Hírek

Java telefon másképp

A *Lucent Technologies* és a *SUN* elkészítette a *Jasper S20*-at, ami alapvetően más koncepcióval használja a *Java*-t, mint a mostani telefonok. Joggal kérdezheti a kedves Olvasó, hogy megéri-e, van-e hely a jelenlegi *Symbian, Windows Mobile* és *Linux* trió mellett. A jelenlegi telefonoknál kétféleképpen futhat egy program: natív vagy *Java* módban. A *Java* mód ott szükségessé tesz pár olyan szintet, amely a *Jasper S20*-nál nincs, hiszen itt minden *Java* program.
A telefonban *ARM 9*-es (*TI OMAP 730*) processzor végzi a munkát ***64 megabájt operatív memória*** és ***64 megabájt flash*** tároló társaságában. Szabványos ***SD kártyával 2 gigabájtig*** bővíthető a kapacitása.
➲ http://www.deviceforge.com/news/NS6109398413.html

Vírusos lehet a MacOS?

Az *Apple* azzal, hogy az *x86*-os világ felé megnyitotta kapuit, potenciálisan vírusfertőzésre adott esélyt.
Ezzel azonban a legtöbb *MacOS X* felhasználó nem számol, hiszen eddig ilyen veszély nem fenyegette őket a *PPC*-s időszakban.

Huszonegyedik századi autótolvajok

Magyarországon még nem jellemző, de tőlünk nyugatabbra már nem tolvajkulccsal vagy feszítővassal, hanem laptoppal járnak az autótolvajok. Mindezt azt teszi lehetővé, hogy a gyújtás, a riasztó és az ajtózárak is távirányíthatóak, így egy megfelelően felszerelt laptoppal is irányíthatóak ezek a rendszerek.
➲ http://www.leftlanenews.com/2006/05/03/gone-in-20-minutes-using-laptops-to-steal-cars/

Elephants Dream

Május 19-én elérhetővé tette az *Orange Open Movie Project* első rövidfilmjét *Creative Commons* jogállással. A film érdekessége, hogy csak nyílt forráskódú szoftvereket használtak. (Ennek listája a film végén a stáblistánál található.) A projekt honlapjáról letölthető a film többféle formátumban is.
➲ http://www.elephantsdream.org/

StarOffice makróvírus

Noha eddig *StarOffice* makróvírusról nem tudtunk, most elkészítettek demonstrációs céllal egyet, ami természetesen nem csinál semmi kártékonyat, csak terjed. Azonban a *StarOffice* a dokumentum betöltésekor rákérdez a makrók futtatására, így ha csak megbízható dokumentumokra engedélyezzük, elkerülhetjük a fertőzést.
➲ http://it.slashdot.org/article.pl?sid=06/06/01/1419216

Samsung okostelefon Linuxszal

Kínában már kapható a *Samsung SCH-i819* mobiltelefonja, amely *Prizm 2.5*-ös *Linuxot* futtat. Ottani nyaralás esetén nyugodtan vásárolhatunk belőle, hiszen a *CDMA* 800 MHz-e mellett az európai 900/1800 MHz-et is támogatja. A kommunikációt egy *Qualcomm MSM6300*-as áramkör bonyolítja, míg az alkalmazások egy 416 MHz-es *Intel PXA270*-es processzoron futnak. A készülék *Class 10-es GPRS* adatátvitelre képes, illetve tartalmaz *GPS* (globális helymeghatározó) vevőt is. Az eszköz *64 megabájt SDRAM*-ot és *128 megabájt* nem felejtő *flash* memóriát kapott, de *micro-SD* memóriakártyával ezt tovább bővíthetjük. A kijelzője 2.4 hüvelykes, felbontása pedig „csak" 240x320 képpont 65 ezer színnel. Természetesen a trendeknek megfelelően nem maradt ki a 2 megapixeles kamera sem. *Bluetooth*, *infra* és *USB* gondoskodik az adatátvitelről.
➲ http://www.linuxdevices.com/news/NS6583352606.html

Debian

2006 december elejére ígérik a *Debian* fejlesztői az új verziót, amely a jelenlegi *Sarge* kódnevű verziót hivatott váltani, amely *2005. június 6*-án jelent meg. Az új verzióban előreláthatólag már *UTF-8-as locale* lesz az alapértelmezett. A másik fontos újítás: javul az eddig hivatalosan hanyagolt *64 bites (AMD, PPC)* processzorok támogatása. 2006. június 30-ig szerepel a támogatott verziók között a *Woody*, de a biztonsági frissítések még az év végéig elérhetőek a *security.debian.org*-on.
➲ http://www.debian.org

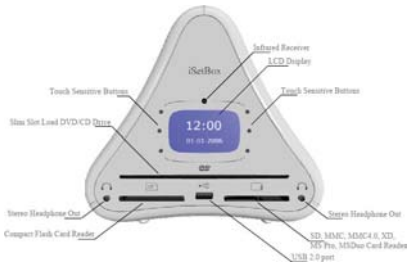
Mini PC

A *Jade Integration* hamarosan bemutatja a *Jack PC*-t, amely nem más, mint egy normál fali aljzat helyére beépíthető mini számítógép. A méretével arányos a *fogyasztása*, ami mindössze *5 watt*. A beépített *RISC* processzora 500 MHz-es, ami asztali gép teljesítményére átszámolva körülbelül *1,2 GHz-nek* felel meg. Elsősorban *vékony kliensnek* szánják, amit az is mutat, hogy „csak" 128 megabájt memória és *64* megabájt flash memória került bele. A beépített videókártya legfeljebb *1280x1024* képpontos felbontást kezel, a *négy USB* port pedig lehetővé teszi a két beviteli forrás (billentyűzet és egér) mellett további perifériák (például USB-s háttértár) csatlakoztatását is. Az eszköz *Windows CE* vagy *Linux* operációs rendszert futtat. Sajnos azonban az ára fordítottan arányos a méretével, 209 font (körülbelül 80 ezer forintba) kerül.
➲ http://www.jadeintegration.com/jackpc.php

Már kapható az Linuxos játék konzol

Május 18-tól már kapható Angliában a *GP2X Linux*-os játékkonzol. Két darab 240 MHz-es processzor hajtja, a programok pedig 64 megabájt memóriával és 64 megabájt flash tárolóval gazdálkodhatnak. A flash tároló *SD kártyával* bővíthető. Az asztali géppel *USB*-n keresztül cserél adatot. Két hagyományos ceruzaelemmel 6 óra használatot ígér a gyártó. A kijelzője egy 320x240 pixeles *LCD*, ami a mai világban már igen kevésnek számít, szerencsére az ára se magas: 190 dollár.
➲ http://gp2x.co.uk/aboutgp2x.html

Tartalomhoz a forma



A bolgár *Media Systems* 2006 negyedik negyedévére ígéri *TV*-hez kapcsolható számítógépét (*set-top box*), amely *Linuxot* futtat, és alkalmas *TV* nézésre, *DVD* lejátszására, zenehallgatásra és internetezésre is. Kezeli a *DVB (Digital Video Broadcast)* adásokat is. A formabontó – háromszög alakú – dobozban egy *PCI slot* kap helyet. Beviteli eszközként *Bluetooth*-os vagy *USB*-s eszközöket (billentyűzet, egér) használhatunk, de használhatjuk saját távirányítóját is.

➲ http://www.linuxdevices.com/news/NS8273097472.html

Túlmelegedés

Az *AMD* lehetséges túlmelegedésről számolt be, amely azonban csak a 2005 vége és 2006 eleje között gyártott *x52-es és x54-es Opteron* processzorok kis százalékát érintheti (körülbelül 3000 darabot), amennyiben extrém körülmények között használják azokat. Elsősorban laborokban jelentkezhet a hiba, ahol az átlagosnál magasabb a hőmérséklet, illetve nagyobb mértékű a lebegőpontos számítások aránya.
➲ http://www.eetimes.com/showArticle.jhtml?articleID=187002047

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.

Mi újság a rendszermag fejlesztése körül

Tekintettel a *GNU General Public License* hármas változatának közelgő megjelenésére természetesen módon merül fel a kérdés, hogy a *Linuxra* vajon továbbra is a második változat vonatkozik majd, vagy itt is megtörténik a váltás. Jelenleg az tűnik a legvalószínűbbnek, hogy nem. *Linus Torvalds* még ha akarná se tudná ellenőrizni a kernel teljes kódjának jogállását. Az évek során több ezer olyan részlet került be a magba, amit mások írtak, agy a velük kapcsolatos szerzői jogokkal is ők rendelkeznek. Ebből pedig az következik, hogy a váltást csak akkor lehetne véghezvinni, ha egyenként mindenki beleegyezne ebbe a saját szellemi tulajdonával kapcsolatban. Bár egyes derűlátó emberek szerint a több ezer szerző felkutatása lehetséges, a dolog tényleges kivitelezése egyelőre lehetetlennek tűnik. Ebből pedig az következik, hogy a belátható jövőben a *Linux* kernel marad a *GPL 2* hatálya alatt.

Mindeközben a fejlesztők egy több és több kernelváltozót és függvényt állítanak át úgy, hogy azok csak olyan harmadik féltől származó meghajtókkal működjenek együtt, amelyek a *GPL* hatálya alá tartoznak. Az ellenőrzést maga a rendszermag végzi úgy, hogy megvizsgálja egy a meghajtó által beállított változó tartalmát, amiben minden gyártó jelezheti, hogy a kérdéses kód milyen jogállású. Ha az érték *GPL*-re utal, a mag hozzáférést enged a korlátozás alá eső szimbólumokhoz is, ellenkező esetben nem működik együtt a meghajtóval. Az „elhajított kő” legutóbb az *AVM*-et találta el, amikor *Greg Kroah-Hartman* az *USB* alrendszert állította át úgy, hogy mostantól csak a *GPL* meghajtókat támogassa.

Az *AVM* köztudottan kizárólag bináris meghajtókat ad ki az általa gyártott hardverekhez, de ez a helyzet most valószínűleg elgondolkodtatja majd. Ami azt illeti a friss módosítás aztán végül mégis kikerült a kernelből, bár az derült ki, hogy *Greg* legfőbb oka a bevezetésére az volt, hogy a mag már régóta lehetővé teszi a felhasználói térben futó *USB* meghajtók írását anélkül, hogy ez a sebesség rovására menne.

Mindazonáltal az egész összeütközésnek egy haszna azért volt: *Greg* létrehozott egy olyan naplózórendszert, ami futtatáskor jelzi, ha egy alrendszer a közeljövőben át fog állni a *GPL* hatálya alá tartozó meghajtók kizárólagos támogatására. Így a rendszergazdák fel tudnak készülni a váltásra.

Willy Tarreau elkezdte összegyűjteni a hasznos 2.4-es foltokat, és elérhetővé tette őket egy központi helyen. A dolog mögött az a viszonylag általános igény húzódik meg, hogy a felhasználók szeretnének egy olyan 2.4-es kernelt, ami egyszerre stabil, és rendelkezik a legújabb szolgáltatásokkal. Mivel a páros/páratlan számozás immár nem utal egyértelműen a stabilitásra, a 2.4-es mag pedig gyakorlatilag véglegesnek tekinthető, ez az igény alapvetően jogos. Azok az érvek pedig, melyek szerint a stabil rendszermagról immár a terjesztések karbantartói gondoskodnak, vagy hogy a *w.x.y.z* fának éppen ez – vagyis a stabilitás és a legújabb szolgáltatások ötvözése – volt a célja egyetlen ok miatt nem meggyőzőek. Igaz ugyan, hogy ezek a rendszermagok egész imponáló futásidőket tudnak produkálni, a működésük azonban helyenként inkonzisztens, verzióról verzióra változik. Összességében ezek a kernelek nem rendelkeznek egy közös, stabil kódbázissal, így nem teszik lehetővé, hogy az alkalmazók és fejlesztők a felhasználói térben megbízhatóan működő programokat hozzanak létre és így stabil szolgáltatásokat nyújtsanak. Bár az kétségtelen, hogy a stabilitás mindig nagyon fontos szerepet játszott a mag fejlesztése során, a csúcsfejlesztők körében mégsem látszik különösebb elmozdulás a megfelelő irányba. Igazából ez a valódi oka annak, hogy *Willy* most színre lépett ezzel a kezdeményezéssel.

A közepesen távoli jövőben az *IDE* meghajtó ki fog kerülni a rendszermagból, és felváltja a *libata*. Ehhez azonban utóbbinak még növelni kell a stabilitását. *Alan Cox* meggyőződése szerint ez az idő még nincs közel, de a *libata* rendszer fejlesztésének egyértelműen ez a célja. Ugyanakkor azt sem szabad elfelejteni, hogy az *IDE* rendszerekkel kapcsolatos rémálomnak nem lehet azzal véget vetni, hogy az egyik kódot kicseréljük egy másikra. Az *IDE* szabvány ugyanis továbbra is meglehetősen laza, a gyártók pedig helyenként saját kényük-kedvük szerint értelmezik. Ráadásul az sem valószínű, hogy ez a helyzet a jövőben jelentősen megváltozna, így az *IDE* eszközök támogatása továbbra is a „nagy kaland” kategóriába tartozik majd. Bármit is hoz tehát a jövő, az biztos, hogy a meghajtó kódjában számos az egyes gyártókra specifikus rész lesz majd. Még ha az *IDE* eszközök gyártói képesek is lennének arra, hogy minden a jövőben gyártott eszköznél ragaszkodjanak egy adott, mindenki által elfogadott és azonos módon értelmezett specifikációhoz, akkor is jó időbe telne, mire el lehetne távolítani a kódból a régi eszközöket támogató részleteket.

Nem nevezhető különösebben váratlannak, hogy a *Reiser 4* fájlrendszer beleszaladt néhány problémába, mikor bebocsátást kért a rendszermag kódjába. A közelmúltban dúló háborúskodás után több kernelfejlesztő egyszerűen feltette a kezét, és azt mondták, hogy a továbbiakban csak akkor hajlandóak kommentálni a *ReiserFS*-sel kapcsolatos foltok működését, ha *Hans Reiser* felhagy az ellenük irányuló támadásokkal. Márpedig a fejlesztői tábor szakmai támogatása nélkül a *Reiser 4*-nek nem sok esélye van, hiszen csak ők látják át mindazokat a követelményeket, amelyeknek minden egyes folt meg kell feleljen, mielőtt bekerülhetne a stabil kernelfába. E nélkül a támogatás nélkül nehéz a jó irányba haladni. Márpedig minél jobban eltér a *Reiser 4* fejlesztése a rendszermag „főfolyásától”, annál nehezebb lesz a végén összhangba hozni vele. Összességében a *ReiserFS 4*-es változata egy idő után nyilván be fog kerülni a kernelbe, de egyértelműen csak az után, hogy a fejlesztők megoldották az egymás közti szociális problémákat, no meg persze azokat a műszaki gondokat, amelyek ezeket kiváltották.

Zack Brown (Linux Journal, 146. szám)

A Xen és a virtualizáció művészete

Virtualizációs technikák – egy gép több virtuális géppé (virtual machine, VM) osztása, „particionálása” több operációs rendszer egyidejű futtatásának támogatásához – már léteznek egy ideje. A virtualizáció megjelenhet hardver és szoftver formájában is. Az IBM elsőként mutatta be a hatvanas években virtuális hardverét (System 360/67), és azóta is folyamatosan fejleszti z/VM nagygépes operációs rendszerének szoftveres virtualizációs technológiáit. Napjainkban a virtualizációs szoftverek terén a legnagyobb lendület a Microsofttól, a VMware-től és a Xen nyílt forráskódú projektből származik (x86 és x64 hardveren).

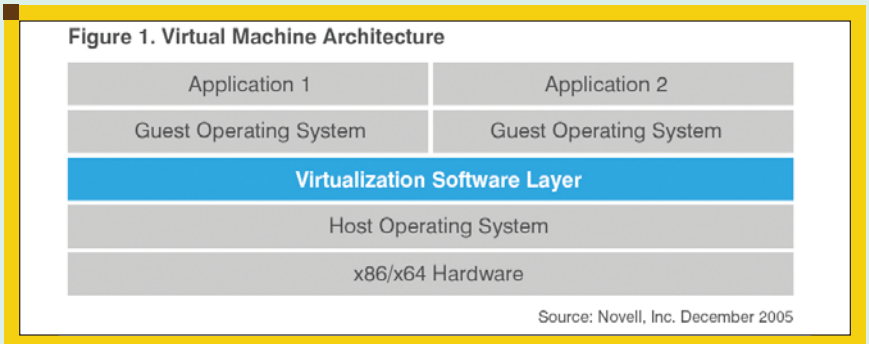
A virtualizáció az informatika minden területét felkavarta. Ennek oka, hogy az egyes virtuális gépeken akár egy, akár több szolgáltatás futtatható, egy szerveren pedig több virtuális gép futhat – így növelhető a szerverek kihasználtsága, csökkenthetők a hardverköltségek és potenciálisan akár a szoftverlicenccköltségek is. A *UNIX*-szerverek kihasználtsága általában 15-20 százalékos és sok más, jellemzően windowsos gép is messze a képességei alatt teljesít, mivel csak egyetlen alkalmazás futtatására használják.
Az *IDC* arra számít, hogy a virtualizációs tevékenységekkel kapcsolatos vásárlások 2009-re az egész világon megközelítik a 15 milliárd dollárt (*IDC* sajtóhír, ***2005. október 18***: ***Increasing the Load: Virtualization Moves Beyond Proof of Concept in the Volume Server Market***). Ezen költségek nagy része a virtualizációs szoftvert futtató hardverből fog származni. Az *IDC* szerint az 500 alkalmazottnál többet foglalkoztató vállalatok több mint 75 százalékánál működnek virtuális kiszolgálók – vagyis virtualizációs szoftvert futtató szerverek. Jelenleg a virtuális kiszolgálókkal kapcsolatos beruházások túlnyomó része *S/390*, *OS/400* és *UNIX* rendszerekben jelenik meg, de a jövőben várhatóan ez átalakul és az *x86/x64* alapú *Linux* és *Windows* rendszerek virtualizációja teszi ki majd a nagyobb részt.

