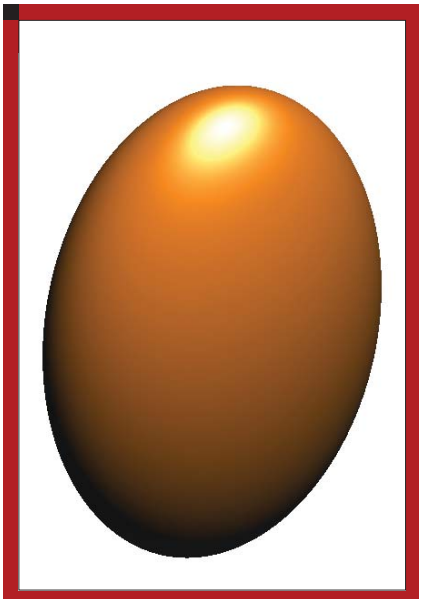
Ha összeírjuk egyetlen formulába az eddig tanultakat, megkapjuk az ***árnyalási egyenlet egyszerűsített alakját***, amely általánosan jellemző a lokális illuminációs modellekre:

$$L = L_e + L_a + \sum (L_d + L_s) + k_r \cdot L_r + k_t \cdot L_t$$

Ahol k_t, k_r az anyag fényáteresztő ill. tükröző színe (melyek definiálják, hogy az anyag milyen intenzitással engedi át magán illetve tükrözi a ***beeső fényt***); L_d, L_s a fenti modellekből ismert diffúz és ***spekuláris*** komponensek; L_a az anyag eredő ***ambiens*** színe ($L_a = i_a k_a$); L_r, L_t *a* felület adott pontjába a környezetből érkező ideálisan tört és tükrözött fényenergia. Végére hagytuk az L_e komponens tárgyalását, amely a felület által a környezetbe sugárzott fény intenzitása, amennyiben az objektum fényt ***emittál*** a környezetbe. Mivel a hagyományos ***ray tracing*** algoritmus csak pontszerű fényforrásokat tud kezelni, ennek igazából nem lesz látványos eredménye.

Másodfokú felületek

Az előző részben tárgyaltuk a gömböt, mint a legegyszerűbb másodfokú felületet. Most további három másodfokú (***kvadratikus***) felülettel ismerkedünk meg. Itt csak a felületek egyenleteit közöljük, a sugárral vett metszéspontokat a gömbhöz hasonló módon, visszahelyettesítéssel és a *t* paraméterre való megoldás keresésével lehet elvégezni.



16. ábra Az ellipszoid

Hengerpalást

Az Y tengellyel egybeeső hossztengelyű, r sugarú végtelen hengerpalást felületét azok a pontok alkotják, melyek XZ síkon vett hossztengelytől mért távolsága r.
Így a hengerpalást minden felületi pontjára teljesül a következő másodfokú egyenlet:

$$\sqrt{P_X^2 + P_Z^2} - r = 0$$

A végtelenbe emelkedő hengerpalást helyett a megszokott változatokat kaphatjuk, ha a koordinátákat a $Y_{max}>Y>0$ egyenlőtlenségekkel korlátozzuk, azaz ha a metszéspont Y koordinátája Y_{max}-nál nagyobb, vagy 0-nál kisebb, a kapott metszéspontot figyelmen kívül hagyjuk.

Kúp

Az Y koordinátatengellyel egybeeső hossztengelyű, h magasságú és a z=0 síkon vett r sugarú végtelen kúp egyenlete

$$\frac{P_X^2 + P_Z^2}{C^2} - (P_Y - h)^2 = 0 \text{ ahol } C = \frac{r}{h}$$

A hengerpalásthoz hasonlóan itt is megkaphatjuk a szokásos alakzatot, ha az Y koordinátákat a fenti egyenlőtlenséghez hasonlóan korlátozzuk.

Paraboloid

A parabola felületét azon pontok alkotják, melyek kiegyenlítik a következő ***kvadratikus*** egyenletet:

$$P_X^2 + P_Z^2 - P_Y = 0$$

Ellipszoid

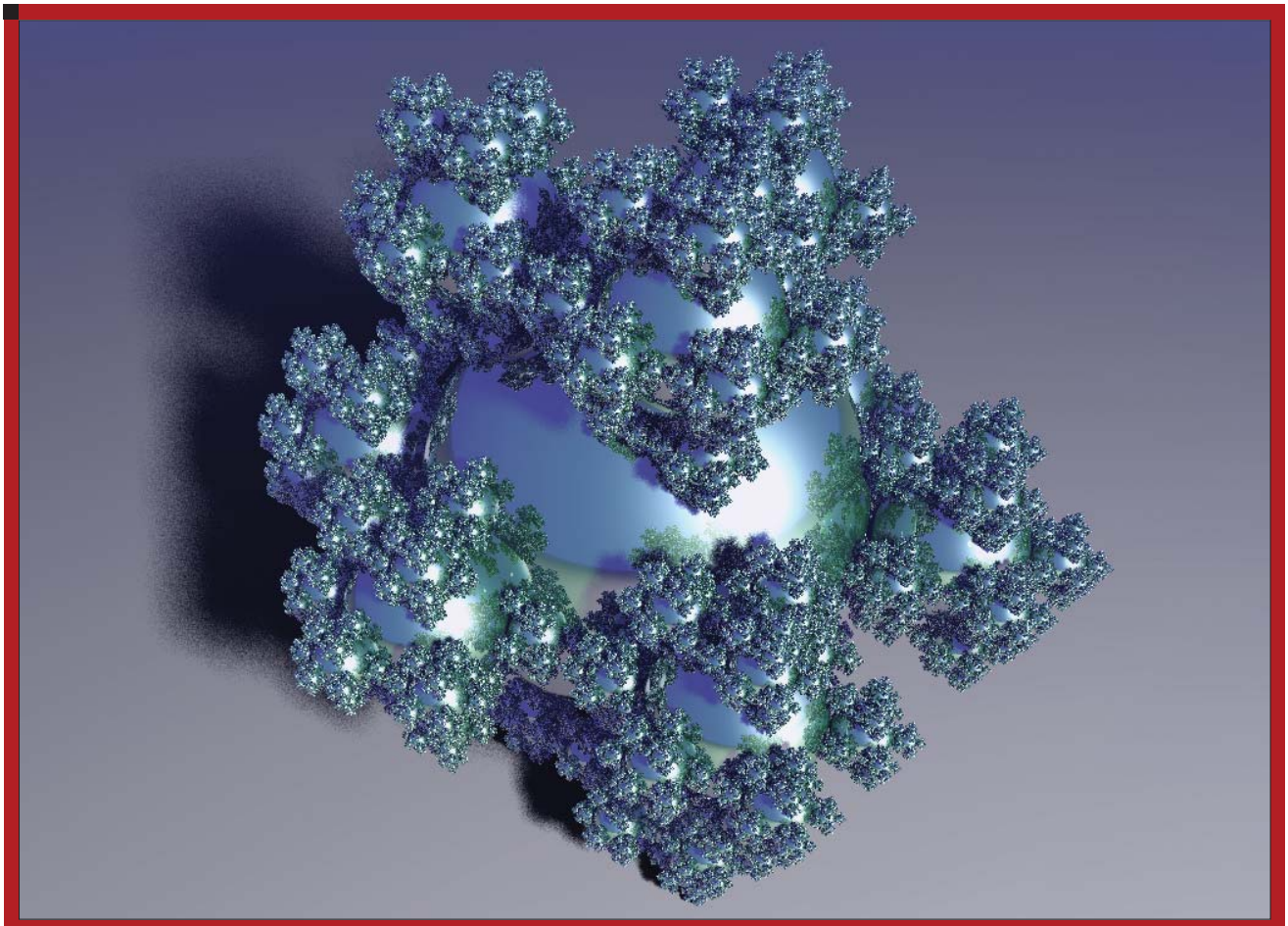
Az ellipszoid egyenlete

$$\frac{P_X^2}{a^2} + \frac{P_Y^2}{b^2} + \frac{P_Z^2}{c^2} - 1 = 0$$

ahol a, b, c az ellipszoid sugarait jelenti.

Felületi normálvektorok

A felületek P pontban vett normálisának előállításához az implicit felületünk egyenletéből kiszámítjuk



17. ábra A „gömbpehely" 6 rekurziós lépés után. Az ábrán látható objektum megközelítőleg 3 millió gömbből áll

a felület P pontbeli ***gradiensvektorát***. Egy felület P pontbeli ***gradiensvektora*** ugyanis mindig merőleges a felületre az adott pontban, tehát a felület P pontbeli ***normálvektoraként*** értelmezhető. A ***gradiensvektor*** komponenseit a felület egyenletének ***X, Y*** és ***Z*** szerinti ***parciális deriváltjai*** alkotják:

$$\nabla f = \frac{\partial f}{\partial X} i + \frac{\partial f}{\partial Y} j + \frac{\partial f}{\partial Z} k$$

Vegyük például az r sugarú, origó középpontú gömböt, ennek egyenletét $r^2 = X^2 + Y^2 + Z^2$ alakra hozva a következő parciális deriváltakat képezhetjük:

$$\frac{\partial f}{\partial X} = 2 \cdot P_X ; \frac{\partial f}{\partial y} = 2 \cdot P_Y , \frac{\partial f}{\partial Z} = 2 \cdot P_Z$$

Az így kapott ***gradiensvektor*** a gömb X,Y,Z pontbeli ***normálvektora***. Hasonló eljárást alkalmazhatunk a fentebb tárgyalt ***implicit felületeknél*** is.

Procedurális objektumok

Egy kis fantáziával és kreativitással különböző elemi felületekből építkezve algoritmikusan létrehozhatunk összetettebb objektumokat, ezeket ***procedurális objektumoknak*** nevezik a szakirodalomban. A *17. ábrán* látható ***„gömbpehely"*** is egy ilyen ***komplex***, ***procedurálisan*** előállított objektum, mely ***rekurzív*** módon generált gömbökből van felépítve. Algoritmikus generálása meglehetősen egyszerű, megfelelően mély ***rekurzióval*** rendkívül részletgazdag és ***impresszív*** látványt nyújt. A ***rekurzió*** minden lépésében az előző ***rekurzió*** által előállított minden egyes gömb körül 12 újabb gömb keletkezik oly módon, hogy azok felülete érinti az előző rekurzióból származó gömb felületét. Az újabb gömbök sugara a szülő (előző ***rekurzióból*** származó) gömb sugaránál kisebb, én a szülő gömb sugarának harmadolásával számoltam. Mivel a gömbök tárolása a számítógép memóriájában történik, valamint a metszéspontkeresés költsége a gömbök számával nő, a rekurziós lépések számát viszonylag rövidre kell fognunk.

Az előző cikkel kapcsolatban kaptam pár olvasói visszajelzést, köszönöm a pozitív, építő jellegű kritikákat. Az ebbe a cikkbe tervezett témák közül hely hiányában kimaradtakat a következő alkalommal igyekszem majd pótolni, ahol lesz szó ***textúrázásról***, ***sugártranszformációkról, a tóruszról*** mint negyedfokú felületről, háromszög alapú modellek ***rendereléséről***, valamint az elosztott sugárkövetésről (***distributed ray tracing***) is ejtünk majd pár szót a tervek szerint.

Szendi Ákos
(akos.szendi@gomortel.hu)

27 éves, szabadúszó programozóként tevékenykedik. A Miskolci Egyetem villamosmérnök szakos hallgatója. Kevéske szabadidejében gitározni tanul vagy épp egy jó könyvet tart a kezében.

SDL – Multimédiás programozói könyvtár (3. rész)

Az előző alkalmak során eseménykezelésből is kaptunk egy kis ízelítőt. Most egy kicsit a fülünknek fogunk kedvezni, kezelésbe vesszük az SDL audio szolgáltatásait, majd az optikai meghajtónkat is felbolygatjuk. A grafika, multimédia, játékfejlesztés nélkülözhetetlen eszköze az időzítő. Erről is fogunk a cikk során tárgyalni.

Az *SDL* igen sok oldalú programozói eszköz. Véleményem szerint a legrövidebb út, ahhoz hogy *Linux* alatt könnyen tudjunk grafikát programozni. Az eddigi ismereteink alapján már nem lehet gondunk az eseményekkel és a videó szolgáltatással. Ideje, hogy belevágjunk az audio szolgáltatás alapjaiba, hiszen végül is egy multimédiás szolgáltatáscsaláddal van dolgunk, amiből nem maradhat ki a „fülbevaló” sem.

Indulhat a koncert

Kép van hang nincs. Mondhatnánk az eddig elhangzottak után, de ne legyünk türelmetlenek! A következőkben az *SDL* audio támogatása kerül előtérbe. Valójában most nem is az *SDL API*-ba tartozó dolgok következnek, az úgynevezett standard *SDL* könyvtárak egyikét fogjuk kicsit szemügyre venni, abból a célból, hogy végre elérjük, hogy *SDL* segítségével programozni tudjuk a hangeszközünket.
Egy egyszerű példát fogunk látni az *SDL_mixer* könyvtár használatára. Itt hívnám fel a figyelmet, hogy a fordítási direktívák között fel kell tüntetnünk majd, hogy az *SDL_mixer* könyvtárat szeretnénk használni a programunkhoz. Tehát egy *SDL_mixer*-es program fordítása így nézhet ki valahogy:

```
g++ `sdl-config --libs
➥--cflags` -lSDL_mixer
➥-pedantic -Wall -ansi
➥main.cpp -o sdl01
```

Ami az eddigiekhez képest változott az a `-lSDL_mixer` felbukkanása.
A kódunk is alakul picit, az eddigiekhez képest. Mivel a most következő eszközök nem tartoznak az *SDL API*-ba ezért nem lesz elég, hogy csak az *SDL.h*-t építsük be a programba, hanem szükségünk lesz az `SDL_mixer.h`-ra is.
Egy általános menete az *SDL*-ben egy hang lejátszásának a következő: hangeszköz megnyitása, hangfájl betöltése a memóriába, lejátszás, majd a hangadatok kitörlése a memóriából ha már nincs rá szükségünk. Lássuk hát a példánkat (*1. lista*).
A *1. listát* tekintve nézzük kicsit részletesebben az egészet, hiszen a kommentárok nem sok mindent

1. Lista Példaprogram hangeszköz használatára SDL segítségével

```
#include <iostream>
#include "SDL.h"
#include "SDL_mixer.h"

int main()
{
   // ez a mutató fogja mutatni a hangadatok
   // helyét a memóriában
    Mix_Music *HANGANYAG = NULL;

   // nyitunk egy ablakot, hogy az
   // eseménykezelést le tudjuk bonyolítani
    SDL_Surface *kepernyo;
    SDL_Event esemeny;
    // a formátumra vonatkozó beállítások
    // eltárolása
    int frekvencia = 22050;
    Uint16 formatum = AUDIO_S16SYS;
    int csatornak = 2;
    int buffer = 4096;

   // az eseménykezeléshez használjuk fel
    int vege = 0;

   // itt már az audio szolgáltatást is
   // inicializálni kell
    if ( SDL_Init(SDL_INIT_AUDIO|SDL_INIT_VIDEO)
➥< 0 )
    {
        std::cout << "Nem tudom indítani az
➥SDL-t: " << SDL_GetError();
```

1. Lista folytatás

```
        exit(1);
    }

    atexit(SDL_Quit);

    // az első lépés, hogy megnyitjuk az adott
    // paraméterekkel a hangeszközt
    if (Mix_OpenAudio(frekvencia,formatum,
➥csatornak,buffer) < 0)
    {
        std::cout << "Nem tudom megnyitni a
➥hangeszközt!\n";
        exit(1);
    }
   // lekérjük, hogy milyen beállításokat tudott
   // eszközölni a hangrendszer
    Mix_QuerySpec(&frekvencia,&formatum,
➥&csatornak);

    // betöltjük a lejátszani kívánt anyagot
    HANGANYAG = Mix_LoadMUS("music.wav");
    Mix_PlayMusic(HANGANYAG,-1);

   // ez mar ismerős kell, hogy legyen
    kepernyo = SDL_SetVideoMode(320,240,0,0);
    while (!vege)
    {
        while(SDL_PollEvent(&esemeny))
        {
            if (esemeny.type == SDL_KEYUP)
            {
                // ESC-re vége a programnak
                if (esemeny.key.keysym.sym ==
➥SDLK_ESCAPE)
                {
                        vege = 1;
                // megállítjuk a lejátszást és
                // felszabadítjuk a hanganyag helyét
                        Mix_HaltMusic();
                        Mix_FreeMusic
➥(HANGANYAG);
                        HANGANYAG = NULL;
                }
            }
            // hogy azért mégse együnk annyi CPU
            // időt
            SDL_Delay(50);
        }
        return 0;
}
```

mondanak el. Az ***SDL_mixer*** könyvtár segítségével ***WAV, MOD, MID, OGG és MP3*** formátumok lejátszására vagyunk képesek. A ***Mix_OpenAudio*** használatával nyitjuk meg a hangeszközünket a paraméterekben megadott formátum lejátszására. A formátum értéke mindig legyen ***AUDIO_S16SYS***. Régebbi ***SDL*** verziókban több variáció is volt, de az elkövetkezendőkben a fejlesztők biztosra mentek és bevezették a ***Mix_QuerySpec*** eljárást, mely visszatér azokkal az értékekkel, melyek ténylegesen használhatóak a hangkártyán. A ***Mix_LoadMUS*** tölti be adott memóriahelyre a lejátszandó anyagot. A lényeges elem a ***Mix_PlayMusic*** melynek első paramétere a hanganyagot tartalmazó memóriaterület mutatója, a második pedig a lejátszások számát adja meg. Ha ennek az értéke -1 akkor ismételgeti az anyagot a rendszer. Az eseménykezelésről már volt szó, azonban van még két új elemünk: ***Mix_HaltMusic*** és ***Mix_FreeMusic***. Az első eljárás leállítja a lejátszást, a második pedig felszabadítja a hangadatok tárolására használt memóriahelyet. Az utóbbinak egy paramétere van mégpedig a felszabadítandó terület mutatója.

A ***Mix_PlayMusic*** párja a ***Mix_PlayChannel*** eljárás, mely annyival több társánál, hogy az első paraméterben a lejátszó csatorna számát adhatjuk meg. Ha -1 értéket adunk meg itt, akkor a rendszer adja automatikusan a csatornát. Ezen eljárás kiválóan alkalmazható két vagy több hanganyag egy időben történő lejátszásához.

Bolygassuk fel az optikai meghajtónkat!

Az ***SDL*** optikai meghajtókat kezelő szolgáltatásai a legszükségesebb elemeket tartalmazzák. Szépen sorban végig fogunk menni mindegyik lehetőségen. Elsősorban valami képet kell kapnunk a rendszerünkhöz csatlakoztatott meghajtókról. Ebben lesz segítségünkre az ***SDL_CDNumDrives*** és az ***SDL_CDName*** függvény. Az elnevezés megint csak igen beszédes, az első lekérdezi az optikai meghajtók számát, a második a rendszerbeli nevüket adja vissza. Ezek alapján már könnyen tudunk olyan programot írni, mely kilistázza rendszerünk optikai meghajtóit, azok nevével együtt (*2. lista*). Mint már megszokhattuk, ebben az esetben is meg kell „nyitni” az eszközt, mielőtt műveleteket hajtanánk végre rajta. Az ***SDL_CDOpen*** függvény segítségével választhatunk ki egy meghajtót, valamit az ***SDL_CDClose*** függvénnyel zárhatjuk le azt. Például nyissuk ki a tálcát a nulladik sorszámú optikai meghajtón (*3. lista*).

Lehetőségünk van kideríteni az ***SDL*** segítségével, hogy adott meghajtóban, milyen típusú adathordozó van, vagy hogy egyáltalán van-e benne valami. Az `SDL_CDStatus` függvény lesz ebben segítségünkre. Paraméterként természetesen a vizsgálandó meghajtó sorszámát várja. A függvény egy `CDStatus` nevű adatstruktúrát ad vissza, mely alapján már tájékozódhatunk a meghajtó állapotáról.

A struktúrát a következő értékek alapján lehet vizsgálni: `CD_TRAYEMPTY`, `CD_STOPPED`, `CD_PLAYING`, `CD_PAUSED`, `CD_ERROR`. Nemsokára látunk egy példát az alkalmazására. Előtte még

2. Lista Optikai meghajtókat kilistázó program

```
#include <iostream>
#include "SDL.h"

int main()
{

    if ( SDL_Init
➥(SDL_INIT_CDROM) < 0 ) {
        std::cout << "Nem tudom
➥indítani az SDL-t:
➥" << SDL_GetError();
        exit(1);
    }

    atexit(SDL_Quit);

    std::cout << "Meghajtók
➥szama: " << SDL_CDNumDrives()
➥<< std::endl;

    for ( int i=0; i<SDL_CDNum
➥Drives(); ++i )
    {
        std::cout << "Meghajtó:
➥" << i << " " <<
➥SDL_CDName(i) << std::endl;
    }

    return 0;
}
```

3. Lista Tálca kinyitása optikai meghajtón

```
/* SDL inicializáció */
...
SDL_CD *cdrom;

cdrom = SDL_CDOpen(0);

SDL_CDEject(0);

SDL_CDClose(cdrom);
...
```

4. Lista Példa audio lemezek kezelésére

```
cdrom = SDL_CDOpen(0);

        // van "valaki" a
        // meghajtóban ?
        if (CD_INDRIVE(SDL_CD
➥Status(cdrom)))
        {
            // cdrom->numtracks
// megadja a lemezen levő
// trackek szamat
            std::cout <<
➥"Trackek száma: " << cdrom-
➥>numtracks << "\n";

            std::cout << "Az
➥első track lejátszása...\n";
            // az első tracket
// lejátszuk
            SDL_CDPlayTracks
➥(cdrom, 0, 0, 1, 0);
        }

    SDL_CDClose(cdrom);
```

tekintsük át, hogy milyen lehetőségeket nyújt az *SDL* az audio anyagot tartalmazó adathordozók kezelésére.

CD-AUDIO támogatása

Az *SDL* négy, kifejezettem audio lemezek kezelését lehetővé tevő függvénnyel rendelkezik. Az első az `SDL_CDPlay`. Ez az egész *CD* lejátszását teszi lehetővé, viszont e mellet létezik még egy nagyon hasonló függvény, az `SDL_CDPlayTracks`, melynek funkciói gazdagabbak és természetesen ugyanazt a funkciót is ellátja mint az előbb említett `SDL_CDPlay`. Ezek miatt inkább erről ejtsünk néhány szót. Az `SDL_CDPlayTracks` rendre a következőket várja paraméterként: a meghajtó száma, kezdő track száma, kezdő frame száma, utoljára lejátszandó track száma, majd ennek a frame száma. A framek audio adategységeket jelölnek. A meghajtó státuszának beolvasása során az *SDL* a `CD_FPS` változóban tárolja számunkra az adott audio lemezhez tartozó egy másodpercre eső framek számát.
Ezek alapján már könnyen megy a következő: játsszuk le a behelyezett audio lemez második sávjának első tíz másodpercét:

```
SDL_CDPlayTracks(cdrom,
➥1,0,0,CD_FPS*10)
```

Az `SDL_CDPause` és az `SDL_CDResume` függvények segítségével, rendre megállíthatjuk illetve folytathatjuk a lejátszást. Mindkét függvény egy `SDL_CD` típusú változót vár paraméterként. Lássunk példaként egy olyan kódot, mely az eddig említett funkciók mindegyikére mutat alkalmazást (*4. lista*).

SDL Timer – Az időzítő

Az optikai meghajtók bűvölése után térjünk át, egy másik igen hasznos témára, az ***időzítők*** használatára.
Az időzítő szolgáltatás tagjai nincsenek sokan: `SDL_GetTicks`, `SDL_Delay`, `SDL_AddTimer`, `SDL_RemoveTimer`. Ezeken kívül még alkalmazható az ***SDL_SetTimer*** funkció is, de ezt a fejlesztők már nem javasolják, mert az elkövetkezendő verziókban ez a funkció már nem fog helyet kapni. Helyette inkább használjuk az említett `SDL_AddTimert`. Lássuk sorjában az időzítő szolgáltatás tagjai.
Az `SDL_GetTicks` függvény az *SDL* inicializálása óta eltelt időt adja vissza ***milliszekundumban***. A visszaadott érték típusa ***Uin32***. Ebből is látszik, hogy nem számolhatja végtelenségig ezt az időt. Körülbelül 49,7 nap után fordul át a *„mutató”*.
Várakozni a program futása közben az `SDL_Delay` utasítással lehetséges, mely egy ***Uint32*** típusú értéket vár paraméterként, melynek értéke a várakozás időtartama megint csak ***milliszekundumban***.
Az egyik legfontosabb az ***SDL_AddTimer***. Segítségével megadott idő intervallumonként hívhatunk meg egy alprogramot. Ennek „párja” az `SDL_RemoveTimer`, mely az előbbi függvény által létrehozott időzítőt szünteti meg.
Ezek alapján, lássunk egy programot mely két másodpercenként kiír egy üzenetet a képernyőre, majd megszünteti az időzítőt és vár még pár másodpercig, hogy tényleg lássuk

5. Lista Az időzített eljárásunk ki fog írni egy üzenetet

```
Uint32 idozito(Uint32
➥intervallum,void *param)
{

    std::cout << "Időzítő
➥üzenete...\n";

    return (intervallum);
}
```

6. lista Példa időzítő alkalmazására

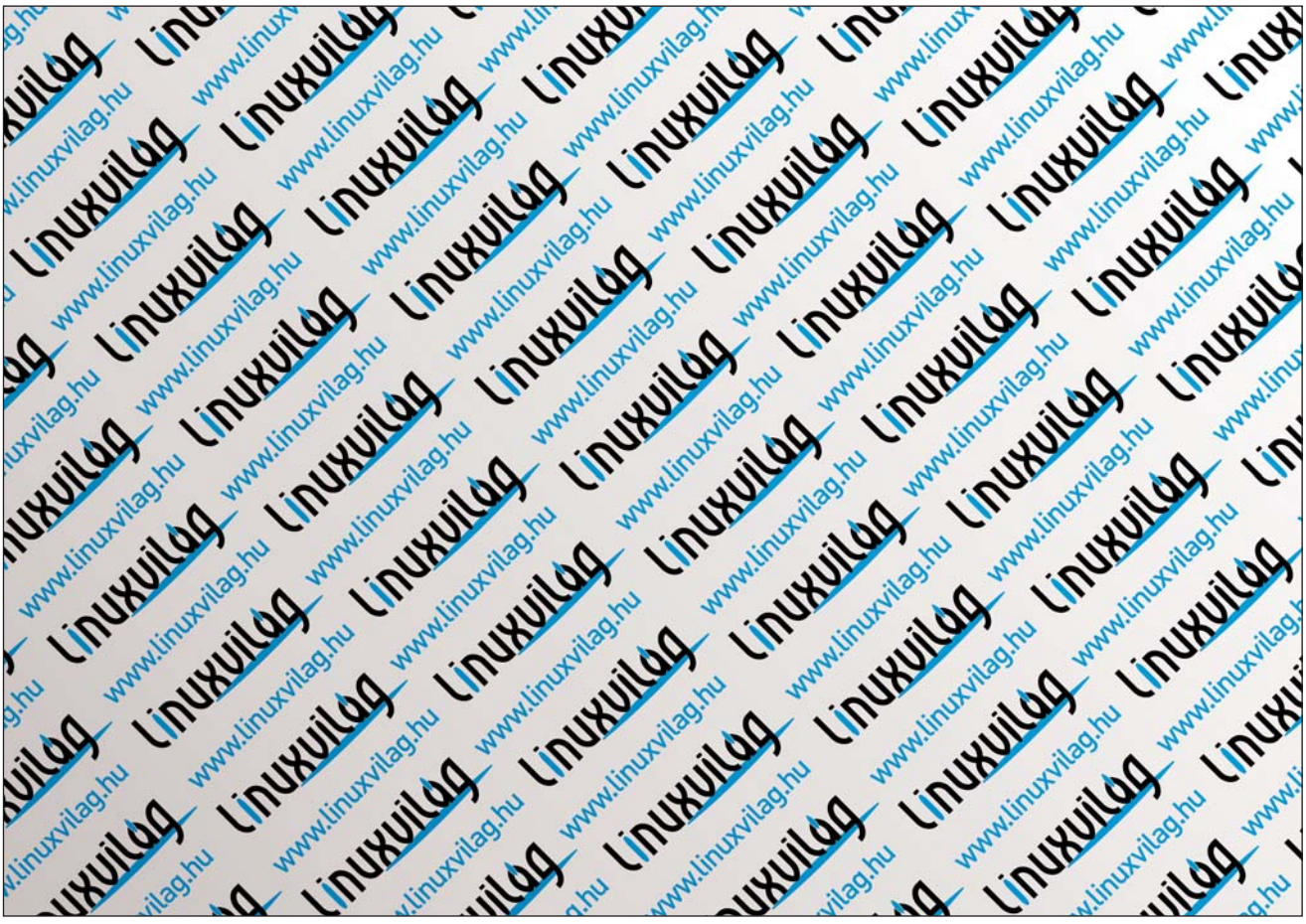
```
...
// inicializáltuk az SDL-t
...
// létrehozzuk az időzítőt, 2
// másodpercenként kapunk
// üzenetet
    SDL_TimerID mytimer =
➥SDL_AddTimer(2000,
➥idozito, NULL);
    // varunk 10 másodpercet,
// addig kapunk üzeneteket
    SDL_Delay(10000);
// töröljük az időzítőt
    SDL_RemoveTimer(mytimer);

    std::cout << "Már nincs
➥timer.\n";
// várunk még egy kicsit, hogy
// lássuk tényleg törölve lett
    SDL_Delay(5000);
...
```

megszűntek az üzenetek. Ez az az eljárás, amit az időzítőnk hívogatni fog (*5. lista*)
Egy időzítő által meghívott eljárásnak teljesítenie kell bizonyos követelményeket. Visszatérési értéke ***Uint32*** típusú kell, hogy legyen. Az első paraméter szinten ilyen típus, és ezt a paramétert vissza kell adnia. Ezek a megkötések az időzítőrendszer implementációjából fakadnak. Most pedig rakjuk össze a programunkat (*6. lista*). Az `SDL_AddTimer` függvény tehát első paraméterként az intervallumot várja (milliszekundum), másodikként a meghívandó eljárás nevét, majd harmadszorra az eljárás által várt paraméterek következnek. Ez utóbbi a mi esetünkben `NULL`, mert nem dolgoz fel paramétereket az időzítőeljárásunk.

SDL végszó

A cikksorozat három részében megismerkedhettünk az ***SDL*** multimédiás programozói könyvtár alapvető elemeivel. Remélhetőleg elég segítséget nyújtottam, ahhoz, hogy az érdeklődők elkezdhessék az ***SDL*** segítségével való fejlesztéseiket. Megtanultuk az eseménykezelés, videó szolgáltatások, optikai meghajtók, időzítők és végül a hangszolgáltatás kezelésének alapjait. Játékfejlesztéshez Linux alatt kiválóan alkalmazható, főleg, hogy együtt használhatjuk az ***OpenGL*** könyvtárral is. Aki ráérzett az ízére, annak sok sikert kívánok a továbbiakban a programozáshoz, hogy sok hasznos multimédiás alkalmazás szülessen a linuxos világot gyarapítva.

Radics Péter



Egy intelligens zenekonverter

a gnormalize

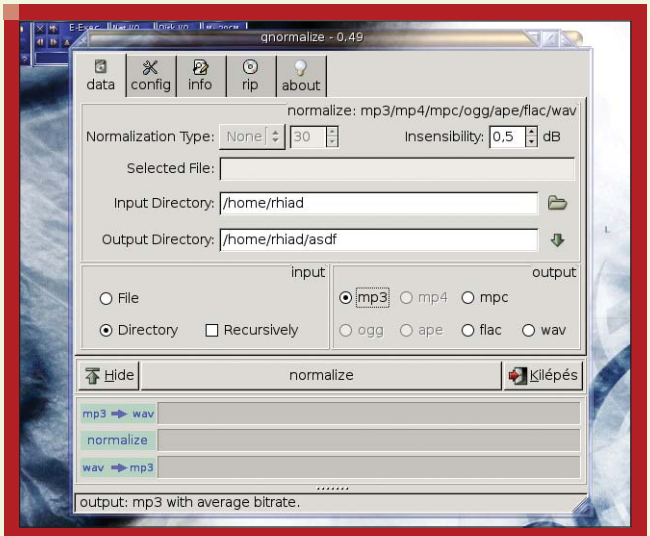
A mai rohanó és drága világban már évek óta népszerű az úgynevezett zenefájlok használata. Sajnos vagy nem, de ezek a formátumok kezdik kiszorítani a piacról a hagyományos hanghordozókat. Ebben a cikkben egy flexibilis audiokonverterrel fogunk megismerkedni, amelyik sok formátumot ismer, GNU/Linuxra íródott, ráadásul sok egyéb hasznos szolgáltatást is magában hordoz.