Virtuálisgép-architektúrák

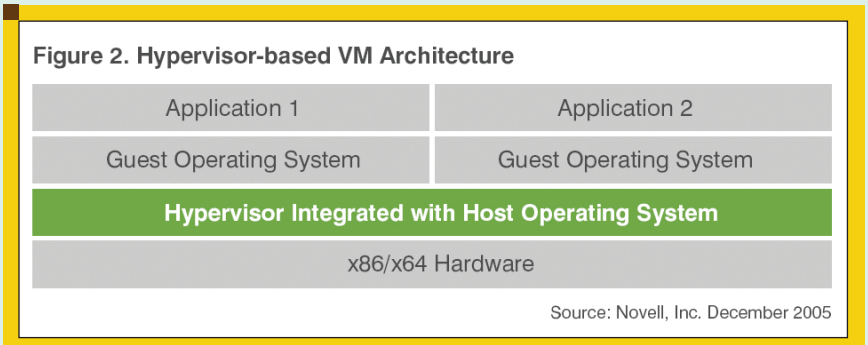
A szoftveres virtualizáció általában egy virtualizációsszoftver-rétegen keresztül kerül megvalósításra.
Ez a réteg, amelyet néha virtuális gép monitornak (*VMM*-nek) is neveznek, számos működő virtuális gép látszatát kelti. A virtuális gépek egy „vendég” operációs rendszerből, egy vagy több telepített alkalmazásból, felügyeleti eszközökből, vírusfelderítő szoftverekből és egyéb eszközökből állnak.
Az egyes virtuális gépek rendelkezhetnek a gazdagép funkcionalitásának egy részével vagy egészével, és a vendég operációs rendszer jellemzően a gazda operációs rendszer illesztőprogramjait és funkcióit használja.
Egy gép felosztása, hogy képes legyen több operációs rendszer egyidejű futtatására, számos kihívást támaszt:

- Az egyes virtuális gépeket el kell tudni szigetelni egymástól.
- A népszerű alkalmazások heterogenitása miatt fontos a sokféle operációs rendszer támogatása.
- A virtualizáció okozta teljesítménycsökkenésnek a lehető legkisebbnek kell lennie.

A manapság legjellemzőbb virtuális gép architektúrát az *1. ábra* mutatja. A virtualizációsszoftver-réteg szabályozza és osztja el a gazdagép és a vendég operációs rendszerek között az erőforrások használatát. A nyílt forráskódú ***Xen 2.0*** és a ***Microsoft Virtualization Server 2005*** egyaránt egy ilyen virtualizációsszoftver-réteg gyakorlati megvalósítása.



1. ábra Ezt az architektúrát néha szokták „hypervisor előtti” virtuális gép architektúrának is emlegetni



2. ábra Hypervisor alapú architektúra

A 2. *ábrán* egy ***hypervisor*** alapú virtuális gép architektúra látható. A ***hypervisor*** technológia – amelyre a ***Xen 3.0*** is épül – egy olyan virtualizációs szoftver, amely integrált a gazdagép operációs rendszerével, például a ***Linuxszal*** vagy ***Windowszal***. ***Hypervisor*** alapú környezetben elsőnek a ***hypervisor*** indul el, és csak eztán a gazdagép operációs rendszere. A ***hypervisor*** lényegében közvetlenül a hardver feletti réteg, amely virtualizálja a virtuális gépek számára az olyan erőforrásokat, mint például a processzor vagy a memória.

A virtualizáció megvalósításai

A virtualizáció hagyományos megvalósításaiban egy virtuális gép a gazdagép összes funkcionalitásával rendelkezik. Ezt nevezik teljes virtualizációnak. Ennek a megoldásnak a nagy előnye, hogy a vendég operációs rendszereket egyáltalán nem kell módosítani, – ilyen például a ***VMware ESX Server*** is. Bizonyos problémák azonban itt is jelentkeznek: az operációs rendszer bizonyos speciális, vezérlési utasításait a helyes virtualizáció érdekében a virtuális gép monitornak kell lekezelnie, ami azonban jó eséllyel lerontja bizonyos műveletek – például egy új alkalmazásfolyamat létrehozásának – teljesítményét.

A virtualizáció egy másik megközelítése az úgynevezett paravirtualizáció. Paravirtualizáció esetén kiküszöbölhetők a teljes virtualizáció teljesítményhátrányai, egy – a rendszer valódi hardveréhez hasonló, de azzal nem teljesen megegyező – virtuális gép-absztrakcióval.

A paravirtualizáció általános megközelítése elvárja a vendég operációs rendszerek módosítását futtatás előtt. E megközelítés esetén a vendég operációs rendszer és a virtualizációsszoftver-réteg nem tud függetlenül lenni; viszont továbbra sem kell módosítani az alkalmazásokat. A ***Xen*** hypervisor paravirtualizációt használ.

Az AMD és az Intel virtualizációs technológiái

Az ***Intel*** és az ***AMD*** számos hardverelemmel – a két cég megoldása ***Virtualization Technology (VT)*** és ***Pacifica*** névre hallgat – segíti a megfelelően konfigurált rendszereken a virtualizációs megoldások kialakítását. (A ***Xen*** projektcsapat az ***Intellel*** és az ***AMD***-vel együttműködve optimalizálják virtualizációs termékeiket, hogy kihasználják a ***VT*** és a ***Pacifica*** előnyeit.) A jelenlegi processzorarchitektúrákban minden szoftver négy privilégiumszinten, ún. „gyűrűben" fut (a 0-3. gyűrűben). Az operációs rendszer hagyományosan a 0. gyűrűben fut, míg az alkalmazások jellemzően a 3. processzorgyűrűben.

Mivel a virtualizációsszoftver-rétegnek privilegizált módon kell kezelnie az erőforrásokat, a szokásos megoldás a ***VT*** előtt az volt, hogy a virtuális gép monitor a 0. gyűrűben futott, a vendég operációs rendszer pedig valamilyen alacsonyabb szintű gyűrűben, például az 1. vagy 3. gyűrűben. A ***VT*** lényegében azt a látszatot kelti, hogy a vendég operációs rendszerek a 0. processzorgyűrűben futnak, míg alatta a virtualizációsszoftver-réteg a -1. gyűrűben.

Következtetések

A virtualizációs szoftverek piacán a verseny egyre szorosabb: a gyártók fej-fej mellett igyekeznek módosítani virtualizációs licenceiken, hogy versenyképesek maradjanak. Jelenleg a Novellé az egyik legegyszerűbb virtualizációs licencrend, valamint alacsonyabb árszínvonalon kínálja megoldásait, mint a konkurens cégek. Virtualizációs licencrendje 2004. augusztusában lépett életbe, amikor megjelent a ***SUSE Linux Enterprise Server 9***. Ennek lényege, hogy egy vagy több virtuális rendszer használata egy fizikai processzoron vagy szerveren nem számít a ***SUSE Linux Enterprise Server 9*** licencrend megsértésének. Például:

- Ha például a ***VMware virtualizációs*** szoftver fut egy ***Windows Server 2003*** rendszer alatt egy kétprocesszoros szerveren, akkor a ***SUSE Linux Enterprise Server 9*** egy vagy több példánya futhat vendég operációs rendszerként egy kétprocesszoros szerveren futó ***SUSE Linux Enterprise Server 9*** áráért.
- Ha a ***SUSE Linux Enterprise Server*** részeként szállított ***Xen*** virtualizációs szoftvert használjuk, akkor korlátozás nélkül, tetszés szerint beállítható a futtatni kívánt vendég ***SUSE Linux Enterprise Server*** operációs rendszerek száma, a ***SUSE Linux Enterprise Server*** adott szerverre vonatkozó áráért. Vagyis egyetlen licenc megvásárlása lefedi a ***SUSE Linux Enterprise Server*** gazdagép, és bármennyi beállított vendég operációs rendszer díját.

A ***Novell*** virtualizációs irányelve nemcsak egyszerű és költséghatékony, hanem az új ***hypervisor*** alapú technológia bevezetése terén is a konkurens cégek előtt jár. A ***Novell*** 2006 január/februárjában már elérhetővé tette a ***Xen 3.x hypervisor*** technológia előzetesét a ***SUSE Linux Enterprise Serverben*** egyes vállalati vásárlók számára, és a ***SUSE Linux Enterprise Server*** következő kiadásába integráltan bekerül a ***Xen 3.x hypervisor*** technológia. A ***RHEL 5*** mintegy hat hónappal később szállítja a ***XEN 3***-at, a ***Microsoft*** pedig egyáltalán nem kínál ***hypervisor*** alapú technológiát a ***Windows Longhorn Server*** második kiadásának megjelenéséig (2008 vége vagy 2009).

Informatika érettségi szabad szoftverekkel – második forduló

A kétszintű vizsgarendszer bevezetésével sokat változott az informatika tantárgy megítélése mind a diákok, mind tanáraik körében. A magasabb követelményeknek köszönhetően ugyan nehezebb is a vizsga, de nagyon sok felsőoktatási intézmény elfogadja felvételi tárgynak a matematika mellett. Így nem csak könnyedén túltehetjük magunkat a kötelezően választható tárgy megjelölésén, de értékes felvételi pontokat is szerezhetünk akár humán szakokon is. Érdemes tehát nekivágni.

Az informatika vizsga alapja a központi követelményrendszer. Mivel a gyakorlati vizsga feladatsorait központilag állítják elő, az eredmények – várhatóan – megbízhatóbbak, és könnyebben összevethetőek lesznek. Azonban, hogy a középiskolák is meg tudjanak felelni a követelményrendszer kívánalmainak, gondos fejlesztési és lebonyolítási munkálatokra volt szükség (adott szoftverek beszerzése, esetleg környező középiskolákban való vizsgáztatás megszervezése stb.), mivel az iskolák eszközparkja és szoftverkörnyezete nem egységes.
Így például, előfordulhat, hogy középszinten először érettségizőknek, csak azoknak a programoknak a használatára van lehetőségük, melyek az adott középiskola rendelkezésére állnak. Tehát nem biztos, hogy lehetőségük lesz éppen *Linux* és *OpenOffice.org* rendszert használni.
Emelt szinten a jelentkezéskor szoftverkörnyezetet kell választani a gyakorlati vizsgához, a megfelelő melléklet csatolásával. Majd a vizsgázót ennek alapján osztják be abba a vizsgát szervező oktatási intézménybe, ahol a választott alkalmazások biztosan elérhetőek.
Mindenesetre, ha a középiskola nem is rendelkezik egy-egy általunk óhajtott szoftverrel, az egyes csoportokban található programok egy ésszerű kombinációjával elérhető lesz, így a feladatok elvileg a legtöbb vizsgázó számára bármely középiskolában megoldhatóak.

Seregszemle

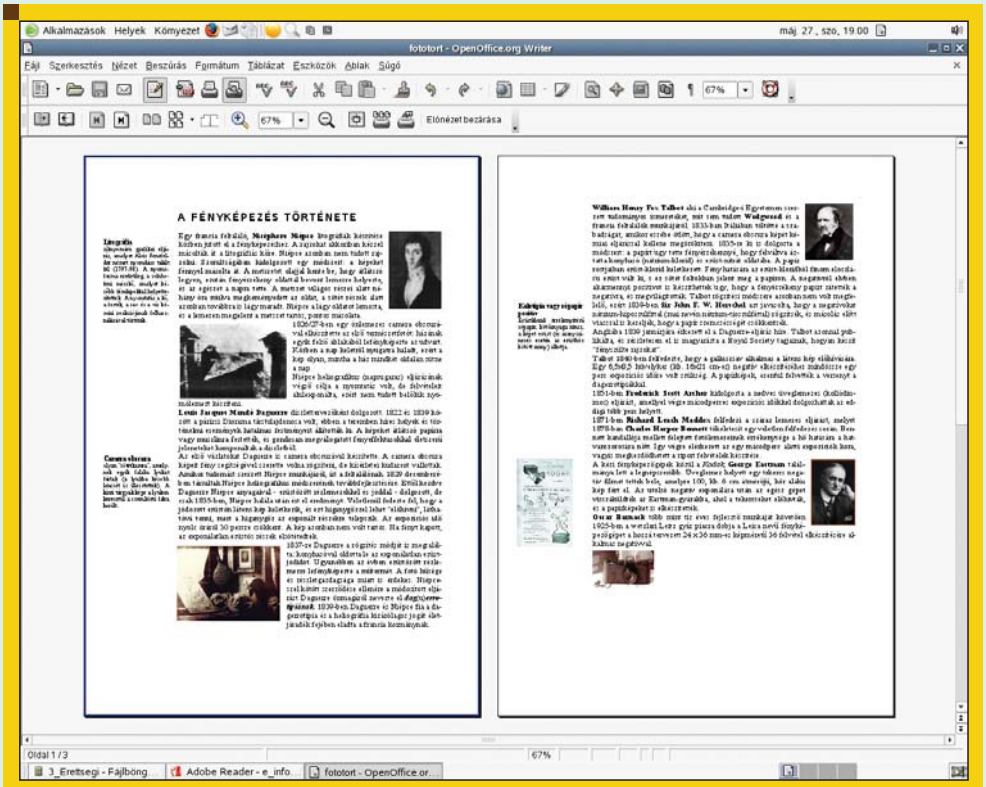
A már sokat emlegetett szoftverlista egy 2 évvel ezelőtti felmérés alapján készült, ahol minden középiskolának lehetősége volt jelezni, milyen környezetben tanulnak a diákok, illetve milyen környezetben képesek az informatika vizsgát lebonyolítani. Ennek köszönhetően minden középiskola megtalálhatja a listán a képzésének megfelelő szoftvereket, így az ott tanuló diákoknak nem okozhat problémát a vizsgára való felkészülés.
Az *OKÉV* szakemberei különösen nagy figyelmet fordítottak az ingyenes és szabad szoftverek áttekintésére, így a listán szereplő programokkal az érettségi részletes követelményében leírtak maradéktalanul megoldhatóak.
Ha egy-két programot mégse találunk a listán, annak az oka lehet az, hogy az adott alkalmazással nem teljesíthető az érettségi követelményrendszere, illetve ennél több szoftver esetén nem teljesülne az egységes értékelés és összevethetőség lehetősége, s így a kétszintű érettségi egyik alapelve sérülne.

Mely szoftverekkel szabad?

Természetesen az említetteken kívül – ahogy a szoftverlista is tartalmazza – más szoftverek is használhatóak, amelyek részét képezik a választott operációs rendszernek vagy az irodai programcsomagnak és megoldható velük az adott érettségifeladat. Ezt főként a *Linux* környezet miatt érdemes megemlíteni, ugyanis egy tipikus *Linux* rendszer is több alkalmas szoftvert tartalmazhat, egy *Microsoft Windows* rendszerrel ellentétben.
Fontos még megjegyeznem, hogy az említett szoftverek között szép számmal akadnak multiplatformosak (például az *OpenOffice.org* és a *Free Pascal* is), vagyis *Microsoft Windows*, sőt *MacOS X* rendszerre is elérhetők, így gyakorlatilag operációs rendszertől függetlenül megoldható az érettségi összes feladata és erről otthon mindenki bármikor maga is megbizonyosodhat.

Operációs rendszer

A *Linux* egyre erőteljesebb térnyerésének köszönhetően az oktatásban is nagy szerepet kaptak a szabad, ingyenesen felhasználható és



1. ábra Szövegszerkesztési feladat megoldása SuSE Linux és OpenOffice.org…

terjeszthető alkalmazások. Hazánkban is (országszerte) egyre több oktatási intézmény használ *Linux* operációs rendszert mind az oktatói tevékenység színesítésére, mind rendszerüzemeltetési feladatok ellátására. Ennek köszönhetően az operációs rendszerek között szerepel több népszerű *Linux* disztribúció is, mint például a *Debian* és a *SuSE*. Így biztosan a számunkra legmegfelelőbb rendszer (akár *MacOS X*) használatával maturálhatunk.

Szövegszerkesztés és táblázatkezelés

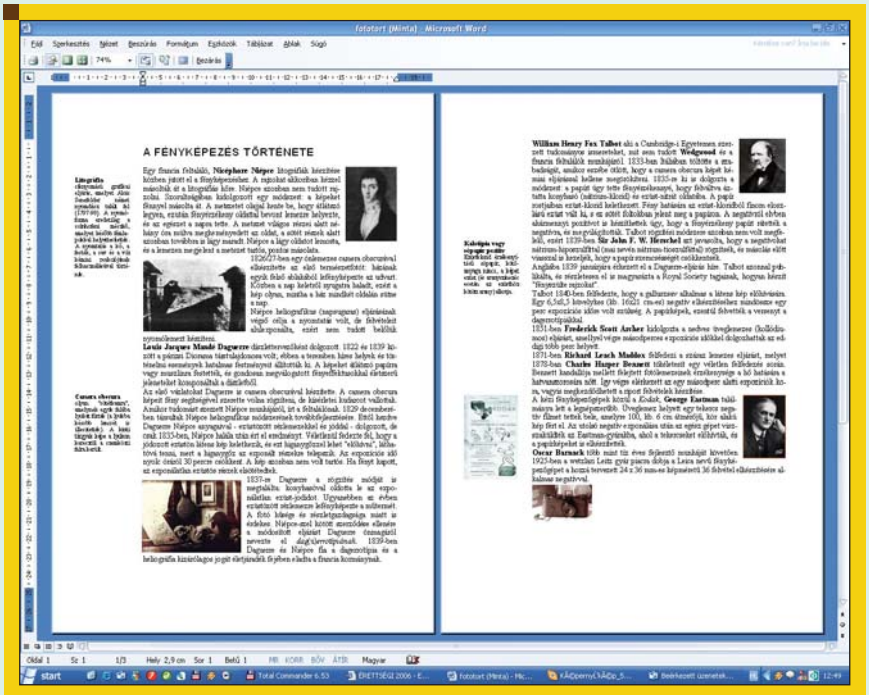
Irodai programcsomagok közül az *OpenOffice.org* és a *StarOffice* áll rendelkezésünkre, melyekkel a szövegszerkesztési és a táblázatkezelési feladatok is játszva megoldhatóak. Idén szövegszerkesztés keretében szöveg és kép fájlok segítségével kellett összeállítani, majd a megadottak szerint formázni egy 3 oldalas dokumentumot. Mivel a táblázatkezelés a vizsga legkisebb témaköre, így erre jár a legkevesebb pont és a feladat megoldása is viszonylag egyszerű függvények, illetve képletek alkalmazását kívánja meg. Idén számok maradékkal történő osztása majd összeadása állt a feladat középpontjában.

Weblapkészítés

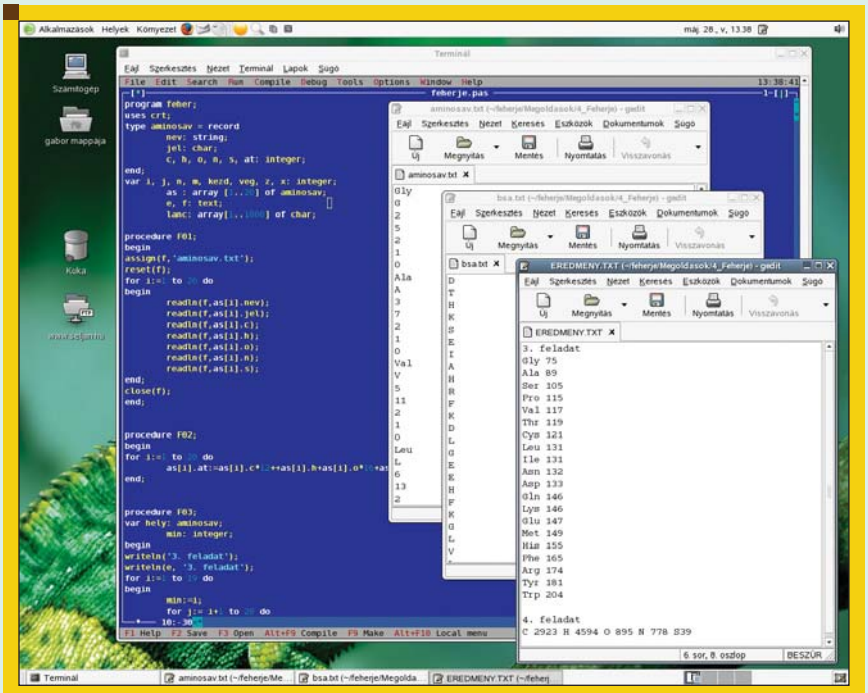
Ugyan emelt szinten ilyen feladat idén nem volt – helyette volt a szövegszerkesztés –, de weblapkészítésnél a *Mozilla Composer* illetve a *Netscape Composer* használható, melyek „Amit látsz, azt kapod” (*WYSIWYG*) módban is, teljes mértékben eleget tesznek a feladatok megoldásához szükséges követelményeknek. Alkalmazásukkal nem szükséges *HTML* kódot írnunk egy *Jegyzettömb* vagy *gedit* szintű egyszerű szövegszerkesztővel, ami – tudásszinttől függetlenül – lényeges időmegtakarítást jelenthet egy vizsgán. A grafikai feladatoknál a *Gimp*, vagyis az *Adobe Photoshop* legkomolyabb szabad szoftveres alternatívája siet a segítségünkre, bár ezek a feladatok most nem igényelték a program használatát.