Sokan kérdezhetik, hogy vajon melyik táborba tartozhatom én? Nos, akkor elmondom. Szerintem nagyon hasznosak a zenefájlok, de ezek nem mehetnek a tradicionális hanghordozók rovására.
Ha jól emlékszem, akkor egy hasonló ömlengést elkövettem már az ***Azureusról*** szóló cikkemben is.
Gondoljuk el, hogy mi lenne akkor, ha mindenki csak letöltögetné kedvencének albumjait?
Eltűnnének a hanghordozók. Eltűnne az ősrégi poros ***bakelit***, eltűnne a nyúlós szalaggal rendelkező ***magnókazetta***, eltűnnének a ***CD-k*** is, mert teljesen feleslegessé válnának. De ez mind semmi, az egész zeneipar eltűnne, mert nem tudnának miből profitot csinálni. Maximum a koncertekből.
Szóval nagy dilemma ez a ***hanghordozó vs. audiofájl*** kérdés. A legrosszabb benne az, hogy nincs megoldás, nincs döntés. A letöltögetések miatt annyira felmentek a ***CD-k*** árai, hogy manapság csak a jómódúbb emberek tehetik meg azt a luxust, hogy hanghordozót vesznek. A szegényebbek letöltik a zenéket, illetve átírt vagy zenefájlokat tartalmazó albumokat birtokolnak.
Természetesen én nem mondhatom meg, hogy mi lenne a helyes döntés, hogy a megfelelő balansz kialakuljon, de valaminek mindenképp történnie kell.
Ennyi bevezető után térjünk rá a tényleges témára, és legyen az az álláspontunk, hogy a zenefájlok jók. Sokfajta típus létezik, ezekről fog pár szó esni a későbbiekben. Személy szerint én az ***MP3*** kiterjesztésű fájlokat preferálom, mivel ezek a legelterjedtebbek, elég jó a minőségük és viszonylag kevés helyet foglalnak. Persze ez megszokás kérdése is.

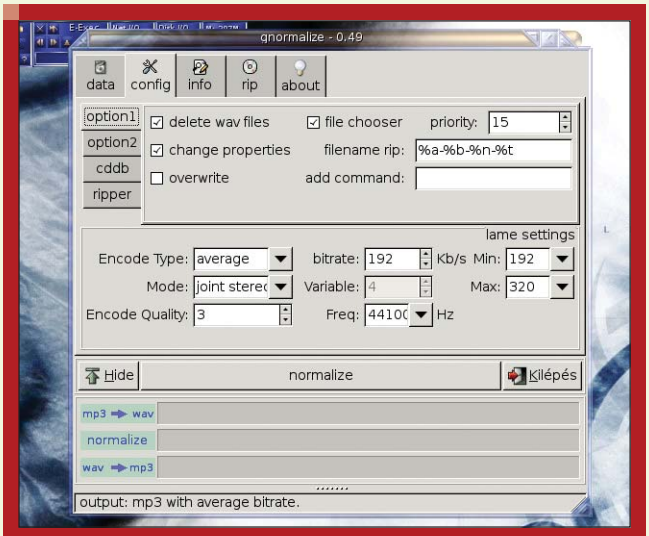
Zenefájlok és formátumok

Ebben a részben megismerkedünk néhány népszerű zeneformátummal (fájlkiterjesztéssel). Természetesen nem fogok mindenbe teljes részletességgel belemenni, de a fontos témákról szó fog esni. Először is, meg kell különböztetni két csoportot: az egyik az úgynevezett ***lossless*** (veszteségmentes tömörítésű), a másik a ***lossy*** (veszteséges) formátum. A ***lossless*** formátumú zenefájlok mindig jóval nagyobbak, mint a veszteségesek. A ***lossy*** fájlok meg nem véletlenül kis méretűek, a minőség a méret rovására megy vagy fordítva.
Vicces, hogy a kedvelt és népszerű formátumok (***MP3, WMA, OGG***) mind veszteségesek. Nemhiába olyan elterjedtek, kis méretük miatt gyorsan letölthetőek/küldhetőek. Bezzeg egy teljes albumot letölteni ***FLAC-ban*** nem két perc.
Nézzük akkor, hogy mely formátumok tartoznak az egyes csoportokba. Veszteségmentes formátumú zenefájlok a következők:

- *Apple Lossless (ALAC – Apple Lossless Audio Codec)*
- *Direct Stream Transfer (DST)*
- *Free Lossless Audio Codec (FLAC)*
- *Mendian Lossless Packing (MLP)*
- *Monkey's Audio (APE)*



1. ábra gnormalize az indítás után



2. ábra A gnormalize beállításai

- *RealAudio Lossless, Shorten (SHN)*
- *True Audio Lossless (TTA)*
- *WavPack Lossless (WV)*
- *Windows Media Losless (WMA Losless)*
- *WAV*

Lássunk néhány veszteséges formátumot is:

- *AAC*
- *ADPCM*
- *ATRAC*
- *Dolby AC3*
- *MP2*
- *MP3*
- *Musepack (MPC)*
- *OGG Vorbis*
- *WMA*

A legelterjedtebb egyértelműen az *MP3 lossy* formátumban, ahogy korábban is említettem, illetve a *WAV losslessben*. Feltörekvő versenyző mostanában a *FLAC* és az *SHN* veszteség nélküli oldalon, viszont a vesztesek oldalán az *MP3* után lohol a *WMA* és az *OGG*. Hogy az *MPC-ről* ne is beszéljek.
Ezt a vicces pár sort csak azért írtam, hogy mindenki lássa: *lossy* és *lossless* harc is folyik. Ez már csak így megy. Mindenki maga döntse el, hogy melyiket preferálja. Én zenészként és zenerajongóként azt állítom, hogy a jó zene nem annyira jó minőségben is tud „ütni”. Ez persze nem azt jelenti, hogy bárkit is megpróbálnék lebeszélni a minőségről. A minőség mindig fontos tényező, de nem szükséges.

Formátumok konvertálása

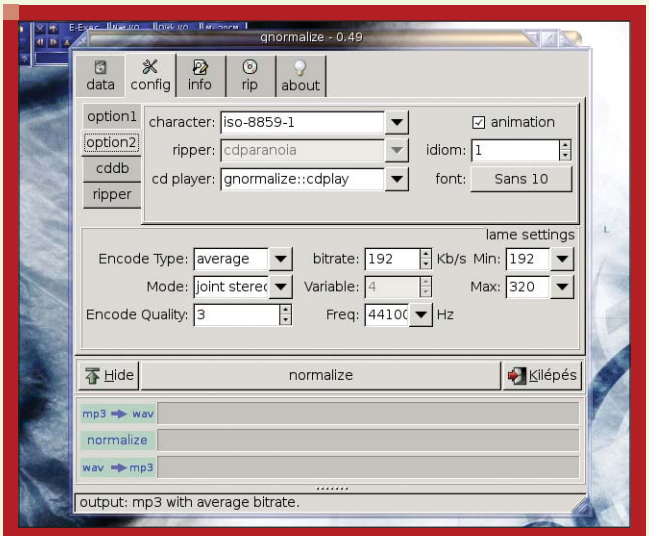
Vajon miért is van szükség konverterre? Egyáltalán mit jelent az, hogy konverter (konvertáló program)? Konvertálni annyit tesz, hogy egyik formátumról a másik formátumra hozni. Maga a fájl gyökeresen megváltozik, viszont a tartalma ugyanaz marad. Tehát ne egy sima átnevezést képzeljünk el. Szükségessége pont az imént említett *lossy-lossless* párharcon alapul. Ha nem lenne harc, nem lenne szükség konverterre.
A személyes tapasztalatom a konverterekkel egészen a régi *windowsos* időkig nyúlik vissza, a mostanra legendássá vált (legalábbis nekem) *Mp3ToWav*, meg *WavToMp3* programokig. Természetesen mivel *Windows* alá kismillió program létezik, ezért létezik kismillió zenekonverter is, bár ezek legtöbbje *shareware*, azaz fizetős. *Linux* alá szintén kismillió konverter létezik (na jó, annyi biztosan nem, mint *Windowsra*), de ezek általában parancssoros *scriptek*, melyek megtanulása komoly idő és erőfeszítés (persze kivételek mindig akadnak). Próbáltam keresni egy olyan programot, ami sok formátumot ismer, könnyű kezelni, ráadásul grafikus. Nem túl hosszas keresgélés után pár éve ráakadtam a *gnormalize-ra*.

gnormalize

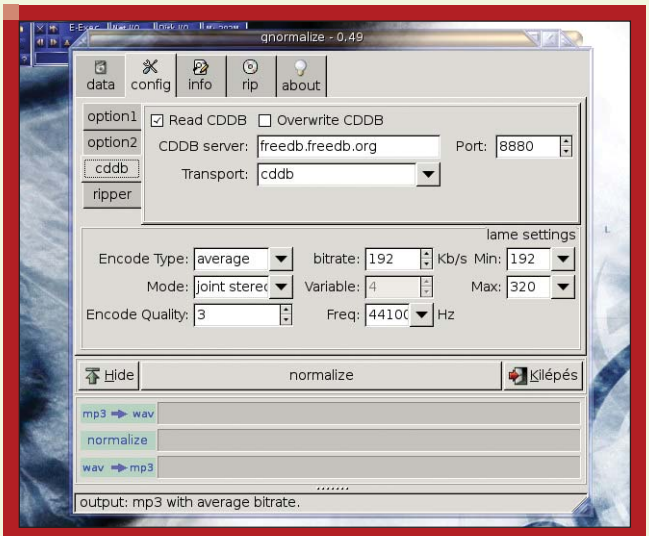
A *gnormalize* (igen, kisbetűvel kell írni, mint szegény *nemecseket*) egy zenefájl-konverter program. Tehát képes az imént elemzett dolgok majdnem mindegyikére. Lényege abban rejlik, hogy minden támogatott formátumot először *WAV*-ba konvertál, majd abból alakítja át zenefájljainkat a kívánt formára. Nagy tudású, könnyen kezelhető, tetszetős kinézetű, ráadásul döbbenetesen stabil. A jelenlegi verziószáma a 0.51, ami a projekt hivatalos honlapjáról letölthető (➲ *http://gnormalize.sf.net*).
A felület a *Gtk2-perl* modult használja, minden grafikus környezetbe illő külsőt varázsolva ezzel a programnak. Nem megszokott dolog, de ez az alkalmazás kizárólag forrásból érhető el. Természetesen léteznek a különböző disztribúciókhoz nemhivatalos csomagok, melyek beszerzését az olvasó leleményességére és a *Google* segítőkészségére bízom. A weboldalon egyetlen *Debian GNU/Linux-hoz* való csomag-lelőhely van feltüntetve. Én mégis javasolnám a forráskódból való telepítést.

Telepítés

Az egész *gnormalize* sztori legkeményebb diója a telepítés, mivel a program rengeteg segédprogramot használ, illetve van pár függősége. No nem kell megijedni, néhány ezek közül opcionális.
Lássuk akkor, hogy mi is a teendőnk! Első lépésként le kell töltenünk az előbb említett forráskódot tartalmazó *tar.bz2* fájlt a *gnormalize* honlapjáról. Ezt egyelőre mentsük el valahova a gépünkre és néhány hosszú pillanat erejéig felejtsük is el. Most következnek a függőségek. Az alkalmazás



■ *3. ábra* Egyéb beállítások



■ *4. ábra* CDDB-beállítások

legfontosabb függőségei a ***perl*** és a ***perl-gtk2*** (ennek a verziója legalább 1.040 kell, hogy legyen). Ezek általában minden disztribúcióhoz megtalálhatóak, legyünk leleményesek! ***Debian GNU/Linux*** alá például az

```
apt-get install perl
```

és az

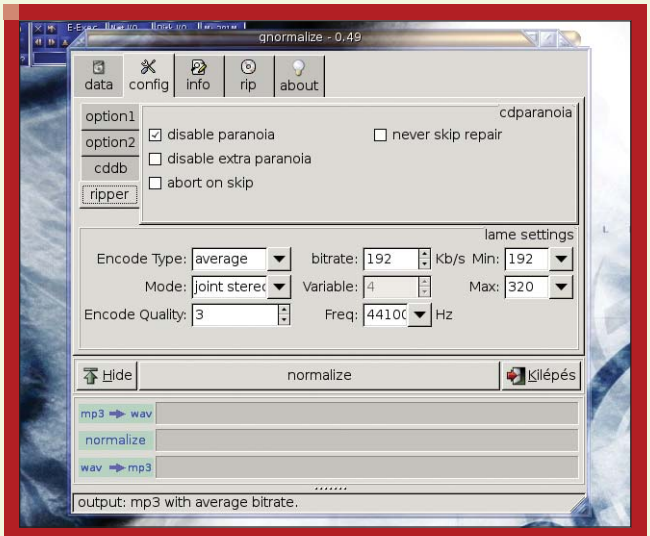
```
apt-get install libgtk2-perl
```

parancsok segítségével telepíthetjük fel ezeket. Érdemes megnézni a második parancsot: általában a ***modulok*** neve (jelen esetben a ***gtk2 perl moduljának*** a neve) ***lib-bel*** kezdődik, tehát így keressünk rá. Jó tanácsként, ha valamilyen ***modult*** (vagy bármilyen programot) nem találunk, használjuk disztribúciónk csomagtelepítőjének keresőjét. ***Debian GNU/Linux*** alatt ezt az

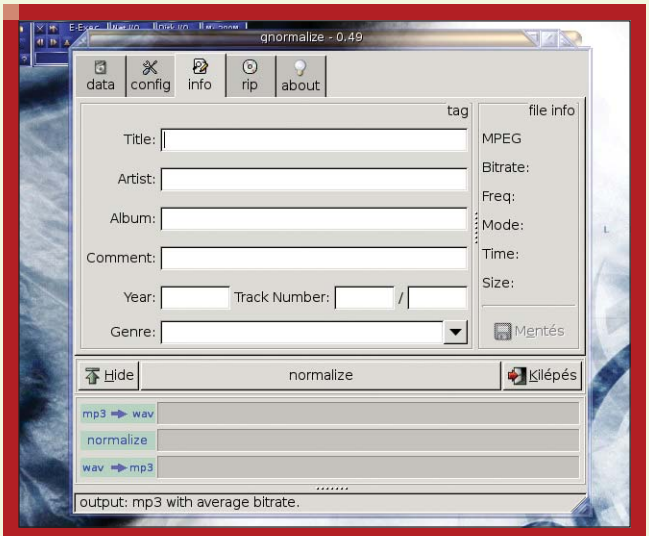
```
apt-cache search <csomagnév>
```

paranccsal tehetjük meg, ahol a `<cso-magnév>` lehet bármilyen ***string*** (szöveg), amit a csomag neve tartalmaz. Természetesen más csomagtelepítők is képesek a keresésre, az ***apt*** csak egy példa.
Ha a két legfontosabb függőség fent van, akkor akár kezdhetnénk is a tényleges telepítést, de én mégis azt javasolnám, hogy előbb a segédprogramok között nézzünk szét. Ezek nélkül ugyanis a ***gnormalize*** szolgáltatásai elég szegényesek.

Amennyiben szeretnénk ***MP3-akat*** kódolni/dekódolni, akkor mindenképp szükségünk van a ***LAME*** nevű ***MP3-kódoló*** programra. Persze sokfajta hasonló alkalmazás létezik, de a ***LAME*** az egyik legjobb, ráadásul a ***gnormalize*** csak ezt támogatja. Elég csak arra gondolni, hogy rengeteg platformon használhatjuk. Többek között ***Windows, DOS, GNU/Linux, MacOS X, *BSD, Solaris, HP-UX, Tru64 Unix, AIX, Irix, SCO Unix, UnixWare, Ultrix, OpenVMS, MacOS Classic, BeOS, QNX, RiscOS, AmigaOS, OS/2, FreeMiNT(Atari)*** rendszereken. Letölteni a ➲ ***http://lame.sourceforge.net/download.php*** oldalról tudjuk.
Mivel ***Audio-CD-ket*** is tud ***rippelni*** (a ***CD-n*** található dalokat zenefájllá alakítani) az alkalmazás, ezért szükségünk lehet a ***cdparanoia*** (vagy ***cdda2wav***) és a ***vorbis-tools*** nevű programokra. Ezek általában csomagként is megtalálhatóak az egyes disztribúciókhoz. Ha mégsem, akkor a ***vorbis-tools*** a ➲ ***http://vorbis.com*** illetve a ***cdparanoia*** a ➲ ***http://www.xiph.org/paranoia/*** oldalakon is elérhetőek. A ***vorbis-tools*** egyébként az ***OGG*** formátum támogatásáért is felelős.
A ***normalize*** nevű program (innen kapta a nevét a ***gnormalize***) arra szolgál, hogy az egyes formátumokból kinyert ***WAV*** fájlokat hangzásbeli egyensúlyba hozza. Erre akkor lehet szükségünk, ha mondjuk egy album különböző hangerősségű fájlokat tartalmaz. A ***normalize*** azonban nem feltétlenül szükséges a ***gnormalize*** használatához. Ha mégis szeretnénk ilyen opciót, akkor a ➲ ***http://normalize.nongnu.org/*** oldalról tölthetjük le.
A ➲ ***http://faac.sourceforge.net/oldsite/*** weblapról letölthetjük a ***faac*** és ***faad*** nevű segédprogramokat a ***gnormalize-hoz***. Előbbi ***WAV*** fájlból készít ***MP4*** formátumot, utóbbi ***MP4-ből WAV-ot***.
Ha szeretnénk ***Musepack***, azaz ***MPC*** támogatást (akasztják a hóhért, de be kell vallanom, hogy én néhány beszerezhetetlen ***Eric Johnson*** kalózkoncertet kizárólag ***MPC-ben*** találtam meg), akkor a ➲ ***http://www.musepack.net/*** site-ról ezt letölthetjük. Innen nekünk az ***mppdec*** és az ***mppenc*** programok kellenek, mivel az egyik kódoló, a másik dekódoló.
A leginkább feltörekvő ***lossless*** formátum támogatásához látogassunk el a ➲ ***http://flac.sourceforge.net/*** honlapra és tegyük magunkévá az alkalmazást.
A szintén ***lossless*** (csak hogy szokjuk a kategóriákat) ***APE*** formátum támogatásához a ***mac*** (***Monkey's Audio Console***) nevű programra van szükségünk. Ez megtalálható a ➲ ***http://sourceforge.net/projects/mac-port/*** oldalon. Néhány nemhivatalos ***RPM*** is készült a macből, ezeket itt érhetjük el: ➲ ***http://plf.zarb.org/*** vagy ➲ ***http://rpm.pbone.net/***.
Audio-CD-k lejátszására is van lehetőségünk a ***gnormalize-zal***, ehhez a ***cdcd*** (vagy másnéven ***libcdaudio***)



■ *5. ábra* CD-ripper és CD-lejátszó beállítások



■ *6. ábra* Információk a konvertálandó fájlról

programra van szükségünk, mely a ➲ *http://libcdaudio.sourceforge.net/* oldalon érhető el.
Természetesen ezen programok csak akkor szükségesek, amennyiben az adott funkciójukat szeretnénk kihasználni. Opcionális dolgokról lévén szó, nem kötelező mindet telepíteni.
A segédprogramok telepítése általában mindenhol ugyanúgy történik. Ellátogatunk a weboldalra, letöltjük a programot, majd kicsomagoljuk. Ezután a régi jól bevált linuxos módszerrel kell telepítenünk, a

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

parancsokkal. Néha az is előfordulhat, hogy a `make install`-ra vagy a `./configure`-ra nem lesz szükségünk. Javaslom, hogy minden segédprogram telepítése előtt alaposan olvassuk el a forráshoz csatolt `INSTALL` és `README` nevű információs fájlokat. Így kizárhatjuk azt a lehetőséget, hogy mi rontottunk el valamit.
Amennyiben a függőségek és a segédprogramok a helyükön vannak, elkezdhetjük a tényleges telepítést. Tömörítsük ki az előbb letöltött *gnormalize-verzió.tar.bz2* fájlt a

```
tar xvjf gnormalize-
➥verzó.tar.bz2
```

paranccsal. A verzió természetesen bármilyen lehet (nálam jelenleg *0.49*).
Létre fog jönni egy *gnormalize-verzió* nevű könyvtár, melybe belépve érdekes dolgokat láthatunk.
Például, néhány segédprogramot már eleve tartalmaz a könyvtár (*Audio-CD* támogatás, *CDDB* támogatás, *MPC* támogatás, *MP3-Info*).
Érdemes itt is, mint mindenhol, elolvasni a csatolt `README` fájlt, melyből kiderül, hogy a program telepítése a pofonegyszerű

```
./install
```

paranccsal történik. Ez a szkript előkonfigurálja, lefordítja, majd telepíti a *gnormalize-t* számítógépünkre.
Ezen felül a telepítő rákérdez minden – *gnormalize* által szállított – segédprogram telepítésére is. Ha minden rendben zajlott, tehát a telepítő nem jelzett hibát, akkor a *gnormalize* máris bevethető.

Használat

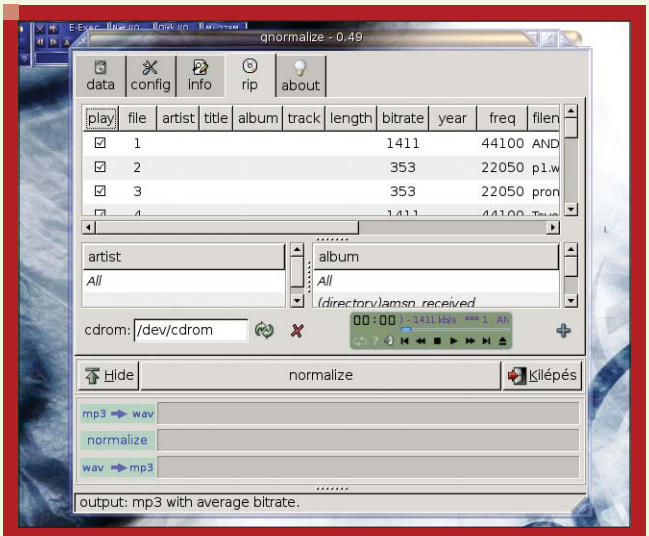
Mivel ezt a programot nem csomagból telepítettük, ezért, hogy megkönnyítsük életünket, létre kell hozni neki menübejegyzést, vagy készíteni kell neki ikont. Természetesen ezek csak kényelmi szempontok, mert bármilyen terminálból indítható az alkalmazás a

```
gnormalize
```

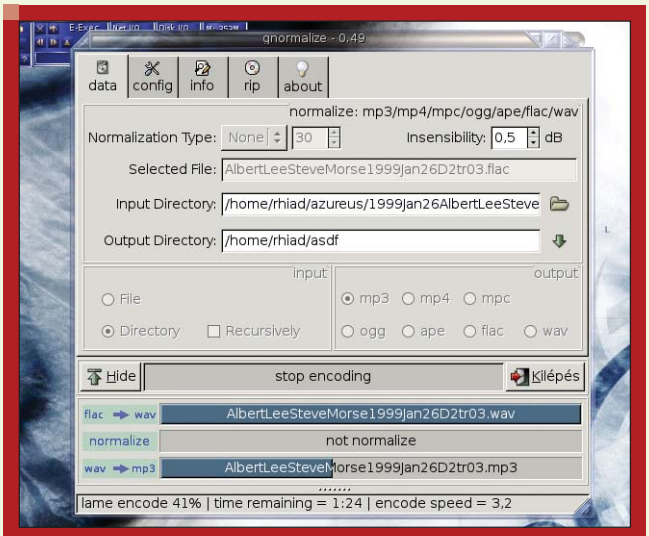
paranccsal. Ha ezt megtettük, egy – a *GTK-nak* köszönhetően – homogén felülettel találjuk szemben magunkat.

Kezelés

Maga a felület 5 fület tartalmaz, melyek közül az első a *„data”*.
Ez fogad bennünket az indítás után, ahogy ez az iménti ábrán is látható. Vegyük végig, hogy mik is találhatóak rajta! Tulajdonképpen az első fül számunkra a legfontosabb, ugyanis minden itt dől el.
A *„Normalization Type”* a *normalize* programra utal, beállíthatjuk a hangerősség mértékét, ráadásul az *„Insensibility”* funkcióval az egyes dalok decibelküszöbét is megadhatjuk. Jómagam ezt a funkciót nem használom, ezért inaktív.
A *„Selected File”* az éppen aktuális, azaz konvertálandó fájl nevét fogja tartalmazni a későbbiekben, *„Input Directory”* és *„Output Directory”* pedig a konvertálandó, valamint a majd már átkonvertált fájlok könyvtárait jelöli. *Output*, azaz kimeneti könyvtárnak érdemes létrehoznunk egy újat (ez jelenleg nálam ~/asdf névre hallgat).
A következő két terület – *„Input”* és *„Output”* – különösen fontos lesz nekünk. Az *Input* területen kiválaszthatjuk, hogy különálló fájlt vagy egy egész könyvtárnyi fájlt akarunk-e konvertálni. A *„Recursively”* funkcióval igény szerint több egymásba ágyazott könyvtárban elhelyezkedő zenefájlokat is konvertálhatunk rekurzívan.
Az *Output* területen azt adhatjuk meg, hogy formátumban szeretnénk viszontlátni konvertálandó fájljainkat (nálam jelenleg *MP3*). Fontos dolog,



■ *7. ábra* gnormalize, mint CD-ripper és CD-lejátszó egyben



■ *8. ábra* gnormalize „bevetésen”

hogy csak azon rádiógombok aktívak itt, melyekhez már feltelepítettük a segédprogramot. Én nem használok sem ***OGG***, sem ***APE***, sem ***MP4*** támogatást, ezért ezeket nem is választhatom a listából.

A ***„Normalize”*** gomb az egész konvertálás elindítására szolgál (ezt az elindulás után a ***„stop encoding”*** felirat fogja felváltani, mellyel megállíthatjuk a folyamatot), az alsó három sáv pedig a procedúra állapotát jelzi. Ha nem használjuk a normalizálási funkciót, akkor a ***„normalize”*** csíkot átugorja a program. A ***„Kilépés”*** gomb pedig magától értetődő.

A következőkben a ***„config”*** füllel fogunk foglalkozni. Megfigyelhetjük, hogy itt a „fül a fülben” esete forog fenn, tehát ezen a fülön belül található még négy.

Az elsőn (***option1***) rögtön fontos beállításokat eszközölhetünk. A ***„delete wav files”*** opció nagy segítség lehet azoknak, akiknek kevés helyük van a merevlemezen. A ***WAV*** fájlok jó nagyok, sok helyet foglalnak, ráadásul ha nem épp ez a kívánt kimeneti formátumunk, akkor feleslegesek. Ezzel a funkcióval a program automatikusan törli a ***WAV-okat***.

A ***„file chooser”*** (fájl-választó) funkciója a nevéből adódik, prioritását is beállíthatjuk.

A zenék kódolásának beállításai garmadával állnak előttünk, jelen esetben a ***LAME*** opcióit látjuk. Nem fűznék kommentárt az egyes értékek beállításaihoz, mindenki tapasztalja ki a saját szájízének megfelelő kombinációt.

Hozzátenném, hogy mindig azon dekóder beállítási értékeit változtathatjuk, amit éppen kijelöltünk outputnak. Tehát, ha mi ***FLAC*** fájllá szeretnénk konvertálni egy másfajta formátumú zenét, akkor természetesen ezen a helyen a ***FLAC*** beállításai szerepelnének.

Ezek a beállítások egyébként az összes alfülön láthatóak, ahogy a konvertálási folyamat is.

A második, ***option2*** nevű fülön inkább kinézeti beállításokat végezhetünk: karakterek és fontok. Továbbá megadhatjuk a program által használt ***CD-ripper***, illetve ***CD-lejátszó*** segédprogramokat is.

A harmadik (***cddb***) és negyedik (***ripper***) fülön egyéb extra lehetőségek közül választhatunk. A ***gnormalize*** képes a ***CDDB*** (***Compact Disc Database***) kezelésére, amely arra hivatott, hogy ***Audio-CD-k*** adatbázisát (számcímek, előadók, stb.) lekérje az interneten keresztül. Bővebb információt a ***CDDB-ről*** a ➲ ***http://en.wikipedia.org/wiki/CDDB*** oldalon találhatunk.

Kedvenc konverterünk nem csak konvertálni tud, hanem képes ***rippelni*** és lejátszani is ***CD-inket***. Ennek a beállításait is finomhangolhatjuk.

A harmadik nagyfülre lépve (***„info”***), információkat találhatunk a zenefájlról, melyet a későbbiekben konvertálni szeretnénk. Többek között láthatjuk a fájl tömörítési értékét, ***bitrátáját***, frekvenciáját és a méretét is.

Fontos és nem elhanyagolható tény, hogy a ***gnormalize*** egy remekbe szabott ***ID3-Tagger*** alkalmazás is. Hogy ez mit takar? Készíthetünk minden egyes fájlhoz ***tag-et***, azaz címkét, mely tartalmazza az adott zene előadóját, albumcímet, zenei stílust, stb.

A következő fül (***„rip”***) maga a ***CD-ripper*** funkció. Én ugyan a ***CD-im rippelésére*** külön erre szakosodott alkalmazást használok (mégpedig a ***Grip*** névre hallgató csodát), de első ránézésre is elég bíztatónak tűnik a dolog. Megjegyzésként hozzátenném, hogy nem csak ***Audio-CD-ket*** játszhatunk le, hanem egy adott könyvtárban lévő zenefájlokat is.

Az 5. és egyben utolsó fül (***„about”***) pedig a ***gnormalize*** névjegye, tanulmányozzuk büszkén, ha már idáig eljutottunk!

Végezetül lássuk a ***gnormalize-t*** „munka” közben.

Összességében azt hiszem egyértelmű, hogy a ***gnormalize*** jóval több, mint egy egyszerű konverter.

Kívánok tehát mindenkinek kellemes és legális konvertálást!

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

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

Térugrás – látványos asztalfelületek

A Compiz egy újfajta ablakkezelő. Elődeitől abban különbözik hogy a modern grafikus kártyák hardveres OpenGL támogatását kihasználva különlegesen látványossá képes tenni a desktopot. Az ablakok áthelyezéskor hullámoznak, az épp nem használtak átlátszóvá válnak, de akár el is kezdhet esni az eső az asztalra.

Bár kevésbé futurisztikus mint a *Linuxvilág* magazin novemberi számában bemutatott ***Looking Glass***, mégis számos előnnyel rendelkeznek az *LG*-hez képest.
Talán a legfontosabb hogy teljes mértékben képes együttműködni a jelenlegi szoftverekkel. Ha kikapcsolnánk a különleges effekteket akkor valószínűleg észre sem vennénk hogy nem a *GNOME* hagyományos ablakkezelője a ***Metacity*** fut.
Ennek köszönhetően a *Compiz* már néhány disztribúciónak része. Ezek közé tartozik a *Novell* által fejlesztett ***SUSE Linux 10*** is.

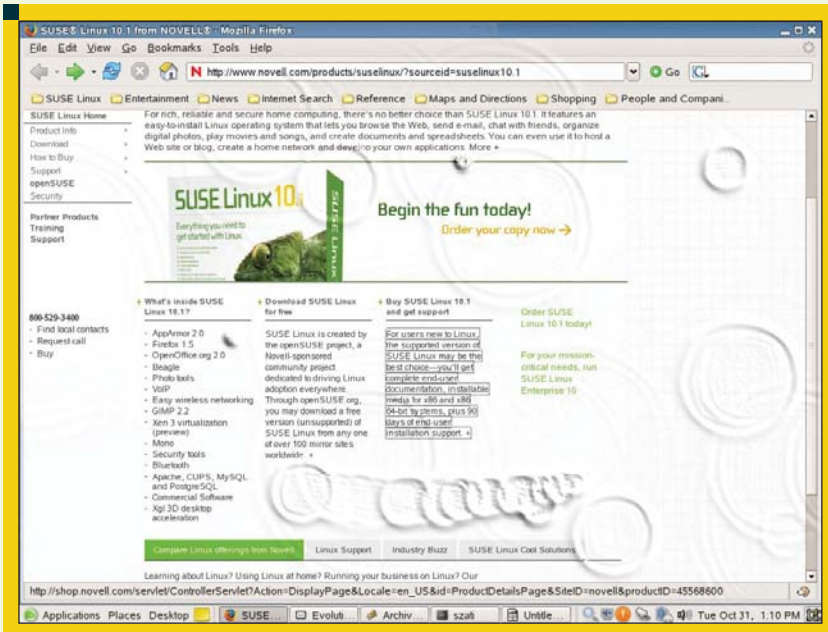
A SUSE Linux 10 és a Compiz

A ***SUSE Linux*** telepítő lemezein megtalálható a *Compiz* és más szükséges szoftverek is, ezért viszonylag gyorsan telepíthető és kipróbálható, így ez az egyik rendszer amit a *Compiz* bemutatáshoz választottam.
A ***SUSE Linux 10.1*** telepítőlemezei a ➲ ***http://en.opensuse.org/Image:Download.png*** címről tölthetőek le. Mindegy hogy a DVD-t vagy az 5 darabos CD készletet töltjük le, a tartalom ekvivalens.
A ***SUSE Linux*** telepítése a grafikus telepítőnek köszönhetően rendkívül egyszerű ezért nem részletezem. Azonban célszerűnek tartom a *GNOME* asztali környezet választását mivel én is ezt telepítettem és ezért a továbbiakban azt feltételezem hogy az olvasó is ezt választotta. A telepítés végeztével ne felejtsük el frissíteni a rendszert az internetről.