Adatbázis-kezelés

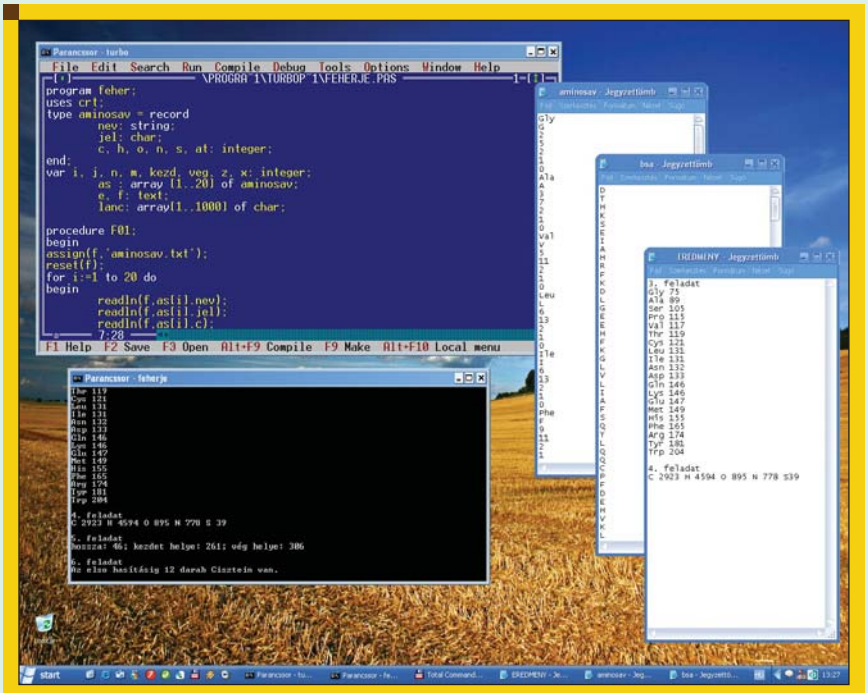
Emelt szinten az adatbázis-kezelési feladatok megoldása jelentős többletismeretet igényel a vizsgázóktól, nem csak a feladatok, hanem az adott szoftverkörnyezet miatt is. Az adatbázis rendszerek közül a *PostgreSQL* és a *mySQL* alkalmas a követelmények teljesítésére. Tehát a lista alapján ezek közül választhatunk hogy *OpenOffice.org* –, vagy *StarOffice*



2. ábra illetve Microsoft Windows és Microsoft Word használatával.



3. ábra Algoritmizálási feladat SuSe Linux és Free Pascal környezetben…



4. ábra illetve Microsoft Windows és Turbo Pascal használatával.

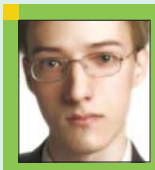
és ***knoda*** (grafikus adatbázis-kezelő ***KDE*** rendszerhez) segítségével kívánunk nekiveselkedni a feladatnak. Idén komplexebb, több táblás rendezett illetve szűrt lekérdezéseket (mindösszesen tízet) kellett készíteni egy adott adatbázison, ami érettségiző diákok, és az egyes érettségi vizsgák adott adatait tartalmazta.

Programozás

Nem kérdéses, hogy az emelt szintű vizsgán az algoritmizálás és adatmodellezés kapja a legnagyobb hangsúlyt, ami középszinten nem is szerepel. A legtöbb pont is ebben a témakörben szerezhető. A feladat adott, a fejlesztői környezet pedig széles körben választható, legyen az a ***Pascal***, ***C***, ***Visual Basic***, vagy éppen ***Perl***. A számos lehetőség mellett – érettségiző szempontból – a legjobb választás talán mégis a ***Pascal*** használata lehet, mivel az interneten található mintafeladatok illetve később a megoldások is általában csak ***Pascal*** környezetben érhetőek el. Idén a fehérjéket alkotó aminosavakhoz kapcsolódóan egy-egy szövegfájlból beolvasott adatok alapján kellett bizonyos műveleteket elvégezni, mint például molekulatömeg számítása, összegképlet meghatározása, majd a kapott adatokat rendezni, összegezni és a képernyőn megjeleníteni illetve fájlba írni.

Summa summarum

A szabad szoftver nap mint nap bizonyítja, hogy nem csak professzionális fejlesztési és mindennapos irodai felhasználás során alkalmazható, hanem a közoktatásban és akár az érettségi vizsgán is megállja helyét a már elterjedt kereskedelmi termékekkel szemben.
Egyre több szabad szoftverre épülő oktatási anyag készül, melynek köszönhetően hamarosan ***ECDL*** jogosítvány is szerezhető ***Linux*** és más alternatív ingyenes programok használatával. Ez a további térnyerés az oktatásban, nagymértékben megkönnyíti a diákok több lépésben történő felkészítését egy informatika érettségire, így majd bátran vállalhatják a vizsga letételét egy szabadabb környezetben is.



Selján Gábor
(gabor@seljan.hu)

Szabadúszó Flash webfejlesztő és a szabadidejében hobbi Linux felhasználó.

KAPCSOLÓDÓ CÍMEK

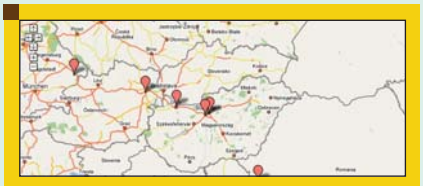
Az idei emelt szintű feladatsor
➲ http://193.225.13.22/e_info_06maj_fl.pdf

és annak megoldása
➲ http://193.225.13.22/e_infomegoldas_06maj_ut.zip

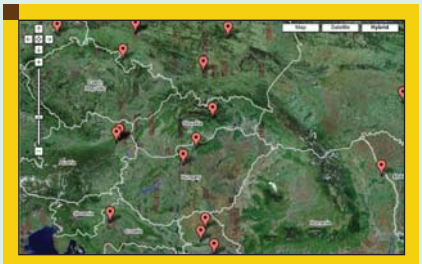
Kódolás egy nyáron át – Google Summer of Code

A keresőóriásnak azon kívül, hogy több, mint negyedmillió számítógépen használ általa testreszabott Linuxot, más kapcsolata is van a szabad szoftverekkel: nemcsak lefölözi a számára érdekes fejlesztésekből a hasznot, hanem vissza is juttat erőforrásokat.

2005-ben több, mint 400 diákot fizetett a *Google*, hogy egy általuk érdekesnek tartott szabadszoftveres problémán dolgozzanak. Idén május 23-ig kellett leadni a pályázatokat a részvételhez, mire az újság olvasóink postaládájába kerül, a diákok a munka derekánál tartanak. A különféle mentorszervezetek (tipikusan egy programhoz kötődően) kiírtak tervezeteket olyan feladatokról, amelyeket szívesen látnának megvalósulni. Példaképpen az *Apache* a *CARP* (*Cache Array Routing Protocol*) implementálást, az *openSUSE a GTK+* alapú *YaST* felület megvalósítását javasolta – többek közt – a lelkes diákoknak.
A diákok számtalan efféle ötlet közül válogathattak, illetve kitalálhattak számukra (és mindenki más számára) érdekes feladatot. A diákok megírták a saját elképzeléseiket a megvalósításól, aztán a mentorok (jellemezően az adott projekt fő fejlesztői) értékelték a pályázatokat, elbírálták, hogy ki az, akinek érdemes bizalmat szavazni a megvalósulást illetően. Tavaly is, idén is pályáztak magyarok, sőt mit több, a tavalyiak el is végezték a megkezdett munkát. A *Google* a diákok erőfeszítéseit 4500$-al, a mentorokét 500 dollárral jutalmazza. Az előző évben két sikeres terv megvalósítója került ki hazánkból: *Henk Csaba* a *FreeBSD* számára készített *SSH* hálózati fájlrendszert és *Elek Márton* új generációs feliratkozás modult készített a *Drupalhoz*.
Idén 6297 pályázat érkezett be a különféle mentor szervezetekhez, ebből körülbelül 630-at fogadtak el. Követve az előző év szép szereplését, idén is vannak magyar résztvevők.



1. ábra Azon tanulók, akik a PlanetSoC weboldalon regisztrálták magukat. Lapzárta előtt nem állt rendelkezésre a tavalyihoz hasonló teljes térkép



2. ábra A sikeresen teljesítő tanulók hazánkban és környékén (2005)

Török Edvin a *clamAV* keretein belül adathalászat elleni megoldásokon dolgozik. *Kövesdán Gábor a FreeBSD ports* gyűjteményének javítását választotta feladatul. *Varga Gábor LDAP* támogatást készít a *Gallery2-höz*. *Rekedt-Nagy József* szintén a *Gallery2-nél* hatékony fabejáró algoritmust implementál.
Az *Inkscape-nél Erdélyi Miklós PDF* exportálást készít. *Lehel Gábor Qt4* alapú *widgeteket* ír a *KOffice 2.0*-hoz. *Szilakszi Bálint* a *Perl 6*-hoz megírja a *DBI* modult. *Farkas Szilveszter* a *Bazaar-NG* verziókövető rendszerhez ír grafikus felületet. Végül e cikk írója a *Drupalnak* fejleszt egy szociális hálózatokat kezelő és megjelenítő modult. Remélhetőleg a résztvevők, akik így bepillantást nyernek a szabad szoftverek fejlesztésének világába, nem fogják szeptember 1-től letenni a billentyűzetet, hanem továbbra is figyelemmel és érdeklődéssel fordulnak a projektjük, a mentor szervezetük és egyáltalán a szoftverfejlesztés efféle szemlélete felé. Van, aki úgy véli, naivitás ezt remélni, sőt, olyan is akad, aki a tavalyi eseményt is bukásnak tekintette (➲ http://weblogs.mozillazine.org/gerv/archives/2006/03/summer_of_code_six_months_on.html).
Valóban lehet, hogy a régi projekteket megnézve nem túl rózsás a helyzet, de az is kétségtelen, hogy több idei mentor tavaly nyertes tanulóként dolgozott. Azaz a szabad szoftveres fejlesztők és csapatépítők tábora bővült. Nyilván lehet százalékokon gondolkodni, osztani és szorozni, a *Google* úgy döntött, idén is megrendezi a *Summer Of Code-ot*. Bizonyos, hogy a keresőóriásnak és a szabad szoftveres világnak nagyon jó alkalom ez a marketingre, ha emellett valódi újdonságokat is kapnak a végfelhasználók (a pályázatokba érdemes volt beleírni, hogy mit nyernek ezzel az adott szoftver használói), az mindenképp jó hír!

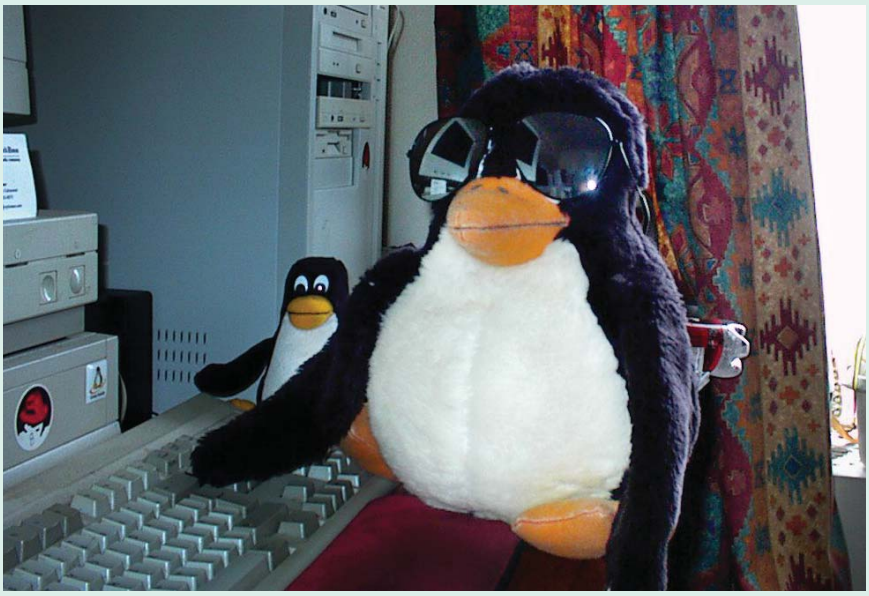
Novák Áron (aaron@szentimre.hu)

BME-VIK-es hallgató, műkedvelő rendszergazda. Jelenleg leginkább a NetBeans-szel és mindenféle hordozható eszközzel foglalkozik, legalábbis mindazokkal, amelyeket meg lehet szólaltatni Linux alatt.

A pingvin és a tű – avagy a Pingvin Projekt

Félreértések elkerülése végett nem narkomán szárnyasok megmentésének szövetségéről szól a tudósítás, sem szénakazalbéli pajzán homo és hetero madártörténetekről – sokkal személyreszabottabb a híranyag: szabjuk személyekre a linux-pingvint!

Megszületett a ***Magyar Pingvin Projekt*** (➲ http://pingvin.mediacenter.hu). Egy csapat plüsspingvinért áhítozó toboroz szertefele lelkesen társakat, hogy tűt, cérnát, ollót és varrónőt ragadva saját bolyhos madarat birtokolhasson. Próbáltam megfejteni



a rejtélyt, ágyvizelős kort túlhaladva mi késztet (majdnem) felnőtt személyeket plüssállatkák (majdnem) mindenáron való megszerzésére, s mielőtt még azt a végkövetkeztetést vonnám, hogy tán némelyek sokkal későbbi leszármazottai a csimpánznak, mint mások, az alábbi kevésbé szigorú találgatásokba bocsátkozom: Feltehetően a csapat ismeretségi körében rövidáruval kézimunkázó varrónénik hosszú combokkal is rendelkeznek. A rövid- és/vagy hosszú távú tervezet elegendő alkalmat nyújt gondos és rendszeres eszmecserék lebonyolítására, melyben a munka hatékonyságát növelendő gondosan és rendszeresen cserélődhetnek a rövid- és hosszúitalok. Természetesen az eszme is jelen van: „Bárki bármennyi munkát el tud végezni, feltéve, ha ez nem az a munka, amit éppen csinálnia kell." No de én, ki már nyakig merültem a pingvinek székelési paramétereibe, tüntetést vizslattam homoszexuális szárnyasok jogaiért, tiszteltem *Nils Olavnak*, a norvég hadsereg pingvinezredesének, szurkoltam a japán madarak sikeres távfutó-mérkőzésénél, kötelességemnek éreztem ismét leoldani az előítéletek láncait, és figyelemmel kísérni a szabásmintákat.

A tervrajzokat tekintve olykor meg kellett állapítanom, ha szép a varrás, még nem biztos, hogy jól sikerült a műtét… Némi fantázia és praktikum elrugaszkodottságára vallott, midőn pingvineket kedvenc háziállatokkal (például kutya, macska, szomszédbácsi) elegyítettek; a kitágult elmék a méreteknek sem mindig szabtak határokat, néhol meg tán a szomszédbácsi ült rá a mintapéldányra.

Ám a játszma nincs elvesztve, míg meg nem nyertük alapon megszülettek a hazai prototípusok is, a projekt életrehívói és támogatói pedig nagyszabású terveikhez továbbra is toborozzák a szabókat-varrókat s leendő Tux-tulajdonosokat.

Halusz Léna

KDevelop

Integrált fejlesztői környezet KDE alapokon

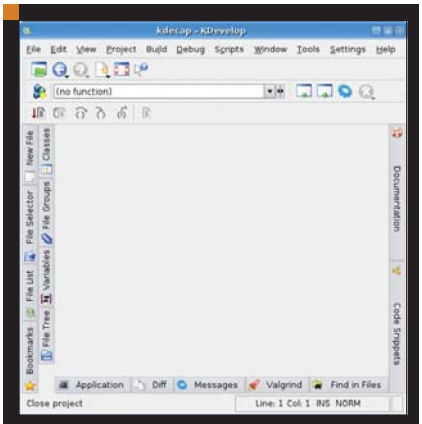
Sokaktól lehetett és néha még mostanság is lehet hallani, hogy a Linux egyik nagy hátránya a kiforrott, magas szintű, könnyen kezelhető integrált fejlesztői környezet(ek) hiánya. Ebben az írásban megpróbáljuk megmutatni, hogy ez a kijelentés ma már egyáltalán nem állja meg a helyét. A Linux alatt ma elérhető fejlesztői környezetek közül itt a KDE alapokon nyugvó KDevelop-ot mutatjuk be.

Ismerkedés

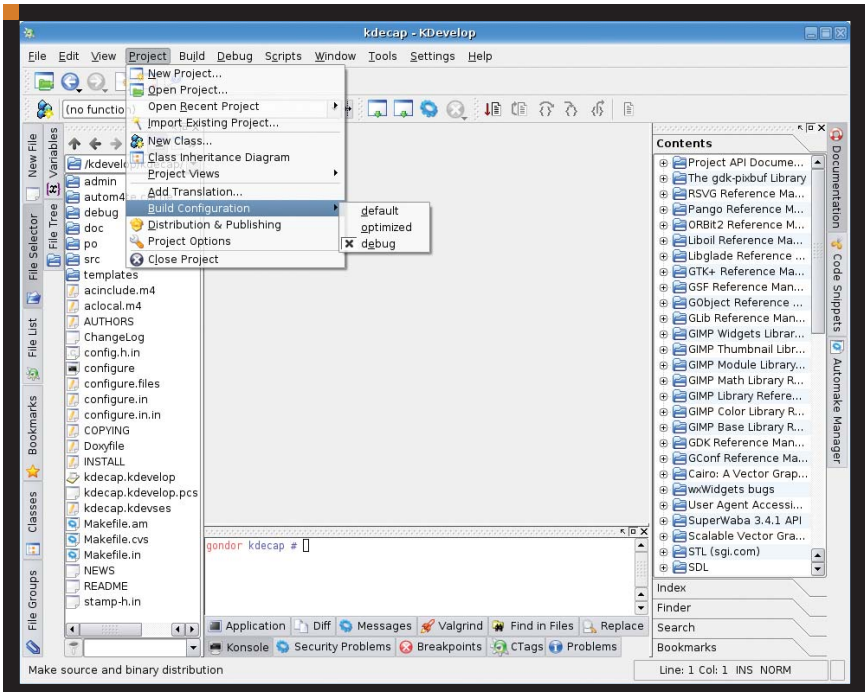
Más operációs rendszerek alatt hosszú évek óta számos alkalmazásfejlesztést segítő eszköz érhető el, ma már inkább hiányuk volna szokatlan. *Linux* alatt, noha legalább ugyanolyan régen elérhetők fejlesztést segítő eszközök és környezetek, kiforrott, magas funkcionalitású, valódi integrált fejlesztői környezetre viszonylag sokáig kellett várnunk. A *KDevelop* egy ilyen környezet, amely több programozási nyelven történő fejlesztést támogat egy rengeteg funkcióval ellátott alkalmazásban. Fő erőssége *KDE/QT* alkalmazások fejlesztése, amely segítéséhez integrált grafikus felületszerkesztőt is tartalmaz (*C++*-ban írt *KDE* alkalmazás esetén, amelyre mi is koncentrálni fogunk).
A *KDevelop* első verziója 1998-ban jelent meg, a *KDE1/QT1* könyvtárakra épülve, a *KDE* grafikus asztali felület részeként. Ezután folyamatos fejlődésben volt része, kétszer is teljes átíráson esett át (*QT1-2* ill. *QT2-3* verzióváltáskor). A ma elérhető stabil verzió a *KDE 3.5.x* része, a *QT3.x* könyvtárakra épül. A hármas verzió óta a *KDE*-s alkalmazásfejlesztési filozófiát követve teljesen *plugin*-alapú, minden eleme a *KParts* technológia révén épül be a *KDevelop*-ba.
Még a forráskód-szerkesztő is, ami a *KWrite*-ban is használt szövegszerkesztő. Ebben az írásban a *KDevelop 3.3.1* verzióját használjuk. Telepítési részletekben nem mélyülünk el, mert *kdevelop* néven minden *KDE*-t támogató disztribúcióban megtalálható.
A *KDevelop* használatához feltétlenül szükséges közvetlen függőségek mellett (amelyek automatikusan települnek a *kdevelop* csomaggal) néhány funkció eléréséhez egyéb külső alkalmazásokra lesz szükség, amelyekről a belső eszközök és funkciók leírásánál szót fogunk ejteni. Az írást nem teljesen kezdő programozóknak szánjuk, sokkal inkább azon olvasóknak, akiknek van már némi programozási tapasztalatuk ill. használtak már fejlesztői környezeteket így rendelkeznek némi összehasonlítási alappal.

Külső

A *KDevelop* elindításakor első ránézésre egy kopár felülettel találkozunk (*1. ábra*). De ez megtévesztő, a puritánnak tűnő külső igazi ezermestert takar. Az *1. ábrán* is látható, hogy a főablak két oldalán és alján találhatók az ú.n. *Tool Dock*-ok. Ezek helye



1. ábra A KDevelop felülete üresen



2. ábra File Selector, Documentation és Konsole view-k megnyitás után

```
183
184   void kdecapWidget::v4l_device_textChanged(const QString&)
185   {
194
183
184   void kdecapWidget::v4l_device_textChanged(const QString&)
185   {
186       v4l_str=v4l_device->text();
187       int pos=menc_pars.find(":device=");
188       if(pos!=-1) {
193   }
194
183
184   void kdecapWidget::v4l_device_textChanged(const QString&)
185   {
186       v4l_str=v4l_device->text();
187       int pos=menc_pars.find(":device=");
188       if(pos!=-1) {
189           int pos2=menc_pars.find(":",pos+1);
190           menc_pars.replace(pos+8,pos2-pos-8,v4l_str);
191           mencoder_params->setText(menc_pars);
192       }
193   }
194
```

3. ábra Elágazások összevonva (fent) és kibontva (lent)

szabadon felcserélhető. Az egyes *dock*-okhoz ú.n. *View*-k tartoznak, amelyek címkéire kattintva ezek az adott *dock*-on belül megnyílnak (2. *ábra*). Ezekben a *dock*-okban rengeteg olyan funkció található, amelyek a munkát segítik (ezekre a későbbiekben térünk ki). Ez az elrendezési megoldás lehetővé teszi, hogy gyorsan elérhetők legyenek és hosszabb használat után nagyon könnyű ezek pozícióit megjegyezni. A *dock*-ok pozícióit és a *view*-k megjelenését a *View* menüből módosíthatjuk. A legfontosabb *view*-k: fájl rendszer (*File Selector*), *File Tree* (a projektünk áttekintése), *Classes* (a projekt osztályai), könyvjelzők (*Bookmarks*), változók futás közbeni értékkövetése (*Variables*), stb.

Beállítások, személyre szabás

A *KDevelop* funkcióit a *Settings* menü *Configure KDevelop* és *Configure Editor* pontjaiban állíthatjuk be. A főablak *dock*-jait és *view*-jait a már említett *View* menüben, az eszköztárakat pedig a *Settings->Toolbars* menüpontban. A *KDevelop* menüinek tartalma változhat, attól függően, hogy milyen programozási nyelvet használ az aktuális projektünk. Az adott nyelv esetén nem elérhető funkciók nem kerülnek be a menükbe. C/C++ esetén érhető el a legtöbb eszköz és funkció.
Egy fontos pont a dokumentációk beállítása (*Settings->Configure Kdevelop ->Documentation*), ahol azt adhatjuk meg, hogy az elérhető dokumentációk közül melyiket használja a *KDevelop* a *Help* összeállításához. A *Help*-et a jobboldali *dock*-on található *Documentation view*-ban érhetjük el, ahol a dokumentáció generálása után böngészhetünk, kereshetünk. A dokumentáció generálásához fel kell telepítenünk a *htdig* csomagot, amelyet majd a *KDevelop* használni fog. Ezen kívül számos formátumozási, szintaxis-kiemelési, szókiegészítési tulajdonságot is személyre szabhatunk.

Támogatott nyelvek

A *Project->New Project* menüpontban kiválaszthatjuk, hogy milyen projektet szeretnénk készíteni. A *KDevelop* számos programnyelvet támogat, többek között *C/C++, Fortran, PHP, Perl, Ruby, Python, Java, Shell script*, stb. Ezek használatához az adott nyelv könyvtárait és fordítóit nekünk kell telepítenünk. *C++*-ban történő *KDE* alkalmazásfejlesztéshez például a *QT* és a *KDE* könyvtárakra és ezek fejlesztői változataira (*-dev* csomagok) van szükség, ill. *gcc/g++* fordítóra. A *QT* könyvtárak lehetővé teszik számos nyelvben történő felhasználásukat, így természetesen nem csak *C++*-ban készíthetünk *KDE*-s alkalmazásokat. Ebben az írásban a *C++*-ban történő fejlesztésre koncentrálunk.

Szerkesztő

Röviden vegyük sorra a *KDevelop* legfontosabb eszközeit, amelyekkel segíti az alkalmazásfejlesztést. Kezdjük talán az egyik központi elemmel, a szövegszerkesztővel. A *KDevelop* szövegszerkesztője mindegyik támogatott nyelvvel képes a szintaxis-kiemelésre. Az egyes elágazások (feltételes utasítások, ciklusok, stb.) összevonására és kibontására is lehetőséget ad, megkönnyítve a kód áttekintését (3. *ábra*).

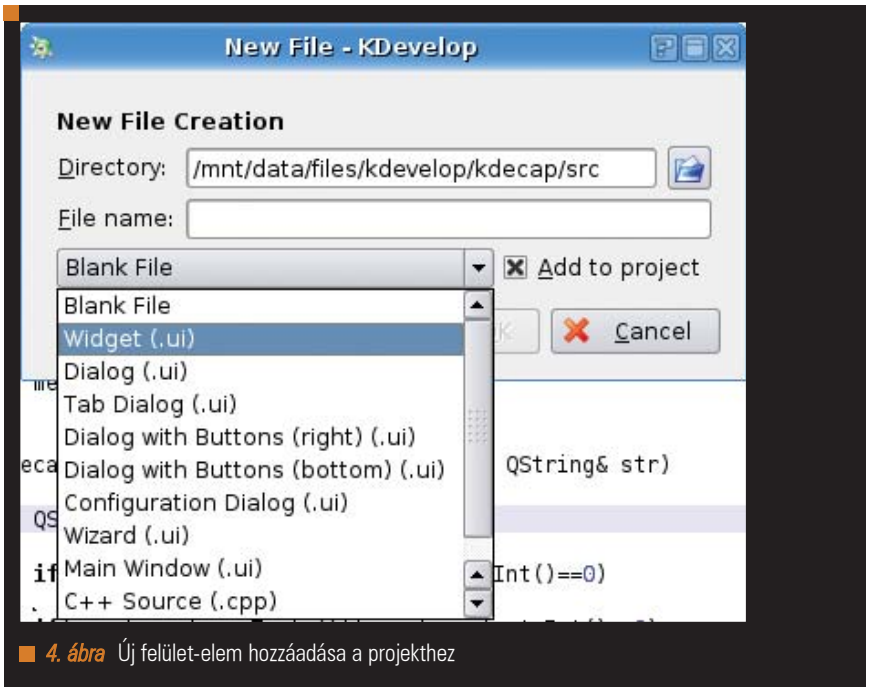
C/C++ kód esetén a ***KDevelop*** néhány karakter begépelése után lehetőségeket kínál a szó automatikus kipótlására (ú.n. ***autocomplete*** funkció), amellyel gyorsabbá válik a kód írása. Más nyelv esetén a szövegszerkesztő szókipótló funkcióira támaszkodhatunk, amelyet a ***Settings->Configure Editor->Plugins*** menüpontban állíthatunk be.

Felület-tervezés

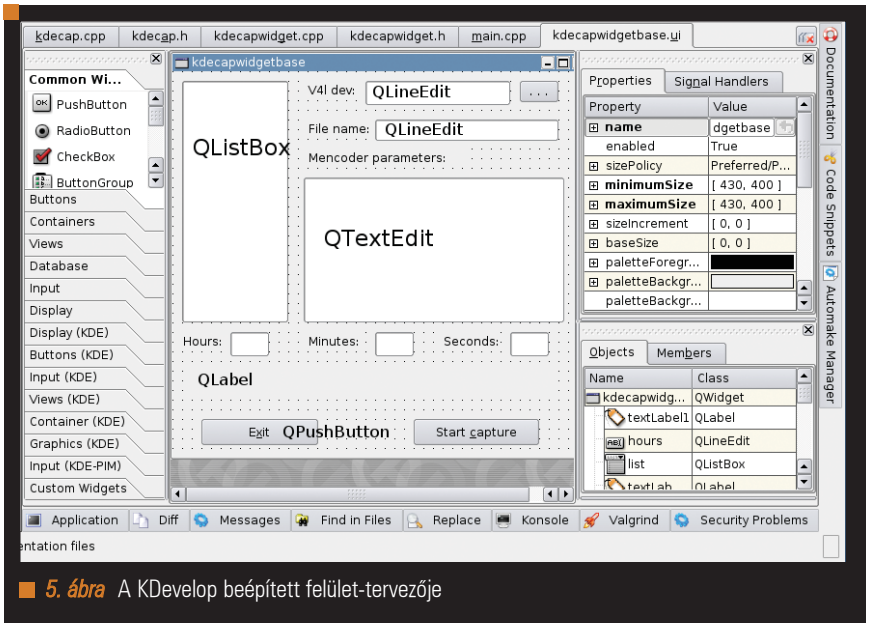
C++-ban készülő ***KDE***-s alkalmazás készítésekor lehetőségünk van vizuálisan szerkeszteni az alkalmazás felhasználói felületét. A 3.x-verziójú ***KDevelop*** beépített ***GUI*** szerkesztővel rendelkezik (korábbi verzióknál erre egy külső alkalmazás, a ***QT Designer*** szolgált). A felületszerkesztő használatához vagy létrehozunk egy ú.n. ***Designer based KDE application***-t, amihez rendelődik egy alapfelület, vagy hozzáadunk létező alkalmazásunkhoz egy felület-elemet.
Ez utóbbit a ***File->New*** menüpontban tehetjük (*4. ábra*).
A felületeket tartalmazó fájlok .ui kiterjesztéssel jelennek meg a projektünkben. A baloldali ***dock->File Groups view***-jában a ***User Interface*** alatt találhatjuk. Az ***.ui*** fájlra kattintva megjelenik a szerkesztő, a ***KDevelop*** szerves részeként (*5. ábra*).
A felületszerkesztőben bal oldalon elérhetjük a felhasználható számos ***widget***-et, amelyeket elhelyezhetünk a készülő felületen, egy-egy lehelyezett ***widget***-re kattintva pedig jobb oldalon szerkeszthetjük a hozzá tartozó paramétereket és tulajdonságokat. Az egyes ***widget***-ek eseményeit (például gomb lenyomása, lista elemének kijelölése, stb.) is innen lekezelhetjük, mégpedig az adott ***widget***-en jobbklikk->***Connections*** kiválasztásával (*6. ábra*). Lehetőségünk van kiválasztani az eseményt fogadó osztályt és a megadni a kezelő függvényt. A ***KDE/QT*** fejlesztői terminológia szerint az eseményt ***signal***-nak, a fogadó/kezelő függvényt pedig ***slot***-nak nevezzük. További eszközöket találhatunk a ***Layout*** és a ***Tools*** menüben.

Szabályos kifejezések szerkesztése

Szabályos kifejezések (***regexp***) kezelésekor (jellemzően szövegfeldolgozási feladatok során) gyakran még haladó programozók is elővesznek egy

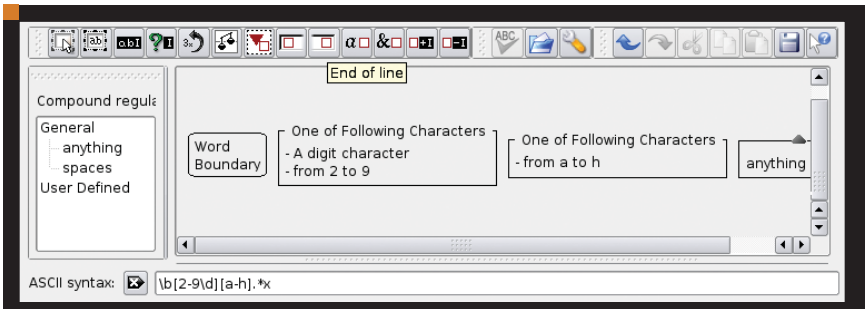


4. ábra Új felület-elem hozzáadása a projekthez

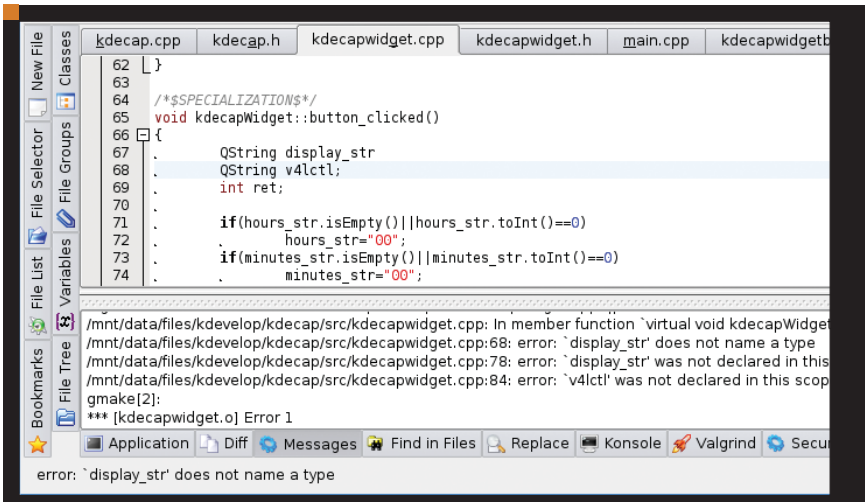


5. ábra A KDevelop beépített felület-tervezője

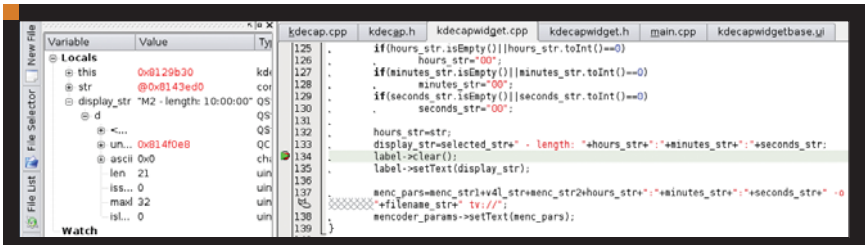
6. ábra Lehelyezett widget eseményéhez kezelőfüggvény rendelése



7. ábra Grafikus/vizuális reguláris kifejezés szerkesztő



8. ábra Hibakeresés – hibára kattintva a hibához ugorhatunk



9. ábra Futás közbeni változókövetés

10. ábra Könyvjelzők használata (baloldali dock-ban) ill. kódrészletek elmentése refactoring-hoz

dokumentációt, könyvet, leírást, vagy egy internetes keresőt, hogy a megfelelő kifejezést állítsák össze a megdöbbentően sokrétű lehetőségek közül. A *KDevelop* ennek segítésére egy vizuális, grafikus felületű reguláris kifejezés szerkesztőt tartalmaz (*7. ábra*), ami a ***Tools->Debug Regular Expression*** menüpontban érhető el (majd a felbukkanó ablakban az Edit kiválasztásával).

Fordítás, futtatás, hibakeresés

Egy megírt programot a ***Build*** menüben tudunk lefordítani ill. futtatni. Ehhez szükségünk lesz az ***automake*** csomagra (a ***kdevelop*** csomaggal együtt települ). A ***Build->Run automake*** menüpont legenerálja az összes dependenciát ami a programunk futtatásához szükséges és létrehoz egy ***configure*** szkriptet amivel mi is és más rendszereken is ellenőrizhetjük ezek meglétét. A program lefordításához szükséges ***makefile***-t is előállítja. Ezután a ***Build->Run Configure*** és ***Build->Build project*** parancsokkal lefordíthatjuk a projektünk és elindíthatjuk a programot. A fordítási kimeneteléről az alsó ***dock Messages*** nevű *view*-jában kapunk információkat. Hibák esetén a hibaüzenetre kattintva az adott sorra ugrik a kurzor (*8. ábra*).

Változókövetés és nyomkövetés

A ***debugger*** használatával (***Debug*** menü, a külső ***gdb*** alkalmazást használja) lehetőségünk van a megírt programunk futás közbeni nyomon követésére. A változók értékeinek megfigyeléséhez a szerkesztőben egy soron jobbklikk->***Toggle Breakpoint*** paranccsal helyezhetünk le megállási pontokat. Futás közben a ***Debug*** menü pontjaival léphetünk a kódban. A megfigyelni kívánt változókat jobbklikk->***Watch*** paranccsal adhatjuk a változókövetési ablakhoz, amelyet a baloldali ***dock***-on találunk ***Variables*** néven (*9. ábra*).