A *Compiz* nem települ alapértelmezetten ezért ezt utólag kell föltenni.
A *SUSE*-ben a csomagokat a ***Yast*** programmal telepíthetjük, tehát adjuk ki a `yast2` parancsot egy terminálba.
A megjelenő ablakban válasszuk a Software Management pontot.
Ezután egy újabb ablak jelenik meg amiben kiválaszthatjuk a telepíteni kívánt csomagokat, jelen esetben a ***compiz, xgl, xgl-hardware-list, gnome-session*** és a ***libwnck*** csomagokat.
Ezt úgy tehetjük meg hogy a kereső segítségével (ha nem látszik akkor a ***Filter*** legördülőmenüben válasszuk a ***Search*** pontot) rákeresünk a nevükre, majd a megjelenő listából a csomag nevére jobb gombbal kattintunk, és az ***Install*** pontot választjuk. Miután ezen lépéssor ismétlésével kijelöltük az összes telepítendő csomagot, nyomjuk meg az ablak jobb alsó sarkában lévő *Accept* gombot. Ennek hatására a kijelölt csomagok települni fognak.
Ha készen vagyunk a telepítéssel akkor jelentkezzünk be normál felhasználóként a *GNOME*-ba.
A *Compizt* a *compiz* csomag részeként települt ***gnome-xgl-settings*** segédprogrammal tudjuk elindítani. Mindegy hogy rendszergazdaként vagy normál felhasználóként tesszük ezt, amint szükség van a root jogosultságokra felhasználót vált a program.
A segédprogram elindításához adjuk ki a `gnome-xgl-settings` parancsot (*1. ábra*).
A program felülete rendkívül egyszerű, néhány információt jelenít meg, és egy gombot tartalmaz. Mint ahogy a mellékelt ábrán is látható többek

1. ábra A gnome-xgl-settings felülete

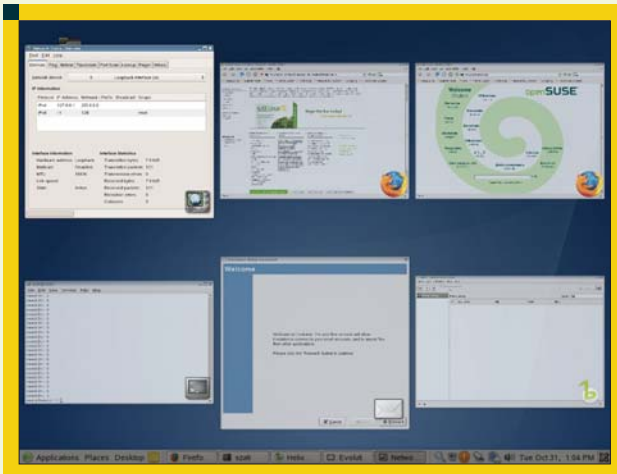
közt azt tudhatjuk meg hogy a számítógépben lévő grafikus kártya szerepel-e az *Xgl* hardver támogatási adatbázisában vagy sem. Ha nem az sem jelent gondot, az én laptopom ***Intel 915GM*** típusú grafikus kártyája nem szerepelt, mégis hibátlanul és akadozás mentesen működött a *Compiz*.
Az egyéb megjelenő információk is a grafikus rendszerre vonatkoznak.
Ha minden feltétel megfelel a *Compiz* indításához, akkor nyomjuk meg az ***Enable Desktop Effects*** gombot.
Ha a gomb felirata az hogy ***Start Sax2***, az azt jelenti, hogy nincsen rendben valami az X beállításaival kapcsolatban. A *Sax2* a *SUSE* X konfiguráló programja, ennek segítségével el tudjuk végezni a beállításokban szükséges módosításokat.
Az hogy esetlegesen melyik beállítás nem megfelelő azt a fehér mezőből tudhatjuk meg.



2. ábra Eső esik az asztalra

Miután megnyomtuk az ***Enable Desktop Effects*** gombot, nem lesz más dolgunk mint kijelentkezni és újra bejelentkezni. Ekkor már a ***Compiz*** indul el.
Most vegyük sorra hogy milyen különleges hatásokat képes a ***Compiz*** az asztalunkra varázsolni.
Nekem a kedvencem az eső effekt. Olyan mintha eső esne egy tó vizére – ami jelen eseteben az asztal – és azon hullámokat keltene. Ha az egeret a ***Win + Shift*** lenyomattartása mellet mozgatjuk az is hullámokat kelt. Az eső effektet a ***Shift + F9*** billentyűkombinációval kapcsolhatjuk ki illetve be (*2. ábra*).
A leghasznosabb effektnek az ablakválasztót találtam. Két lehetőség is van, az egyik ***Windows Vista***, a másik ***MacOS X*** szerű. Előbbi az Alt + Tab billentyűkombináció hatására lép működésbe (*3. ábra*). A képernyő közepén egy fehér átlátszó csíkon sorba megjelennek a futó programok ablakainak kicsinyített példányai.
Az Alt változatlan nyomva tartása mellet a Tab billentyű nyomogatásával lehet köztük váltani. Miután fölengedtük az Alt gombot a kiválasztott ablak aktívvá válik. Ha azt ***Alt + Shift + Tab***-al indítjuk ezt a funkciót akkor az összes virtuális desktopon lévő ablak közül választhatunk.
A ***MacOS X*** szerű ablakválasztót a ***Pause*** gomb megnyomásával aktiválhatjuk (*4. ábra*). Ekkor az ablakok összezsugorodnak és egymás mellé helyezkednek az asztalon. Ezután az az ablak válik aktívvá amelyikre kattintunk.
A virtuális desktopok közti váltás is igen látványos. Mindegyik desktop egy kocka egy-egy oldalán kapott helyet, váltani köztük pedig a kocka forgatásával lehet. Ezt az ***Alt + Ctrl + Bal egér gomb*** nyomva tartásával és az egér mozgatásával tehetjük meg. Ugyan ez érhető el az ***Alt + Ctrl*** és bal vagy jobb nyíl lenyomásával is (***5. ábra***).
A desktopok közt váltani nem csak a kocka forgatásával lehet. Az ***Alt + Ctrl + Lefele nyíl*** lenyomása után filmszalag szerűen egymás mellé rendeződnek a virtuális desktopok. Itt az oldalra nyilakkal lehet váltani köztük (*6. ábra*).
Ha egy ablakot át szeretnénk vinni az egyik desktopról a másikra akkor egyszerűen kezdjük el húzni az ablakot, a képernyő széléhez érve automatikusan átfordul a kocka.
Az aktív ablak átlátszóságát az Alt + Egér görgővel állíthatjuk 0-tól majdnem 100 százalékig.
A ***Zoom*** effekt, mint ahogy a neve is sugallja, arra való hogy a desktop egyes részeire ráközelíthessünk. Ezt a funkciót a ***Win + jobb egérgombbal*** lehet bekapcsolni, vagy a ***Win + Egérgörgővel***.
Ha túl gyorsnak bizonyul valamelyik effekt ahhoz hogy „gyönyörködjünk benne” akkor le is lassíthatjuk őket a ***Shift*** és az ***F10*** együttes lenyomásával. Normál sebességre az előbb említett billentyűk ismételt lenyomásával lehet visszaváltani.

3. ábra Vista és …



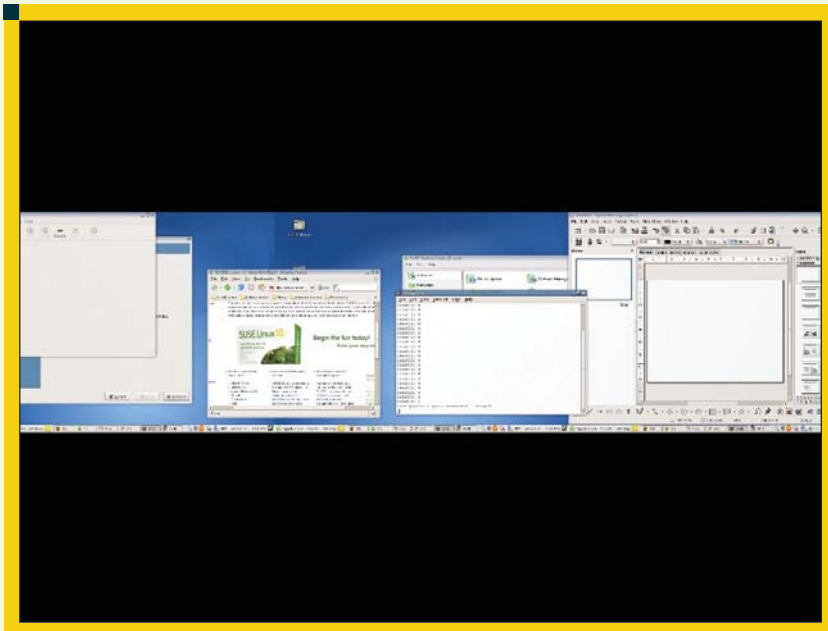
4. ábra … MacOS X szerű ablakváltás

Számos effektet nem kell külön billentyűkombinációval aktiválni. Ilyen például a rongyszerűen mozgó ablakok, vagy az elhalványuló menük.
Az effektek egy része konfigurálható a *gnome-xgl-settings* segítségével, erre azonban most nem térek ki.
Szövegben és képekben képtelenség visszaadni a *Compiz* látványosságát, de remélem sikerült egy kis ízelítőt adnom belőle.
További információ a *SUSE Linuxról* és a *Compizról* a ➲ *http://en.opensuse.org/Compiz* oldalon található.
Most rátérünk a hogyan-okra és a miértekre, majd végezetül egy másik ablakkezelőt a *Berylt* is bemutatok.

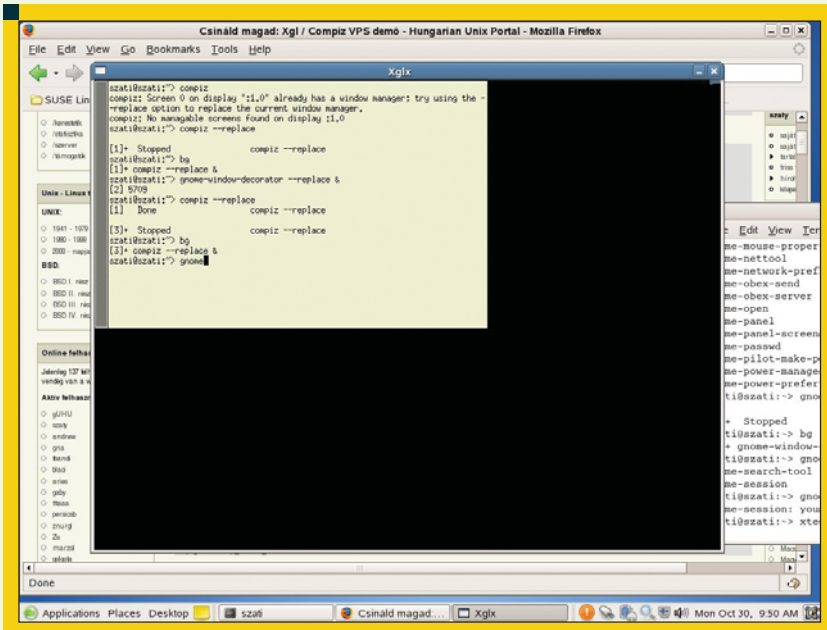
Hogyan működik a Compiz?

Tapasztaltabb *Linux* felhasználókban felmerülhet a logikus kérdés, miért van szüksége a *gnome-xgl-settingsnek* root jogosultságokra ahhoz hogy a *Metacityt* lecserélje a *Compizra*, annak ellenére hogy minden felhasználó maga választhatja meg ablakkezelőjét? Nos a válasz viszonylag egyszerű a *gnome-xgl-settings* nemcsak a `compiz --replace` parancsot adja ki hanem egy speciális X szervert az *Xgl*-t is elindítja és alapértelmezettnek állítja be, ehhez kellenek a rendszergazdai jogok.
Az *Xgl*-t a *Novell* fejlesztette ki, a különlegessége az hogy a modern grafikus kártyák hardveres *OpenGL* támogatását kihasználva jeleníti a grafikus elemeket. Azonban az *Xgl* – jelen állapotában – önállóan nem képes működni, a normál *Xorg* tetején fut. Ezt nagyjából úgy kell elképzelni mint az *Xnest*-et, a futó *Xorg* szerver egy ablakában elindul egy újabb X szerver az *Xgl*, ami az *Xorg* szerver *OpenGL*-ért felelős kiterjesztését a *GLX*-et használva hardveres gyorsítást biztosít a megjelenítéshez abban az ablakban amiben fut.
Az hogy az *Xgl*-t nem ablakban látjuk az annak köszönhető hogy az indításkor meg lett neki adva a *–fullscreen* kapcsoló, így teljes képernyős módban indul mint például a játékok.
Ha terminálból szeretnénk elindítani az *Xgl*-t akkor nem a `startx` hanem a `Xgl` parancsot kell kiadni. (Ha *ATI* kártyával rendelkezünk akkor valószínűleg a `Xgl :1 -fullscreen -ac -accel xv -accel glx:pbuffer &` parancs lesz a megfelelő, *Nvidia* esetén pedig az `Xgl :1 -fullscreen -ac -accel xv -accel glx:pbuffer &`.) Ekkor az *Xgl* automatikusan elindít egy *Xorg*-ot is, ezzel nem nekünk kell vesződni.
Jelenleg fejlesztés alatt áll egy *Xegl* nevű backend az *Xgl*-hez, aminek segítségével az *Xgl* képes lesz közvetlen a *framebufferen* futni, tehát nem kelleni majd hozzá a hagyományos *Xorg*.
A különleges desktop effekteket a *Compiz* hozza létre, kihasználva az *Xgl* hardveres *OpenGL* gyorsítását.
A *Copmiz* az ablakkezelők egy különleges csoportjának a *compositing window mangereknek* egyik első linuxos példánya.
A *compositing* az a folyamat amikor a legkülönbözőbb képekből egy újat hoznak létre. Talán legismertebb példa erre az *TV*-s időjárás jelentés amikor a tárkép nincsen a meteorológus mögött a valóságban, az számítógépes utómunka eredményeképpen kerül oda. Ezt a folyamatot megvalósító számítógépes program a *compositing manager*. A *Compiz* tulajdonképpen egy ilyen *compositing manager* és egy

5. ábra A virtuális desktopok közt a kocka forgatásával …



6. ábra … vagy a filmszalag tekerésével válthatunk



7. ábra Az Xgl a -fullscreen kapcsoló nélkül ablakban fut

ablakkezelő „összeépítve". A compositing manager felel a különleges effektekért, az ablakkezelő pedig az ablakkezelési feladatokért.
A ***Compiz*** nemcsak az előzőekben felsorolt effektek keltésére képes, hanem bővítőmodulok (plug-in) segítségével további képességekkel is felvértezhető.

Az AIGLX

Az ***Xgl*** számos hiányossággal is rendelkezik. A jelenlegi felépítésének az az előnye hogy az ***Xorg***-ra támaszkodik ezért nem szükséges törődni a grafikus hardverek különbözőségével, azokat az ***Xorg*** elrejti az ***Xgl*** elől, ezért a jelenlegi driverekkel tökéletesen működik az ***Xgl***. Ennek a struktúrának viszont ez a hátránya is egyben, az ***Xgl*** a grafikuskártyák legegyszerűbb plusz szolgáltatásait sem tudja kihasználni, például az ***Xgl*** hivatalosan nem képes több monitort kezelni. (Gyakorlatilag megoldható a többmonitoros üzemmód is de nehezebben mint a hagyományos ***Xorg*** vagy az ***AIGLX*** esetében.)
Ez a probléma megoldható ha a ***GLX*** backendet lecseréljük ***Xegl***-re. Itt viszont rögtön jelentkezik a következő nehézség, az ***Xegl*** az ***EGL API***-n alapul. Jelenleg azonban csupán egy grafikus kártya az ***ATI Radeon R200*** drivere támogatja ezt az ***API***-t.
Ezeknek a problémáknak a láttán a ***Fedora Core*** fejlesztőcsapata egy másik irányba indult el, módosították az ***Xorgot*** és a ***Mesa*** csomagot. Ennek a munkának az eredménye képessé tette az ***Xorg 7.1***-et ***OpenGL*** effectek produkálására a hagyományos desktopon. Azonban nem tértek le olyan mértékben az eredeti ***Xorg***-tól hogy ez meggátolja a jelenlegi driverek hibátlan működését.
Ezt a módosítást ***AIGLX***-nek hívják. Az ***AIGLX*** a legtöbb hardveren gond nélkül működik, az ***ATI*** kártyák közül a ***Radeon 7000***-től kezdve szinte mindenen fut. Hasonló a helyzet az ***Nvidia*** kártyák estében is, elvileg mindegyikkel működik, egyetlen fontos feltétel hogy 1.0-9625 (jelenleg még béta állapotú) verziójú vagy újabb ***nVidia*** meghajtóval kell rendelkezni.
A részletesebb hardvertámogatási információk a ➲ ***http://fedoraproject.org/wiki/RenderingProject/aiglx*** oldalon találhatóak.
A ***Compiz*** természetesen hibátlanul együttműködik az ***AIGLX***-el is, a végeredménybe semmi különbség nem látszik az ***AIGLX*** és az ***XGL*** közt.
A ***Debian GNU/Linux unstable*** ága tartalmazza az Xorg 7.1-et, ezért az ***AIGLX*** gyakorlati alkalmazását ezen a disztribúción mutatom be.

A Debian és a desktop effectek

A ***Debian*** telepítése megérne egy külön cikket ezért, itt nem részletezem, a továbbiakban feltételezem hogy az olvasó rendelkezik egy ***Debian unstable*** alaprendszerrel. (Legkönnyebben úgy kaphatunk egy ilyen rendszert hogy először egy ***testinget*** telepítünk a ***weekly build*** első CD-jéről majd ezt frissítjük ***unstablere***.)
Tehát miután telepítettünk egy ***Debian unstable*** alaprendszert a következő dolgunk az lesz hogy telepítsük a ***xserver-xorg***, a ***gnome-core***, az ***xfonts-base***, a ***gdm***, és a ***compiz*** csomagokat. Miután ezek települtek, és megfelelően beállítottuk az X szervert, ha szükséges föltelepítettük a számítógépünkben lévő grafikus kártya driverét, bekapcsolhatjuk az ***AIGLX***-et. Ezt úgy tehetjük meg hogy a következős sorokat hozzáadjuk az ***xorg.conf*** végéhez:

```
Section "Extensions"
    Option          "Composite"
➥"Enable"
EndSection
```

Ha ***Nvidia*** gyártmányú grafikus kártya van számítógépünkben akkor még a következő három sort is hozzá kell adni az ***xorg.conf "Device"*** szekciójához:

```
Option          "RenderAccel"
➥"true"
Option          "AllowGLXWith
➥Composite" "true"
Option          "AddARGBG
➥LXVisuals" "true"
```

Ezek után nincs is más dolgunk mit újraindítani az X szervert majd egy ***gnome-terminal*** ablakban kiadni a `compiz –replace &` parancsot.
Ha azt szeretnénk hogy mindig automatikusan elinduljon a Compiz, akkor adjuk ki a `gnome-session-save` parancsot. Ez elmenti az épp futó programok listáját és ezentúl mindig elindítja őket bejelentkezés után.
A ***Debian compiz*** csomagja sajnos nem tartalmaz grafikus konfiguráló eszközt mint amilyen a ***gnome-xgl-setting*** volt a ***SUSE*** estében, ezért a ***gconf*** kulcsok állításával lehet a viselkedését befolyásolni. (Ehhez egy grafikus eszköz a ***gconf-editor***, ami az azonos nevű csomaggal telepíthető.) Az alapbeállítások azonosak a ***SUSE***-és ***Compiz*** alapbeállításaival ezért itt még egyszer nem tárgyalom a különböző effekteket.
Többmonitoros üzemmódban különösen látványos az ***AIGLX – Berly*** páros.

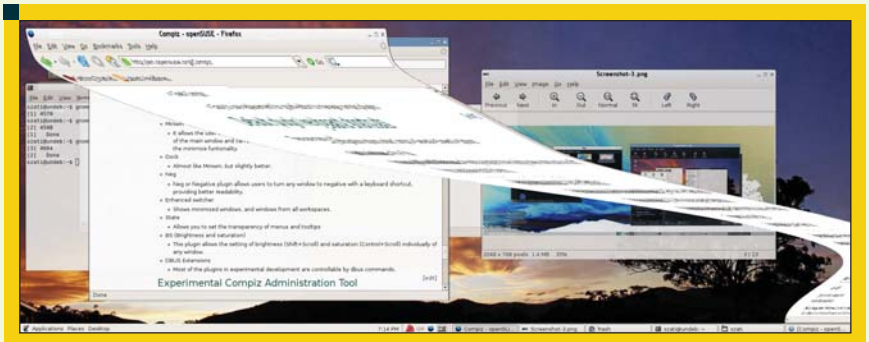
Én az *Nvidia* grafikus kártyám *TwinView* opcióját használtam, de elvileg Xineramaval is megoldható a többmonitoros üzemmód. A hatás fokozza ha egy jó panorámaképet állítunk be háttérképnek (*8. ábra*).

Beryl

A *Beryl* egy másik compositing window manager. A fejlesztés eredetileg a *Compiz* forkjából indult de mára már számos különbség van a két ablakkezelő közt. Ezeknek a fejlesztéseknek az eredményeképpen a *Beryl* jobban együttműködik más desktop környezettekkel is nemcsak a *GNOME*-val. Mivel a *Compiz* a beállításait a *gconf*-ból a *GNOME* konfigurációs rendszeréből tölti be ezért azt mindenképpen muszáj volt telepíteni még akkor is ha egyáltalán nem szándékoztunk *GNOME*-t használni. Emellett a *Beryl* saját univerzális *window decoratoral* (az a program ami a kereteket rajzolja az ablakok köré) rendelkezik, amit *emerald*-nak hívnak.
Számos fejlődés tapasztalható az látványosság terén is, például nem csak elhalványulnak a menük hanem közben hullámoznak is, vagy a minimalizálás közben az ablak „elfolyik" a tálcára (*8. ábra*).
A *Beryl* jelenleg nem része a *Debiannak*, ezért a ➲ *http://vally8.free.fr/Beryl/* címről kell letölteni az összes itt lévő csomagot. Miután ez megvan a *dpkg* segítségével kell telepíteni őket:

```
dpkg -i <csomag_fájlok_nevei>
```

A *Beryllel* érkezik egy *beryl-manager* nevű program is. Ez tulajdonképpen egy *applet* ami állandóan látható a figyelmeztető területen (notification area). Az ikonjára kattintva (egy piros gyémántot ábrázol) jobb egér gombbal, egy menü jelenik meg, ebből minden beállítás elvégezhető, nem kell a *gconfal* bajlódni (*9. ábra*).
Ahhoz hogy elindítsuk a *Berylt*, a *Select Window Manager* menüpontban válasszuk ki. Mint a *Compiz* esetében itt is a `gnome-session-save` paranccsal utasíthatjuk a *GNOME*-t hogy a *Beryl*-t indítsa el automatikusan a bejelentkezés után.
Visszatérve a *beryl-manager* menüjéhez, az első menüponttal a *Beryl* grafikus konfiguráló programja indítható el.

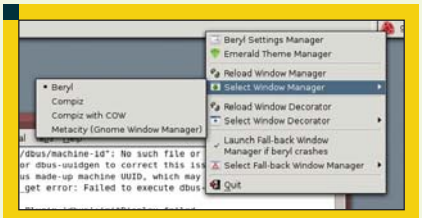


8. ábra Minimalizálás közben az ablakok elfolynak a tálcára

Itt többek közt azt állíthatjuk be hogy melyik *plug-in* működjön és hogy milyen billentyűkombinációra milyen effekt jelenjen meg. Mivel innen könnyen megtudható a billentyűkiosztás és alapértelmezetten hasonlít a *Compizéra* ezért ezt nem ismertetem külön.
A második menüponttal az emerald témaválasztója indítható el. Itt tudjuk beállítani a használni kívánt emerald témát. Ahhoz hogy az emerald legyen a window decorator, még ki kell választani a *Select Window Decorator* menüpontnál az *emeraldot*.
Előfordulhat hogy az emerald nem működik. Ekkor próbáljuk meg a parancssorból elindítani az emerald paranccsal , ilyenkor ide fogja kiírni a hibaüzenetet. Ha arra „panaszkodik" hogy nem talál néhány programkönyvtárat, akkor a ➲ *http://packages.debian.org/* segítségével keressük meg hogy melyik csomag része a keresett programkönyvtár és rakjuk fel a megfelelő csomagot.
A *Compiz* és a *Berly* nem „zavarják" egymást nyugodtan lehet őket egyszerre telepíteni, érdemes kipróbálni mind a kettőt.

Összegzés

A *Beryl* és – elvileg – a *Compiz* is képes együttműködni a *KDE*-vel is, erre terjedelmi okok miatt nem tértem ki. Akit érdekel ez a téma az látogassa meg a ➲ *http://www.novell.com/coolsolutions/feature/17174.html* oldalt, itt elég alapos képernyőképekkel ellátott útmutatót fog találni.
A nemrég megjelent GNOME 2.16-al szállított *Metacity* is képes különleges effektek produkálására, azonban ez még annyira új hogy nem része a *Debian unstable* ágának, ezért nincs szó róla a cikkben. Mire jelen írás



9. ábra A beryl-manager menüje

a Tisztelt Olvasóhoz jut valószínűleg már ez is rendelkezésre fog állni *Debian* csomagban, ezt is lehet próbálgatni.
Összességében nekem nagyon tetszenek ezek a látványos desktop effektek, bár gyakorlati hasznuk kétséges. Személyszerint a legjobb párosításnak az *AIGLX Beryl-t* tartom. Miután többmonitoros üzemmódban használtam egy napig, furcsa volt visszaülni a windowsos laptopom elé.

Szilágyi Attila (szati1@invitel.hu)
Néhány éve használ Linuxot. Alapvetően minden ezzel a témával kapcsolatos felhasználási terület érdekli és szívesen fogadja bárki kérdést, észrevételét.

KAPCSOLÓDÓ CÍMEK

Az SUSE Linux honlapja:
➲ www.opensuse.org

A Debian GNU/Linux honlapja:
➲ www.debian.org

Az 1.0-9625 nvidia driver lelőhelye:
➲ http://www.nzone.com/object/nzone_downloads_rel70betadriver.html

A Beryl honlapja:
➲ http://www.beryl-project.org/

Geometria Linux-módra – A DrGeo program alapjai

Általános és középiskolai matematika óráink szerves részei voltak a geometriai szerkesztések. Mindenki emlékszik arra, hogy amikor a vonalzója vagy a körzője akár 1 millimétert is elcsúszott, rögtön nem a kívánt alakzat lett a végeredmény, nem illeszkedtek az alakzatok a pontokra. Ezen szerkesztési probléma megoldására született néhány számítógépes alkotás, melynek egyik jeles képviselője a DrGeo.

A szerkesztési feladatok (és a matematikatanárok) mindig nagy pontosságot követelnek meg a helyes végeredményhez. Eszközeink (körző, vonalzó, stb.) pontatlansága sajnos ez ellen dolgozik. Viszont a számítógép ilyen tévedéseket nem csinál. Ezt kihasználva születtek különféle geometriai szerkesztőprogramok, közöttük Linux alá is elérhető (ingyenes) szoftverek, többek között a *DrGeo*.
A program hivatalos honlapja a ➲ *http://www.ofset.org/drgeo/* címen található, ahonnan a forráskódot, valamint a belőle készült *Debian* és *Fedora* csomagot tölthetjük le. Ezen felül a legtöbb disztribúció készítői csomagot is készítettek, így a telepítése a legtöbb esetben a csomagkezelővel könnyen megoldható.
A legutolsó verzió a cikk írásának idején az 1.1.0-s (2005. júliusi kiadás), és a honlapon olvasható, 2006. október 20-án megjelent hír szerint a fejlesztő (sajnos) felhagy a fejlesztéssel, és át kívánja adni a projektet.

Első lépések

A szerkesztések eredményei *png, LaTeX* és *PostScript* formátumba exportálhatóak, így egyéb dokumentumainkban felhasználhatjuk. Telepítés után a program terminálból a

```
drgeo
```

paranccsal indítható. Induláskor egy üres oldalt kapunk, amelyen egy új szerkesztést kezdhetünk a *Fájl* menüpont *Új* alpontja után az *Alakzat* lehetőséget kiválasztva. Ezután megjelenik a szerkesztéshez szükséges eszköztár is.
A program „szerkesztési elve" a következő: megadhatunk ún. „szabad pontokat", tehát amelyek helye nem rögzített, nem függ semmitől. Ezeket felhasználva lehet építkezni „kötött pontok" megadásával, illetve egyéb alakzatok szerkesztésével. Kötött pontot lehet megadni például két (tetszőleges) pont által meghatározott szakasz felezőpontjaként („középpont"), két görbe metszeteként, stb. Ezeket nyilván nem lehet szabadon mozgatni, mivel ők lényegében egy szabály eredményeként születtek. Viszont ha a felezőpont esetén az egyik „végpontot" mozgatom, akkor a felezőpont helyzete is változni fog.
Lehetőségünk van az alapvető geometriai alakzatok (szakasz, egyenes, félegyenes, kör, körív, vektor, sokszög) szerkesztésére, különféle geometriai transzformációk (tengelyes és középpontos tükrözés, eltolás, forgatás, átméretezés) végrehajtására, merőleges és párhuzamos egyenesek (egy lépéses) szerkesztésére.
Első pillantásra mindez nem tűnhet soknak, de nem szabad lebecsülni ezeket az „alapszerkesztések" mennyiségét, mivel mi se vagyunk képesek egy körzővel, és két vonalzóval, mégis mennyi mindent lehet ezekből felépíteni.

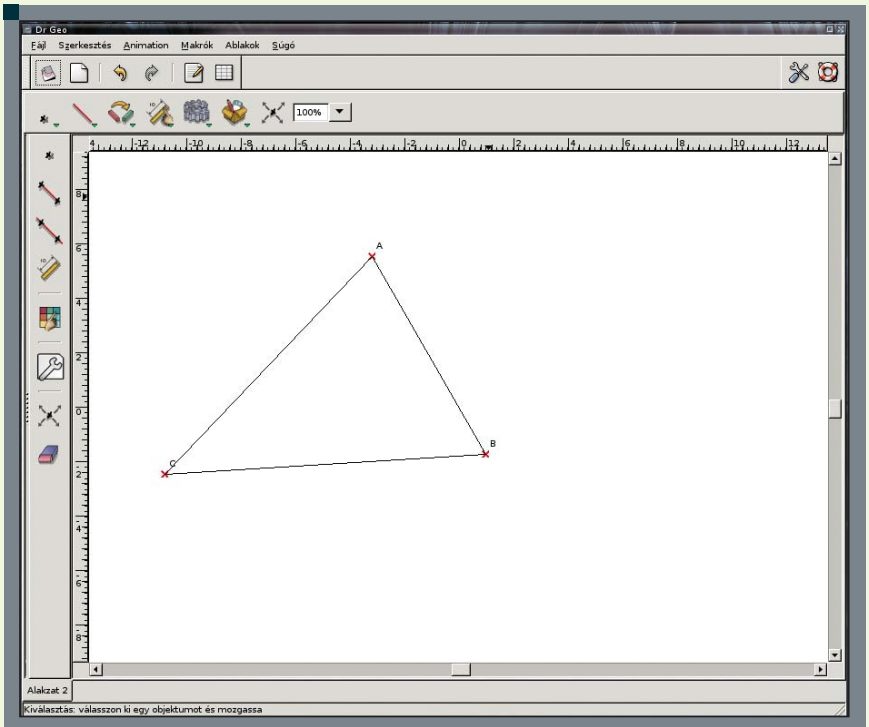
A háromszög körei

Talán az első, érdekes és nem túlzottan egyszerű szerkesztés a háromszögbe és a háromszög köré írható kör megszerkesztése volt. Tegyük mi is ezt meg, csak most körző, vonalzó és ceruza használata helyett egér és billentyűzet segítségével!
Azok kedvéért, akiknek nem a geometria volt a kedvence az órákon (ilyenek bizonyára kevesen vannak), néhány szóban elevenítsük fel, hogyan is kaphatjuk meg ezen köröket!
A háromszög köré írható kör (tehát egy olyan kör, amelyen a háromszög mindhárom csúcsa rajta van) középpontja a háromszög oldalainak felező merőlegeseinek metszéspontja, míg a beírható körének (ami egy olyan kör, melynek a háromszög oldalai a körnek egy-egy érintője) pedig a belső szögek szögfelezőinek metszéspontja.