Könyvjelzők

Lehetőségünk van könyvjelzők elhelyezésére a kódban (***Bookmarks->Set bookmarks*** menüpont, vagy ***Ctrl+B***), amelyek a baloldali ***dock Bookmarks view***-jába kerülnek és egy kattintással a kód adott pontjára ugorhatunk (*10. ábra*).

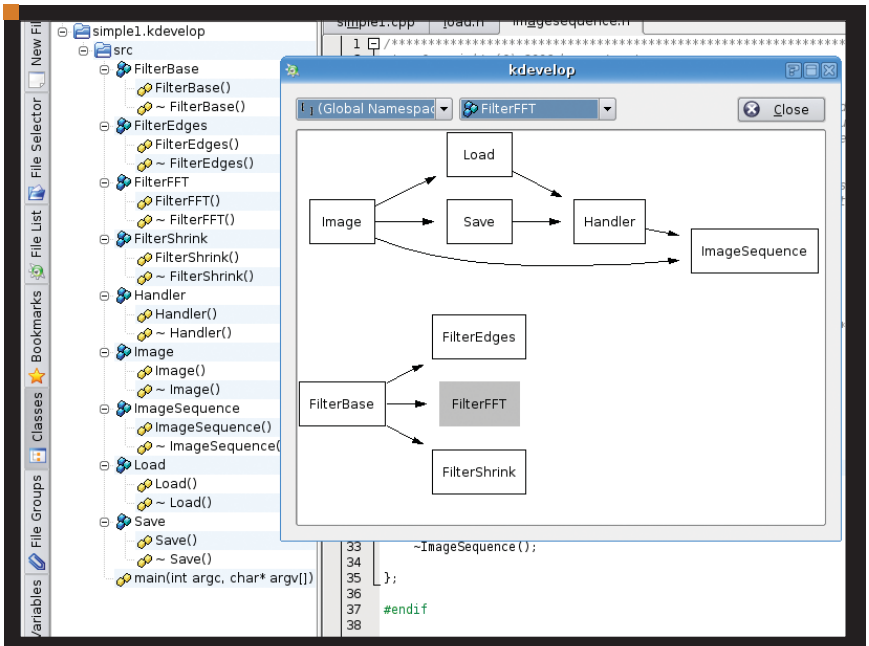
Kód-újrafelhasználás

Szintén egy nagyon fontos képesség, hogy ú.n. ***refactoring***-ot (kód újrafelhasználás) segítendő kódrészleteket kiemelhetünk a forrásból és egy kód listához adhatjuk, ahonnan később könnyedén felhasználhatjuk ezeket a jobboldali ***dock Code Snippets view***-jából (*10. ábra*). A kódrészletek hozzáadásához meg kell nyitnunk a ***Code Snippets view***-t majd jobbklikk->***Add Item***-et választanunk.

Osztályok

Programtervezéshez egy fontos eszköz a ***Project*** menü ***Class Inheritance Diagram*** pontja, amely a külső

doxygen és *graphviz* alkalmazás használatával egy dialógusban megjeleníti készülő alkalmazásunk osztályhierarchiáját (*11. ábra*). Új osztályokat a *Project->New Class* menüpontjával adhatunk hozzá a projekthez, amelyben az osztály minden tulajdonságát könnyen szerkeszthetjük. Az egyes osztályokat a baloldali *dock->Classes view*-jában érhetjük el.



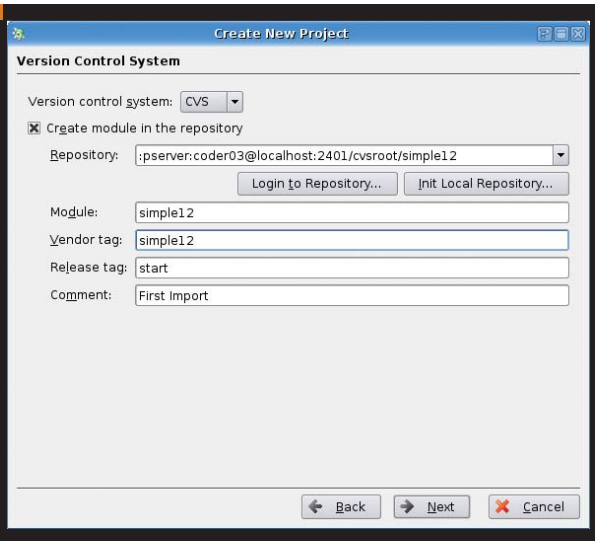
11. ábra Osztályhierarchia megjelenítése

Verziókövetés

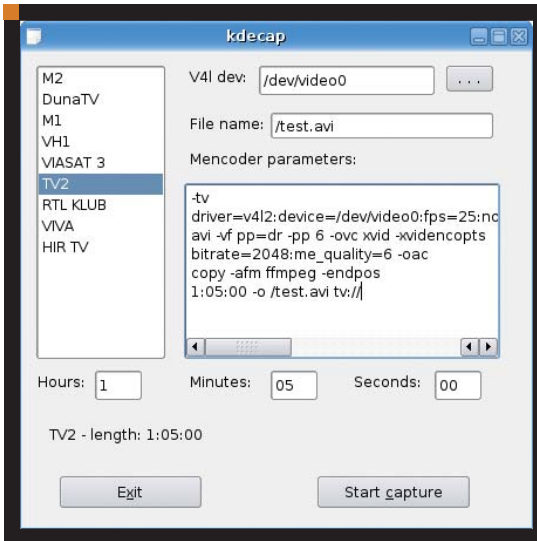
Nagyobb projektek, főleg ha többen részt vesznek programírásban, elképzelhetetlenek valamiféle verziókövetési rendszer használata nélkül. Ennek megfelelően a *KDevelop* is támogat verziókövetési szerverekhez történő csatlakozást és ezek használatát. Projekt létrehozásakor az első lépéseknél már lehetőségünk van valamelyik támogatott verziókövető szerverhez csatlakozni ill. megadni az eléréseket (*12. ábra*). Már létező projekt esetén a *Project->Project Options->Version control* pontban állíthatók be a használni kívánt verziókövető rendszer adatai. A *KDevelop CVS*-t, *SubVersion*-t (*SVN*), *Perforce*-t és *Clearcase*-t is támogat *plugin*-eken keresztül.

Példa: KDE-s alkalmazás C++-ban

Mivel ez az írás nem a *KDE*-s alkalmazásfejlesztésről szól, hanem a *KDevelop* bemutatásáról, nem szándékozunk elmélyülni a *KDE/QT* könyvtárak is programozási interfészek (*API*-k) lelkivilágában. Tekintsük inkább át egy egyszerű dialógus-alapú *KDE*-s alkalmazás elkészítésének egyes lépéseit. Nevezzük ezt az alkalmazást *kdecap*nek. Célja az, hogy tv *tuner* egy csatornájának adását rögzítse. A *13. ábra* mutatja az elkészült alkalmazást. Egy dialógusablak köré épül, amiben néhány widget-et (gomb, lista, szövegbevitel) helyezünk el a rögzítési paraméterek beállításához. Rögzítéshez a *mencoder*-t használjuk, ami az *Mplayer* csomag része. Létrehozunk egy *Simple Designer* based *KDE Application*-t a *Project->New Project* menüpontban. A baloldali *dock Classes*, *File Tree* és *File Groups view*-iban átböngészhetjük a létrejött projekt összes elemét. Majd a *File Groups*-ban válasszuk ki a *User Interface* alatti .ui fájlt, és a felülettervezőbe kerülünk. A *5. ábrán* látható módon *widget*-eket helyezünk el: *QListBox*, *QTextEdit*, *QLineEdit* ill. *QPushButton* elemeket. Az egyes elemekhez szükség lesz események kezelésére: ha az óra, perc, másodperc mezőkben megváltozik az adat, ha a listában megváltozik a kijelölés, ha megváltozik a *tuner* eszköz kijelölése, vagy ha lenyomnak egy gombot, ezekről mind tudnunk kel. Egy *widget*-re



12. ábra Verziókövetési paraméterek



13. ábra A példa alkalmazás futás közben

kattintva az szerkesztőablak jobb oldalán a ***Signal Handler*** fülön láthatók a gomb eseményei (*14. ábra*). Kiválasztva egyet (pl. a ***clicked*** eseményt) megadhatunk egy kezelőfüggvényt, majd oda is ugrunk a függvény törzséhez.
Például ha a ***tuner*** eszköz melletti „. . .” feliratú gombra kattintanak, akkor nyissunk egy dialógus ablakot amiben a felhasználó kiválaszthatja a ***tuner*** eszközt amiről rögzíteni szeretne. Az ezt kezelő függvény a *14. ábra* jobb oldalán látható, a megnyitott dialógus pedig a *15. ábrán*.
Sajnos a teljes forrás túl hosszú lenne, így utolsó példaként álljon itt a rögzítés elindításának kezelése. A lehelyezett, és az ábrán ***Start Capture***-nek nevezett gombra jobbklikkelve és a ***Connections***-t kiválasztva rendeljünk a gomb ***clicked*** eseményéhez egy függvényt és nevezzük ***button_clicked()***-nek, ahogy az a *6. ábrán* látható. Ekkor létrejön a kezelőfüggvény és oda is ugrik a kurzor (*16. ábra*).
A ***KDE*** alapjait jelentő ***QT*** könyvtárakból használható osztályok, függvények, a ***KDE*** fejlesztői könyvtárainak függvényei a vonatkozó dokumentációkban mind elérhetők, ill. a cikk végi hivatkozásokban is megtalálhatók.

Összefoglalás

Röviden, de megpróbáltunk a ***KDevelop*** legfontosabb tulajdonságaiba betekintést adni, elsősorban ***C++*** programozás esetén. Az utóbbi években, és leginkább a 3.x verziók megjelenése óta a ***KDevelop*** már kinőtt a hobbiprogramozók köréből, és a megbízhatóan használható számos integrált eszköz és funkció révén a számos kisebb tudású forráskódszerkesztőt is végleg maga mögé utasította. Természetesen elsősorban ***KDE/QT*** alkalmazások fejlesztésekor tudjuk kihasználni a benne rejlő lehetőségeket, de ez nem meglepő hiszen eredendően is az volt a cél, hogy létrejöjjön egy ***KDE*** alkalmazások készítésére használható integrált fejlesztői környezet. A fejlesztés ma is nagy ütemben folytatódik, és remélhetőleg egyre többen fedezik fel a benne rejlő lehetőségeket.

14. ábra Lehelyezett gomb lenyomási eseményének kezelése

15. ábra KfileDialog-gal megnyitott dialógusablak fájl kiválasztásához

```
65  void kdecapWidget::button_clicked()
66  {
67          QString display_str;
68          QString v4lctl;
69          int ret;
70
71          if(hours_str.isEmpty()||hours_str.toInt()==0)
72                  hours_str="00";
73          if(minutes_str.isEmpty()||minutes_str.toInt()==0)
74                  minutes_str="00";
75          if(seconds_str.isEmpty()||seconds_str.toInt()==0)
76                  seconds_str="00";
77
78          display_str=selected_str+" - length: "+hours_str+
79                  ":"+minutes_str+":"+seconds_str;
80          label->clear();
81          label->setText(display_str);
82
83          // start capture
84          v4lctl="v4lctl setchannel "+chan_str+" volume mute off";
85          FILE *ex=fopen("/tmp/kdecap_temp","w");
86          fprintf(ex,v4lctl); fprintf(ex,"\nmencoder ");
87          fprintf(ex,menc_pars); fprintf(ex,"\n");
88          fclose(ex);
89          ret=system("chmod ugo+rwx /tmp/kdecap_temp");
90          ret=system("/tmp/kdecap_temp");
91
92          KMessageBox::information(this,QString("Capture finished."));
93  }
```

16. ábra A rögzítés elkezdésekor lenyomott gomb eseményének kezelése

Kovács Levente
(leventek@gmail.com)

26 éves informatikus- és villamosmérnök. Évek óta használ különféle Linux disztribúciókat. Fontosnak tartja a nyílt forrású szoftverek és fejlesztés előnyeinek megismertetését az emberekkel.

KAPCSOLÓDÓ CÍMEK

- www.kdevelop.org
- doc.trolltech.com
- developer.kde.org
- women.kde.org/articles/tutorials/kdevelop3

Platformfüggetlen szoftverfejlesztés – A Lazarus

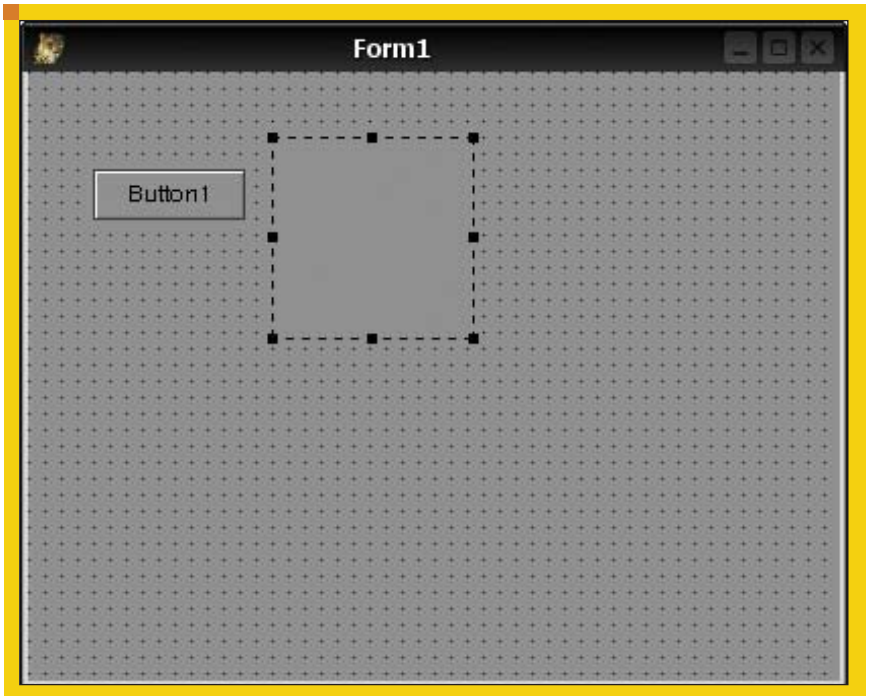
Az utóbbi években a Linux egyre inkább kezdi kinőni azokat a „gyermekbetegségeit” amelyek miatt nem volt alkalmas otthoni felhasználásra. Most, hogy egyre több helyen jelenik meg a Linux és már nem csak kiszolgáló szerepét tölti be elérkezett az ideje, hogy a programozók célplatformnak tekintsék és a hosszú hónapokon keresztül fejlesztett programjaikat egyetlen kattintással vihessék át a Pingvinre.

A *Delphivel* dolgozó fejlesztők számára született meg a *Lazarus*, az első igazi linuxos *RAD* eszköz amely alkalmas linuxos, windowsos, sőt maces alkalmazások fejlesztésére is.

Mik azok a 3G nyelvek?

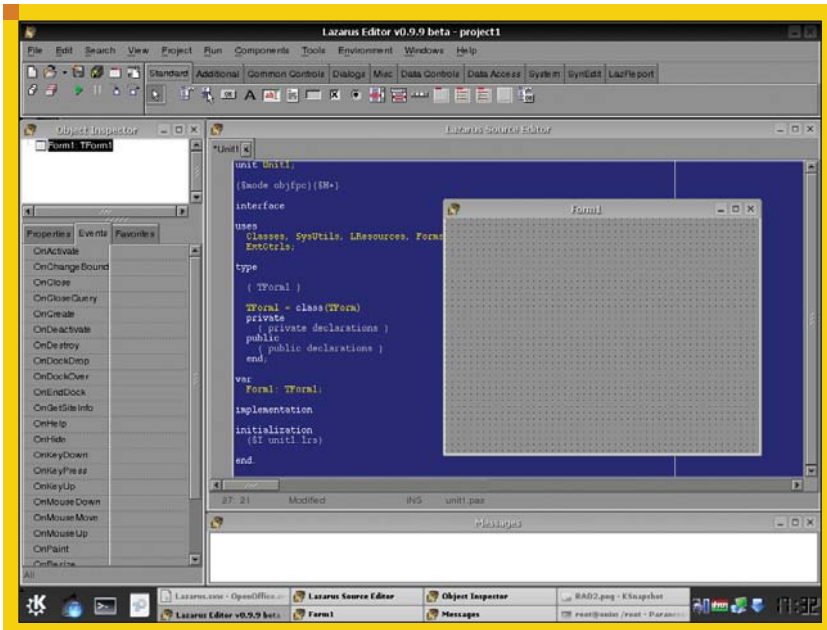
A *3G* nyelvek napjaink egyre gyorsabban terjedő programozási rendszerei melyek objektum-orientált szemléletük és eseményvezérelt programozásuk miatt napról-napra nagyobb teret hódítanak maguknak. Az eseményvezérlés nagyon megkönnyíti a programozó munkáját és rengetek monoton, felesleges programozással töltött időt spórol meg számára. A rendszer minden egyes objektumhoz (amely lehet egy gomb, egy kép, vagy akár egy adatbázis) ***esemény***eket (***events***) rendel hozzá melyeknek bekövetkezését követően a programozó határozhatja meg, hogy mi történjék. Például ha a felhasználó rákattint a gombra, akkor jelenjen meg egy ablak vagy történjék bármi más. Ennek csak a programozó fantáziája és az igények szabhatnak határt. Természetesen az objektumok mindegyike csak rá jellemző eseményekkel rendelkezik.
Gondoljunk csak bele: egy adatbázisra nem tudunk rákattintani, de egy képen sem igazán tudunk lefuttatni egy *SQL* lekérdezést. Az események mellett az objektumok rendelkeznek

1. ábra Egy ilyen palettán találhatók a komponensek



2. ábra Egy ablak amelyen egy gomb és egy üres képkeret látható

tulajdonságokkal (***properties***) is. A tulajdonság lehet a gomb mérete, a rajta található szöveg, vagy adatbázis esetén a hozzá tartozó jelszó/felhasználónév páros is. A tulajdonságok és események módosíthatók tervezési időben (***design-time***) vagy a program futásakor (***runtime***). Az objektumok elhelyezése a fejlett *RAD (Rapid Application Development*



3. ábra A Lazarus

= Gyors Alkalmazásfejlesztés) rendszerekben bámulatosan egyszerűen történik: egy „palettáról" a programozó a megfelelő helyre teszi az objektumokat, majd hozzárendeli az eseményekhez a hozzájuk tartozó metódusokat és végül lefordítja a programot.