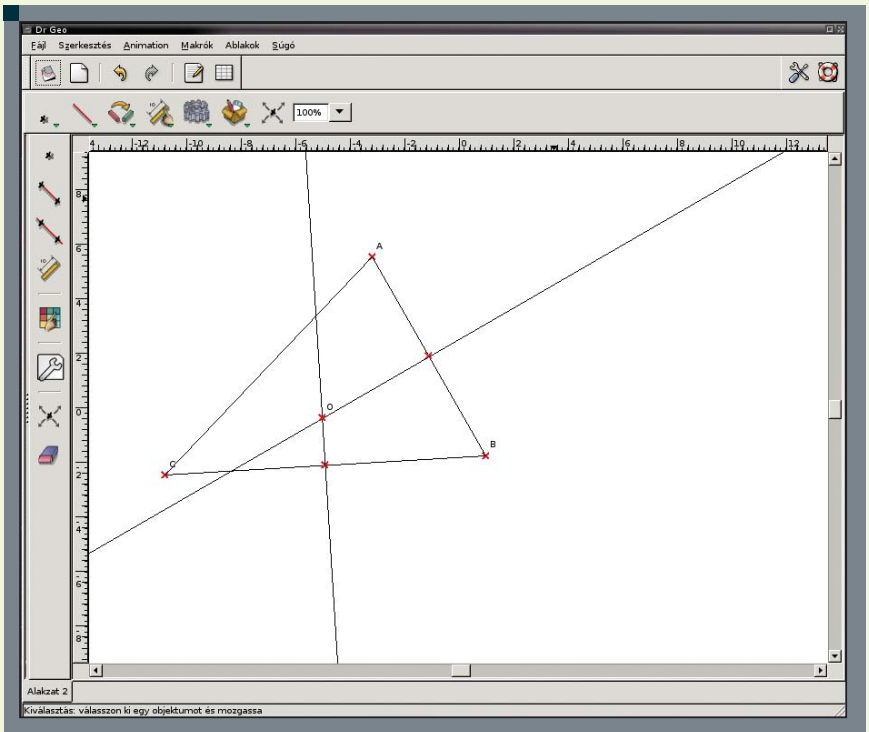
A szerkesztés

Először is egy háromszöget kellene produkálni, amelynek meg akarjuk szerkeszteni a köreit. Ez mi sem egyszerűbb: felveszünk három szabad pontot (akár az eszköztárat, akár a szerkesztési felületen jobb gombbal kattintva a *helyi menüből* a *Pont/Szabad pont* parancsot használva),

majd ezeket összekötjük egy-egy szakasszal. Ha akarjuk (már miért ne?), akkor a pontoknak nevet is adhatunk, mondjuk az *A, B, C* neveket (helyi menüben az ***Egyéb/Kinézet***-et választva, majd a pontra kattintva előjön egy párbeszédablak, ahol a színen, formán, méreten és láthatóságon kívül az objektum nevét is beállíthatjuk).
A beállítás után a szakaszok automatikusan kapnak egy-egy nevet, az *AB*-szakasz például *[AB]*-t (tehát a konvenciót betartva). Ha végeztünk, akkor az *1. ábrához* hasonlót láthatunk.
Geometriai ismereteinkből tudhatjuk, hogy az oldalfelező merőlegesek egy pontban metszik egymást (ha nem hisszük el, szerkesszük meg mindhármat, majd a háromszög csúcsait mozgassuk, és vegyük észre, hogy a három merőleges metszéspontja mindig egybeesik), tehát elegendő két oldal oldalfelező merőlegesét megszerkeszteni. Ehhez először határozzuk meg a felezőpontját mondjuk az *AB* és a *BC* szakaszoknak (***Helyi menü/Pont/Középpont***, majd kijelöljük az *AB* szakaszt, majd a *BC* szakaszt)! Ezután állítsunk merőlegest mindkét szakaszra a felezőpontban (***Helyi menü/Transzformáció/Merőleges egyenes,*** majd kijelöljük az *AB* felezőpontját és az *AB* szakaszt, hasonlóan a *BC* szakasz esetében is)! Jelöljük ki a két felező merőleges metszéspontját (***Helyi menü/Pont/Metszéspont***, majd kijelöljük a két egyenest)! Nevezzük el az így keletkezett pontot az előbb leírt módon mondjuk *O*-nak! Ha ez kész van, akkor a *2. ábrához* hasonló állapotban vagyunk.
Ezután már csak a kört kell megszerkeszteni. A ***Helyi menü/Görbe/Kör*** parancs hívása után jelöljük ki az *O* pontot középpontnak és mondjuk a *B* pontot a körvonal egy pontjának! Vigyázzunk, mivel a *B* pont két szakaszon is rajta van, így véletlenül az egyik szakaszt jelölhetjük ki, ami azt jelenti, hogy a kör sugarát a szakasz hosszának választjuk. Ha a *B* pontot az egér bal gombját nyomva tartjuk, akkor az előugró menüből válasszuk ki a *B* pontot! Ezzel a háromszög köré írt kör megszerkesztve.
Egy kicsit túl sok vonalat látunk az ábrán, nem tűnik ki a lényeg. Ezen lehet segíteni: a már említett ***Helyi menü/Egyéb/Kinézet*** parancsát használva a feleslegesnek tartott

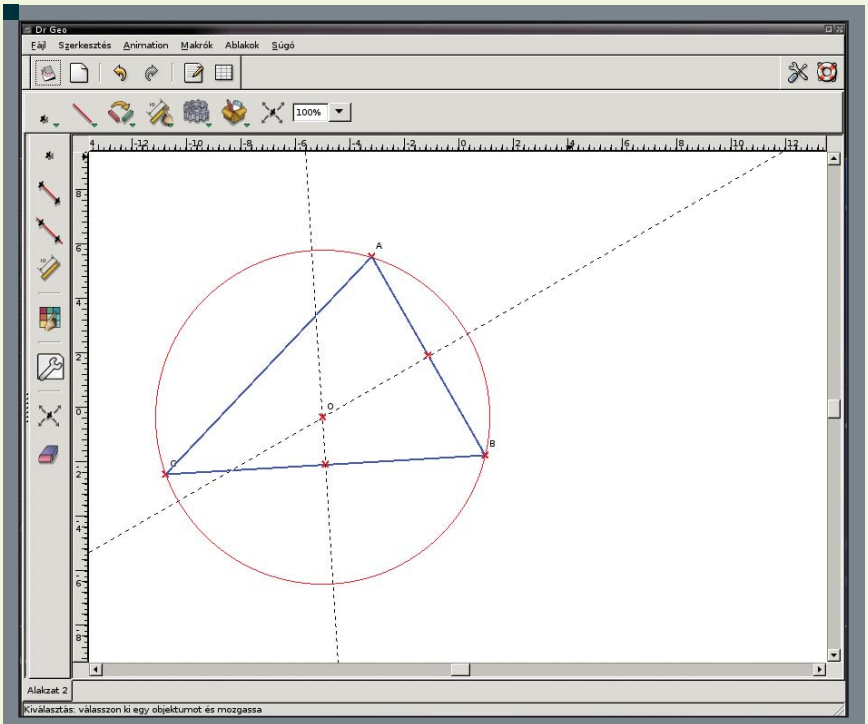


1. ábra A kiindulási háromszög

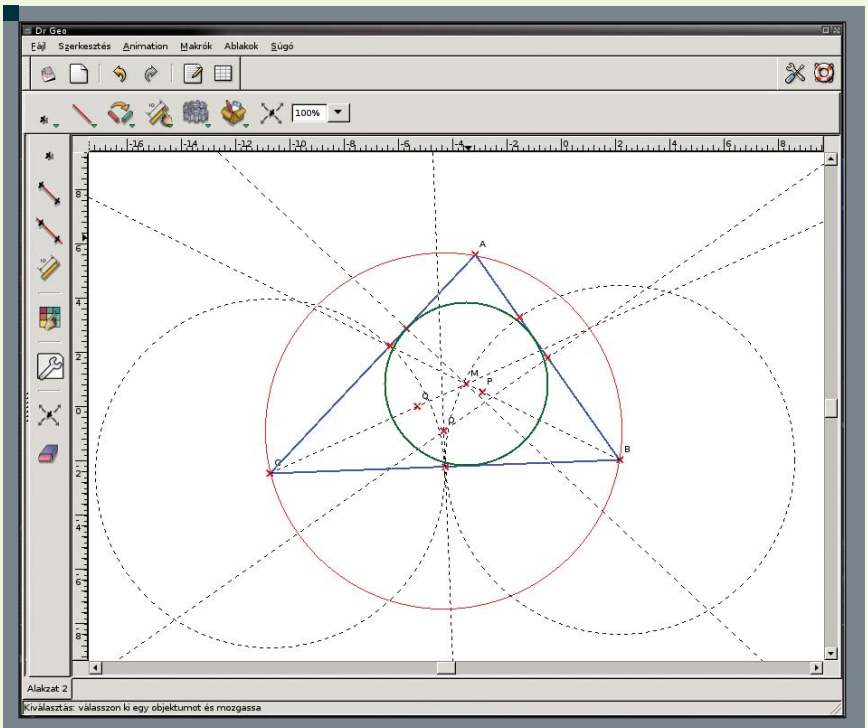


2. ábra A háromszög és oldalfelező merőlegesei

vonalakat („segédvonalakat”) eltüntethetjük, vagy szaggatottá változtathatunk, a háromszög oldalait kiemelhetjük egy szín használatával, valamint a kört is. Egy ilyen megoldás látható a *3. ábrán*.
A beírható kör megszerkesztéséhez először a szögfelezők metszéspontját kell megszerkeszteni, ahhoz pedig magukat a szögfelezőket.
Ehhez szerkesszünk egy *B* középpontú, tetszőleges sugarú kört! Mivel az, hogy „tetszőleges” sugarú, nem precíz fogalom, így legyen egy olyan kör, amelynek *B* a középpontja és pl. a *BC* felezőpontja legyen rajta



3. ábra Körülírt kör



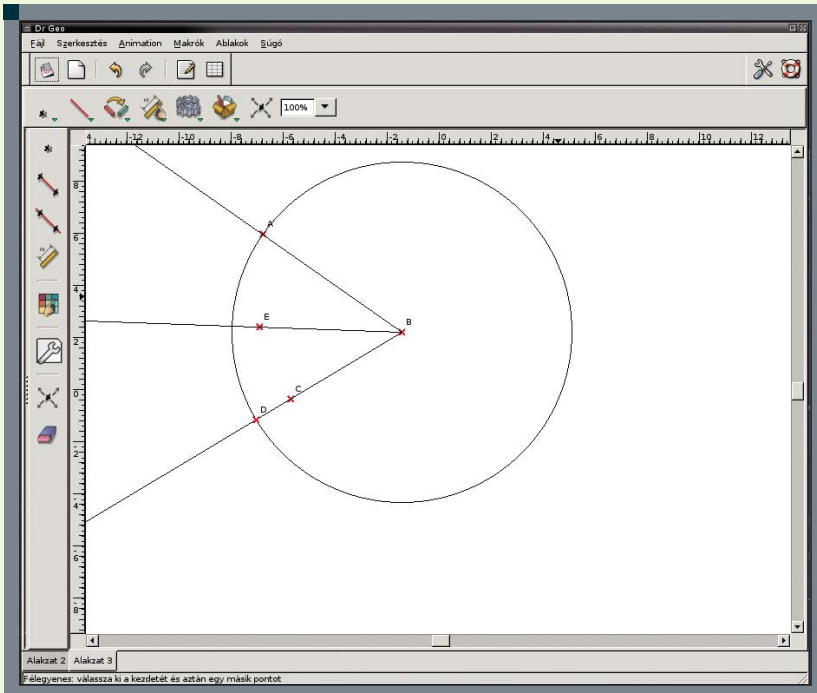
4. ábra A háromszög és a körei

a körvonalon. Ezt hasonlóan szerkeszthetjük meg, mint az elébb a körülírt kört a középpontja és egy pontja által. Képezzük az ***AB*** szakasz és a kör középpontját! Ezen pont és a ***BC*** felezőpontja „közepét” szerkesszük meg (***Helyi menü/Pont/Középpont***, majd a két pontot kijelöljük)! Legyen ez a ***P*** pont! A ***B*** pontból induló, ***P***-n átmenő félegyenes (***Helyi menü/Görbe/Félegyenes***) épp a szögfelező lesz. Ugyanezt megcsinálva a ***C*** pontra, kapjuk a ***Q*** pontot és a ***C*** csúcsnál lévő szög felezőjét. Képezzük a két szögfelező metszéspontját, legyen ez az ***M*** pont!

Mivel az ábra kezd már kaotikussá válni, így a segédköröket és a szögfelezőket tüntessük el (a *4. ábrán* csak szaggatott vonalak vannak jelen, hogy a kedves Olvasó lássa a szerkesztés közbülső lépését)!
Ezután az ***M*** pontból (amely a belső szögfelezők metszéspontja) bocsássunk merőlegest az egyik oldalra (***Helyi menü/Transzformáció/Merőleges egyenes***, ***M*** pont és mondjuk az ***AC*** szakasz kijelölése), mivel az érintő merőleges az érintési pontba húzott sugárra! Ha meghatározzuk ezen merőleges egyenes és (esetünkben) az ***AC*** szakasz metszéspontját, akkor megkapjuk a beírt kör egy pontját, innen pedig a beírt kört az eddigiek alapján megrajzolhatjuk. A végeredmény egy kicsit csinosítva a *4. ábrán* látható.
A kész ábrán a háromszög csúcsait, illetve oldalait/köreit mozgathatjuk, nagyíthatjuk, közben figyelve, mi történik az ábrával.

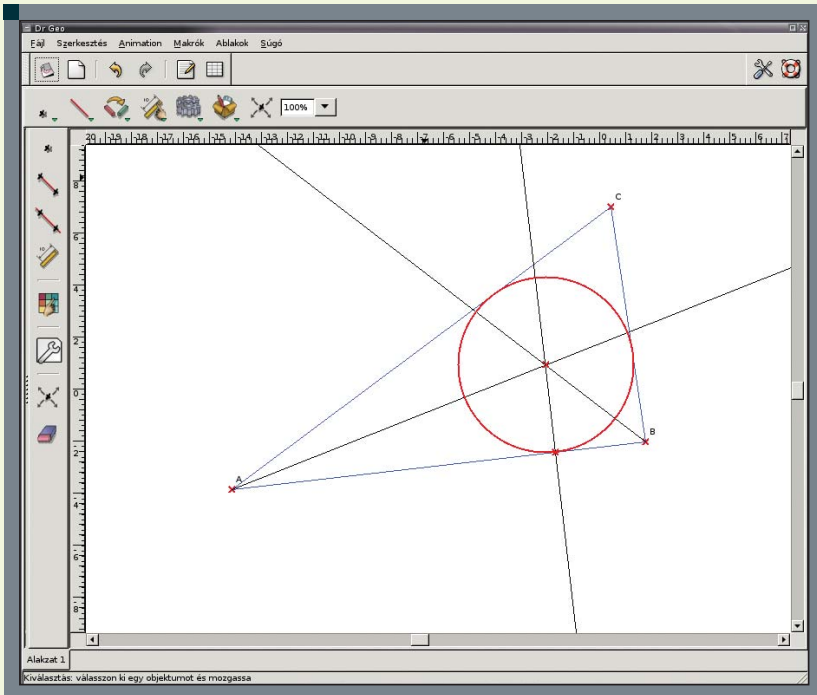
Makrók használata

Mivel a szögfelező megszerkesztése elég hosszadalmas volt (főleg a már amúgy is sok vonalat tartalmazó ábrán), és ráadásul kétszer kellett megcsinálni (sőt, egyrészt későbbi szerkesztéseink során is nagy hasznát vehetjük, másrészt nem teljesen tökéletes, amelynek az okára az ábra „variálgatása” során rájöhetünk), így felmerülhet az a kérdés, hogy a szerkesztést nem lehetne-e automatizálni. A válasz természetesen: igen. Az „automatizálást” itt is makrónak hívják, és itt az idő, hogy rögzítsük első makrónkat, amely szögfelezőt szerkeszt.
A makró felvételéhez először meg kell szerkeszteni egy példányban a szögfelezőt, amelyet a program fog amolyan sablonként kezelni. Tehát vegyünk fel három pontot egy új szerkesztőlapon, nevezzük ki az egyik pontot (legyen ***B***) a szög csúcsának, és szerkesszük meg az ebből a pontból kiinduló, a másik két ponton átmenő félegyeneseket. Ezek lesznek a szög szárai. Ezután szerkesszünk egy kört, amelynek a középpontja a szög ***B*** csúcsa, egy rajta fekvő pont a szög egyik szárán lévő A pont! Majd képezzük a kör és a másik szögszár metszetét, amelyet nevezzünk el ***D*** pontnak, majd a ***D*** és az A pont közepét szerkesszük meg, és nevezzük el ***E***-nek! Húzzuk meg a ***B***



5. ábra Szögfelező

pontból kiinduló és az ***E*** ponton átmenő félegyenest! Ez lesz a szögfelező. Az eredmény az *5. ábrán* látható. Ezután itt az ideje, hogy megmondjuk a ***DrGeo***-nak, hogy ebből egy makrót szeretnénk csinálni. Ehhez kattintsunk a felső eszköztáron levő, fogaskerékszerű ikonra (***Makrók*** gyorstipp megjelenik, amikor az egérkurzort fölé visszük), és válasszuk ki a megjelenő ikonok közül a felsőt (***Makró létrehozása***)! A megjelenő ablak utasításait elolvasva, az ***Előre*** gombra kattintva bemenetnek válasszuk az ***A***, ***B*** és ***C*** pontokat (azért ebben a sorrendben, hogy majd tudjuk, hogy a szög csúcsát „jelképező" pont a középső legyen a sorban). Ezután újra ***Előre***, majd kimeneti paraméternek válasszuk a szögfelezőt (***BE*** félegyenes). A következő ablakban



6. ábra Beírt kör makróval megvalósítva

adjuk meg a makró nevét (mondjuk ***Szögfelező***), és néhány szóban a leírását (a pontok sorrendjét például javaslom beleírni, hogy később biztosan tudjuk, hogyan kell megadni a pontokat), majd az ***Alkalmaz*** gombbal elmenthetjük a makrónkat.

Ahhoz, hogy később is tudjuk a makróinkat, el kell mentenünk, ezt pedig a ***Fájl/Több mentése*** menüpontot kiválasztva tehetjük meg. Használat előtt az elmentett makrókat tartalmazó fájlt be kell tölteni!

Lássuk, hogyan lehet makrókat használni! Ha már megvan a ***Szögfelező*** makrónk, akkor vegyünk fel egy ***ABC*** háromszöget (oldalaival együtt), és a makró használatával szerkesszük meg két szög szögfelezőjét! Ez történhet a ***Makrók*** menüpontból a ***Szögfelező*** makró kiválasztásával, vagy pedig szintén a ***Makró***-ikonra kattintva, az ***Előkészített makró végrehajtása*** ikont kiválasztva, ezek után pedig a három csúcsot megadni (vigyázzunk a sorrendre!).

A szögfelezők metszéspontjának meghatározása után bocsássunk egy merőlegest az egyik oldalra, és ezután a fentebb leírt módon megszerkesztjük a beírt háromszöget. A végeredményt (ebből semmit sem eltüntetve, a háromszöget és a kört kiemelve) a *6. ábrán* láthatjuk.

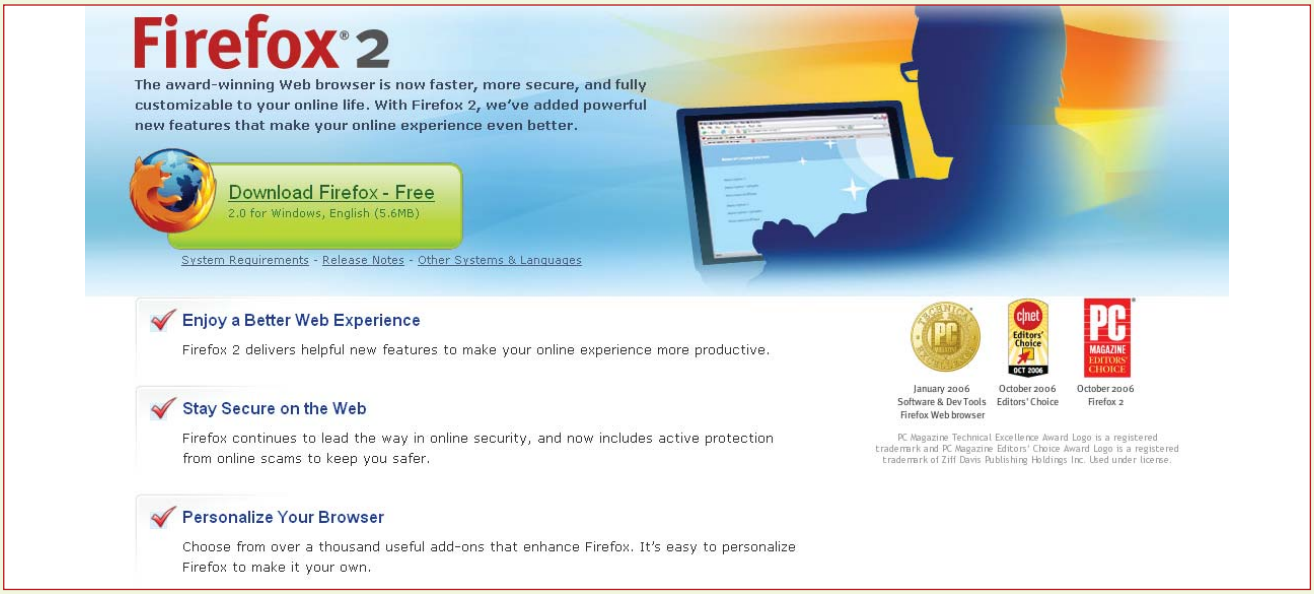
Összefoglalás

A fenti néhány példa jól demonstrálja a ***DrGeo*** képességeit és hasznosságát. A szoftver nem csak konkrét geometriai (házi) feladat esetén lehet hasznos, hanem egyéb „szabályos" ábrák (pl. mechanikai ábrák) szerkesztésekor is. A program lehetőségei még szerteágazóbbak, egy programnyelv segítségével még precízebb, még általánosabb eljárásokat is megadhatunk, valamint animációk is készíthetők vele.

Udvari Zsolt
(udvzsolt@gmail.com)
25 éves vagyok, egy gimnáziumban tanítok matematikát és fizikát. A Linuxszal először 2004 elején találkoztam, az UHU Kamionja volt, ami elgázolt. Azóta 4-5 disztribúciót hosszabban is használtam, jelenleg egy saját építésű LFS-t nyúzok.

Firefox 2.0 – A „tűzróka” újratöltve

A Firefox életútjának egy újabb mérföldkőhöz érkezett, hiszen október 24-én megjelent 2.0-ás verziója. Egy ilyen váltáskor jó esetben kívülről belülről megújul egy program. Lássuk a Firefox esetében mi változott és mi nem.



A program frissítése elméletileg a 1.5 verziótól kezdve a ***Súgó/Frissítések*** menü alatt automatikusan elindítható, de nekem ez az út nem működött (program szerint nincsenek elérhető frissítések), így a ➲ *firefox.hu* oldalról szereztem be a telepítőkészletet. Feltelepítése után a program leellenőrzi az eddig használt kiterjesztéseket, és ami nem felel meg abból újabbat keres, majd telepít vagy pedig letiltja az adott bővítményt. Ezzel nem volt problémám, szerencsére továbbra is tudom használni mindegyiket.

A böngésző elindulása után egy új modern felület fogadja a felhasználót, s egyből fel is ajánlja egy magyar nyelvű szótár telepítését, mely egy új lehetőséghez, a böngészőbe beírt szövegek (fórumok, e-mailek, blogok) helyesírás ellenőrzéshez kell. Eddigi tapasztalataim szerint ez a szótár még nem pontos, a legtöbb probléma az igekötős igékkel van, hiszen mindet két szóba akarja írni, de ezek ellenére is dicséretes kezdeményezésnek számít. Helyesírás-ellenőrzés, szavak hozzáadása a szótárhoz szövegszerkesztőknél megszokott módon történik.

Megújult a fülek kezelése

További újdonság, hogy minden fülre külön bezáró gomb került, mely megoldás hasonlít az *Operaban* lévőre. Ezt megváltoztatni a következőképpen lehet: A címsorba írjuk be:

```
about:config
```

majd keressünk rá a

```
browser.tabs.closeButtons ->
➥ szám
```

sorra. Itt a `szám` értéke lehet `0` , amikor csak az aktív fülön látszik a gomb, `1` amikor minden fülön látszik (ez az alapértelmezett), vagy `2`, amikor egyiken sem jelenik meg.

Ennek a megoldásnak előnye, hogy egy fül bezárásához nem kell azt megnyitni, (tehát senki sem látja meg tartalmát) hanem elég csak a bezárógombjára kattintani.

Az eddigi bezárógomb helyére egy lenyíló menü került, melyben felsorolásra kerül a különböző füleken megnyitott tartalom. Lehetőség van a bezárt oldalak ismételt megnyitására is, melyek elérhetőek a ***Előzmények/Nemrég bezárt lapok*** korábbi verziókban nem szereplő új menüpont alatt. Ennek tartalma a böngésző bezárása után elveszik, viszont az ***Előzmények*** alatt megtaláljuk az előzőleg megnyitott oldalak listáját, hogy ez is törlődjön ki kell venni

a pipát az *Eszközök->Beállítások->Adatvédelem->Az elmúlt napban látogatott weboldalak megjegyzése* sor elől.

Témák és kiterjesztések

A témák és kiterjesztések kezelése egyszerűbbé vált. Közös kezelőfelületük a eszközök/kiegészítők alatt érhető el. Itt ábécé szerint kerülnek rendezésre, és nem telepítési sorrend szerint. Egyrészről lehetőségek között szerepel a kiterjesztések beállítása, letiltása és a törlése is, másrészről az alapértelmezett téma beállítása, eltávolítása. Egy új kiegészítő telepítése után szükséges a böngésző újraindítása, amit egy megjelenő gomb segítségével gyorsan meg is tehetünk, ha ez megtörténik akkor az eredetileg megnyitott oldalak visszaállítódnak. Ha valakinek nem tetszik az új téma, akkor lehetősége van a régi visszaállításra, mely a ➲ *addons.mozilla.org/firefox/3479/* oldalon található meg. Telepítése után a csere az előzőekben ismertet menüpont alatt tehető meg.

RSS

Még mindig egyszerűen és könnyen van lehetőség az *RSS* hírforrásokra való feliratkozásra. De most már elég csak rákattintani az adott oldalnál címsorban megjelenő narancssárga *RSS* ikonra. Miután ezt megtettük integrálhatjuk a hírforrást az élő könyvjelzőkbe. Ezután adott oldal frissüléséről itt is értesülhetünk, nem szükséges felkeresni azt, ha pedig a megjelenő szalagcím felkeltette érdeklődésünket egy kattintással az adott cikkre kerülhetünk. Az *Eszközök->Beállítások->Hírforrások* alatt beállíthatjuk az alapértelmezett hírforrás kezelőnket, melyek közül három *on-line* szolgáltatót alapból is felajánl a program.

Biztonság

A *Mozilla* mindig arra törekedett, hogy a *Firefox* legyen böngészőpiacon megtalálható programok közül a legbiztonságosabb, s ez marketing politikájának is mindig szerves részét képezte. A 2.0-ás verzióban jelent meg az úgynevezett adathalász támadások elleni védelem (*phising protection*), ami akkor lép működésbe, ha böngésző úgy értékeli, hogy a felhasználó megtévesztő oldalra tévedt, ekkor egy behajtani tilos tábla jelenik meg a címsorban, melynél egy figyelmeztető ablakban olvashatjuk, hogyha adatokat adunk meg ezen az oldalon, akkor azokat ellophatják, és javasolja, hogy egy keresőoldal segítségével találjuk meg a számunkra szükséges oldalt. Ezen kívül a program nem engedi, hogy egy weboldalról a felhasználó jóváhagyása nélkül bármilyen program települjön a gépre.
Sokak szerint elmaradt az új verzióváltástól várt nagy durranás, de szerintem erre nem is nagyon volt szükség, mivel a *Firefox* eddig is teljes körű szolgáltatást nyújtott alapfunkciói és bővítményei segítségével, s végezetül szerintem bevált azaz álláspont, hogy a böngésző lehetséges új funkcióit előbb kiterjesztések formájában tesztelik és csak utána építik be a programba.

Fekete Imre
(imre.fekete@gmail.com)

Programtervező-matematikusként végeztem a Debreceni Egyetemen. A Linuxtól kezdetben idegenkedtem, de ma már csak azt tudom mondani róla, hogy remek rendszer.

Parancsnok a fedélzeten!

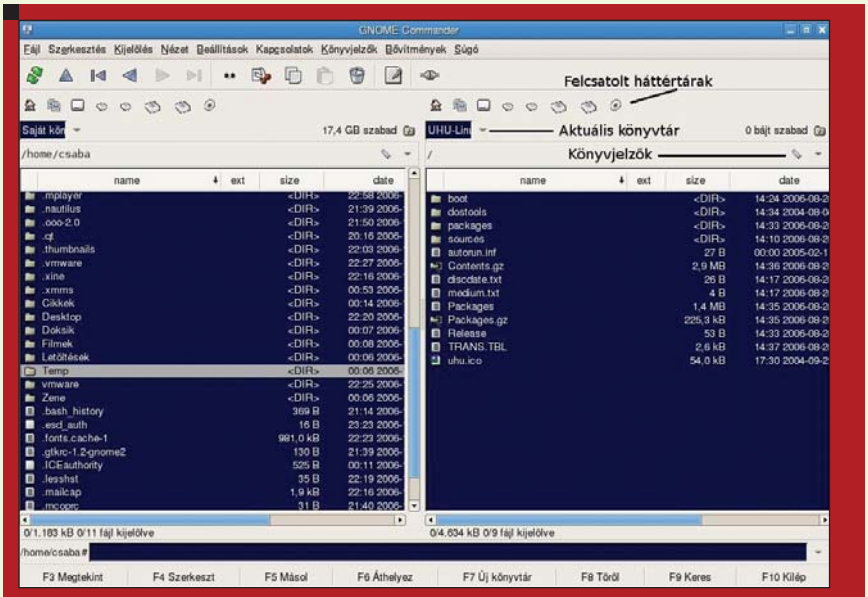
A Gnome Commander a GNOME asztali grafikus felület alapértelmezett kétpaneles fájlkezelője. Fejlesztése az idők során többször is szünetelt, azonban az utóbbi időben a készítők ismét felkarolták a projektet, és gőzerővel dolgoznak a hibajavítások mellett az új tulajdonságok beépítésén is. Hogy mit várhatunk most a programtól? Az alábbiakban kiderül.