A Linux és a 3G nyelvek

Windows alatt két domináns *RAD* rendszert találhatunk, de ezek mindegyike több száz vagy ezer dollárba kerül. A legelterjedtebb a *Borland Delphi* és a *Microsoft* által kiadott *Microsoft Visual Studio*. Ezek hatalmas, komplex alkalmazások melyekben rengeteg olyan szolgáltatás és kiegészítő program található amit a programozók nem túl gyakran vagy egyáltalán nem használnak. Ezek a programok csak natív *Windowsos* alkalmazások előállítására képesek. Ez teljesen érthető cégpolitikai szempont. A *Microsoftnak* nem érdeke a *Linux* támogatása. A *Borland* cég ezzel ellentétben tett egy kísérletet a Linuxos programozók „megkörnyékezésére" és kiadta a *Delphi* linuxos kistestvérét a *Kylixot*. Sajnos a *Kylix* a mai napig gyermekbetegségekkel küzd (és otthoni programozók számára megfizethetetlenül drága is).
A hajszás telepítéstől kezdve az instabil kód generálásáig sorolhatnánk azokat a hibáit amik nem teszik lehetővé vele a az összetett, hordozható alkalmazások fejlesztését.

Szerencsére ezen a területen is akadtak olyan programozók akik nem voltak megelégedve a *Linux RAD*-os ellátottságával és munkához láttak, hogy létrehozzanak egy – a *Delphivel* vetekedő – ingyenes szoftvert. Így született meg a *Free Pascal* alapokon nyugvó *Kassandra* mely később a *Lazarus* nevet kapta.

Free Pascal? Lazarus?

1996 júniusában egy német programozó, *Florian Klaempfl* írt egy kísérleti jellegű *Pascal* fordító programot, mert a *Borland Pascallal* írt alkalmazásai nem fordultak le *Linux* alatt. A projekt az *FPK* (azaz *Free Pascal Kompiler*) nevet kapta. Látva a felhasználók lelkesedését a terv tovább nőtt, új célt tűztek ki maguk elé: egy olyan platformfüggetlen *Pascal* fordító létrehozására szánták el magukat amely minden ismert platformot támogat és az így készített alkalmazások egy-két sor módosítással vihetők át egyik operációs rendszerről a másikra, ráadásul képes igen kevés módosítással *Delphiben* írt kódok lefordítására is a támogatott platformokon.
A *Lazarus* a *Free Pascallal* összefonódó fejlesztés amely a *Delphi* keretrendszerét és működését alapul véve hatékony fejlesztői eszközt ad a programozók kezébe. A *Lazarus* jelenleg a 0.9.10-es stabil kiadásnál tart, de ha minden a fejlesztők tervei szerint alakul akkor Karácsonykor kézbe vehetjük az 1.0.0-ás kiadást.

A Lazarus és a Free Pascal telepítése

Mielőtt nekiállnánk a telepítésnek, nem árt ha beszerezzük valahonnan a fordítót és a keretrendszert! Látogassunk el a *Free Pascal* honlapjára (➲ *www.freepascal.org*) és töltsük le a nekünk tetsző tükör szerverről a teljes *Free Pascal* csomagot. Találhatók olyan csomagok amelyekben csak bináris vagy csak forrás található de nekünk most nem ez kell. Kis keresgélés után rálelhetünk az *„Everything in 1 big package" (Minden 1 nagy csomagban)* nevű 26,4 Mbyte-os állományra. Töltsük le és tömörítsük ki egy számunkra szimpatikus könyvtárba. Én a */opt/fpc*-t javasolnám, de ez szinte teljesen mindegy, csak jól jegyezzük meg, hogy hova került, mert később szükségünk lesz rá. A kitömörítés valahogy így történik:

```
tar -xvf fpc-2.0.0.i386
➥-linux.tar
```

Ha ezzel elkészültünk, adjuk ki az

```
sh install.sh
```

parancsot és feleljünk minden kérdésre legjobb tudásunk szerint, egyetlen kikötés van csak: minden csomagot telepítsünk amire rákérdez a telepítő!
Ezután nincs más feladatunk mint a *Lazarus* beszerzése a ➲ *http://lazarus.freepascal.org* oldalról. A bátrabbak megpróbálkozhatnak az aktuális *CVS* forrással is, de ezt nem igazán tanácsolnám, mert nagyon instabil. A *Lazarus*-t vagy *tar.gz* formátumban, vagy *rpm* csomag formátumban tudjuk letölteni. Én vallási okokból a tar-hoz vonzódom, ezért most ezt mutatnám be.
Letöltés után másoljuk a tömörített állományt valahova a *Free Pascal* forrás közelébe és adjuk ki a

```
gunzip -d lazarus-0.9.10tar.gz
tar -xvf lazarus-0.9.10.tar
```

parancsokat.
Sajnos a telepítés még nem ért véget, mert a *Lazarusról* gyorsan kiderül, hogy nem bináris hanem forrás állapotban töltöttük le. A fordítás közben fellépő rendellenességek elkerülése

végett telepítsük a *libgtk1.2-dev* és a *libgdk-pixbuf-dev* csomagokat. Lépjünk be a *lazarus* könyvtárba és adjuk ki a `make` parancsot.
Ha minden terveink szerint alakul akkor minden fennakadás nélkül befejeződik a fordítás. Ha valamilyen csomagot még mindig hiányol akkor azt jelezni fogja nekünk, hogy pótolhassuk a hiányosságot.

Utolsó simítások

A telepítés befejezése után nincs más dolgunk mint elindítani a *Lazarust* a `./startlazarus` paranccsal.
Egy felbukkanó kis ablak arról fog minket tájékoztatni, hogy a könyvtárak nincsenek megfelelően beállítva. Ezt az ablakot nyugodtan tüntessük el. Ha mindent jól csináltunk pár másodperc elteltével ezt kell látnunk (*3. ábra*).
Most végezzük el az utolsó finomításokat, hogy a keretrendszer ne panaszkodjon többet. Én a *Lazarust* és a *Free Pascalt* is a */opt* könyvtárba szoktam telepíteni, ez jól látható a beállításaimon is. Kattintsunk az *Environment* menüre, azon belül az *Environment Options* menüpontra és állítsuk be az üresen hagyott mezőket *(4. ábra)*.
Azok kedvéért akik számára nem lenne egyértelmű, hogy melyik beállítás micsoda:

- *Lazarus directory*: Az a könyvár ahol a `startlazarus` bináris állomány van.
- *Compiler path*: A `ppc386` elérési útvonala. `whereis ppc386` azonnal megmondja hol találjuk.
- *FPC source directory*: Itt a *Free Pascal* forrásának könyvtárát adjuk meg. Azt ahol az *fcl* és *rtl* alkönyvtárak vannak.
- *Make path*: A standard `make` elérési útvonala. A `whereis make` itt is segít!
- *Directory for building...*: Átmeneti fájlok tárolására. Kézenfekvő a */tmp*.

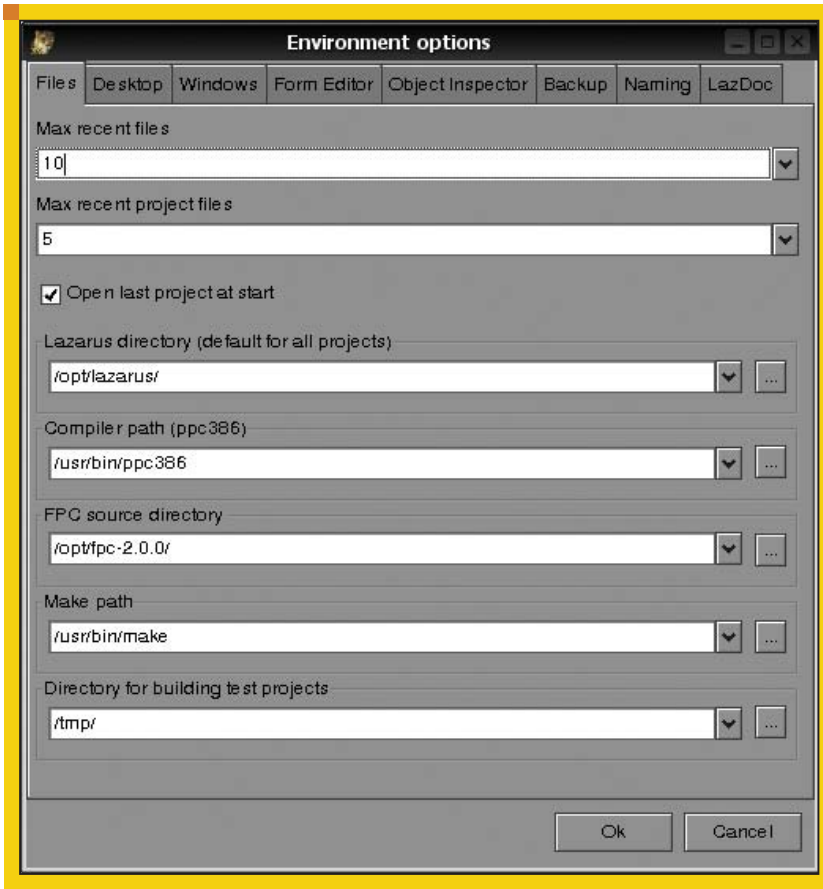
Ismerkedés

A program kezelőfelületét öt ablakra oszthatjuk. Az első a főmenüt és a komponenseket tartalmazó fejléc. Ezen a hosszúkás ablakon találhatóak azok az „alkotóelemek” amelyek a rendelkezésünkre állnak a fejlesztés folyamán. A könnyebb elérhetőség végett csoportosítva vannak az objektumok:

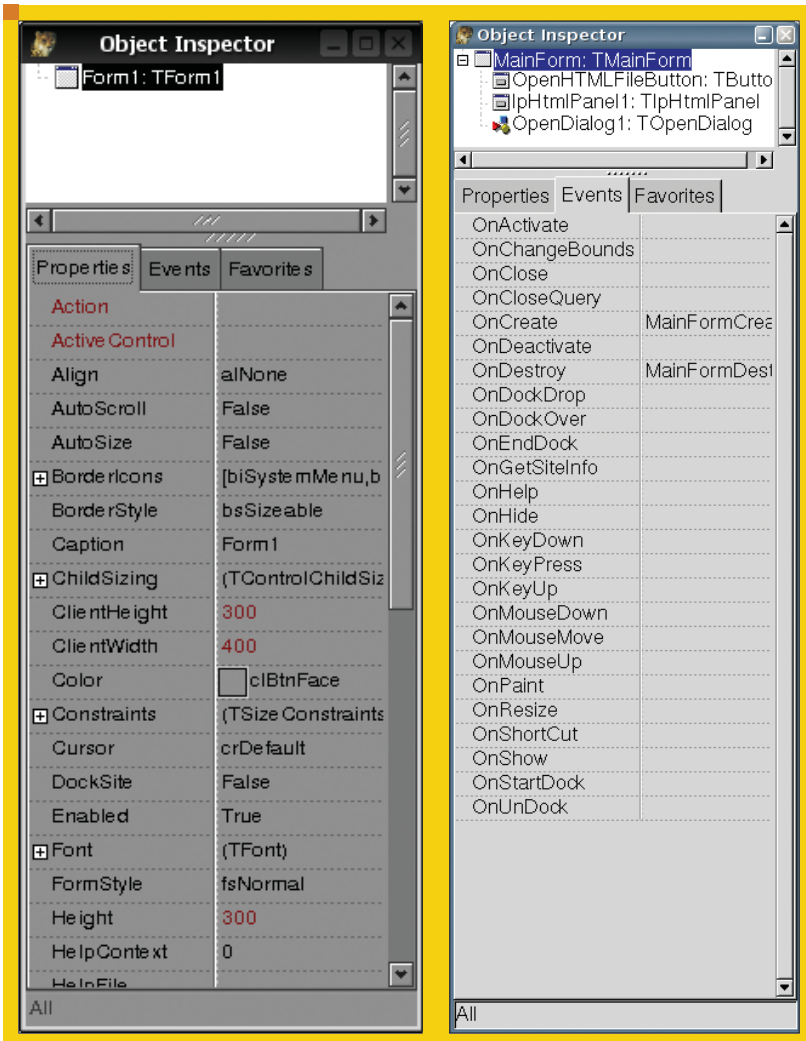
- *Standard*: Gomb, képek, jelölő négyzet, stb.
- *Additional*: Keretek, táblázatok, „rajztábla”, stb.
- *Common*: Fülekkel ellátott oldalak, fa megjelenítések, stb.
- *Dialogs*: Nyomtatási-, Fájl megnyitási és mentési ablakok.
- *Misc*: Naptár, színváltó.
- *Data Controls*: Adatbázisokhoz adatmegjelenítési komponensek.
- *Data Access:* Adatforrás adatbázisokhoz.
- *System*: Időzítők.

Ezek a komponensek találhatók az alap telepítésben, de napról-napra növekszik a számuk, mert a *Lazarus* csapat bizonyos tagjai újabb és újabb komponenseket „hoznak át” *Delphiből*. Ha jobban körülnézünk a honlapon akkor a *Documentation* menüpont alatt rábukkanhatunk egy olyan oldalra amely az eddig portolt komponenseket tartalmazza.

Adatbázis komponensek a legelterjedtebb rendszerekhez: *Interbase (Firebird)*, *PostgreSQL*, *MySQL*, *OpenGL* komponensek a *3D*-s grafikához és még sorolhatnánk.
A baloldali *Object Inspector* elnevezésű ablak segít minket a már elhelyezett komponensek tulajdonságainak és eseményeinek megváltoztatásában.
Ha egy komponenst helyezünk a főablakra és utána rákattintunk akkor az adatai betöltődnek az *Inspectorba*.
Minden komponens rendelkezik névvel (*Name*) amely egy egyedi azonosító. Ne próbáljunk meg több `Button1` nevű gombot létrehozni, mert a program nem hagyja. Az tulajdonságokat és eseményeket nem fogom felsorolni, mert ahány objektum, annyi különböző lehetséges, de a nevek magukért beszélnek. Például az *OnClick* esemény akkor következik be ha rákattintunk az adott objektumra. Az esemény neve mögött megadhatjuk annak a függvénynek a nevét ami akkor fog meghívódni ha az adott esemény bekövetkezik. Érdemes beszédes neveket használni, mert ha sok objektum



4. ábra Az utolsó simítások



5. ábra Az Object Inspector tulajdonság és esemény módosító felülete

halmozódik fel, nehéz átlátni a forráskódot. Ha egy függvényt már hozzárendeltünk egy eseményhez, akkor a *Lazarus* létrehozza a függvény vázát, nekünk már csak annyi a dolgunk, hogy megírjuk a kódot hozzá.
A következő ablak (amely indításkor Form1) névre hallgat, az a keret amelyre a komponenseinket helyezhetjük. A program futtatásakor a komponensek úgy fognak megjelenni ahogy az ablakra helyeztük őket.
A főablak egy szövegszerkesztő amelyben a programunk forráskódját tudjuk írni, módosítani. Azok aki már használtak bármilyen *RAD* környezetet, vagy írtak már programot bármilyen más felületen azok számára nem lesz ismeretlen a dolog.

De itt csak mindenféle gombok vannak! Hol az többi komponens?

A cikk elején már ejtettünk egy pár szót a komponensekről, amik a *Delphi*, a *Kylix* és a *Lazarus* alapját képezik. Ezeknek a komponenseknek a segítségével gyorsíthatjuk a munkát és lényegesen lerövidíthetjük a programfejlesztés idejét. Mielőtt nagyon belemásznék a komponensek lelkivilágába, kalandozzunk el egy kicsit más irányba.
A 90-es években egy *Jim Starkey* nevű úriember írt egy adatbáziskezelő rendszert a saját cége számára (*Groton Database Systems*) amely később a Borland tulajdonába került. A *Borlandnak* annyira megtetszett a fejlesztés, hogy egy alvállalatot hozott létre a számára. Ez az adatbázis-kezelő motor nem volt más mint az *InterBase*. Az *InterBase* fő támogatói és egyben „fogyasztói" igen neves cégek voltak: *Ericsson*, *Egyesült Államok Védelmi Minisztériuma*, tőzsdék. 1999-ben a *Borland* bejelentette, hogy az *InterBase* a továbbiakban nyílt forráskódú fejlesztésként fog tovább létezni és 2000-ben ki is adták a teljes forráskódot milliónyi programozó örömére. Pár évvel később a *Borlandtól* kilépett szakemberek megalapították az *IBPhoenix* nevű céget ami a *FireBird* (az *InterBase* nyílt forráskódú neve) fejlesztésével és támogatásával foglalkozott. Sajnos a *Borland* pár éve letett a *FireBird* támogatásáról és visszatértek az *InterBase*-hez.
Hogy miért meséltem el mindezt? Azért mert a *FireBird* talán az egyik legjobban támogatott *RDBMS* rendszer amihez találunk *Lazarusba* integrálható komponenseket.
Aki igazán hatékony és megbízható adatbázisra szeretné építeni az alkalmazásait annak bátran ajánlhatom. Természetesen azok sincsenek hátrányban akik már meglévő – nem *FireBird* – adatbázisaikat szeretnék valamilyen módon kezelni. Számtalan komponens létezik *PostgreSQL*, *MySQL* és egyéb adatbázis rendszerekhez való csatlakozásra.
A másik érdekes fejlesztés a *GLScene* nevű *OpenGL* komponens gyűjtemény amelyet a *Delphi JEDI* csoport – egy kiváló *Delphi* programozókat tömörítő társaság – hozott létre. Többek között az ő fejlesztésük a *JVCL* csomag, a *DirectX for Delphi* és a *GLScene* is. Ezek közül a *GLScene*-t szeretném kiemelni, mert a többi komponens gyűjtemény sajnos nem portolható *Linux* alá. A komponens gyűjtemény segítségével könnyedén tudunk *3D*-s alkalmazásokat fejleszteni. Támogatja az *idSoftware* által létrehozott *MD2* és *MD3* objektumok megjelenítését, animálását, időjárás effektusok (eső, köd, hó, villám, tűz) létrehozását és még számtalan egyéb dolgot.
Azok akik sem adatbázisokkal, sem *3D*-s grafikával nem szeretnének foglalkozni megemlítenék egy harmadik családot a *Synapse Ararat*-ot. Ez nem *Lazarus*-hoz lett kifejlesztve, így nem vizuális komponens, csak használható „unit"-ok. Az *Ararat* segítségével szinte gyerekjátékká válik bármilyen hálózati szoftver fejlesztése. Részletes és jól érthető dokumentációt találunk a honlapjukon, rengeteg példával alátámasztva.

Remek! Letöltöttem, de hogyan telepítsem?