Bevezető

Az informatika, ezen belül az operációs rendszerek és segédprogramok fejlődésével a szokásos, napi rutinnak számító feladatok is egyre inkább lerövidültek. A kétpaneles fájlkezelők megjelenésének hála a felhasználóknak már nem kellett fél képernyőnyi sorokat begépelniük ahhoz, hogy egy szimpla másolást elindítsanak.
A fájlkezelés a kurzormozgató billentyűk, a funkcióbillentyűk és az *Enter* használatára redukálódott. A *Norton Commander*, a *Dos Navigator*, a *Far Manager* vagy a *Midnight Commander* programnevek még ma is ismerősen csengenek, nagy valószínűséggel egyiket-másikat még használjuk is. Ezek a karakteres felületű programok adták a táptalajt a manapság leginkább használatos olyan grafikus felületű fájlkezelőknek, mint a *Windows* alatt futó *Total Commander*, a *KDE* felülethez tartozó *Krusader*, és a *GNOME*-os *Gnome Commander*. A célom bemutatni a sok fájlkezelő közül egy olyan alternatívát, amely a mindennapi feladatok elvégzéséhez gyors és hatékony segítséget nyújt.

Bemutatás

A *Gnome Commander* honlapján, ami a ➲ *http://www.nongnu.org/gcmd/* címen érhető el, ezen sorok írásakor jelent meg az 1.2.1-es verziószámú stabil változat. Érdemes megemlíteni, hogy a program fejlesztése két ágon folyik. A fejlesztők az 1.2.x jelölést használják a stabil kiadások megkülönböztetésére, míg a fejlesztői ág az 1.3.x jelölést kapta. Ezen fejlesztői ágból emelik át a friss tulajdonságokat és a hibajavításokat a stabil ágba, ahogyan ez most az 1.2.1-es kiadás esetén is történt. Az alkotók elmondása szerint programjuk azoknak a haladó szintű felhasználóknak készült, akik a munkájuk során a fájlkezelésre szeretnek fókuszálni. Ez abból is látszik, hogy a honlap szerint a *Gnome Commander* nem azoknak való, akik szeretik, ha időjárás előrejelző is van a fájlkezelőjük eszköztárában. Való igaz, ez a program sokkal inkább egy éles kés, mintsem egy svájci bicska, azonban sokszor pont az ebből adódó gyorsaság és áttekinthetőség segíti a könnyebb feladatmegoldást. A program *Dvornik László*, *Kelemen Gábor* és *Sári Gábor* jóvoltából magyarul szól hozzánk. Köszönet érte!



1. ábra A Gnome Commander

Telepítés

A *Gnome Commander* előre lefordított csomagként megtalálható minden *Linux* disztribúcióhoz.
Azoknak azonban, akik szeretik a legújabb funkciókat használni, forrásból kell telepíteniük kedvenc fájlkezelőjüket. A tömörített forráskódot a ➲ *http://www.nongnu.org/gcmd/download.html* oldalról tölthetjük le. Kitömörítés után az *INSTALL* fájl utasításait követve a forráskód

1. táblázat *Néhány hasznos billentyűkombináció*

SHIFT+F2	Könyvtárak összehasonlítása
SHIFT+F4	Új fájl szerkesztése
SHIFT+F5	Másolat készítése a fájlról az adott könyvtárba (biztonsági másolat készítésére alkalmas)
CTRL+SHIFT+F5	Szimbolikus link létrehozása az aktív panelen lévő könyvtárból az inaktív panelen lévő könyvtárba
SHIFT+ENTER	A parancs végrehajtása után a terminált nyitva hagyja (látható a parancs kimenete)
ALT+ENTER	A kijelölt fájl tulajdonságainak megjelenítése
ALT+1	Váltás a felcsatolt háttértárak között a bal oldali panelben
ALT+2	Váltás a felcsatolt háttértárak között a jobb oldali panelben
CTRL+G	Gyors FTP kapcsolódás
CTRL+P	Az aktuális könyvtár teljes elérési útjának hozzáadása a parancssorhoz
CTRL+ENTER	A kijelölt fájl nevének hozzáadása a parancssorhoz
CTRL+SHIFT+ENTER	A kijelölt fájlnév teljes elérési úttal való hozzáadása a parancssorhoz
CTRL+SHIFT+H	Rejtett fájlok megjelenítése / elrejtése
CTRL+DOWN	A parancssorban kiadott utasítások listája
CTRL+"."	Forrás = Cél, azaz az inaktív panelben is az aktív panel tartalma jelenik meg

könyvtárában kiadott következő néhány paranccsal percek alatt telepíthetjük a programot:

```
$./configure
$make
$make check
#make install
$make clean
```

A `make install` utasítást természetesen root-ként bejelentkezve kell kiadni.

Ha már fut a Gnome Commander

A kezdő felhasználók első és talán legfontosabb kapaszkodója a *Linux*os világban egy olyan fájlkezelő rendszer, amely felépítése és működése hasonlít a *Windows* alatt megszokott legnépszerűbb hasonszőrű programhoz, a *Total Commander*hez. Első ránézésre a *Gnome Commander* rengeteg kezelésbeli tulajdonsága megegyezik *Windows*os társáéval. A funkcióbillentyűk parancsai mindössze két helyen térnek el, ezzel is fokozva a hatékonyságot. Az *F2* segítségével gyorsabban nevezhetünk át egy állományt, mint a *Shift+F6* billentyűkombinációval, míg az *F9* az *ALT+F7*-hez hasonlóan keresésre szolgál.
A *Backspace* itt is egy könyvtárral léptet minket fentebb a könyvtárfában, míg a *Space* ugyanúgy kijelöl illetve az adott könyvtár méretét írja ki. A különböző kijelöléseket is a már jól ismert billentyűkkel vagy billentyűkombinációkkal tehetjük meg. Érdekesség, hogy míg a *Fel* és *Le* billentyűkkel tudunk egy adott könyvtárban navigálni, addig a *Bal* nyíllal egy szinttel feljebb tudunk lépni, míg a *Jobb* nyíllal az adott könyvtárat nyitjuk meg.
A *Gnome Commander* menürendszerét végignézve jövünk rá a fentebb említett éles késes hasonlat találó mivoltára. Az általa biztosított lehetőségek egy-két kivételtől eltekintve nem lépik át az alapvető fájlkezelés határait.
A menüpontok bemutatása meghaladná a cikk határait, így azok felfedezése az olvasóra vár. Megnyugtatásként közlöm, hogy a felépítése logikus és átlátható, a kezdőknek sem jelent majd problémát az eligazodás. Fontos, hogy ahogyan már fentebb is írtam, a *Gnome Commandert* készítői a gyakorlottabb felhasználóknak szánták. Ez a leginkább ott nyilvánul meg, hogy nem minden funkciója érhető el menüből, hanem csak billentyűkombinációk által. Ezért a gyorsabb és hatékonyabb munkavégzés érdekében vessünk egy pillantást a ➲ *http://www.nongnu.org/gcmd/keys.html* oldalra.

Testreszabás

Mindenki szereti a saját szájíze szerint beállítani a naponta használt programjait. Ezt elősegítendő vessünk egy hosszabb pillantást a *Beállítások* menüpont érdekesebb részeire.

- A *Beállítások* ablak *Általános* fülén adhatjuk meg többek között, hogy a jobb egérgombot kijelölésre vagy egy gyorsmenü előhívására szeretnénk-e használni. Itt állítható még többek között a könyvtáron belüli gyors keresés billentyűkombinációja is.

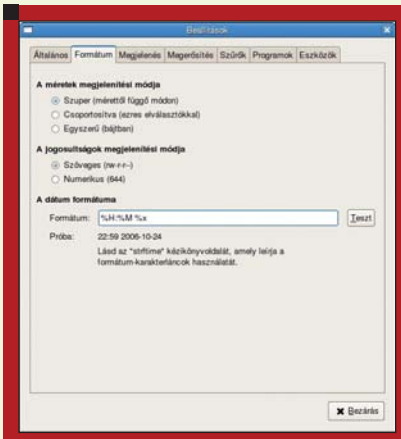
- A *Formátum* fülön A *méretek megjelenítési módja* alatt dönthetjük el, hogyan szeretnénk állományaink méretét viszontlátni. Én a *Szuper* lehetőséget kedvelem, így az áttekinthetőség kedvéért a fájlméretet bájtként (B), kilobájtként (kB), megabájtként (MB) vagy gigabájtként (GB) látjuk. A fájlok jogosultságainak számok formájában történő megjelenítése is itt található.

- A *Megjelenés* fül a beállítási lehetőségek közül a legveszélyesebb. Az ember ha belefeledkezik, hajlamos egy jó félórát is elszöszmötölni vele.

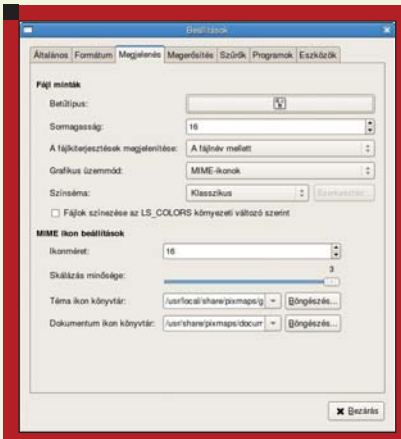
- A *Megerősítés* fülön megtett megfelelő beállításokkal sok kellemetlenségtől óvhatjuk meg magunkat. Erősen ajánlott a *Megerősítés törlés előtt* jelölőnégyzet bejelölése, továbbá a másolás és mozgatás során fellépő felülírások előtti rákérdezés is.

- A *Szűrők* fülön állíthatjuk be, hogy a mindennapi használat során milyen típusú fájlokat szeretnénk figyelmen kívül hagyni azáltal, hogy nem jelenítjük meg őket.

2. ábra Általános beállítások



3. ábra A Formátum fül

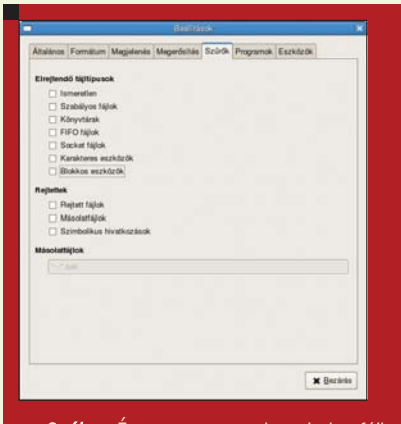


4. ábra A Gnome Commander „ruhája"

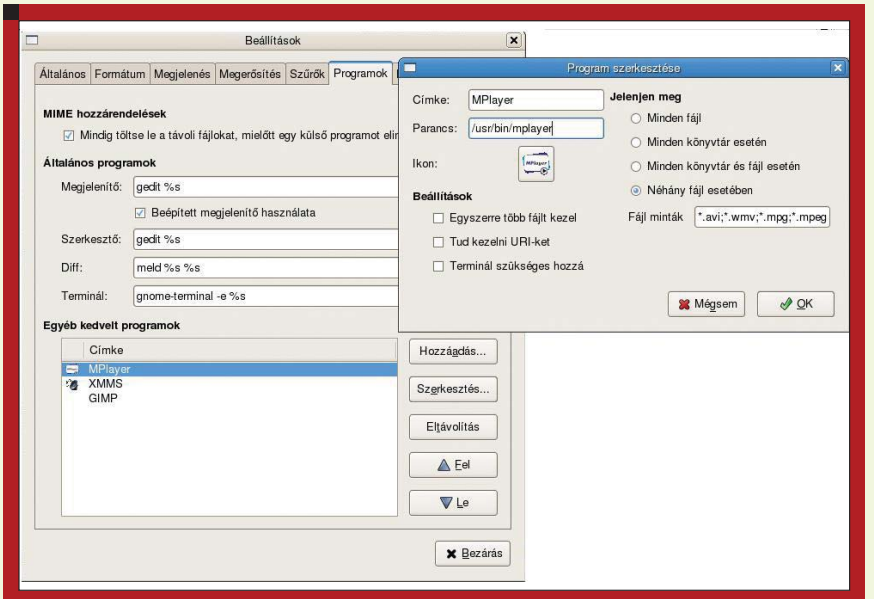
- A *Programok* fülön belül az általános programok megadásán túl az általunk kiválasztott fájlkiterjesztésekhez rendelhetünk hozzá különféle programokat. Ezeket a jobb egérgombbal aktiválható gyorsmenüből érhetjük el.

- Az *Eszközök* fülön adhatjuk meg a számítógépünkhöz csatlakoztatott tárolókat, a csatolási ponttal és az eszköz nevével együtt. Elméletileg *automount* használata esetén ezt nyugodtan kihagyhatjuk, a *Gnome Commander* automatikusan jelzi, és egy, az adott eszközre jellemző ikonnal bővíti az eszköztárat, ha új háttértárat csatlakoztattunk a számítógéphez. A gyakorlat azonban azt mutatja, hogy néha egy-egy partíciót nem ismer fel, noha az be van csatolva a rendszerbe. Ilyenkor lesz szükségünk erre a lehetőségre.

5. ábra Megerősítés a biztonságosabb adatkezelésért



6. ábra Én azt szeretem, ha minden fájlt láthatok



7. ábra Itt állíthatjuk be a gyakran használt vagy kedvenc programjainkat

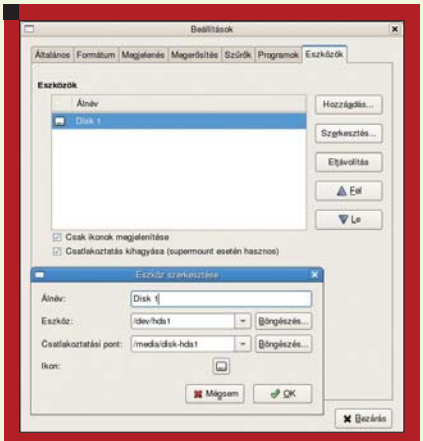
Gyakorlati példa helyett

Ahelyett, hogy elmagyaráznám, hogyan tudunk fel illetve lefelé mozogni egy adott könyvtárban, vagy másolni egy fájlt, inkább megosztom pár gyakorlati tapasztalatomat.

- A *Gnome Commander* ugyan tud FTP kapcsolatot létesíteni, azonban a megszakadt letöltést még nem tudja folytatni. Ezt vegyük figyelembe mielőtt nagyobb méretű adat letöltésébe kezdünk!

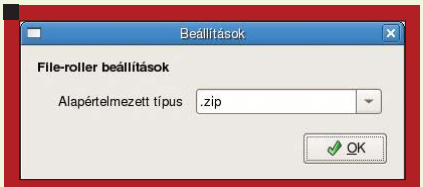
- A *Bővítmények -> Bővítmények beállítása...* alatt engedélyezzük és állítsuk be a *File Roller* nevű programot, amivel a tömörített fájlokat kezeljük. Mivel a *File Roller* nagyon sok formátumot ismer, itt csak azt kell megadnunk, hogy az archiválás milyen típusú tömörítéssel történjen.

- Ha a rendszergazdai jogok miatt szükséges a *sudo* használata, a parancssorból *ENTER* helyett



8. ábra Azon partíció hozzáadása, amit a Gnome Commander nem ismert fel automatikusan

9. ábra Az FTP kapcsolat beállítása csak pár egyszerű lépés



10. ábra Archiváláshoz a .zip formátumot használom

a *SHIFT+ENTER* kombinációval futtassuk a parancsot, így nem tűnik el a terminálablak, hanem módunkban áll megadni a megfelelő jelszót.

- Jogosultságok és fájlhozzárendelések változtatásának a legegyszerűbb eszköze a ***Fájl*** menü/ ***Tulajdonságok*** menüpontja (*ALT+ENTER*).
- Könyvjelzők hozzáadásával könnyebben érhetjük el a gyakran használt könyvtárakat, ezzel is időt takarítva meg. Érdemes a ***Könyvjelzők*** menüpontot használni, mert az eszköztári gomb csak az adott partíción belüli könyvjelzőket jeleníti meg az összes helyett.

Zárszó

Noha a *Gnome Commander*re az egyszerűség és néhány helyen a kiforratlanság a jellemző, amin a fejlesztőknek még lesz mit dolgozniuk, az 1.2.1-es kiadást elnézve jó úton haladnak. Szerencsére a negatívumokat feledteti velünk a könnyű kezelhetőség, az áttekinthetőség, a gyors munkavégzést segítő billentyűkombinációk, a testre szabhatóság és a lelkes és gyorsan dolgozó fejlesztői csapat. Ha valakinek egy sallangmentes fájlkezelő programra van szüksége, a *Gnome Commander*t csak ajánlani tudom!

Leszkoven Csaba

Merre tovább Ubuntu?

Október 26-án jelent meg az Ubuntu Linux új kiadása, az 6.10-es verzió vagy Edgy Eft. Azonnal elkezdődtek a következő kiadás munkái is. Az Ubuntu kezdettől fogva szoros munkatempót diktál, ezért és a mögötte lévő stabil anyagi bázisnak köszönhetően jó eséllyel számíthat arra, hogy valamikor napjaink legnépszerűbb operációs rendszere, a Microsoft Windows vetélytársa legyen.

Talán ma még túlzó optimizmusnak tűnik ez, de az ***Ubuntu*** deklarált céljai éppen efelé az ígéretes irányba mozdítják el. 2004 októberében, amikor megjelent az első stabil verzió, villámgyorsan népszerűvé vált, olyan gyors karriert futott be, amit még előtte ***Unix*** disztribúció nem produkált. A világ linuxos felének jelentős része láthatóan éppen erre várt: egy ***Debian*** alapú, könnyen kezelhető, rendszeresen frissülő, óriási csomagkészlettel rendelkező és biztosan szabad, ingyenes operációs rendszerre. ***Mark Shuttleworth*** világhírnevét (az első afrikai űrturista volt) és informatikai cégeinek sikerét a ***Linux*** desktop világának érdekében használta fel, ami az általában csupán lelkesedésből fejlesztett vagy kereskedelmi célú disztribúciók mellett sokat lendített az ***Ubuntu*** népszerűségén. Bizonyára különleges nevét sem felejti el könnyen az ***Afrikán*** kívül élő népesség, a mögötte lévő filozófia, az emberség, a másikra figyelés, a szabadság pedig sokaknak mutat példát arra, hogy a globalizációnak lehetnek pozitív hatásai is.

Az ***Ubuntu*** valóban nagyszerű célokat tűzött ki: a Debian hagyományát folytatva mindig szabad és ingyenes marad, segíti szülő projektjének fejlesztését is, rendszeresen jelentkezik újabb kiadásokkal, amelyek mindig igyekeznek a legújabb szoftververziókkal és a legújabb technikákkal működni, emellett stabil, a használhatóságra és a kinézetre egyaránt nagy gondot fordító desktop rendszert alkot. A vállalati szférát is megcélozva, ***Dapper Drake*** kódnevű, 2006 júniusában megjelent kiadásával elkezdődött az úgynevezett ***Long Term Support (LTS)***, ami az előző verziókhoz 18 hónapos támogatásához képest a desktop verziót 2009-ig, a szerver verziót pedig 2011-ig támogatja. A szerver verzió az ***Ubuntut*** univerzális operációs rendszerré teszi, hiszen így nem csak a desktop felhasználók élvezhetik a gyors fejlesztés és stabilitás előnyeit. Egy ***Ubuntu*** szerver a ***Debian*** szerverhez képest azzal az előnnyel rendelkezik, hogy frissebb és gyorsabban is frissül. Ez lehet hátrány is olyan helyeken, ahol ez nem kívánatos, tehát kritikus, nagy megbízhatóságú rendszereken.

Csupán két év telt el az ***Ubuntu*** első megjelenése óta és mégis töretlenül az első helyen szerepel a ➲ ***distowatch.com*** disztribúciókkal foglalkozó oldal statisztikájában. Olyan, sokkal régebben népszerűeket előz meg, mint az ***OpenSuse***, a ***Fedora*** vagy a ***Mandriva***. A mögötte álló cég, a ***Canonical*** igazán sikeresnek mondhatja legfőbb termékét, így 2006-ban a ***Linuxawards*** a legjobb ***Linux*** disztribúciónak nyilváníthatta. Az ***Edgy Eft*** kapcsán – amire a ***Dapper*** kiadás elhúzódása folytán alig 4 hónap fejlesztési idő jutott – , röviden nézzük meg, merre is tart az ***Ubuntu*** fejlesztése, mi várható a közeljövőben.

Újdonságok, tervek

A 6.10-es verzió nem kapott hosszú távú támogatást, hiszen ez a verzió inkább az új technológiákra

fókuszált, ***LTS support*** tehát a következő kiadásnak (***7.04 - Feisty Fawn***) jár. Az egyik érdekesség az eddig szinte egyeduralkodó ***init*** rendszer, a ***System-V*** vagy ***sysvinit*** teljes lecserélése volt, az új technika neve: ***upstart***. Az ***init*** felelős a szolgáltatások elindításáért és leállításáért.

Az ***upstart*** esemény alapú, ami lehet például a rendszerindulás, a gyökér fájlrendszer írhatóvá válása, ha egy blokkeszközt adnak hozzá a rendszerhez, egy fájlrendszer csatolása, egy bizonyos időpont vagy ismétlődő időperiódus, ha egy másik folyamat elindul vagy leáll, egy fájl megváltozik, fájlok kerülnek a sorba, egy hálózati eszköz detektálása vagy ha az alap átjáró megváltozik. De eseményt bármely folyamat is önállóan generálhat. Az ***upstart*** ezekre képes reagálni és az ennek megfelelő szolgáltatást elindítani, leállítani vagy újraindítani. Az ***upstart***-ot valószínűleg hasznosnak találják majd, akiknek gyakran kell ***USB*** eszközöket csatlakoztatni, a rendszeradminisztrátorok akik azt szeretnék, hogy bizonyos folyamatok akkor is automatikusan újrainduljanak, ha előzőleg valamilyen ok miatt leálltak, az ***iPod*** felhasználók akik szeretnék, ha az eszköz csatlakoztatásakor automatikusan elinduljon egy bizonyos szoftver, az adatbázis adminisztrátorok, akik szeretnék, ha a rendszer leállásakor automatikusan mentésre kerülne az adatbázis és így tovább, rengeteg olyan feladat van, ahol az ***upstart*** jobban teljesít, mint az eddigi rendszer. A modern kernelekkel is hatékonyabban együttműködik. Elvileg a bootolási folyamatnak is fel kell általa gyorsulni. Az ***Edgy***-ben tulajdonképpen ennek az alapozása kezdődött el, így egyes dolgok csak a következő kiadásban fognak működni. További információk a ➲ http://upstart.ubuntu.com címen érhetőek el.

Az ***Ubuntu*** nagyon fontos iránynak tartja a multimédiás tartalmak minél jobb támogatását, ami a következő kiadásokban is fontos cél lesz. Ennek érdekében került például most bele a Novell kiváló terméke az ***F-Spot*** képmenedzser szoftver, aminek hasznos szolgáltatása az idővonal, az alapvető képszerkesztési funkciók vagy a könnyű exportálás ***Flickr***-be, ***Picasawebre***, webgalériába vagy ***CD***-re. A nem támogatott kodekek és egyéb alkalmazások is valószínűleg elérhetőek majd a ***Feisty*** kiadásban, miután az ***universe*** és ***multiverse*** tárolók már alapértelmezésben nyitva lesznek. Ezentúl egyszerűen használhatjuk a ***Xen*** virtuális gépet ***Ubuntu*** alatt, mivel bekerültek a csomagok közé a ***Xen Ubuntu*** kernelek, dokumentációk és szoftverek. Természetesen a ***VMware Player*** is az alaprendszer része.

Sokan szeretnének laptopjukon is ***Linuxot*** használni, de azok támogatása még messze nem tökéletes. Így a következő kiadás egyik fontos iránya a laptopok jobb integrálása. Jelenleg az ***Acer***, ***Apple***, ***Asus***, ***Averatec***, ***Compaq***, ***Dell***, ***ECS***, ***Fujitsu***, ***Gateway***, ***Gericom***, ***Hewlett-Packard***, ***IBM***, ***Packard Bell***, ***Sony*** és ***Toshiba*** laptopok bizonyos típusai támogatottak ***Ubuntu*** alatt. Bizonyosak lehetünk benne, hogy a kinézet és a felhasználó életének minél kényelmesebbé tétele is hosszú távon fontos cél marad, így az ***Edgy***-ben is találkozhatunk lekerekített sarkú ablakokkal, új bootképernyővel, látványos, elegáns bejelentkező képernyővel, grafikus lemezhasználat mérővel. A ***Feisty*** a tervek szerint még nagyobb hangsúlyt fektet majd a látványelemekre. Nemcsak mint kiváló desktop alkalmazás, de a különböző vállalati szegmensek elérése is szorgalmazott cél, az ***Ubuntu Server*** eddig sem vallott kudarcot a kis- és középkategóriában, de szeretnék elérni a nagy- és csúcsszerverek szintjét is.

Már eddig is a legkönnyebben telepíthető ***Linux*** operációs rendszernek tartották, de a telepítés a jövőben még egyszerűbbé válik amellett, hogy a haladó felhasználók és ***IT*** szakemberek számára egyre részletesebb beállítási lehetőségeket is nyújt, mint például a ***RAID*** vagy az ***LVM***. A grafikus, könnyen használható beállítási lehetőségek is dinamikusan fognak bővülni, mint a különböző hálózati beállítások, a biztonsági eszközök beállításai vagy a grafikus rendszer részletes testreszabása. A ***Live CD*** és a telepítő rendszer összevonása azt eredményezi, hogy nem kell foglalkoznunk külön letöltéssel és ***CD***- vagy ***DVD***-írással, hogy telepítés nélküli rendszerünk is legyen, hanem egy ***CD***-n megkapjuk mindezeket.

A telepítés és a szoftverek menedzselése is erőteljes fejlesztés alatt áll, ma is többféle módszer közül választhatunk, egyszerűbb és részletesebb telepítő, szoftvermenedzselő szoftverek közül. A ***Synaptic*** a haladó felhasználóknak, míg az Alkalmazások telepítés és eltávolítása eszköz az átlag desktop felhasználó részére nyújt egyszerű kiválasztási és telepítési/eltávolítási lehetőséget.

Több tekintetben is közeledik a ***Mac OS X***-hez az ***Ubuntu***: nagy hangsúlyt fektet a grafikus felületre, a grafikus eszközökre, a letisztult és elegáns kinézetre, a fejlett keresésre vagy a speciális effektekre. Ezeket bár a rendszergazdák nem szokták komolyan venni, nagyon fontosak lehetnek a felhasználók megnyerése szempontjából.

Egyre inkább úgy tűnik, az ***Ubuntu*** nem kíván előnyben részesíteni egyféle ablakkezelőt, hanem párhuzamos fejlesztői csoportokkal támogatja a legfontosabbakat: a ***Gnome***, a ***KDE*** és az ***Xfce*** egyaránt választható és így mintegy felkínálja a kipróbálás és választás lehetőségét.

A minőség fenntartása és ellenőrzését és komolyan veszi a Canonical. Sokan azért választják az ***Ubuntut***, mert bizonyosak kívánnak lenni abban, hogy egy megbízható és folyamatos fejlesztés áll mögötte és ez töretlenül folytatódik is a jövőben. Mindezek miatt az otthoni és a vállalati szférában is nagy jövő előtt áll ***Mark Shuttleworth*** emberségre és a közös munkára alapozott operációs rendszere.

Molnár Norbert
(molnar.norbert@gmail.com)
35 éves, rendszergazdaként dolgozik, 5 éve foglalkozik Linuxszal. Főként a szabad szoftverek és a számítógépes biztonság érdekli. Budapesten él feleségével és 2 éves kisfiával. Hobbija a csillagászat és a filozófia – lehetőleg jó vörösbor mellett.

GnuPlot – Adatok ábrázolása mesteri fokon (2. rész)

Az első részben a GnuPlot alapjaival és alapparancsaival ismerkedhettünk meg. Az ottani információk alapján még nemigen tűnik ki a GnuPlot előnye a különféle táblázatkezelők (például az OpenOffice.org Calc része) diagramkészítőivel szemben.

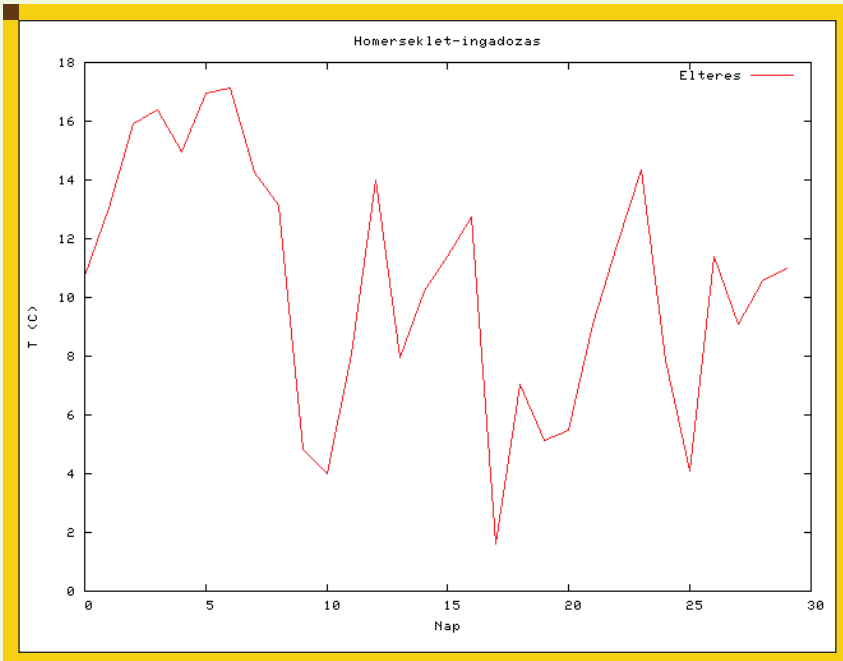
Emlékeztető

Az első részben megismerkedtünk a *gnuplot*-tal, majd ennek örömére egy meteorológiai adatsort kezdtünk szemléletessé tenni. Az utolsó ábránk egy grafikon volt, amelynek vízszintes tengelyén 2005. áprilisának napjai voltak, míg a függőleges tengelyen hőmérsékleti adatok. Az ábrán három görbe szerepelt: a napi átlag-, minimum- és maximum-hőmérséklet. Itt már láthattuk a napi hőmérséklet-ingadozást, bár ez még nem volt az „igazi”, a lehető legpraktikusabb ábrázolási módja az ingadozásnak. Keressünk erre egy jobb megoldást!