Amennyiben megtaláltuk és letöltöttük a nekünk tetsző komponenseket, már csak telepíteni kell őket, hogy használhassuk. Tömörítsük ki a letöltött fájlt egy könyvtárba és másoljuk a könyvtárat a */lazarus/componenst* alá. Ha ezzel elkészültünk indítsuk el a *Lazarust* és a *Components* menüből válasszuk ki az *Open Component Package (lpk)* pontot. Keressük meg a kitömörített komponenshez tartozó *lpk* fájlt és válasszuk ki. Ha minden rendben ment, akkor megjelenik egy kis ablak amelyben először a *Compile*, majd az *Install Package* pontot kell választanunk. A *Lazarus* meg fogja kérdezni, hogy „újra akarjuk-e fordítani az egész keretrendszert a csomag telepítéséhez". Természetesen igen! Pár másodperc töltögetés és a *Lazarus* újraindulása után máris használatra készen áll az komponensünk.
Persze telepítés közben merülhetnek fel hibák (eltérő verziószámú fordító, rosszul beállított elérési útvonalak, hiányzó függvény könyvtárak...). Ezeknek a hibáknak a kiküszöbölésére a jobb érzésű komponens készítők mellékelnek *README* állományt a szerzeményükhöz amit érdemes még a telepítés előtt átolvasni. Ez különösen igaz a *ZeosDBO* csomagra amit botor módon a telepítési útmutató nélkül próbáltam életre kelteni. Két munkanapom ment rá.
Ha a hibára nem találunk megoldást a dokumentációban akkor irány a ➲ *http://lazarus.freepascal.org* ahol megtaláljuk a fórumot.
Itt pár perc alatt elháríthatjuk a hibát. Szerencsére a fórumot a rendszer fejlesztői és a komponensek írói is folyamatosan látogatják ezért szakértői segítségre számíthatunk. Ha nem hibáról van szó, csak bármilyen jellegű kérdése van a Kedves Olvasónak akkor is bátran írjon!

Hibák

Annak ellenére, hogy a *Lazarus* nagyon dinamikusan fejlődik és egymást érik az újabb és újabb kiadások, rengeteg gyermekbetegsége van még. Az első és talán legnagyobb hibája a telepítés bonyolultsága.
Persze a *Linuxhoz* szokott programozók nem fogják zokon venni, ha nem lesz hozzá „kattintgatós" telepítő, de hogy *Windows* alatt is forrásból kell telepíteni – ha azt akarjuk, hogy működjön – kicsit meglepett.
A cikk olvasói között többnyire olyanok vannak akik *Linux* alatt szeretnék kipróbálni, vagy használni a *Lazarust*, ezért az egyik legidegesítőbb hiba kijavítását szeretném most bemutatni. Ez nagyon fontos lehet, ugyanis a *Linux* és a *Windows* eltérő szál-kezelése (*threading*) miatt a programjaink nem fognak lefordulni, vagy egyéb megmagyarázhatatlan hibákat fognak produkálni. A legrosszabb eset amikor egy szálakat használó komponens telepítése után a *Lazarus* összeomlik. A javításhoz keressük meg a *Lazarus* könyvtáron belül az *IDE* alkönyvtárat, azon belül is a *lazarus.pp* fájlt és módosítsuk a fájl deklarációs részét. Valahol a 43. sor körül kell megtalálnunk!

Ilyen volt:

```
{off $DEFINE IDE_MEM_CHECK}

uses
 //cmem,
 {$IFDEF IDE_MEM_CHECK}
```

Ilyen lett:

```
{off $DEFINE IDE_MEM_CHECK}

uses
 cthreads,
 //cmem,
 {$IFDEF IDE_MEM_CHECK}
```

A `uses` kulcsszó után adjuk meg a `cthreads` unit nevet. Ha mást is módosítunk a fájlban, akkor is nagyon figyeljünk oda, hogy mindig a `cthreads` legyen a *Lazarus* által betöltött első egység!
Sok időnek kell még eltelnie mire az *UIB*, *FIBL*, *ZeosDBO* és *GLScene* is hivatalosan bekerülnek a *Lazarusba* és már telepítés után rögtön használhatóvá válnak. Addig marad a manuálisan történő telepítgetés és az esetleges hibák kézi elhárítása.
A *debugger*-en is van még mit csiszolni, de amíg nem lesz olyan mint a *Delphiben* addig nem fog megjelenni az 1.0.0-ás verzió. Legalábbis ezt állítják a fejlesztők.

Zárszó

Hosszú évek *Delphi*-s tapasztalatai után már nagyon hiányzott egy ilyen rendszer. Napról-napra próbálom nyomon követni a *Lazarus* fejlődését és nagyon bízom benne, hogy a fejlesztők lelkesedése nem fog alább hagyni az első hivatalos, stabil kiadása után sem. Már mostani állapotában is tökéletesen használható kisebb programok fejlesztésére, ha az ember nem adja fel az első hibánál.
Csak így tovább!

Sok sikert!

Tóth Péter
(thotacc@drotnet.hu)

A BMF hallgatója vagyok, mellette egy kis-és középvállalatok informatikai rendszereinek Linuxos átállításával és szoftverfejlesztéssel foglalkozó cég informatikai vezetőjeként tevékenykedem. Kevés szabadidőmet barátnőmmel és barátaimmal töltöm egy-két sör társaságában.

KAPCSOLÓDÓ CÍMEK

A Free Pascal hivatalos oldal:
➲ http://www.freepascal.org

A Lazarus hivatalos oldala:
➲ http://lazarus.freepascal.org

Egy szinte minden RDBMS-t támogató komponenscsalád:
➲ http://sourceforge.net/projects/zeoslib/

Free Interbase Component:
➲ http://sourceforge.net/projects/fibl/

Az egyik legjobb és leggyorsabban fejlődő FireBird komponens:
➲ http://progdigy.com/

Hálózatkezeléshez a Synapse Ararat-ot javaslom:
➲ http://www.ararat.cz/synapse/features.htm

GLScene – 3D grafika OpenGL-el:
➲ http://wiki.lazarus.freepascal.org/index.php/GLScene

TMPack – hasznos, apró komponensek:
➲ http://lazarus.freepascal.org

A Sony PlayStation játékai Linux alatt

A konzolok világában otthonosan mozgó játékosok bizonyára nagy tisztelettel tekintenek a Sony első generációs játékkonzoljára, melyből fénykorában a neves gyártó milliós nagyságrendben értékesített világszerte. Nem véletlenül, hiszen baráti árával, rengeteg játékával, „moddolhatóságával” tömegek szórakozását szolgálta.

A konzolról, dióhéjban

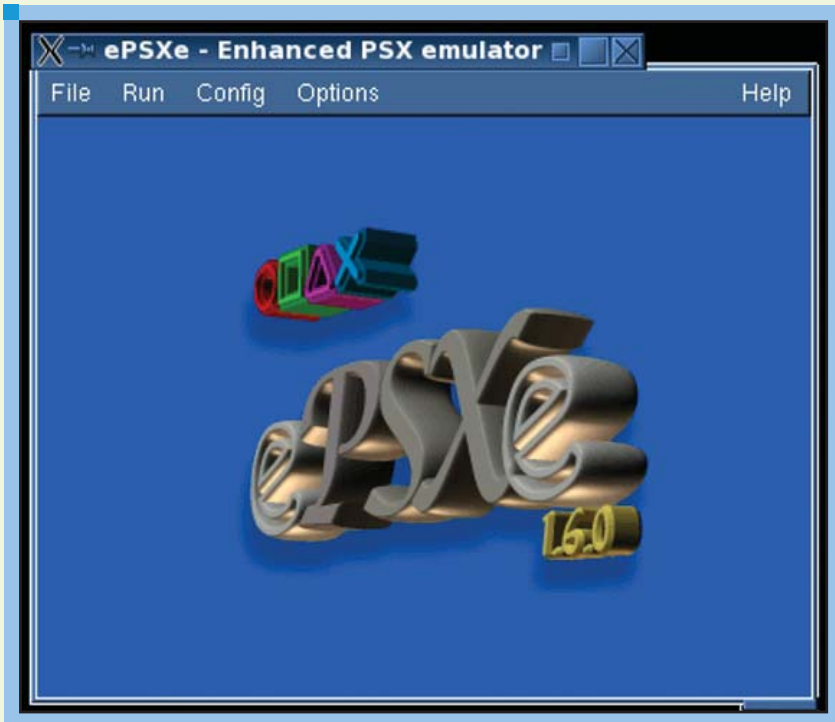
A játékgép képességei mai mércével már igencsak szerénynek tűnnek, ennek ellenére nagyon jókat lehet vele játszani, hála azoknak a profi programozóknak, akik többnyire olyan programokkal rukkoltak elő, melyek gyakorlatilag a ***PSX*** „platform” minden képességét kiaknázták. Őszintén szólva nem vagyok megrögzött konzol-fanatikus, ennek ellenére néhány évvel ezelőtt még mérhető időt töltöttem ilyen masina előtt ülve.

Ha valaki megkérdezné tőlem, hogy mit szerettem rajta, akkor valószínűleg nem is tudnék egyértelműen válaszolni. Utólag visszagondolva talán azért becsültem meg ezt a konzolgépet, mert több szempontból formabontó volt, nagyon egyszerűen lehetett használni, megelégedett egy egyszerű televízió ***RCA*** bemenetével, és „mi tagadás” azért is, mert az általa ígért játékélményt igen hatékonyan fokozta szenzációs vibráló kontrollere. Bár a vezérlőről kialakított véleményemet csupán szubjektív szempontokra támasztom, személy szerint úgy gondolom, hogy ebben az eszközben mind a mai napig komoly potenciál rejtőzik. Egyszerűen mesteri ez a kiegészítő: ha valakinek rövid gyakorlás után „ráállt” a keze, nem hiszem, hogy valaha is igazán elégedett lesz bármilyen egyéb ***gamepad***-ekkel.

No, de térjünk vissza az eredeti témához, és nézzük, mit rejt valójában a ***Sony „PS One”*** belseje! ***R3000 RISC*** processzort (***33MHz*** üzemi órajelen), ***28MByte*** memóriát (melyből ***16Mbyt***e a rendszerhez tartozik, ***8MByte*** a videóvezérlőhöz, ***4Mbyte*** pedig a hangkeltő egységhez), valamint egy halk működésű, felültöltős ***CDROM***-ot.

A konzol hangját leginkább ***reverb*** effektek segítik emészthetővé tenni, ennek ellenére audió-képességei a ***sztereó*** szintjén megállnak, grafikai lehetőségei pedig ma már leginkább egy szerény, szoftverből renderelt képi világot idéznek. A népszerű játékgép életében két ruhát is kapott formatervezőitől: az első szériák nagyobb, sötétebb dobozban lettek terjesztve, a későbbi darabok pedig egy kisebb, vajszínű „karosszériát” kaptak. Utóbbi verzió néha még megtalálható a multiáruházak polcain (igen alacsony áron), azonban a készülék csupán már ***retro*** szerepet tölt be, néhány év múlva pedig talán már csak a szakmai múzeumokban fogunk összefutni vele. Az emulátorok körében azonban

1. ábra A Sony első generációs konzolja



2. ábra Az ePSXe grafikus felülete

továbbra is él, mivel utánzására jó néhány projekt hivatott, érintve akár a *Linux* rendszereket is. A mostani cikkben egy kiforrott, külső *plugin* rendszerre támaszkodó emulátort fogok röviden bemutatni.

ePSXe

Az *Enhanced PSX Emulator* a *PC* platform egyik vezető *Sony Playstation* emulációs projektje. A program a hivatalos lapján, a ➲ *http://www.epsxe.com* oldalon érhető el, bináris formában. *Linux* verzióján felül természetesen létezik belőle *Win32* változat is, mely tudásában nem tér el az előző változattól, azonban a működéshez szükséges modulrendszer már sok tekintetben különbözik.
Látogassunk el az említett oldalra, melynek *„Downloads”* gombját használva töltsük le a linuxos archívot (a cikk írásakor a *v1.6.0* azonosítót viseli). A *tarball* mérete „mellbevágó”, hiszen alig éri el a *200 KByte* méretet, mindez azonban ne tévesszen meg senkit sem, a működéséhez szükséges kiegészítő állományok ennek a méretnek többszörösét fogják kitenni. Csomagoljuk ki a letöltött anyagot! Ha jobban megnézzük, egy indítható állományt és egy könyvtárszerkezetet találunk benne.

Az *epsxe* binárist elindítva egy egyszerű, igen lényegre törő grafikus interfész bukkan fel az asztalunkon, melyben a *„Config”* menüpont kitüntetett figyelmet érdemel. Ennek almenüit végigböngészve könnyen észlelhető lesz, hogy az emulátor az utánzott konzol *ROM BIOS* rutinjaira támaszkodik, valamint képi és hangleképezése egyaránt idegen modulokat használ. Mivel a letöltött projekt egyik elemet sem tartalmazza, így ezeket külön kell beszerezni, mégpedig a következők szerint.
A *BIOS* lenyomat (többek között szervízcéllal) elérhető az egyik vezető on-line *driver* gyűjteményben is, nevezetesen a ➲ *http://driverguide.com* oldalon. Az ingyenes regisztrációt követően a *„Basic Search”* funkciót használva keressünk *scph1001* nevű állományt!
Miután letöltöttük a tömörített fájl, tartalmát *spch1001.bin* néven csomagoljuk az *ePSXe* főkönyvtárának */bios* nevű almappájába.
Az emulátor linuxos verziójához szükséges képi és hangmodulokat *Pete Bernert* munkájaként érdemes keresni, a ➲ *http://www.pbernert.com* oldalon. A megjelenítésért felelős *plugin*-ek igény szerint *SDL*, *Mesa Indirect*, *OpenGL* támogatásra alapoznak, a hangszolgáltatás kiegészítése pedig *OSS* leképezést használ. Hang terén nincs sok választásunk, mivel csupán ez a verzió létezik belőle, azonban a grafika terén már más a helyzet. Töltsük le a nekünk leginkább megfelelő változatot!
A mai *3D* grafikus hardverek gyakoriságát tekintve leginkább az *OpenGL* alapú változatot érdemes használni, de természetesen munkára fogható akár a *Mesa* alapú renderelés is,

3. ábra A virtuális PSX és az Ape Escape, KDE asztalon

mely nélkülözi a *3D* megoldás valódi, hardveres segítségét. A *plugin* archívokat a következők szerint bontsuk ki: a tömörített fájlokban található **.txt* és **.cfg* állományokat másoljuk az *ePSXe* megfelelő */config* útjára, az összes többi részét pedig a */plugin* mappába. Immár csaknem minden teendőt elvégeztünk, egy apró beavatkozást azonban mindenképp meg kell még ejtenünk: az emulátor fő mappájában található, rejtett attribútumokkal rendelkező (első indításkor létrejött) *epsxerc* állományban a *CdRomDevice* és *CdRomMountPath* értékeket változtassuk meg a rendszerünk megfelelő értékeire (az én esetemben ez */dev/hdc* és */mnt/dvdrw* bejegyzéseket jelent). Minderre azért van szükség, mert a menükből ezt a beállítást jelenleg sajnos nem lehet megtenni.

Használatba vétel

Ha minden leírt művelettel végeztünk, a projekt fő mappájában található bináris állomány indításával, a `./epsxe` parancs kiadásával indíthatjuk virtuális játékkonzolunkat. A néhány sorral előbb említett konfigurációs menükben tallózzuk ki a megfelelő könyvtárba másolt *BIOS* lenyomatot, majd a *„Video”* és *„Sound”* menükben meghatározandó szükséges kiegészítőket is. Fűzzük be a kedvenc játékunk lemezét rendszerükbe (az emulátor könnyedén megbirkózik a *supermount* mechanizmussal is), majd válasszuk a fájl menü *„RUN CDROM”* lehetőségét. Mint az a lehetőségek között látható lesz, akár a merevlemezen lévő **.iso* lemeztükröket is fel tudjuk használni célunkra, akár közvetlenül, hurok rendszerű csatolás mellőzésével. A vezérlőt és memóriakártyát érintő beállításokat szintén a *„Config”* menüben találhatjuk meg, az *„Options”* fül alatt pedig többek között a *PAL* és *NTSC* szabványnak megfelelő változtatásokat eszközölhetünk.

A tapasztalatok

Igazság szerint nem hiszem, hogy a leírtakat különösebben kommentálnom kellene. Ennek a könnyen és ingyenesen elérhető, viszonylag egyszerűen használható projektnek köszönhetően a *Sony* játékgépe tovább él *Linux* platformon is. Mindez szerencsére nem kíván erős „gazdagépet”, egy mai mértékkel gyengének mondható *1700MHz* órajelű *Celeron* processzor *256MByte* központi tár, első generációs *nVidia GeForce* grafikus kártya segítségével kompromisszumoktól mentesen élvezhetjük a program adta lehetőségeket. *Szabad Rendszeren* mindazonáltal nem csak ez a *Playstation* emulációs projekt létezik, így hasznos lehet keresgélni a világhálón, hiszen a választás lehetősége mindenki előtt nyitva áll. Érdemes élni ezzel a lehetőséggel, mivel kedvenc rendszerünket százas nagyságrendben segíthetjük ki (kifejezetten szórakoztató) játékokkal.
Tartalmas kikapcsolódást kívánok minden érdeklődőnek!

Kovács Zsolt (kovi@linuxforum.hu)

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

Fájlcsere és egyenrangú hálózatok Linux alatt

Manapság az internetről szinte bármilyen adatot letölthetünk. A határ úgymond a csillagos ég. Természetesen ennek csak a töredéke legális és Linuxhoz kapcsolódó. Ebben a cikkben megpróbálom összefoglalni a fájlcserélgetésről szóló gondolataimat általánosságban, illetve kitérek egy Linux alatt használatos fájlcserélgetésre szakosodott alkalmazás bemutatására.