Műveletek az adatokkal

Először is gondoljuk meg, hogy mi jellemzi legjobban a napi hőingadozást! Rövidebb-hosszabb elmélkedés után rájövünk, hogy a napi legmagasabb és legalacsonyabb hőmérséklet közötti eltérés. Tehát akkor a maximum- és minimum-hőmérséklet különbségét kellene ábrázolni! Lehetséges-e az a *gnuplot*-ban, hogy két „oszlop” különbségét ábrázoljuk? Persze, és ráadásul nem is kell egy külön oszlopot létrehozni, amelyben a különbségek szerepelnek, hanem „dinamikusan” is lehet csinálni. Akkor indítsunk egy *gnuplot*-ot, majd az alábbi parancsokat adjuk meg neki:

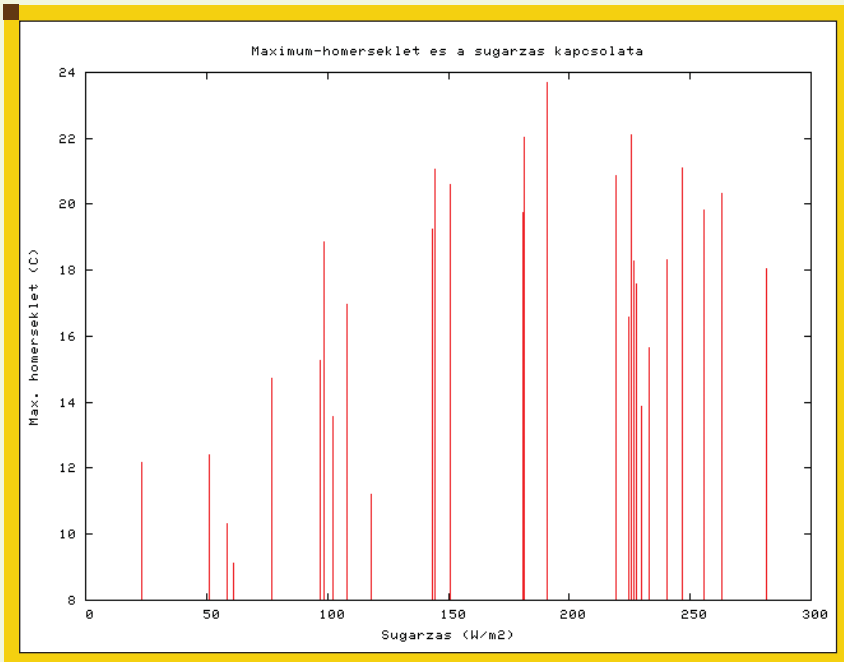
```
set title "Homerseklet-
➥ingadozas"
set xlabel "Nap"
set ylabel "T (C)"
plot "2005-apr.txt" using ($3)-
➥($4) with lines title "Elteres"
```



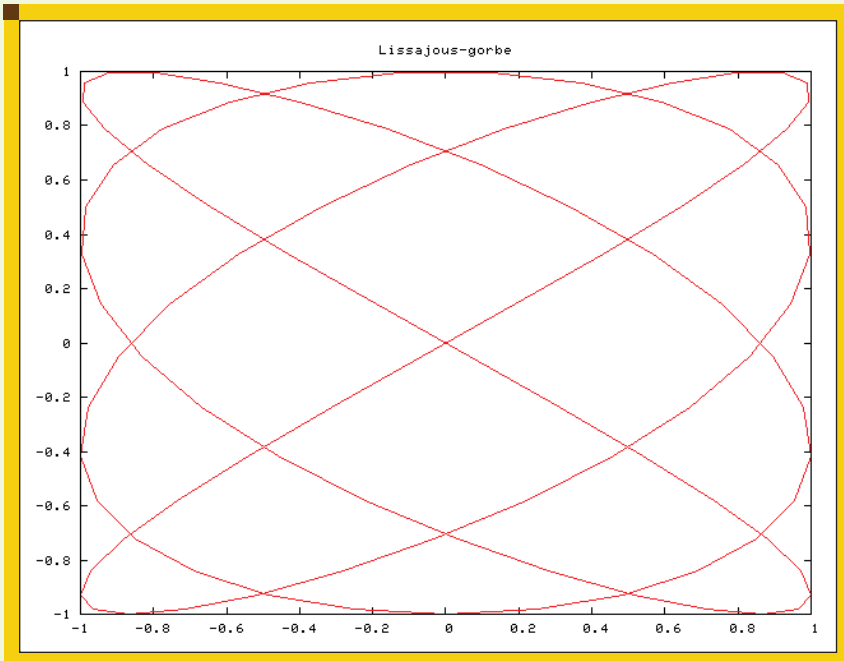
1. ábra Hőmérséklet-ingadozás

A végeredmény az *1. ábrán* látható. Az új dolog az eddigiekhez képest az utolsó sorban rejlik, mégpedig a `($3)-($4)`részben. Ez az, ami utasítja a programot arra, hogy a harmadik oszlop adataiból vonja ki az első oszlop adatait, és azt használja (`using`). Itt már tényleg nagyon jól látszódik az, amit már az előző részben még csak „szemre” állapítottunk meg: április 10. körül kicsi volt az ingadozás, míg a hónap elején viszonylag nagy. Hasonlóan használható a többi matematikai művelet, képlet, formula. Az alapműveletek jelei a „megszokottak”, a hatványozásé a kettő csillag (például a `($3)**2` a 3. oszlop négyzetét jelenti), a trigonometrikus és egyéb operandusok szintén a szokásosak (`sin, cos, log, ...`), csak arra kell figyelni, hogy ezek egy zárójelen belül legyenek (pl. a `using ( sin($3) )` a harmadik oszlop szinuszát ábrázolja), különben a *gnuplot* nem fogja értelmezni.

Olyan eset is előfordulhat, hogy nem az első oszlop függvényében akarom ábrázolni egy másik oszlop(ok) adatait, hanem pl. a második oszlop függvényében. Ez akkor fordulhat elő, ha a fájl készítője nem gondolt arra, hogy mi később a méréseit ábrázolni szeretnénk, vagy pedig egyszerűen egy más összefüggést is szeretnénk vizsgálni.



2. ábra Maximum-hőmérséklet és a sugárzás kapcsolata



3. ábra Lissajous-görbe

Ezen problémát megoldatjuk fájlmódosítással (például *awk*-val a második oszlopot első oszlopként kiírjuk), de minek ágyúval lőni verébre, hiszen erre is van megoldás a *gnuplot* berkein belül.
Ha továbbra is a jól bevált adathalmazunkat használjuk, akkor arra is kíváncsiak lehetünk, hogy a napsugárzás és a maximum-hőmérséklet között van-e valami összefüggés. Nyilván azt sejtjük, hogy minél nagyobb az átlagos sugárzás, annál nagyobb a maximum-hőmérséklet. Ahhoz, hogy ezt lássuk is, ábrázoljuk! Először gondoljuk meg azt, hogy ha az első oszlop lenne a napsugárzás és a második a maximum-hőmérséklet, akkor mit is tennénk? Nyilván ugyanazt, mint eddig is. Ha ezt megpróbáljuk (érdemes megnézni!), akkor nem egy szép görbét kapunk, hanem egy „összevisszaságot”, amelynek az az oka, hogy a program az első adatpárnak megfeleltetett pontot köti össze (`with lines`) a második adatpár pontjával, a másodikat a harmadikkal, stb. Ha az *x*-tengely adatai nincsenek a fájlban nagyság szerint rendezve, akkor kapunk ilyen „hibát”. Persze ez sokszögek rajzolására nagyon jó.
Visszatérve a problémánkra, a megoldást a következő *gnuplot*-parancssorozat adja:

```
set title  "Maximum-homerseklet
➥es a sugarzas kapcsolata"
set xlabel "Sugarzas (W/m2)"
set ylabel "Max. homerseklet
➥(C)"
plot "2005-apr.txt" using
➥($6):($3) with impulses
➥title ""
```

A 2. *ábrán* nagyjából igazolva is látjuk „elméletünket”, miszerint több napsugárzás nagyobb maximum-hőmérsékletet eredményez. Nyilván mivel az időjárás nagyon sok mindentől függ, így csak tendenciát állapíthatunk meg.
Már látjuk is, hogy hogyan is kell megadni, hogy mi is a megoldás: a `using` után szereplő `($6):($3)` kifejezés. Tehát egy kettőspont és egy előtte álló `($6)` mondja meg, hogy mi a hatodik oszlop függvényében szeretnénk ábrázolni. Azt is észrevehetjük, hogy a `with lines` helyett `with impulses` szerepel, a már fentebb említett ok miatt, valamint a `title` után egy „üres sztring” következik, ami azt eredményezi, hogy nincs jelmagyarázat.
Az *x*-tengelyt megadó oszlopon is végezhetünk műveleteket, így készíthetünk pl. logaritmikus ábrázolást is. Megjegyzem, erre van egy standard mód is, a `set logscale` parancs.
A következőekben elsősorban a matematikával és fizikával foglalkozók találhatnak hasznos információkat.

Paraméteres görbék és felületek

Először is a paraméteres görbék ábrázolása: azon görbék, amelyek *x* és *y* koordinátája is egy-egy egyváltozós függvény (ilyen például a középiskolában is tanult „egyenes paraméteres egyenlete”), amelynek változóját általában *t*-vel jelöljük. A *gnuplot* is ezt használja (amiről tájékoztat is minket), viszont át is írhatjuk a `set dummy parameter-neve` paranccsal.

Paraméteres ábrázolásba a `set parametric` begépelése után léphetünk, kilépni pedig a `set noparametric`-kel lehet. Egy Lissajous-görbe (egymásra merőleges rezgések együttes „pályája”) előállítása:

```
set parametric
set title "Lissajous-gorbe"
plot [-pi:pi] sin(4*t),cos(3*t)
➥title ""
```

Egy másik érdekes terület a nem-Descartes-koordináták használata. Két dimenzióban (a síkon) a polárkoordinátarendszert szokták még használni (itt is kettő paraméter jellemzi a pont helyzetét: a pont és az origó távolsága valamint az őket összekötő szakasz és a vízszintes által bezárt szög).
A *4. ábrán* látható virágot a következőképp lehet „legyártani”:

```
set polar
set title "Virag"
plot [-pi:pi] 1+cos(5*t)
➥title ""
```

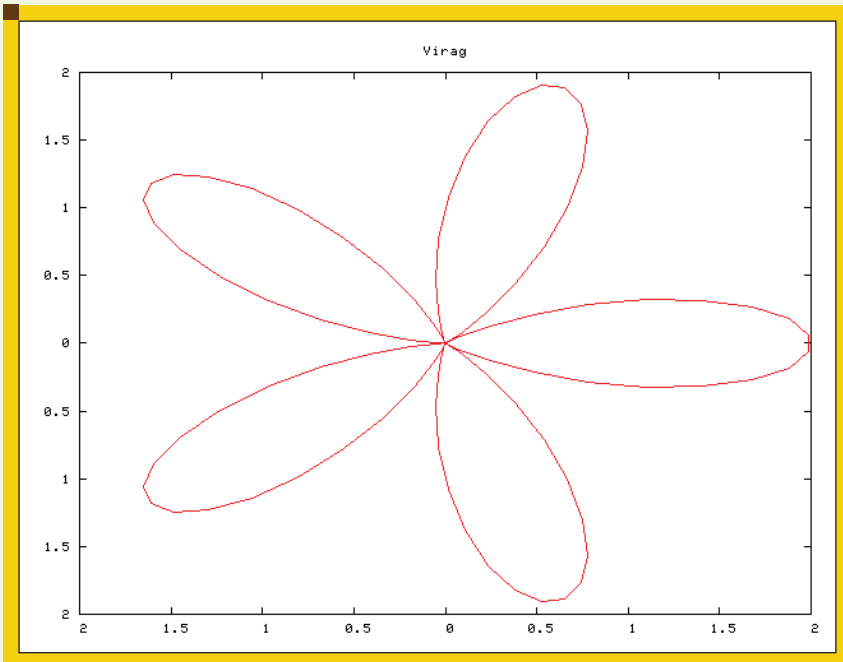
Felületeket az `splot` paranccsal tudunk létrehozni, a változókat ***x***-szel és ***y***-nal jelöljük. A paraméterezése hasonló a `plot` parancséhoz, egy-két újdonsággal bővítve (lásd a `help splot` súgót). Az ***x11*** terminálban még arra is van lehetőségünk, hogy az ábrát az egérrel forgassuk.
Szebb (szemléletesebb) képet ad, ha a ***pm3d*** környezetet állítjuk be:

```
set pm3d
set title "Paraboloid"
splot x**2+y**2 title ""
```

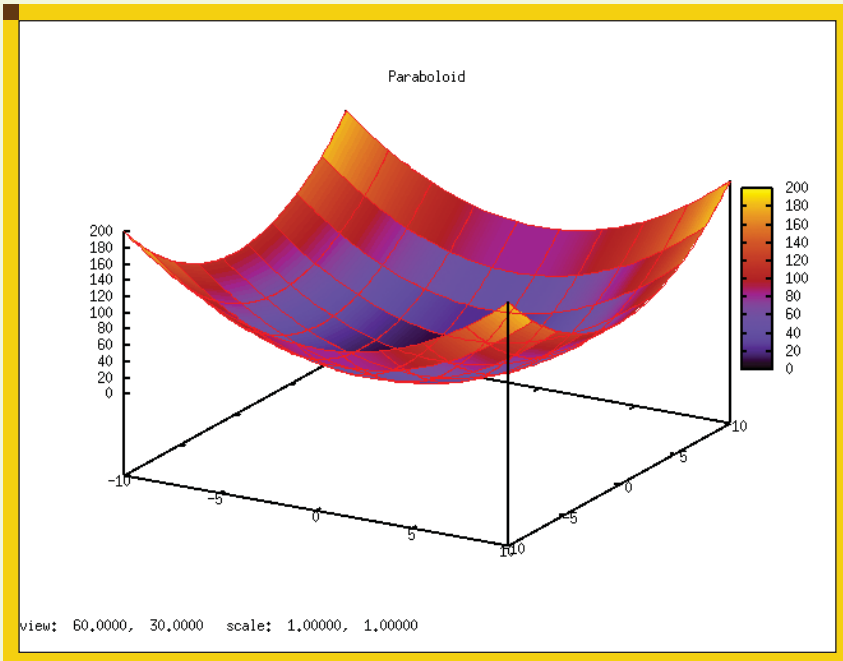
Térbeli polár- (***spherical***) és hengerkoordinátarendszer (***cylindrical***) a `set mapping` paranccsal állítható be. Ezek kezelése is a fentiekhez hasonló.

Zárszó

A ***gnuplot*** képességei a fentiekkel még egyáltalán nem merülnek ki, nagyon sok lehetősége van még benne. Például a vonalak színeit, típusát, az adattartományt, címkék, jelmagyarázatok helyzetét, ... állíthatjuk, sőt, animációk készítésére is alkalmas. Érdemes a honlapján levő és a forráskóddal (vagy a csomagban) szállított demókat és példákat tanulmányozni.
Összességében tehát egy nagyon jól használható, nagyon jól átgondolt eszköz kerül a kezünkbe a ***gnuplot*** személyében. A kiforrottságát és széleskörű használhatóságát mi sem jellemzi jobban, mint az, hogy rengeteg egyéb (grafikus és nem grafikus, új és kevésbé új) program használja, valamint a legújabb, 4.0-ás verziója 2004. április 16-án jelent meg.



4. ábra Virág polárkoordinátákkal



5. ábra Paraboloid

Udvari Zsolt
(udvzsolt@gmail.com)
25 éves vagyok, egy gimnáziumban tanítok matematikát és fizikát. A Linuxszal először 2004 elején találkoztam, az UHU Kamionja volt, ami elgázolt. Azóta 4-5 disztribúciót hosszabban is használtam, jelenleg egy saját építésű LFS-t nyúzok.

RSSH

Biztonság korlátok között

Az ssh igen népszerű alkalmazás, amely egy sokféleképpen használható, titkosított csatornát biztosít 2 gép között. Ez a rugalmasság azonban egy többfelhasználós rendszeren biztonsági problémákat is eredményezhet. Az rssh egy olyan segédprogram, amely ennek megoldásában siet a segítségünkre.

Gyakori kívánság, hogy a felhasználók titkosított csatornán mozgathassák állományaikat a gépek között. Noha több *ftp* kiszolgáló is biztosít *SSL* támogatást, ez bizonyos környezetekben problémát okozhat, például ha 2 tűzfal is áll az útban. Szerencsére az *ssh*, az *sftp* illetve *scp* programok segítségével, egyetlen *TCP* kapu (*port*) használatával biztosítja a titkosított átvitelt.

Egy apró probléma azért marad: amíg az *ftp* kiszolgálók esetében a felhasználók beérik egy másra nem használható héjprogrammal (*shell*), például */bin/false*, addig az *ssh* szolgáltatásainak használatához egy tisztességes *shell*, például */bin/bash* is szükséges. Így azonban, aki például *sftp*-vel be tud jelentkezni, az *ssh*-val is be tud, ill. az *sftp* nem tudja bezárni a felhasználót a saját *home* könyvtárába (*chroot*), ami az *ftp* kiszolgálók alapszolgáltatása. Az *rssh* azonban mindkét problémát orvosolni képes.

A telepítéshez töltsük le a legfrissebb verziót a ➲ *http://www.pizzashack.org/rssh/* oldalról, majd csomagoljuk ki, fordítsuk le, és telepítsük a következő módon:

```
tar zxvf rssh-2.3.2.tar.gz
cd rssh-2.3.2
./configure
make
su -c 'make install'
```

Első lépésben állítsuk be a `bela` nevű felhasználó héjprogramját:

```
usermod -s /usr/local/bin/
➥rssh bela
```

Ezután szerkesszük az *rssh* konfigurációs állományát (*/usr/local/etc/rssh.conf*), és adjuk hozzá a következő sort:

```
user = "bela:022:
➥00010:"
```

A felhasználó név után az *umask* paraméter szerepel, majd annak a meghatározása, hogy Béla milyen módon jelentkezhet be a gépünkre. Ahol 0 szerepel, az a protokoll nem engedélyezett, ahol pedig 1, az igen. Az egyes mezők jelentése a következő:

```
rsync:rdist:cvs:sftp:scp
```

Béla tehát kizárólag *sftp*-vel jelentkezhet be. Ha ezután mégis *ssh*-val próbál meg bejelentkezni, akkor egy ehhez hasonló üzenetet kap:

```
$ssh -l bela szerver.ceg.hu
bela@szerver.ceg.hu's password:

This account is restricted by
➥rssh.
Allowed commands: sftp

If you believe this is in
➥error, please contact your
➥system administrator.

Connection to szerver.ceg.hu
➥closed.
```

Béla arról kapott tájékoztatást, hogy az ő felhasználói fiókját az *rssh* korlátozta le úgy, hogy csak az sftp program használata engedélyezett. Jogorvoslatért pedig a rendszergazdához

fordulhat. A naplóban pedig az alábbi üzenet jelent meg:

```
Nov 24 11:31:55 plutonium
➥rssh[2128]: user bela attempted
➥to log in with a shell
```

Lépjünk be most *sftp*-vel:

```
$sftp bela@szerver.ceg.hu
Connecting to szerver.ceg.hu...
bela@szerver.ceg.hu's password:
sftp>
```

A naplóban ezúttal az alábbi listában látható üzenet jelent meg, amely arról tájékoztat, hogy Béla ezúttal *sftp*-vel jelentkezett be:

```
Nov 24 11:32:06 plutonium
➥sshd[2138]: subsystem request
➥for sftp
Nov 24 11:32:06 plutonium
➥rssh[2139]: setting log
➥facility to LOG_USER
Nov 24 11:32:06 plutonium
➥rssh[2139]: setting umask to
➥022
Nov 24 11:32:06 plutonium
➥rssh[2139]: line 52:
➥configuring user bela
Nov 24 11:32:06 plutonium
➥rssh[2139]: setting bela's
➥umask to 022
Nov 24 11:32:06 plutonium
➥rssh[2139]: allowing sftp to
➥user bela
```

A következő listában szereplő konfigurációs részlet segítségével házirendet (*policy*) alakíthatunk ki. Az alapértelmezett házirend szerint semmit sem engedünk meg a felhasználóknak.

```
#allowscp
#allowsftp
#allowcvs
#allowrdist
#allowrsync
```

Ezután az *rssh.conf* állományban definiáljuk, hogy az egyes felhasználók, milyen módon jelentkezhetnek be. Én hasznosnak tartom azt, ha a legtöbb felhasználó által használt módokat engedélyezzük, és csak a megkülönböztetett felhasználókat definiáljuk külön. Néhány példa a következő listában látható:

```
user=bela:011:00100:   # csak
➥cvs
user=geza:011:01000:   # csak
➥rdist
user=joska:011:10000:  # csak
➥rsync
```

Chroot

Az *rssh* arra is lehetőséget ad, hogy a felhasználókat bezárjuk egy adott könyvtárba. Az eljárás nem egyszerű, mert egy komplett futtató környezetet kell felépíteni hozzá. *RedHat* alapú rendszerek esetén használhatjuk ehhez a `mkchroot.sh` héjprogramot, más *Linux* disztribúciók esetén ezt magunknak kell megtennünk.
Az *rssh* ehhez a funkcióhoz az `rssh_chroot_helper` programot használja. Ez úgy működik, hogy ez utóbbi kiolvassa a konfigurációs állományból az adott felhasználóhoz rendelt *chroot* könyvtárt, majd végrehajtja a *chroot()* függvényt, majd az *execv()* rendszerhívással futtatja az adott *ssh* szolgáltatáshoz tartozó démont, például az *sftp-server*-t.
Az *rssh* alkalmas az összes támogatott szolgáltatás (*rdist, rsync, cvs, sftp, scp*) ketrecbe zárására. Adott esetben szükség lehet, hogy az *ssh*-val bejelentkezett felhasználókat is bezárjuk. Ezt azonban nem támogatja az *rssh*. Ez legkönnyebben (?) az *sshd* démonon belül lehetne megvalósítható, esetleg egy másik segédprogram segítségével. Növelhető úgy is a biztonság, ha az *rssh*-val korlátozott felhasználókat egy külön csoportba tesszük (például *rssh*), és a csoportnak csak a feltétlen szükséges binárisokhoz adunk hozzáférést. Ez azonban a kezdeti beállításokon túl utólagos is meglehetősen munkaigényes, például a frissítések után kézzel, esetleg egy erre a célra írt héjprogrammal, kell a jogosultságokat ismét beállítani. A *chroot* környezet kialakításához szükséges részletes tudnivalókat a *CHROOT* állományban találjuk meg. Sok sikert kívánok ehhez, nekem sajnos ez nem sikerült. A feladat nehézségét az is jelzi, hogy a levelező listán a legtöbb kérdés ezzel a funkcióval kapcsolatos.

A jövő

A program írója szerint az *rssh* készen van, nem várható a funkciók további bővülése, legfeljebb a napvilágra került hibákat (*bug*) javítja ki. A biztonságot azzal is növelni lehetne még, ha az *rssh_chroot_helper* program kriptográfiailag leellenőrizné, hogy valóban az *rssh* program hívta meg. A program írója ennek hiányát ilyen irányú tapasztalatainak elégtelen voltával magyarázta meg.
A mai heterogén platformok világában probléma lehet, hogy a **BSD* platformokon hiányzik a *wordexp()* függvény – pedig ez *POSIX.2* része – amely a parancssori argumentumok növelésére használható. A program írója azonban beleunt abba, hogy *BSD* rendszerekre megírja ezt a funkciót. Így aki *Linux* mellett például *FreeBSD* rendszereket is üzemeltet, annak be kell szereznie ezt a függvényt, mondjuk a *glibc2* implementációból. További nehézség lehet az, ha az *sftp* kapcsolatot *Windows* platformról is igénybe akarják venni az ügyfelek, mert nem mindegyik alkalmazást támogatja az *rssh*. A program írója szerint a *WinSCP* javított, és a legutolsó változata már képes vele együttműködni. A szerző egyébként a *FileZilla* ill. a *SecureFX* programokat javasolja.

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

NVClock – nVidia tuning Linuxon

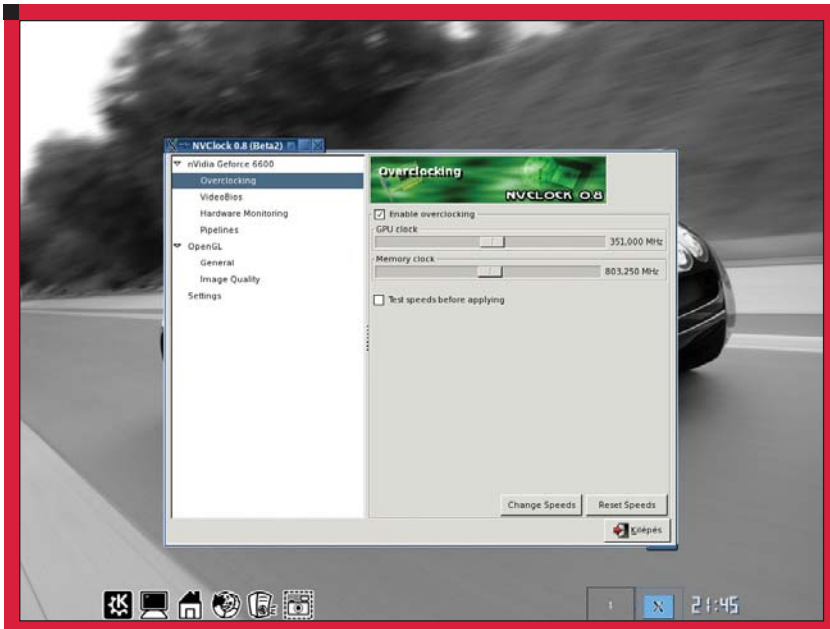
A Linux-felhasználók számítógépeiben általában nVidia grafikus hardver lapul. Nem véletlenül, hiszen a favorit gyártó köztudottan a konkurencia előtt jár, a számunkra fontos támogatás terén. A 3D eszközeik finomhangolásról és a túlhajtásról azonban méltánytalanul kevés szó esik...

nVidia, és megint csak nVidia

Ha valaki komolyabb *3D* grafikus munkát végez *Linuxon*, vagy éppen *OpenGL* programozási felületre támaszkodó játékokkal játszik, akkor bizonyára egyetért a bevezetővel. Akár alaplapi csipkészletbe égetett grafikus vezérlőről, akár *PCI-Express* / *AGP* sínbe helyezett kártyáról legyen szó, az *nVidia* hardvereit nagyjából két perc alatt rá lehet bírni a gyári meghajtók használatára – ezáltal a népszerű *3D* leírónyelv ismeretére is. Nem mintha a konkurencia képtelen lenne kielégítő támogatást nyújtani megoldásaihoz, de a cikk írásakor összességében (hatékonyság, kényelem és kompatibilitás terén) én is mérhetőnek érzem az említett gyártó linuxos népszerűségi fölényét. Szerencsére a *GeForce* osztályú grafikus hardvereket nem kell különösebben bemutatnom az Olvasónak. Hiszen az ide vonatkozó szilíciumtechnikai újdonságokat az *Internet* segítségével bárki könnyedén nyomon követheti, a telepítés és a használatba vétel témája pedig már többször szerepelt e magazin hasábjain is. Az újabb meghajtók installálási módja pontosan egyezik az előző generációban megismertekkel, a kódok változási listája pedig a gyártó honlapján elég jól van dokumentálva. Ezért ezt a részt hanyagolni fogom. Ellenben a cikk teljes megértéséhez minimális szakmai ismeretet fogok feltételezni.

A feladat felelőssége

Szinte biztosan akad olyan érdeklődő, aki már összefutott a következő jelenséggel: egy bizonyos szoftver (leginkább egy nagy igényű játékprogram) adott számítógépen nem fut kielégítő sebességgel. Ennek a kellemetlen helyzetnek több oka is lehet, kezdve a *CPU* gyengeségétől, a memória szerény méretén át egészen valamely szűkös „adat-keresztmetszetig". Természetesen az okok között szerepel a grafikus vezérlő elégtelensége is, ami a cikk szempontjából kiemelt fontosságú. Hogy miért? Utóbbi esetben a tehetősebbek azonnal megrohanják a boltok polcait (egy modernebb darabért), mások belenyugszanak a lehangoló helyzetbe, de vannak, akik különböző trükköket vetnek be orvoslásképpen...

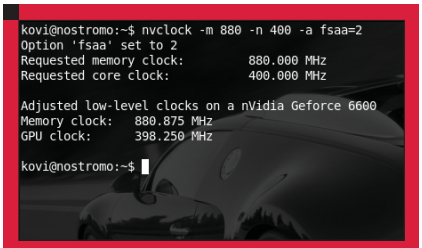


1. ábra A GTK alapú megoldás, KDE asztalon

Ezek a „trükkök" jellemzően arra épülnek, hogy a minőségi munkákról elhíresült, neves kártyagyártók termékeiben sokszor kihasználatlan (de kihasználható) teljesítménytöbblet lapul. A feladat adott, csalogassuk hát elő a kiaknázatlan lehetőségeket! Mielőtt azonban belevágnék a téma ismertetésébe, két dolgot le kell szögeznem. Az előbb írt teljesítménytöbblettől nem szabad csodát várni. Ha egy program a hardver alapvető képességeinek hiányában nem élvezhető, akkor nincs értelme sem finomhangolással, sem pedig tuninggal próbálkozni. A másik dolog amire ki kell térjek, annak felelősségteljes hozzáállásra és ésszerű megoldások keresésére kell ösztönöznie az

1. táblázat *Az NVClock használata*

`nvclock -m`	A kártyán lévő memória órajelének beállítása (MHz)
`nvclock -n`	A grafikus mag sebességének beállítása (MHz)
`nvclock -r`	Az eredeti értékek visszaállítása (BIOS alapján)
`nvclock -s`	A jelenleg érvényes órajelek megjelenítése
`nvclock -d`	Nyomkövetési (debug) információk visszajelzésének kapcsolása
`nvclock -c`	Az állítandó kártya száma (többkártyás rendszereknél)
`nvclock -f`	Kényszerített (force) mód, vagyis a hangolás kikényszerítése
`nvclock -F`	A hűtőventilátor szabályozása (10-100%)
`nvclock -P`	Az extra pixel-csatornák bekapcsolása (ha van)
`nvclock -Q`	A PCI buszazonosító felülbírálása
`nvclock -S`	A smartdimmer fényerő szabályozása (15-100%)
`nvclock -T`	A grafikus mag hőmérsékletének kijelzése
`nvclock -i`	A hardver részletes információinak kijelzése
`nvclock -a`	Az nVidia meghajtó opcióinak beállítása (pl. FSAA)
`nvclock -q`	Adott nVidia opció lehetséges értékeinek kijelzése
`nvclock -l`	Használható nVidia / OpenGL opciók kijelzése
`nvclock -x`	Másik X megjelenítőre váltás



2. ábra A konzolos verzió használata

Olvasót. Arról van szó, hogy egy olyan videokártyát, melynek a központi egységét nem ritkán 20 millió tranzisztor építi fel, oktalan és szakszerűtlen „beavatkozással" pillanatok alatt a másvilágra lehet küldeni. Márpedig a leégett, elfüstölt, vagy éppen csak elszíneződött alkatrészek a garanciális ügyintézést (szinte) minden üzletben kizárják. Röviden összefoglalva az intelmeket: a továbbiakban leírt dolgokkal csak ésszerű keretek között, saját lehetőségeihez mérten és várható eredmények reményében próbálkozzon bárki.

Az NVClock felélesztése

A *Win32* környezet felhasználói bőségesen el vannak látva mindazokkal az eszközökkel, amivel egy *nVidia* grafikus hardver teljesítménye emelhető. A megoldások számát tekintve *Linux* alatt már kevésbé jó az ellátottság, szerencsére azonban akad olyan Szabad projekt, mely letisztult, lényegre törő és hatékony – ezzel el is jutottunk az *NVClock*-ig... A szoftver honlapja a ➲ *http://www.linuxhardware.org/nvclock/* URL mögött található, innen érhető el a *GPL* licenc szerint terjesztett forráskódja.
Töltsük le az *nvclock_verzió.tar.gz* tarballt, majd a kicsomagolt archívban adjuk ki a szokásos

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

parancsokat (az utolsót természetesen *root* jogkörrel kell megtenni). Ennek hatására felépülnek és helyükre (*/usr/bin*) kerülnek az *ELF* binárisok, melyek közül az egyik a konzolos futásért, a másik a *GTK* alapú *GUI*-val ellátott használatért felel. A szöveges felület hívei ezután a felhasználóként kiadott

`nvclock -`

a hangolás grafikus lehetőségét szorgalmazók pedig az

`nvclock_gtk`

paranccsal juthatnak közelebb a kész megoldáshoz (feltéve, hogy a konfiguráló szkript talált használható *GTK* könyvtárat). Fontos, hogy az *NVClock* teljes körű funkcionalitása egy használható (induláskor betöltött) gyári *nVidia* modult feltételez a rendszerünkben!

És a gyakorlat...

Akár a grafikus interfésszel ellátott verziót, akár a terminálon használható megoldást választja az érdeklődő, nagyon egyszerű használatot kell észben tartania. A program három célt szolgál: először is a grafikus hardverről ad részletes információkat, másodszor lehetőséget biztosít ennek finomhangolására, végül pedig (de nem utolsó sorban) a túlhajtásra fókuszál. A fő opciókat a konzolos megoldásnál úgy lehet életre hívni, hogy az `nvclock` parancsot felparaméterezzük a mellékelt táblázat szerint. A *GTK* alapú ablakos megoldás nem szorul különösebb magyarázatra, az ide kapcsolódó ábrából reményeim szerint látszik a projekt lényegre törő felépítése.
A program nem képes végleges változásokat kieszközölni, ezek a beavatkozások továbbra is csak a *ROM BIOS* módosításával érhetőek el. Ennek ellenére bizonyos értelemben alkothatunk maradandót, hiszen az *NVClock* a *Linux* indulásakor automatikusan betölthető (a megfelelő *init* szintek módosításával, vagy esetemben akár a *KDE* indító szolgáltatásával). A beállított paraméterek mindig a megfelelő */home/$/.nvclock/config* állományban vannak lerögzítve, tehát a kívánt értékek az aktuális felhasználói profiltól is függhetnek. E fájlok kézi szerkesztésétől óva intek mindenkit: bizonyos sorai-ban, a nehezen értelmezhető értékek megváltoztatásával könnyen előidézhető az adott hardver totális leégése.
Ezek után nézzük az életszerű példát! Nekem egy *Asus* márkájú, *Top Silencer* videokártyám van: ez annyit tesz, hogy a nyák ellentétes oldalára került a grafikus processzor (esetemben egy középszerű *GeForce 6600*). A mag így kevésbé melegszik, hiszen (álló házba építve) a termelt hő felfelé szabadon távozhat. Az sem mellékes, hogy ez a típusú *6600*-as amúgy sem melegedős, a ventilátor helyett egy hatalmas radiátor felel a hangtalan hűtésért. Az *Asus* konzervatív és profi politikájára jellemzően a válogatott grafikus mag gyári órajelen fut, mint ahogyan a felhasznált memória is hivatalos ajánlás alapján van időzítve – miközben

nagyobb sebességen is hibátlanul működik. Kiadom tehát egy konzolon az

```
nvclock -m 880 -n 400 -a fsaa=2
```

parancsot, ami által a grafikus mag sebességét (a memóriához hasonlóan) a gyári értékek fölé emelem 15%-al. Mindemellett a példában lebutítom a teljes képernyős élsimítás képességét is (a mintavételezési érték rögzítésével), így a *3D* szoftverből történő automatikus vezérlésnek is keresztbe teszek. Az eredmény: közel 10 *Celsius* fokkal melegebb *GPU*, közel 15%-al gyorsabban futó *Unreal Tournament 2004* (ebben a 15%-ban természetesen érződik az *FSAA* kapcsolása is, mely 1024x768 felbontásban nagyjából 5 százalékot segít nekem).
Persze játék végén *nvclock -r* paranccsal visszaállítok mindent az eredeti állapotba. Ha a *GTK* alapú megoldást választom, ez a teljes folyamat öt kattintás árán kivitelezhető...

Figyelem!

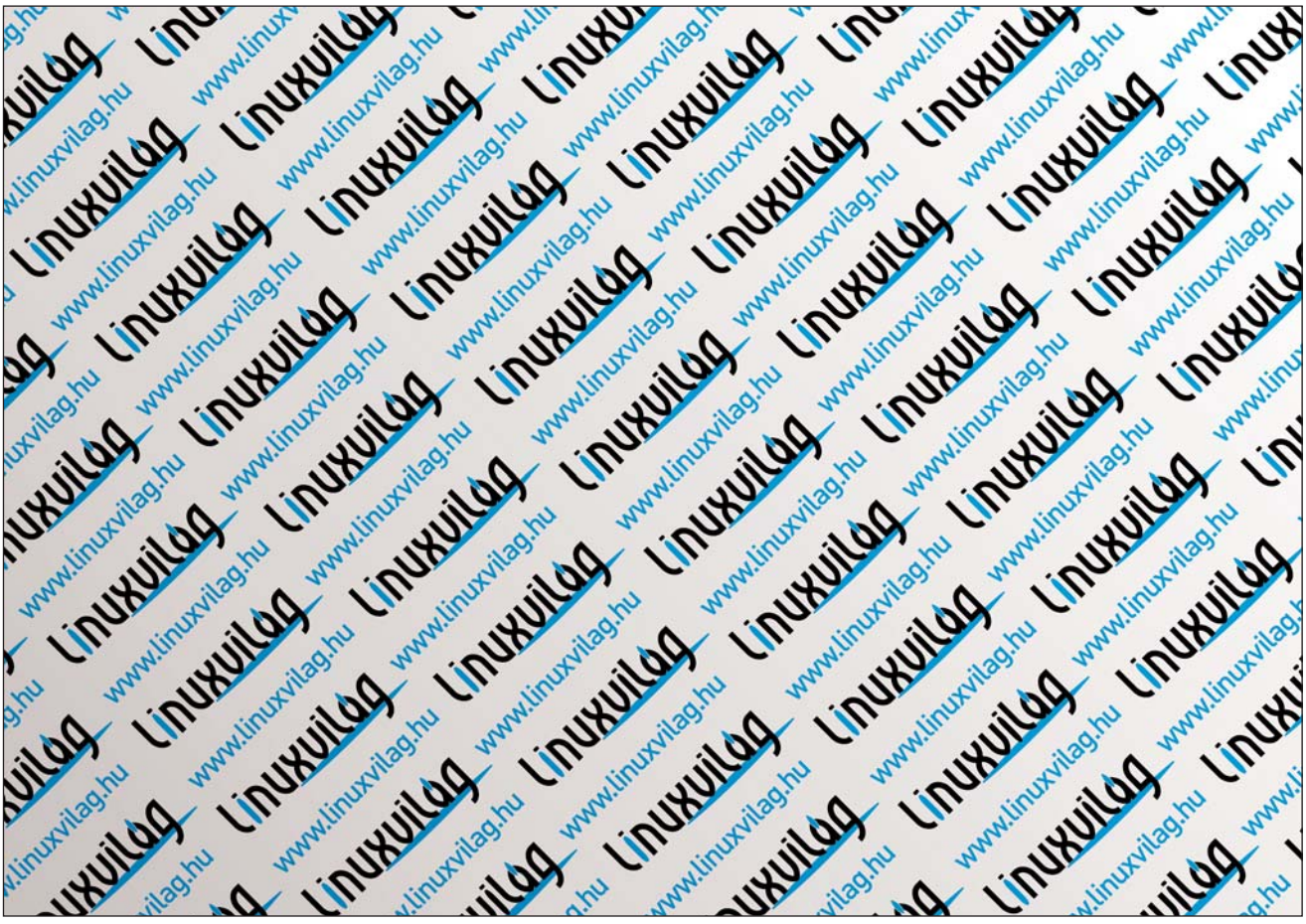
Nem győzöm hangsúlyozni: minden változtatás kiemelt figyelemmel hajtandó végre! Egy csendesre állított (alacsony fordulatú) ventilátor, egy túl gyorsra állított mag pillanatok alatt komoly anyagi kárt képes okozni. Hiszen a *GPU* tranzisztoraiból, vagy a memória celláiból csak egynek kell tönkremennie, és máris kész a baj.
Szerényebb minőségű hardvereken inkább ne eszközöljünk semmilyen változtatást, hiszen egy (válogatás útján) gyengébbnek ítélt grafikus mag elemei általában nem viselik zokszó nélkül a gyorsabb nyitástzárást. Abban az esetben lehet érdekes az ismertetett program, ha éppen annyi teljesítménytöbblet hiányzik egy játék élvezhető futásához, mint amennyi kisajtolható egy jobb fajta *nVidia* alapú kártyából – ésszerű keretek között. Mindemellett az esetlegesen bekövetkezett balesetekért a felhasználó az egyetlen felelős: sem a gyártók, sem pedig az *NVClock* fejlesztői nem vállalják fel ezt a „tisztséget” (teljesen érthető módon én sem tarthatom a hátam az említett példa személyre szabott sikeréért). Így a program *README* állományát minden leendő használó olvassa el.

Vélemények

A túlhúzás mértékéről és létjogosultságáról megoszlanak a vélemények. Van, aki igazi megszállottként próbálja sebesebb tempóra kényszeríteni hardverét, de akad olyan is, akit kiráz a hideg ettől a dologtól. Nem célom állást foglalni a kérdésben, de a magam óvatos megközelítésével sokkal inkább a középutat keresem: ritkán hajtom túl a (lassan kiöregedő) grafikus kártyámat. Egy dologgal azonban már én sem vagyok kibékülve: az előbb említett első csoportból néhányan sportot űznek ebből a „játékból”, és akkor is tuningra adják a fejüket, ha igazából nincs is szükségük rá.
Ha valaki legalább ezt az egy kapitális hibát igyekszik elkerülni, akkor már meg is tette az első lépést az *NVClock* értelmes használata felé.
Sikeres, és főként „balesetmentes” próbákat mindenkinek!

Kovács Zsolt (kovi@linuxforum.hu)

Quake fanatikus. Négy éve a debreceni linuxosok egyike. Töretlenül hisz a Slackware terjesztésben.



TSClient – Kattintás egy távoli asztalon

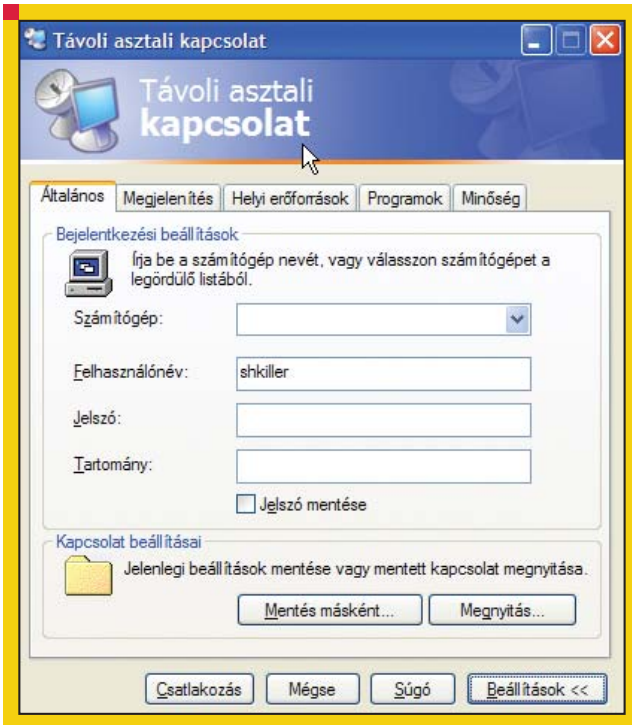
Egyszerű, könnyen kezelhető grafikus előtét rdesktop, VNC, és egyéb távoli asztal kezelő programokhoz.

Távoli munkaasztal eléréséhez jobbnál jobb nyílt forrású alkalmazásokat (például *RealVNC, rdesktop*) találhatunk az Interneten, melyek általában nagyon jó minőségű, megbízható szoftverek, igaz többségük használata nehézkes, használatuk több parancssori kapcsoló ismeretét is igényli. „Bezzeg" a *Windows*-os kollégáknak ott a *„Távoli asztal elérése"*... Szerencsére a *Linux*-os felhasználók sem maradnak *„Click & Connect"* élmény nélkül, hiszen segítségül hívhatják munkájuk megkönnyítésére a *tsclient*-et, mellyel grafikusan állíthatjuk be *rdesktop* illetve *VNC* kapcsolatainkat, mindezt a *Windows* környezetben megszokott módon. Funkcióiban, kinézetében szinte teljesen megegyezik *Windows*-os „testvérével".

Csatlakozzunk, de hova, hogyan?

Ez az első dolog, amit a program indításakor el kell döntenünk. Hogyan, hova és milyen felhasználóként akarunk csatlakozni. A *„Számítógép"* mezőbe az elérni kívánt távoli gép nevét kell megadni. A *„Protokoll"* legördülő menüből választhatjuk ki a használni kívánt kapcsolat típusát, melyek a következők lehetnek:

- *RDP*: A jó öreg *Remote Desktop Protocoll*, segítségével *Windows Terminál Szerverekhez* kapcsolódhatunk. Ezt használva az *rdesktop* program hozza létre a kapcsolatot.
- *RDPv5*: *TCP* alapú, *Windows 2000*-től támogatott *RDP* protokoll. Használatakor szintén az *rdesktop* program indul el a háttérben.
- *VNC*: *Virtual Network Computing* protokollt kiválasztva *VNC* szervert futtató kiszolgálóhoz csatlakozhatunk. A *„Csatlakozás"* gombra kattintva egy `vncviewer`-t próbál elindítani, vagyis nincs meghatározva a *VNC* kliens típusa, lényeg, hogy a futtatandó program elérhető legyen. Támogatott *VNC*



1. ábra Windows „Távoli asztal elérése..."

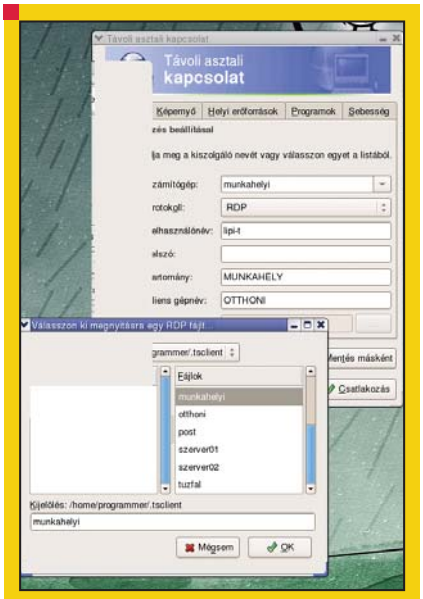
2. ábra ...és a letisztult GTK+ megfelelője

parancssori kapcsolók, melyeket a *tsclient* képes átadni: `-fullscreen`, `-geometry`, `-depth` and `-viewonly`.

- *XDMCP*: *„X Display Manager Control Protocol”*, *X-Terminal* szerver eléréséhez használatos. Az alkalmazások futtatása itt is kiszolgáló oldalon történik.

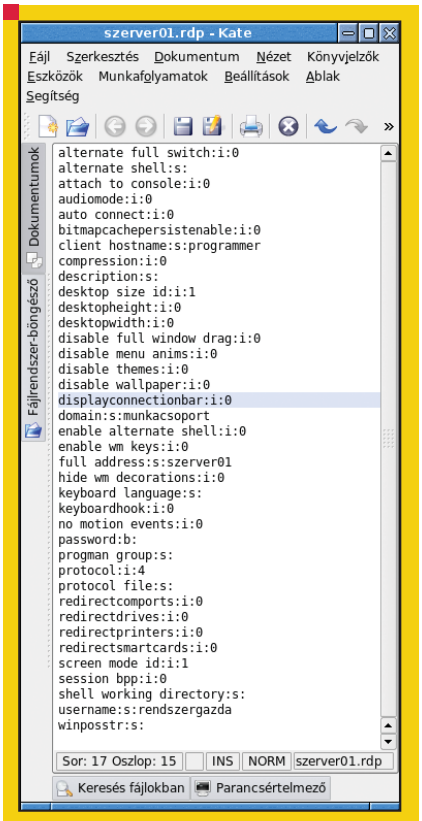
- *ICA*: *Citrix* 1995-ben kiadott protokollja. Segítségével a „vastag” kliensek is „vékony” kliensként képesek működni, hiszen a az alkalmazások futtatását teljes egészében a kiszolgáló végzi. Alacsony sávszélességet igényel, *Unix* és *Windows* rendszereken egyaránt elérhető.

„Felhasználó” és a *„Jelszó”* bevitelnél értelemszerűen a használni kívánt felhasználói nevet és a hozzátartozó jelszavat adhatjuk meg (*VNC* kapcsolat esetén csak azonosítást megkövetelő kapcsolatnál van értelme a felhasználói név megadásának). *„Domén”* és a *„Kliens név”* adatai *RDP* kapcsolatok esetén lehetnek hasznosak (Megkövetelt domén és kliensnév esetén). A *„Megnyitás”* és a *„Mentés”* gombok hamar a barátaink lesznek, segítségükkel előre beállított profiljainkat tudjuk kezelni, későbbi felhasználás céljából.
Alapértelmezetten minden profil a `~/.tsclient` mappába van mentve .rdp kiterjesztéssel, szöveges konfigurációs állományként, így akár egy szövegszerkesztővel is módosíthatjuk beállításainkat.
A *„Képernyő”* fülön az asztal méretét, illetve színmélységét állíthatjuk, illetve a teljes képernyős módot is engedélyezhetjük *VNC* és *RDP* kapcsolat esetén is. *„A (-F) kapcsoló használata teljes képernyőhöz”* opciót engedélyezzük, ha az *„unified patch”*-et is használjuk *rdesktop*-unkhoz (speciális billentyűzetek esetén használatos ez a frissítés). A „Helyi erőforrások” fülön a hangkezelés módját szabályozhatjuk (helyi kimenet/távoli kimenet/hang tiltása), illetve a használni kívánt billentyűkiosztás kódját adhatjuk meg (en, hu). Ezen beállítások csak az *rdesktop*-ra hatnak.
A *„Programok”* fülön az *RDP* kapcsolódáskor futtatandó program neve, illetve a munka könyvtára állítható be.



3. ábra Profilok kezelése gyorsítja és egyszerűsíti munkánkat

„Sebesség” fülön *RDP* kapcsolataink minőségét és sebességét befolyásolhatjuk. A *„Bitmap gyorsítótár engedélyezése”* hatására az *rdesktop* helyi gyorstárat hoz létre, mely a grafikus objektumok kinézetét tárolja. *„Mozgási események tiltása”*-kor *RDP* kapcsolat esetén nem kerül átvitelre az egérmutató mozgása (sávszélességet spórolva), míg *VNC* esetén ez az opció megfelel a *„View only”*, csak nézet beállításnak.
Az *„Ablakkezelő billentyű hozzárendeléseinek engedélyezése”* eltakarja a kliens oldalon futó ablakkezelő billentyű kombinációit, így a gyorsbillentyűket a kliens oldal értelmezi (*ALT+TAB*). *„Ablakkezelő dekorációinak elrejtése”*-t bepipálva nem jelenik meg az ablakkeret a távoli asztal körül (az *rdesktop* egy hasznos tulajdonsága, ha saját fejlesztésű programból szeretnénk indítani azt).
Az *„Kapcsolódás a konzolhoz”* csak *RDP* kapcsolat és *Windows 2003 Server* vagy újabb verziójú operációs rendszerek esetén működik. Hatására képesek leszünk kapcsolódni a kiszolgáló valós munkaasztalára. Mivel alkotói eleve *Gnome* alá fejlesztették, ezért nem meglepő, hogy ebben a környezetben kisalkalmazásként (*applet*) is használhatjuk, megkönnyítve a *tsclient* indítását. Természetesen más grafikus rendszerekből is használhatjuk. Én például *KDE* alatt próbáltam ki.



4. ábra A profil fájl sok egyéb érdekességet is tartalmaz, melyeket közvetlenül nem állíthatunk

Végeredmény

Nagyon jó és hálás barátja ez a programocska mindazoknak a felhasználóknak, akik munkájuk során többször is rákényszerülnek arra, hogy távoli számítógépeken is dolgozzanak, viszont a parancssor túl rideg és bonyolult számukra, bár összetett kapcsolatok létesítésére továbbra is marad a kézzel finomított parancssori kapcsolók véget nem érő sora.

Gráma Tibor
(tibor.grama@hoya.lmh.hu)

1997 óta „Linuxozik”, UHU hívő. Szabadidejében gyermekeivel és vizsla kutyáival játszik, ha éppen nem kertészkedik vagy horgászik.

KAPCSOLÓDÓ CÍMEK

- http://www.gnomepro.com/tsclient/

Wormux

Kukacok után, szabadon

A kilencvenes évek derekán járunk. A játékpiac jeles szereplőjévé avanzsált Worms sorozat rövid idő alatt kultuszt épít maga köré: oldalnézeti stratégiai játékokról van szó, ahol tetszetős pályákon, felfegyverzett kukacokkal kell felülkerekedni az ellenfél csapatán.

Nosztalgikus érzéssel gondolok vissza ezekre az időkre... Régen volt, jó volt – és ugyan régen elmúlt, bennem is mély nyomot hagyott a (kezdetben) különcnek ható sorozat. Nem mondhatnám, hogy sok időt töltöttem bármely epizód előtt, de összességében néhány száz órát én is „kukacoskodtam" a számítógép, vagy valamelyik cimborám ellenében. A program varázsa a rajzfilmszerű grafikában, a taktikus küzdelem vicces megközelítésében egyaránt megnyilvánult – csakúgy, mint a remek hanghatásokban. Aztán a kukacok sajnos átköltöztek a divatos *3D* világba, ahol engem speciel már nem tudtak úgy elkápráztatni, mint az elődeik. De természetesen nem csak én vagyok jó véleménnyel a sorozat kezdeti darabjairól, így világszerte százezernyi rajongó lelheti örömét a „kukacháború" Szabad klónjában – és ezzel el is jutottunk a *Wormux* projekthez.

Tudni illik...

A *Wormux* alapkódja *Lawrence Azzoug* csapatának munkája (a fejlesztésben egyébként elég sokan érintettek, tekintettel a *GPL* licenc lehetőségeire). A program fejlettsége még nem érte el a stabil szintet, ennek ellenére komolyabb problémáktól mentesen használható. A cikk írásakor fellelhető legfrissebb kiadást *v0.8alfa1* azonosítóval jelölik. A *Wormux* egyébiránt elérhető *Linux* és *Win32* felületre is (forráskód és bináris formában egyaránt) – a különböző platformra készített csomagok pedig képesek együttműködni, így a többjátékos mód vegyes hálózatokban is használható.
Apropó, több játékos mód! Természetesen ez a lehetőség egyetlen számítógép előtt ülve is kihasználható – egészen négy csoport/gépig. E remek kikapcsolódás bárkinek javasolt, akit vonz a kategória varázsa: a *Worms* játékok hangulata, az ott megismert lehetőségek még hét évvel az ezredforduló után is képesek tömegeket csábítani a képernyők elé! Talán fontos lehet, hogy a játékosok itt nem kukacokat irányítanak, hanem *Linux-pingvineket*, *BSD-démonokat*, *Emacs-ökröket*, *Gimp-rókákat*, *OOo-madarakat*, és így tovább...

A kontroll maradt a régi, jól bevált módszernél: adott csapat tagjait a kurzorbillentyűkkel mozgathatjuk jobbra-balra. A kívánt helyre érve, majd jobb egérklikket nyomva felbukkan a használható fegyverek, eszközök listája. Innen kiválasztva a szükségeset, a fel-le gombokkal tudjuk irányba állítani a célkeresztet. Végül csak az marad hátra, hogy a *„Space"* megnyomásával aktiváljuk a kívánt eszközhöz tartozó tevékenységet.
A friss kiadásban 27 csodaszép pályát, 13 választható csapatot programoztak nekünk, közel húsz fegyverrel – nem beszélve a „ninja" csáklyáról, *JetPackról*, légkalapácsról: egyhamar nem lehet megunni a küzdelmeket...

Akarom!

A szükséges állományok a ➲ http://www.wormux.org honlapról indulva érhetőek el. Itt a fő disztribúciókhoz kapcsolható *rpm*, *deb*, *tgz* bináris csomagok mellett a forráskódok is beszerezhetőek. Nekem sajnos nincsenek jó tapasztalataim az előre fordított verziókkal (sercegő hang, szemetelő kép, hibaüzenetek

a konzolon), így ajánlásom szerint inkább mi magunk építsük fel a játékot! Töltsük le a *wormux_verzió.tar.gz* tarballt, majd a kicsomagolt forrás archívban adjuk ki *root* jogkörrel a `./configure`, `make`, `make install` parancsokat! (Ahhoz, hogy a játékot kompromisszumoktól mentesen használhassuk, szükségünk lesz a rendszerünkben lévő *libSDL*, *SDL_net*, *SDL_mixer*, *SDL_ttf* komponensekre, valamint a *libXML* csomagokra: ezek jelenléte nélkül a konfiguráló szkript le sem fut!). Ha minden fájl az alapértelmezett helyére került, felhasználóként kiadott `wormux` paranccsal indítható a móka. Ugye, milyen egyszerű?

1. ábra Íme a Wormux, KDE asztalon

Egy csata története

Azért, hogy betekintést nyerjünk ebbe a világba, ismerjük meg egy két csapatos háború rövid lefolyását! Engedelmetekkel felveszem a kommentátor szerepét:
„Kedves Közönség, előttem a pálya, mindkét félnek három egysége van készenlétben. Egyikük *Wilber* osztályú (a *Gimp-rókák*), a másik *Tuxokból* áll (*Linux-pingvinek*). Megkezdődik a mérkőzés, *Wilber* vezér indít: az első rókáját megpróbálja olyan helyzetbe mozgatni, hogy az ellenfél ne lásson rá egykönnyen, majd egy jól irányzott aknavetős lövéssel puhítani kezdi a rossz helyen lévő csapattársa alatti

2. ábra A pályán épp sajt szállingózik...

3. ábra Az ellenfelem fegyvert választ

4. ábra Győzelmi statisztika

pályaelemet. A másik csapat ezt észlelve megtámadja a rókát, melynek segíteni próbál az aknavetős egység: távcsöves puskával eresztenek két lövést a szerencsétlenbe, melyből az egyik célt téveszt. Ezzel a tévesztéssel azonban meg is pecsételődik a madár sorsa! A sértett fél egy automata ***Bazookával*** válaszolva kivégzi a pingvint: úgy ér célt a lövedéke, hogy a sarki-madár komolyan megsérül, ráadásul felrobban a pályaelem alatta, így a mélybe zuhan. A megfogyatkozott „szárnyas-csapat" felszívja magát, és ***JetPackkel*** repít egy egységet közvetlenül a gyilkos mellé. A harmadik róka eközben olyan helyzetet keres, hogy szabadon rálásson a két megmaradt madárra – nincs könnyű helyzetben, az egyiküknek komoly fedezéke van. Az előbb gyilkossá vált ragadozó mellé repült pingvin dühében előránt a zsebéből egy görényt, és rászabadítja ellenfelére. Micsoda bosszú! Zöld füst gomolyog, szinte a lelátókig terjeng a bűz! A róka nem bírja sokáig a szagokat, a másvilágra költözik: már csak a sírköve tudatja egykori létét. Maradt tehát két róka, két pingvin. A pingvinek tudják, hogy még semmi sem dőlt el: egyikük aknát telepít maga elé. Itt most ideig biztonságban lesz: nem lehet rálátni, mert a pálya közepén egy éktelenül nagy tehén leledzik. Az egyik ***Wilber*** osztagos a ***JetPackért*** nyúl, és türelmetlenül átszáll a tehénen... de elrontja a landolást! A földre zuhan, a megsérült ravaszdit pedig kivégzi az előbb letett taposóakna. Az állás tehát egy-kettő."

De most... recsegés zavarja meg a közvetítést. Mi ez a hang? Hová lett a kommentátor?

„Tisztelt Közönség! A hátralévő közvetítést megszakítjuk. A közvetítő egyszerűen letette a mikrofont, és szinte fejvesztve szaladt haza, hogy ő is csatázhasson egyet. Utolsó szavaival, futtában kiáltva kért meg minket, hogy tolmácsoljuk szavait: arra buzdított minden Linuxvilág Olvasót, hogy hívja át magához egyik szomszédját, cimboráját, szülőjét, bárkit – és tapasztalják meg egymás ellenében ezt a nem mindennapi mókát!„

Végszó

A viccet félretéve, nagyon nehezen született meg ez a rövidke írás. Gépelés közben, a kettes számú asztalomon folyamatosan futott egy ***Wormux:*** gyakoroltam, és megint csak gyakoroltam. Azért igyekeztem ennyire, mert a zsigereimben akartam érezni az aknavető lövedékének röppályáját, no meg a ***JetPack*** kezelését is. Én leginkább ezek használatával próbálok felülkerekedni a cimboráimon, akikkel folyamatosan csatázunk. Elvégre hogy nézne ki, ha a többiek két – három körből a másvilágra küldenének?

Végezetül, egy dologra még ki kell térnem: bár a cikkben sokszor szerepelnek a „meghalt", „lelőtte", „kivégezte" szavak, ennek ellenére kiskorú játékosokat is nyugodt szívvel oda lehet engedni a program közelébe. Az agresszivitás itt csak képletesen értendő – minden mozgásfázis és effekt roppant aranyosan és emészthetően van kivitelezve, vért pedig keresve sem találni!

Kovács Zsolt (kovi@linuxforum.hu)

Quake fanatikus. Négy éve a debreceni linuxosok egyike. Töretlenül hisz a Slackware terjesztésben.

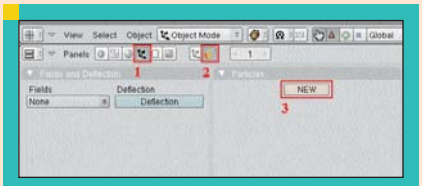
A Blender használata (10. rész)
Finomságok

Az elmúlt egy évben kilencszer írtam bevezetőt. Hogy őszinte legyek, mind a kilencszer bajban voltam és most sincs könnyebb dolgom. A mostani a tizedik, és minden valószínűséggel a legutolsó cikkem a Blender sorozatban, ami szintén csak nehezíti a dolgom.

Afféle búcsúzás képen megpróbáltam néhány igazi csemegét összegyűjteni, amelyekre mindezidáig nem jutott idő, de úgy érzem túlságosan sokról maradnánk le.
Néhány egyszerű példán keresztül bemutatom a ***Blender*** „részecske motorját", amivel könnyen tudunk tűzijátékot, esőt, füvet, stb. modellezni, majd a hasonlóan hasznos folyadék szimulációval is megismerkedhetünk. A cikk második felében betekintést nyerhetünk a ***Blender Python*** nyelven programozható felületébe, megtudjuk mi az amiben a ***Blender*** messze túlszárnyalja a „professzionális", fizetős (és egyplatformos) alkalmazásokat, és végül vetünk egy pillantást a ***Blenderbe*** épített játékmotorra is, amivel a programozó lelkületűek a ***Blender*** adottságait kihasználva gyönyörű játékokat írhatnak.

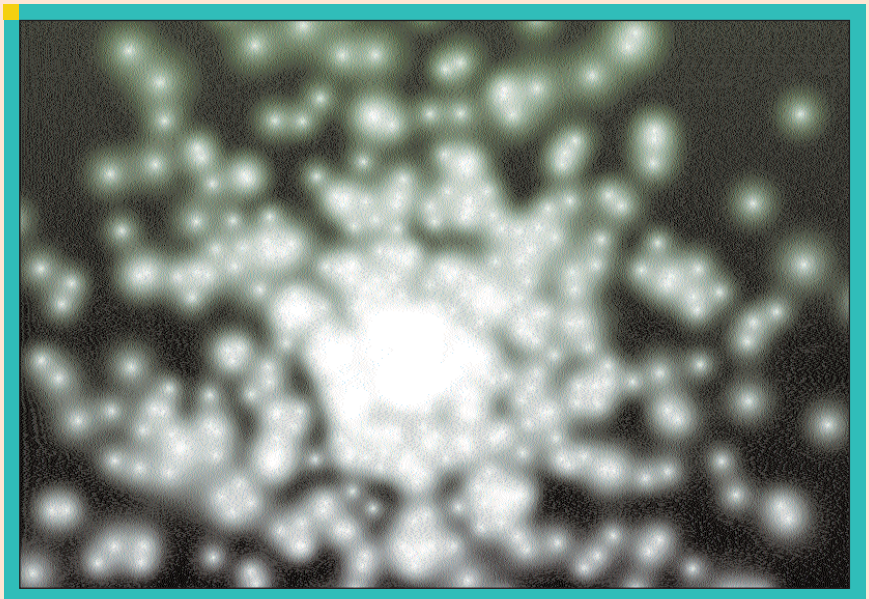
Particle engine

A név kicsit csalóka, ugyanis a ***particle*** szó magyarul részecskét jelent, de a ***Blenderben*** ez tulajdonképpen bármilyen objektum lehet. Az alapvető felállás szerint egy általunk megadott felületről induló részecskékre ható fizikai erőhatásokat szimulálhatjuk, így gyönyörű effektekkel bővíthetjük az animációnkat. Az egész akkor válik igazán gyönyörűvé, mikor rájövünk, hogy a részecskének nevezett mozgó objektum gyakorlatilag bármi lehet (tényleg bármi). Mindezt kombinálva a ránézésre is sok beállítási opcióval (*2. ábra*), az eredmény tényleg csak a fantáziánkon múlik. Legelső lépésként létre kell hoznunk egy objektumot, ami létrehozza a részecskéket. Ő lesz az emitter. Tetszőleges mesh lehet, de ajánlott mindig a célnak legmegfelelőbbet használni. Ha például egy szörnyecskének szeretnénk bundát varázsolni (igen, olyat is lehet), maga a szörny lehet a részecskéket kibocsátó objektum, de egyszerűbb esetekben egy ***plane*** (egy egyszerű négyszöglap) is megteszi. Hogy a részecskék a mesh felületén (face), vagy a vertexeken (vagy esetleg mindkettőn) keletkeznek, a ***Particles*** nevű panelen ***„From"*** részénél a ***Verts***

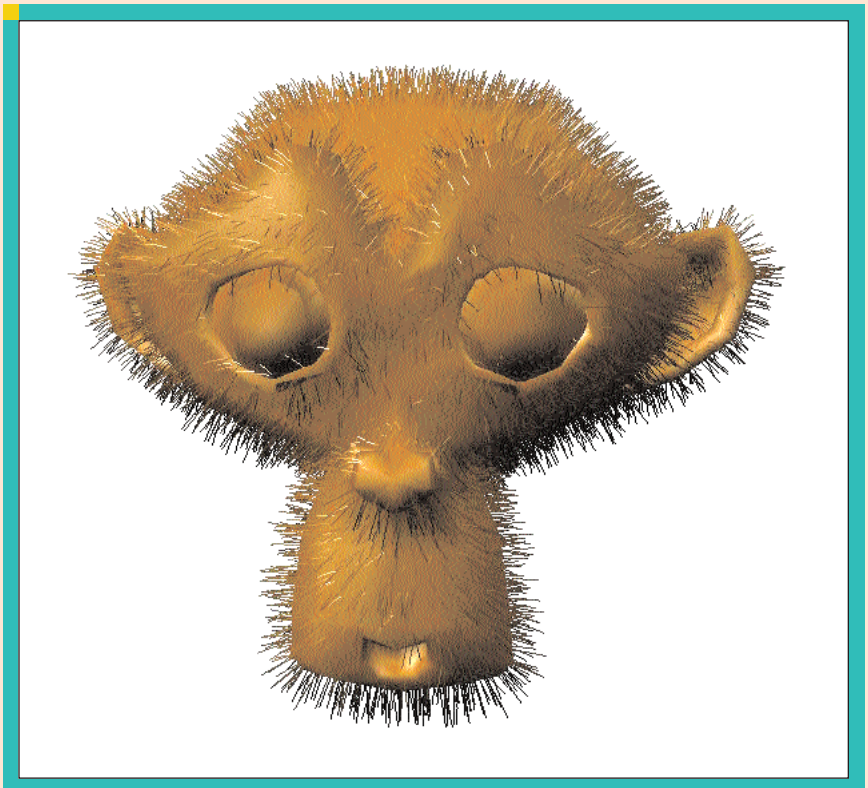


1. ábra Particle Engine bekapcsolása

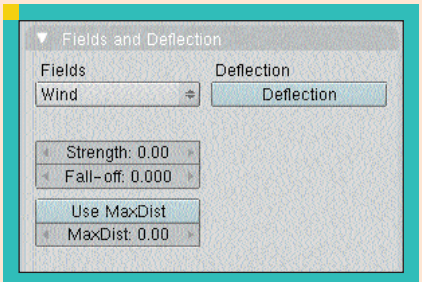
2. ábra Beállítások (Particles, Particle Motion)



3. ábra Példa részecske szimulációra



4. ábra Static particles



5. ábra Fields and Deflection panel

illetve ***Faces*** gombokkal állíthatjuk be. Ugyanitt beállíthatjuk az is, hogy vertexek esetén melyik ***Vertex Group***-ot (vertexek egy csoportja) kívánjuk használni. Ha nem adunk meg csoportot, akkor az összes vertex emitterként fog működni.

Mivel a részecskék születnek, élnek, majd halnak, ezért meg kell adnunk hogy pontosan mikor és mennyi részecske keletkezzen, és azok milyen sokáig létezzenek. Az *„Emit”* felirat alatt található ***Amount*** gombbal az összesen megjelenő részecskék számát szabályozhatjuk. Ezek az animációban a ***Start*** és ***End*** beállításoknál megadott idő alatt keletkeznek, és a ***Life*** gombon beállított ideig léteznek (*3. ábra*). Alapállapotban, ha egy mesh-t emitterré avatunk, az a mesh a renderelt animáción nem látszik. Ha mégis ezt szeretnénk, a ***Display*** felirat alatt kapcsoljuk be a ***Mesh*** gombot. Ugyanitt található a material nevű beállítás, ami kicsit trükkösen működik. A ***Mesh*** anyagbeállításainál több csoportot hozhatunk létre, és megadhatjuk, hogy az egyes face-ek melyik csoporthoz tartozzanak. Itt ugyanezekből az anyagokból választhatunk ki egyet, ami majd a részecskékre vonatkozik. A ***Static*** beállítással (mint a neve is mutatja) statikus részecskéket hozhatunk létre. Ezek a részecskék a *4. ábrán* látható módon helyezkednek el, és a ***Vector*** gombbal kombinálva kiválóan alkalmasak szőr vagy fű szimulálására.

Részecskék mozgása

Megpróbáltuk, de nem működik? Nem mozognak a részecskék? Nos, ez csupán annyit jelent, hogy a ***Blender*** a valóságnak megfelelően működik. ***Newton*** törvénye kimondja, hogy minden test (a részecskét tekintsük pontszerű testnek) nyugalomban marad, vagy egyenes vonalú egyenletes mozgást végez mindaddig, amíg egy másik test vagy erő nem hat rá. Létrejövő részecskéinknek nincs kezdeti sebessége, és mivel nem hat rájuk semmiféle erőhatás, ezért egyáltalán nem mozognak.

Sok lehetőségünk van, hogy a részecskéket mozgásra bírjuk. A ***Particle Motion*** panelen kezdősebességet adhatunk a részecskéknek és statikus erőhatásokat (például gravitáció) állíthatunk be, illetve létrehozhatunk különböző mezőket, amik dinamikusan hatnak a részecskékre.

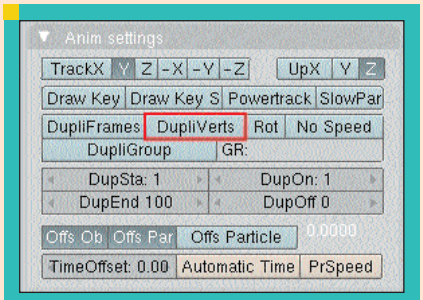
A statikus beállításokat a ***Particle Motion*** panelen (*2. ábra*) végezhetjük el. A ***Force*** résznél beállíthatjuk a részecskékre ható X, Y és Z irányú erőket, a ***Velocity*** résznél pedig a részecskék kezdősebességét, és kezdő haladási irányát. A ***Normal a*** részecskék indulási sebessége. Nagyobb szám nagyobb sebességet jelent. Ha azt szeretnénk, hogy adott irányba gyorsabban mozogjanak a részecskék, használhatunk egy textúrát is, ami megadja az indulási sebességet. A kezdő haladási irány általában az emitter középpontjából kifelé mutató vektor vagy a face normálvektora, de ha unalmasnak találjuk, a ***Random*** érték állításával véletlenszerű irányba indíthatunk részecskéket. Minél nagyobb ez az érték, a kezdő haladási irány annál jobbal eltérhet a normálistól.

Dinamikus erőhatások

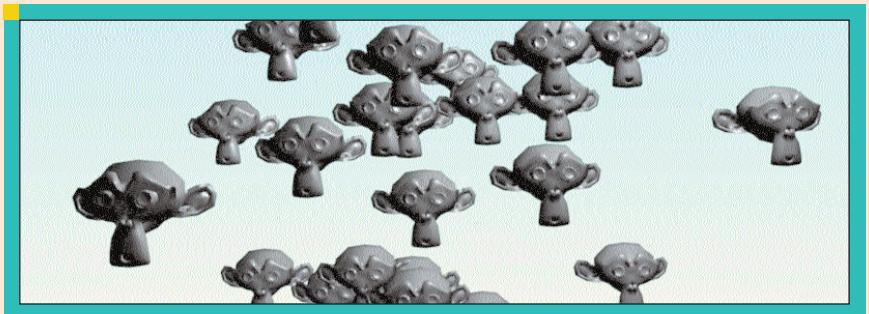
A statikus, mindig ugyanazt az útvonalat bejáró részecskék unalmasak. A ***Random*** beállítással javíthatunk ezen a hiányosságon, de az igazi megoldás az úgynevezett mezők (***fields***) használata. ***Blenderben*** ez tetszőleges objektumot jelenthet. Ezek beállításait a ***Fields and Deflection*** panelen végezhetjük el. A szabadságot az adja, hogy ezeket a mezőket objektumok szimbolizálják, vagyis animálhatók. Így hozhatunk létre például változó erősségű és irányú szelet, ami változó irányba fújja a füstöt vagy a felhőinket.

DupliVerts

Az előbb azt állítottam, hogy részecske bármi lehet, ennek ellenére még mindig csak pontoknál és vonalaknál tartunk. A ***DupliVerts*** opció segítségével azonban akár angyalkák is repkedhetnek a képernyőn részecskék helyett. Semmi más dolgunk nincs, mint elkészíteni a kívánt objektumot, szülőnek kiválasztani az emittert és ez utóbbi objektumon bekapcsolni a ***DupliVerts*** opciót a *6. ábrán* látható módon.



6. ábra DupliVerts



7. ábra DupliVerts (működés közben)

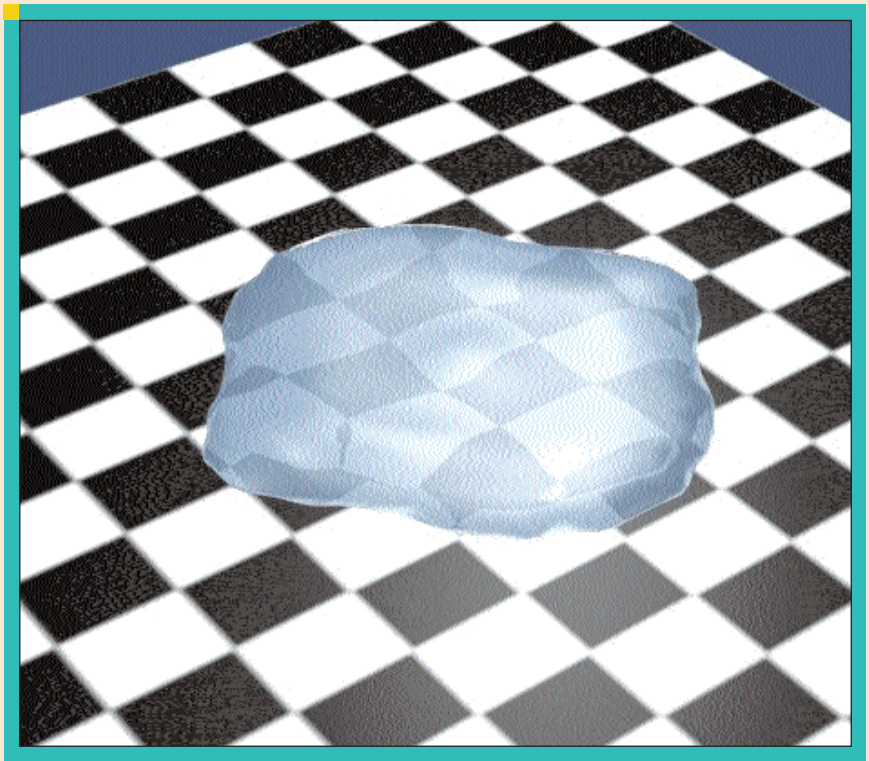
Fluid Simulation

A 2.4-es verzió újdonsága a ***Fluid Simulation***, amely kiválóan alkalmas sör,bor,víz,vér és hasonló folyékony állagú anyagok szimulációjára, és ami azt illeti mindezt egész ügyesen csinálja. Az egyetlen negatívum az ehhez szükséges memória és ***CPU*** idő, ugyanis egy összetettebb folyadék-animáció kiszámolása legalább akkora gépigényt igényel, mint egy bonyolultabb raytracing. De lehet hogy csak én számítógépem maradi, és elkelne egy kis bővítés.
A ***Fluid Simulation*** panelen (*8. ábra*) összesen öt gomb közül választhatunk. Az itt kiválasztott opció határozza majd meg, hogy az épp kijelölt mesh milyen szerepet fog betölteni a szimuláció során.
A legszükségesebb a ***Domain***.
A ***Domainnek*** kikiáltott objektum (legtöbbször egy kocka) mintegy behatárolja a szimuláció színterét, így ami a ***domain*** objektumon kívül van, egyáltalán nem vesz részt a szimulációban. Ez maga után vonja azt is, hogy minden bizonnyal a ***Domain*** objektum lesz a legnagyobb.
A ***Fluid***-ként megjelölt objektum(ok) fogják betölteni a folyadék szerepét.
A szimuláció kezdetekor egész egyszerűen elkezdenek vízként viselkedni.
Az ***Obstacle*** magyarra fordítva valamiféle akadályt jelent. A mi esetünkben olyan objektumot, amin a víz nem, vagy csak részben folyik át.
A fenti háromféle objektummal már neki is állhatnánk pancsolni, de van még két beállítási mód, ami sokat segíthet. Az ***Inflow***-nak titulált objektum folyadékforrás – vizet hoz létre –, az ***Outflow*** pedig afféle lefolyóként funkcionál, vagyis vizet von el a szimulációból.
Minden szimulációban résztvevő objektum beállítását külön szabályozhatjuk, továbbá a szimuláció részletességét, a folyadék fizikai viselkedését (víz, olaj, stb) és a gravitációt is mi állíthatjuk be.
A szimulációt a ***Domain*** objektum ***BAKE*** gombjával indíthatjuk el. Ilyenkor a ***Blender*** a megadott ideig próbálja fizikailag modellezni a folyadék mozgását. Mindeközben a jobb felső sarokban láthatjuk, pontosan hányadik frame-nél tart a szimuláció.
Nincs más dolgunk, mint hátradőlni és várni, vagy erre az időre beidőzíteni az ebédet (mint én tettem), ugyanis a szimuláció frame-enként egy úgy mesh-t hoz létre, ami majd a folyadékot reprezentálja és ez bizony eltarthat egy darabig…

Soft Body

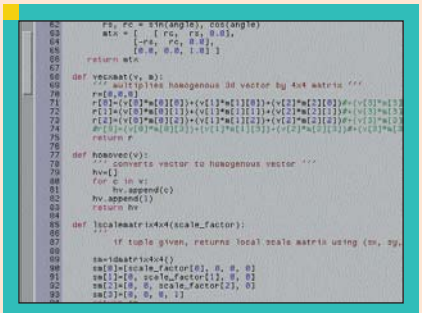
A ***Soft Body*** a ***2.37***-es verziótól található meg a ***Blenderben***. A puha, rugalmas anyagok fizikai viselkedését modellezi úgy, hogy animáció közben kiszámolja

8. ábra Fluid Simulation panel



9. ábra Fluid Simulation

10. ábra Soft Body



11. ábra A Blender python editora

a vertexekre ható fizikai hatásokat, és aszerint mozgatja el azokat.
Az *10. ábrán* a ***Soft Body*** panelt láthatjuk. A felső gombokkal állíthatjuk be a szimuláció fizikai tulajdonságait.
A ***Fraction*** az általános közegellenállást, a ***Mass*** a tömeget (nagyobb tömeg lassabban mozog) a ***Grav*** pedig a -Z irányba ható gravitációs erőt jelenti. A ***Speed*** beállítással a szimuláció sebességét, az ***ErrorLimit***-el pedíg a precizitást (kisebb szám, nagyobb precizitást, de több számolást jelent) állíthatjuk be.
Mivel semmi sem nyúlik akármeddig, ezért kell bizonyos korlátozásokat bevezetni. A ***Goal*** gomb bekapcsolásával, a rugalmas anyagokhoz híven minden részecske igyekszik a fizikai hatások ellenére visszatérni az animáció által meghatározott pozícióba.
Megadhatunk egy ***Vertex Groupot***, amire nem hat a fizika, csak az animáció, ilyenkor a többi vertex ehhez a csoporthoz igyekszik igazodni.
További korlátozásokat jelentenek a mesh élei. Ezek mint a rugalmas gumikötelek, próbálják összetartani a vertexeket. Ráadásul ezeknek a beállításait is szabályozhatjuk.
A ***Blender*** egyetlen hiányossága, hogy a ***Soft Body***-t csak mesh-re lehet alkalmazni. Vertexekkel dolgozik, így nem lehet statikus részecskéket sem animálni (a mozgó szőrszálakat is máshogy kell megoldani). Ettől az apróságtól eltekintve azonban a ***Soft Body*** egy igencsak kezes apróság, persze csak a megfelelő kezekben.

Játék a kígyóval

A ***Python*** névre hallgató találmány egy nagyon jól használható szkript nyelvet takar. Legfőbb jellemzői az objektum orientáltság, különböző platformok támogatása, a java-hoz hasonlóan működő bájtkódos fordítás, az elegáns szintaktika és a bővíthetőség. Ezek a tulajdonságai tették elterjedté, és emiatt választották a ***Blender*** szkript nyelvévé.
A legtöbb professzionális, kereskedelmi ***3D*** animációs program rendelkezik valamilyen saját szkriptnyelvvel, de mindnek korlátozottak a lehetőségei. Azt hiszem nem állítok valótlant, ha azt mondom: a ***Blender Python API*** bizony fényévekkel megelőzi a versenytársakat, és a folyamatos fejlesztések révén a lehetőségek száma csak nőni fog.
A ***Blender*** minden indításkor ellenőrzi, hogy megtalálható-e a számítógépünkön a ***Python*** futtatókörnyezet. Ennek hiányát vagy meglétét közli, de a program futtatását nem teszi ettől függővé (nélküle is fut), azonban a ***Python*** futtatókörnyezet nélkül nem használhatjuk ki a ***Blenderben*** rejlő rejtett lehetőségeket. A ***Blender*** biztosít ugyanis számunkra egy programozható függvénykönyvtárat, aminek segítségével szinte bármit megtehetünk.
Objektumokat, mesheket hozhatunk létre, azok beállításait tetszőlegesen megváltoztathatjuk, mozgathatjuk őket, stb. mindeközben pedig saját készítésű felhasználói felülettel tarthatjuk a kapcsolatot a felhasználóval.
Játékot írunk, és saját formátumba szeretnénk exportálni a modelleket? A ***Python API***-val probléma nélkül megoldható. Esetleg importálni szeretnénk egy szöveges formában tárolt ***motion capture*** animációt? Semmi akadálya.
A ***Pythonban*** írt bővítményekhez saját felhasználói felületet alakíthatunk ki, így az együgyű felhasználók is használni tudják. Számos, a ***Python API***-ra textúra illetve mesh generátort tölthetünk le az internetről, amik könnyebbé (de legalábbis élvezetesebbé) teszik a munkánkat.

„Ha már a kezünkben van egy jól használható 3Ds függvénykönyvtár, miért ne használhatnánk fel interaktív alkalmazások írására?"

Valami ilyesmi gondolat motoszkálhatott a fejlesztők fejében, amikor megalkották a ***Blender*** játékmotorját. Ennek segítségével valós időben renderelt interaktív játékokat hozhatunk létre a ***Blender*** grafikai lehetőségeit felhasználva. Természetesen figyelembe kell vennünk a számítógép sebességét, ugyanis a másodpercenkénti 25-30 kép megjelenítéséhez ugyanennyi renderelésre is van szükség, ami eleve kizárja erőforrásigényes effektek és raytracing alkalmazását.

Végszó

A ***Blender*** egy nagyon sokrétű, összetett alkalmazás. Egyetlen cikksorozatban nem lehet bemutatni minden részletét, és ha lehetne is, a sorozat elkészültekor már elavult lenne. A nyílt forráskódú programokhoz híven a ***Blender*** folyamatosan fejlődik, minden verzióban találhatóak újdonságok és sajnos néha hibák is. Mindezekkel azonban együtt lehet élni és amint azt az ***Orange Project*** is bebizonyította, a ***Blender*** alkalmas professzionális munkára, és tudásban felveszi a versenyt a versenytársaival.

Szalai András
(sly87@freestart.hu)

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

KAPCSOLÓDÓ CÍMEK

A Blender hivatalon honlapja:
➲ http://www.blender.org

Python scriptek:
➲ http://www.blender.org/cms/Python_Scripts.3.0.html

További információk:
➲ http://mediawiki.blender.org/

Serious Sam – avagy „Komoly Samu” kalandjai

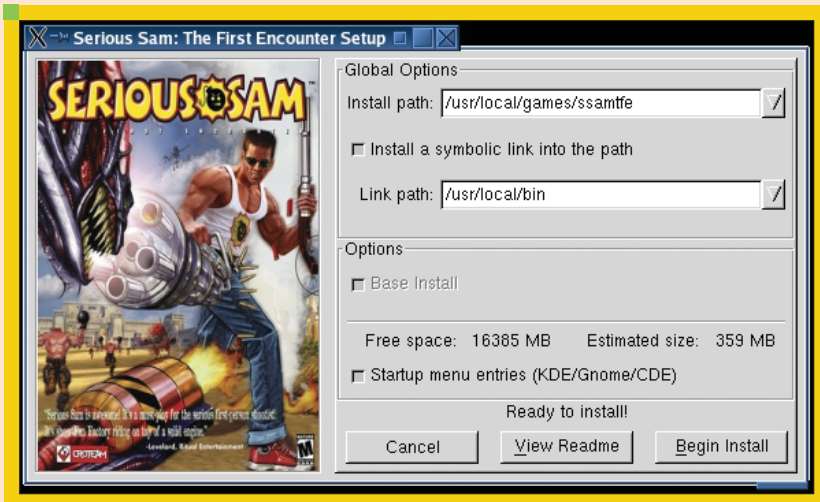
A név alapján kevesen gondolnának „komoly” játékra, miközben a horvát illetőségű fejlesztők 2001-ben az év egyik legjobb FPS projektjét adták el ezzel a címmel.

Nem komolytalan

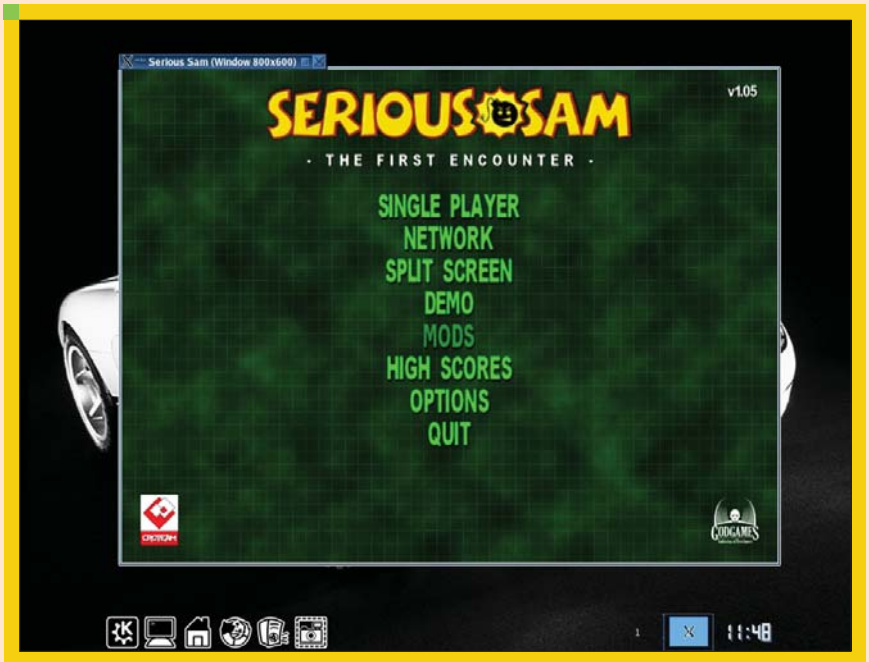
2001-et írunk: a *Quake3* láz kimondatlanul csillapodóban van, az *Unreal* programozói pedig még javában készítik az aktuális, multis nagyágyút. *„Egy olyan játék kellene, amiben van egy kicsi ebből is - abból is, esetleg ha a kódot ki lehetne hegyezni egy eszetlen hentelésre, akkor nagyot alakíthatnánk”...* Talán ezek a gondolatok járhattak a horvátországi *Croteam* tagjainak a fejében, amikor megalkották a művüket. Ha így történt, akkor teljes a siker: a *Serious Sam* messze van a *Quake* sorozattól, nem is húz az *Unreal* vonal felé, miközben menetében több tekintetben is tetten érhető a (többek között *Doom* által is kivitelezett) tömeges öldöklés. Főbb jellemzői tekintetében az eredmény mellbevágó lett, szerencsére szó nincs komolytalanságról.

Megszállottak előnyben

A játékprogram kerettörténete eléggé felületes, ezt a rész a projekt leggyengébb eleme. Egy jövőbéli meséről van szó, miszerint társaink rátaláltak az ősi civilizáció egyik ránk hagyott örökségére: idő és tér áthidalásáról szóló feljegyzések kerültek az emberiség birtokába. Jó szokás szerint, az ismeretlenbe beleugorva hamar kipróbálásra került a dokumentumok alapján kivitelezett technológia. Egy ideig minden rendben haladt, ám egy megnyitott kapun keresztül később különféle idegen és brutális lények özönlöttek a világunkba. Rövidesen tisztázódott a megoldás: csupán egy esély maradt a túlélésre, miszerint meg kell szerezni egy speciális tárgyat, ami bezárhatja az átjárót. Erre a feladatra kell vállalkoznia



1. ábra A játék linuxos telepítője



2. ábra A puritán menürendszer

3. ábra Mészárlás indul, KDE asztalon

4. ábra Két tucat ellenfél

a potenciális *Sam* jelölteknek, pontosabban a tárgy négy darabban szétszórt részeinek a begyűjtésére. Az aktuális helyszínre a főszereplő egy időgép segítségével érkezik meg...
A gyenge mesét elfelejtve azonban nem lehet okunk panaszra: „jó közepes" képességű grafikai motor, egyedi fizika, ötletes pályák és mellbevágóan sok ellenfél biztosítja a szórakozást. Hozzáteszem, hogy e kissé különc öldöklés megszállottságot feltételez: a megrögzött *Quake* rajongók egyáltalán nem biztos, hogy végigjátsszák a játékot, csakúgy, mint ahogy talán az *Unreal* hívei sem. Azonban akit tornacipős, laza főhősként feldob, ha egy egyszerűbb helyzetben is húsz ellenfél ront egy időben az életére, ne keressen tovább: erre a játékra van szüksége!

Életre hívni...

Mivel a projekt alapvetően *Win32* felületre készült, egy ideig nem rendelkezett linuxos binárissal. Azonban később, amikor a *Croteam* saját fejlesztésű motorja már kellőképpen kiforrott, valamint látszott, hogy a játékosok egyáltalán nem felejtetik el a rém lényegre törő projektet, megszületett a natív verzió. A portolást az *Icculus* gárdája végezte el, akiknek a munkáját legkönnyebben a megfelelő *Loki -* alapú telepítővel tudjuk üzembe állítani.
A szükséges állomány a *Loki Installers for Linux Gamers* csapatának honlapján, a ➲ *http://liflg.org/?catid=6&gameid=71* címen érhető el.
A nagy karriert befutott telepítőt *root* jogkörrel le kell futtatni: először bekéri a windowsos játéklemezt, majd a diszkről kinyert adatcsomagokat az átírt motorral együtt a helyére teszi (*/usr/local/games/ssamtfe*). A telepítő használatának nem függvénye a grafikus interfész: *X kiszolgáló* nélkül indítva egy ízléses, szöveges felületen is életre lehet hívni a kódot. Miután minden fájl a helyére került, felhasználóként kiadott `ssamtfe` paranccsal indítható a „vadászat". Fontos tudni, hogy a linuxos kliens jelenleg béta állapotú (*v1.05beta3*), ennek ellenére csaknem hibamentes. Egy „hiányossága" azonban szemet szúr: sajnos a szabad rendszer ügyfele nem kompatibilis *Windows* hajtotta szerverekkel és kliensekkel, így vegyes hálózatokon nem használható a *Multiplayer* üzemmód! Az átirat szerencsére nem kíván erős gépet: ahhoz, hogy *Sam* kalandjait kompromisszumoktól mentesen tapasztalhassuk meg, csupán *1GHz* órajelű *x86* processzorra, *256MByte* központi memóriára, és egy középszerű *3D* grafikai hardverre van szükségünk (működő *GLX* vagy *DRI* kapoccsal) – no meg egy *Linux* disztribúcióra.

... És életben maradni

Mint említettem, a program erőssége az ötletes, sajátos belső világa. Azonban az sem mellékes, hogy igazi és hamisítatlan „tömegmészárlás-szimulációról" van szó: a mai, divatos megvalósítások talán összesen sem rendelkeznek ennyi ellenféllel. Itt nem számít kirívónak az, amikor (használható fedezék nélküli) nyílt terepen egyszerre több tucat, kamikaze jellegű ellenfél ront ránk, bombákkal megrakva...
Külső és belső térben, késsel, pisztollyal, dupla csöves puskával, vagy

éppen rakétavetővel a kézben pillanatok alatt megtapasztalható mindez: közepes fokozaton már elég komoly kihívást jelent a pályák teljesítése. Egyébként ellenfeleink kasztjaiból is akad bőven, hiszen a fej nélküli alakoktól kezdve, pókokon keresztül, csontvázakon át a lépegetőkig széles a repertoár. Minden adott: a képernyő előtt töltött idő kész túlélőgyakorlat.

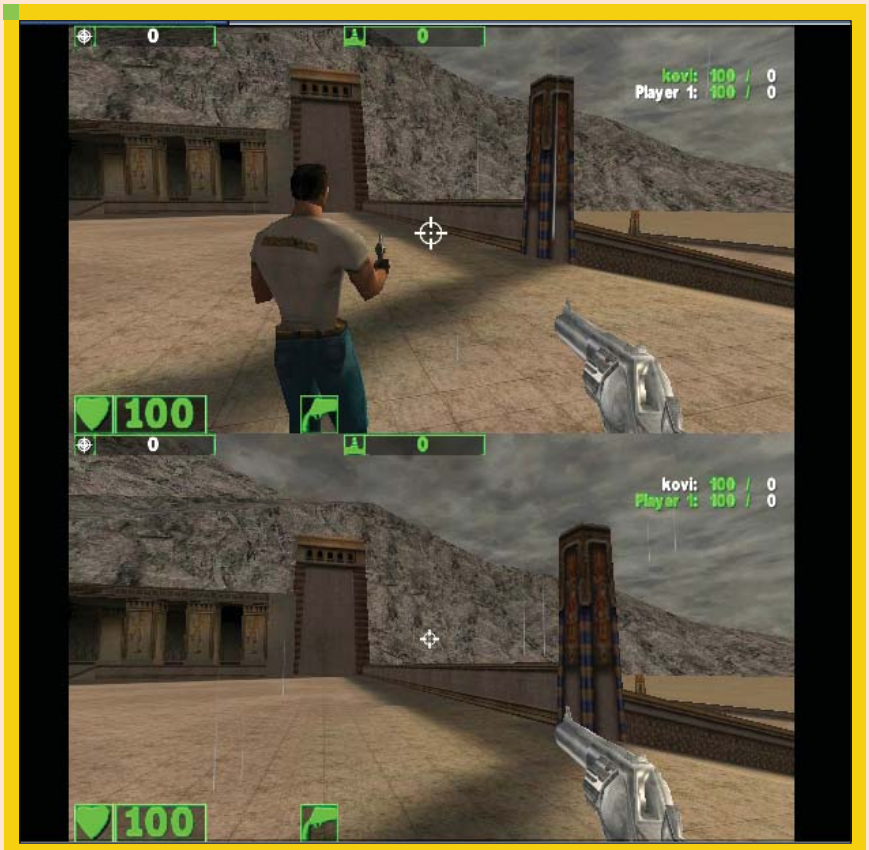
A dolog szépségét rontandó, három (idegesítő) sajátossággal sokan össze fognak futni: ellenfeleink gyakran a semmiből kerülnek elő, a dinamikusan változó zenei kíséret rövid időn belül idegőrlővé válik, valamint az öldöklés néhol már erősen *arcade* jelleget vesz. Meglátásom szerint a program emiatt bizonyos pontokon monotonnak tűnik, de a negatívumok mit sem rontanak azon, hogy a *Serious Sam* igazi kultuszjátékként vonult a történelembe. (Később, a második epizód pedig erre a maradandó emlékre építve kaszált nagyot. Apropó, második rész! Ennek szintén van linuxos kliense, de a cikk írásakor még erősen próbaverziós állapotú: éppen bétateszterek munkájára vár. Üzembiztos kiadás esetén erre majd bővebben kitérek.)

Ötletek, apró ötletek

Arról, hogy a játék ne csak a páratlan tömegmészárlás miatt legyen beszédtéma, a fejlesztők apró ötletekkel gondoskodtak. Kiragadott példaként nagyon tetszik az a helyzet, amikor egy gördülő sziklagolyó elől menekülve a karakterem *Indiana Jones* dallamokat fütyörészik... De a hálózati mód is megér egy misét: a megszokott stílusokon felül lehetőséget ad az egyjátékos szál kooperatív megoldására is. Érdekességként akár osztott képernyőn is megoldható a pusztítás, így egy számítógép előtt ülve akár négyen is segíthetik egymást. Utóbbi lehetőség nem mindennapi ebben a kategóriában! Természetesen a linuxos hálózatok lehetőségeire támaszkodva a játék indítható dedikált szerver üzemmódban is (`cd /usr/local/games/ssamtfe/Bin, ./ssam_lnxded <konfig>`).

Gyakorlatilag minden adott a könnyed kikapcsolódáshoz: aki csak tud, éljen e lehetőséggel – csalódás kizárva! Sőt, a *Quake*, *Unreal* rajongóknak sem kell túlzottan idegenkednie a képek láttán: magamból kiindulva annyit elmondhatok, hogy egy-két sör társaságában jó néhány délutánt agyonütöttem a *Serious Sam* világában!

5. ábra Kitartás és reflex dolga



6. ábra Egy Linux, két játékos

Kovács Zsolt (kovi@linuxforum.hu)

Quake fanatikus. Négy éve a debreceni linuxosok egyike. Töretlenül hisz a Slackware terjesztésben.