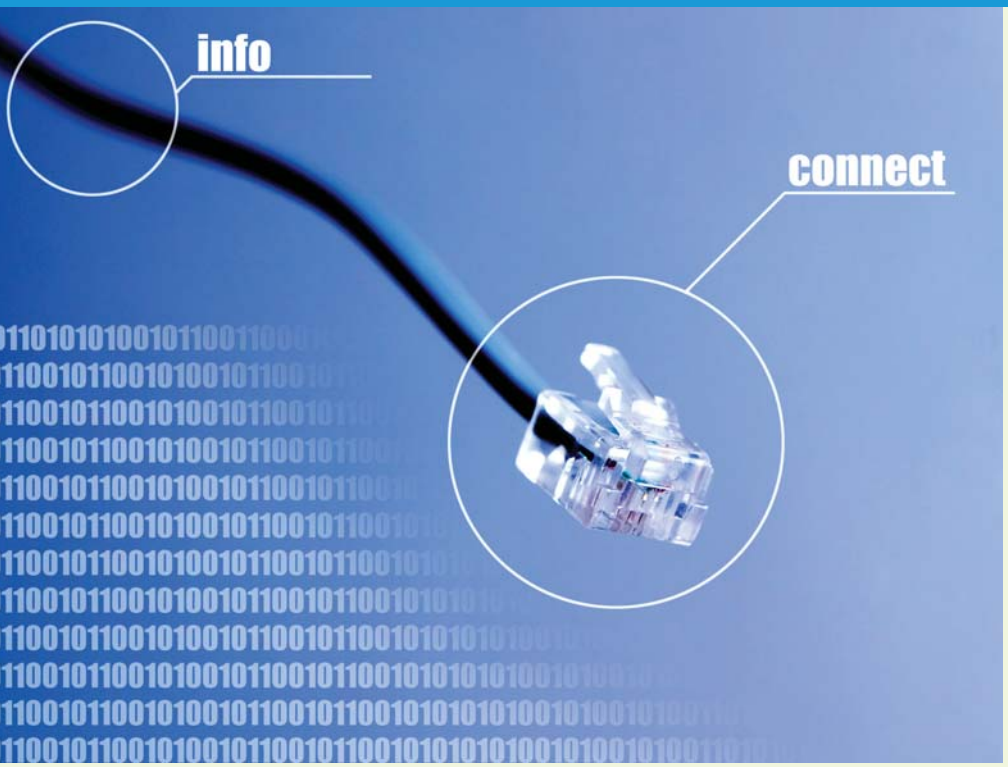
A fájlcseréről általában

Lássuk csak, hogyan is kezdődött ez az egész. Amennyire vissza tudok emlékezni, az első fájlcserélgetős élmenyem a hírhedt ***Napster***hez kapcsolódik. Gondolom sokan emlékeznek erre, ez a program talán az első volt, amivel zenéket lehetett letölteni az internetről a saját gépünkre. Vele egy időben futott a ***Linux***os ***Gnutella***, amit bevallom őszintén, én sosem használtam (pláne abban az időben még csak „szaglásztam" a ***Linux***ot). Nos, a ***Napster*** becsődölt, amikor ***Lars Ulrich***, a ***Metallica*** dobosa fellépett ellenük (még a magyar tv-k híradója is leközölte a hírt). Ebből mind zenész, mind rajongói berkekben közbotrány lett, hogy mit képzel magáról ez a felfuvalkodott multimilliomos rocksztár, amikor már így is annyi pénze van. Mit érdekli az őt, hogy mások ingyen töltenek le ***Metallica*** dalokat, albumokat? Nos, ebben van igazság, mivel ***Lars*** barátunknak tényleg nem kell aggódnia a pénz miatt, azonban teljesen egyet lehet érteni vele abban, hogy ha mindenki mindent letölt a netről ingyen, akkor az emberek (zenészek, programozók, filmesek, stb...) miből fognak megélni?
Ezen azóta is töpreng a világ, de a megoldást még senki sem találta meg. Nyugodt szívvel állíthatom, hogy majdnem minden internetet használó egyén (beleértve magamat is) a világon, „kalózkodott" már a fent említett módon legalább egyszer. Ez teljesen természetes, mert általában az emberek fogékonyak az új dolgok kipróbálására. Viszont tudni kell különbséget tenni lopás és kipróbálás között.
A ***Napster***t követték az újabbnál újabb kliensek, mint például a ***Windows***-os ***iMesh***, ***KaZaA***, illetve a ***Linux***os ***Limewire*** és hasonszőrű társai. Kifejlődött a ***peer-to-peer*** (vagy ***p2p***) technológia, ami nagyon egyszerűen fogalmazva annyit tesz, hogy vannak főszerverek, amikhez kapcsolódunk, de ezzel szemben a saját gépünk is szerverként funkcionál. Így tőlünk is tölthetnek a hálózatban lévők, és mivel minden gép szerver, ezért a hálózatot „lebuktatni" nem egyszerű. Magyarán szólva nem érdemes. Sőt, úgy is fogalmazhatnék, hogy a „sok lúd disznót győz" elv teljesen ráhúzható a ***peer-to-peer*** rendszerre. Sajnálatos módon ezt sokan ki is használják.

Néhány régi fájlcserélő programban kötelező volt megadni egy ***shared folder***-t, ahol azok a fájlok vannak amit tőlünk letölthetnek a rendszeren keresztül. Például a manapság is népszerű ***Direct Connect***ben (avagy ***dc***, illetve ***Windows*** alatt ***dc++***) kötelező megosztanunk fájlokat ahhoz, hogy mi is tölthessünk, tehát a megosztás szükségszerű feltétele a töltésnek.
Az újabb alkalmazásokban nem kell feltétlenül megosztanunk semmit. Az éppen aktuális letöltéseink funkcionálnak megosztott állományokként.

Fájlcserélgetés és Linux

A hosszú bevezető után most essen szó arról, hogy e cikk mit is keres ebben a lapban. Elárulom, nem véletlenül szerepel a linuxos cikkek között.



Sok linuxos cég felismerte, hogy az *ftp* szerverek bizonyos határon túl nem terhelhetők, ezért gondolt egy merészet, és elkezdte fájlcserélőkön keresztül is terjeszteni a letölthető *iso* fájlokat.
Hogy miért is tették ezt a merész lépést? Nos, az *ftp* szervereknek van egy maximális terhelhetősége, amit egy új disztribúció kiadásakor a sok ezer felhasználó bizony átlép. Ekkor jönnek a „szerver túlterhelt, próbáld meg 5 perc múlva" és hasonló hibaüzenetek, illetve jobb esetben a csigalassúságú letöltés. Viszont ha *peer-to-peer* rendszereken is elérhetők az állományok, akkor azok sokkal előbb és sokkal gyorsabban eljutnak a célközönséghez, mint ha csak *ftp*-n keresztül lehetne őket elérni.
Nézzünk erre egy példát. Jómagam *Debian GNU/Linux*-ot használok pár éve, ezért a példám is a *Debian*ról fog szólni. Ha ellátogatunk a *Debian* honlapjára (➲ http://debian.org) és kiválasztjuk a *„Debian beszerzése"* menüpont alatti *„ISO-CD-fájlok"* pontot, akkor észrevehetjük, hogy rengetegféleképpen hozzájuthatunk a terjesztéshez. A lap közepe táján a következőt pillanthatjuk meg: *„CD/DVD képfájlok letöltése a BitTorrent segítségével. A Bittorrent peer to peer rendszer lehetővé teszi, hogy egyszerre több felhasználó együttműködve töltsön le képfájlokat, úgy hogy szervereink terhelése közben minimális legyen."*

Természetesen nem csak a *Debian* terjeszti a képfájlokat *torrent* segítségével, hanem minden nagyobb disztribúció is.

Mi is az a torrent?

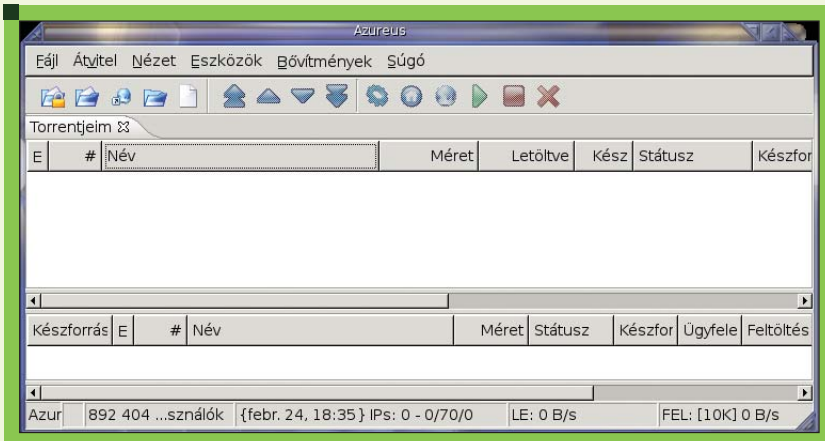
Sokunkban felmerülhet ez a kérdés. Sokan már hallottak/olvastak is róla, mégsem tudják, hogy valójában mi is ez. A *torrent* programok fájlcserélgető szoftverek. *Peer-to-peer* technológiát használnak, ezáltal elősegítik a gyors és biztos letöltést.

A *„torrent"*-et gyűjtőnévként használjuk, ilyen konkrét néven nem fogunk találni programot. Hogy miért ez a gyűjtőnév? Kifejlesztettek egy olyan programot, ami információs fájlokból nyeri a letölteni kívánt alkalmazás/zene/videó paramétereit (például az állomány hosszúságát), illetve a kapcsolathoz szükséges szerver adatait. Ezt a fájlt *.torrent* kiterjesztéssel látták el, innen a név. Nekünk nincs más dolgunk, mint letölteni ezt az információs fájlt, majd egy *torrent* programmal megnyitni. Ezután rövid időn belül megkezdődik a tényleges letöltés.
A *torrent*-ügyfeleknek (klienseknek) rengeteg fajtája van. Hogy csak néhány nevet említsek: *BitTorrent, BitTornado, Azureus*.
A következőkben a *Linux* rendszereken elterjedt *Azureus*-t fogom bemutatni részletesebben.

Azureus

Az *Azureus* (➲ http://azureus.sourceforge.net) egy *torrent*-kliens. Jelenlegi verziószáma 2.4.0.0. Számos platformra elérhető, többek között: *Linux Intel/AMD64/PPC, Windows 9x/XP, OSX, Solaris*. Felhasználóbarátságát mi sem bizonyítja jobban, mint hogy rengeteg különféle nyelven „tud", természetesen magyarul is. Ez hatalmas segítség lehet az idegen nyelveken nem beszélő felhasználóknak.

1. ábra Az Azrueus indulása is magával ragadó élmény



2. ábra A kezelőfelület

Java telepítése

Mivel ez a program *java*-ban íródott, ezért még mielőtt bármit csinálnánk, ellenőrizzük le, hogy a gépünkön megtalálható-e a *java*. Amennyiben nem, akkor le kell töltenünk és fel kell telepítenünk. Ez egy roppant egyszerű folyamat. Akár az *Azureus* honlapjáról is elérhető a *Sun Java* honlapja (➲ http://java.com/getjava), ahol hozzájuthatunk a legfrissebb binárishoz. A *java* telepítőfájl az

```
sh jre-1_5_0_06-linux-i586.bin
```

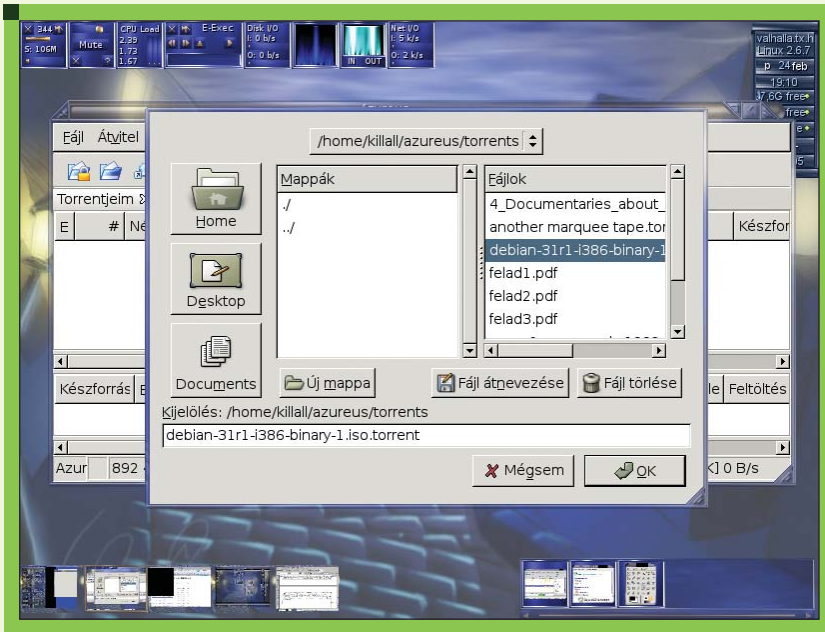
paranccsal kicsomagolható, amennyiben *Intel* processzoros gépünk van és a legújabb *java* verziót választottuk. Akár *root*ként, akár *user*ként hajtjuk végre a parancsot, a kicsomagolt könyvtár abban a könyvtárban marad, ahol a telepítő binárist futtatjuk. Ez általában a *home* könyvtárunk. Biztonsági okokból célszerű áthelyezni ezt a könyvtárat a */usr/lib*-be az

```
mv ~/jre1.5.0_06 /usr/lib/
```

paranccsal. Ezek után készítsünk *java* nevű szimbolikus linket a könyvtárról a */usr* alá az

```
ln -s /usr/lib/jre1.5.0_06
➥/usr/java
```

paranccsal. Ez a lépés annyiban könnyíti meg a munkánkat, hogy az *Azureus* telepítése után nem kell egyből konfigurációs fájlokat szerkeszteni a *java PATH* (elérési út) beállítása érdekében. Amennyiben feltelepült a *java* a gépünkre, belefoghatunk a tényleges telepítésbe, ami még a *java* telepítésénél is egyszerűbb.



3. ábra .torrent állomány megnyitása

Telepítés

Első dolgunk az *Azureus* honlapjáról letölteni a legfrissebb installációs állományt, ami jelen esetben egy tömörített *.tar.bz2* fájl.
A telepítés csupán a

```
tar xvjf
➥Azureus_2.4.0.0_linux.tar.bz2
```

parancsból áll, mely kicsomagolja az állomány tartalmát abba a könyvtárba (most célszerű a *home* könyvtárunkat használni erre a célra), ahol a tömörített fájl van.

Indítás

Az *Azureus* indításához (*1. ábra*) a következő pofonegyszerű parancsokra van szükségünk:

```
cd ~/azureus
```
(ha a *home* könyvtárunkba csomagoltuk ki), majd

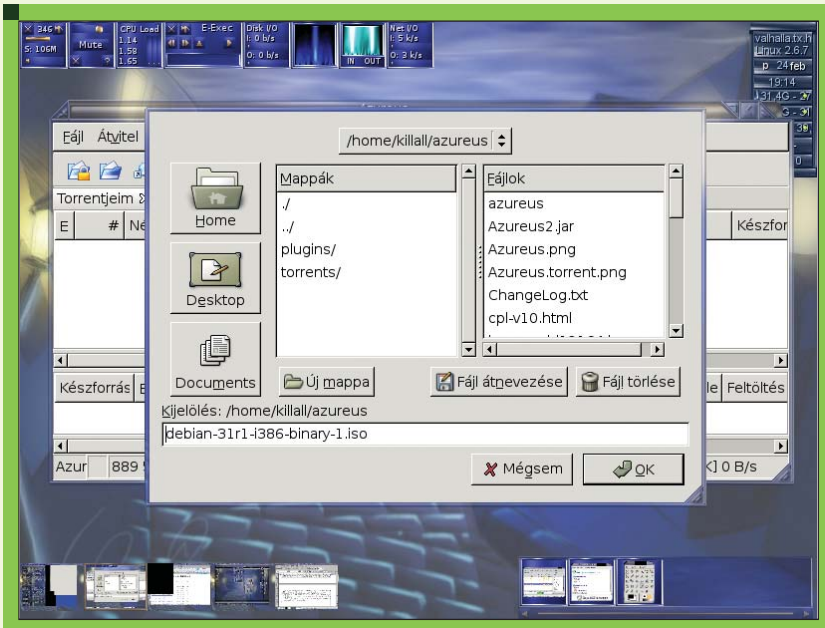
```
./azureus
```

Használatba vétel

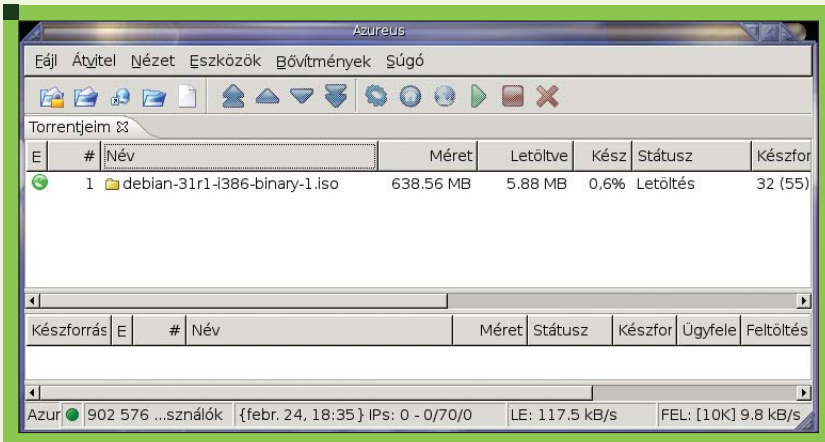
Köszönhetően a *java*nak, az *Azureus* kezelőfelülete mentes mind a *GTK*, mind a *QT* minden jellemzőjétől, így egy igazán homogén, ám egyszerű kezelőfelületen kezdhetünk hozzá a tényleges „munkához". Természetesen azért használok idézőjelet, mert mostantól kezdve tényleg nincs más dolgunk, mint néhányat kattintani, majd hátradőlve élvezni, amint a bitek áramlanak az éterből a gépünk merevlemezére.
Nézzük meg egy kicsit közelebbről ezt az alkalmazást és dobjuk egyből a mély vízbe. Azaz, kezdjük el használni arra, amire kitalálták: töltsünk le vele valamit!
Ezt demonstrálva, csak hogy maradjak a megszokottnál, a *Debian GNU/Linux* jelenlegi stabil kiadásának (*3.1*, azaz *sarge*) első telepítő CD-je lesz a cél.

Lássuk a teendőinket!

Ez a pont nagy vonalakban már szerepel ezen cikk hasábjain, bár akkor egy másik példát szemléltettem vele.



4. ábra A program rákérdez a letöltendő fájl végleges helyére



5. ábra A munkánk jól megérdemelt gyümölcse

Látogassunk el a ➲ http://debian.org webhelyre, majd a menüből válasszuk ki az *iso* fájlokra vonatkozó pontot. Ekkor megjelenik egy tetemes lista, ami tájékoztat bennünket arról, hogy milyen úton-módon juthatunk hozzá a képfájlokhoz. Válasszuk a *BitTorrent*-en keresztüli letöltés pontot, majd mentsük el a *.torrent* kiterjesztésű állományt. Célszerű a *~/azureus/torrents* könyvtárba menteni a *torrent* fájlokat, mivel az *Azureus* ott fogja őket alapból keresni. Nyissuk meg az imént letöltött állományt az *Azureus*-szal. Természetesen lehet a *.torrent* fájlt társítani az *Azureus*hoz, de én mégis a régi, fapados jól bevált módszert ajánlanám (*ctrl+o*, vagy *Fájl/Megnyitás*).
Az *Azureus* ezután rá fog kérdezni a letöltendő fájl mentési helyére. Alapértelmezettként minden befejezett letöltés a *~/azureus* könyvtárba kerül. Amint ezt „leokéztuk”, a tényleges letöltés megkezdődik. Hátradőlünk és megvárjuk, amíg a töltés befejeződik.
Ilyen pofonegyszerű az egész. Nincs szükségünk semmiféle megosztásra, mivel azt osztjuk meg, amit már letöltöttünk, mondhatni *„on-the-fly”* módon. A használatbavételhez még beállításokra sincs szükségünk. Szerintem mindenki ilyen programról szokott álmodni.
Természetesen rengeteg különféle beállítási lehetőségünk van. Például blokkolhatunk nem kívánt *IP*-címeket, készíthetünk saját megosztást, saját *torrent* fájl-t, fontossági listába szedhetjük a letöltéseket, amennyiben egyszerre több dolgot töltünk.
Amint már említettem, a program az utolsó megjegyzésig magyarított, el van látva *thumbnail*-ekkel, illetve mindent világosan nyomon követhetünk rajta.
Aki szeretne jobban belemélyedni a beállítások apró rejtelmeibe, annak tudom ajánlani a projekt honlapján lévő on-line dokumentációt (➲ http://azureus.sourceforge.net/doc/Azureus%20User%20Guide.htm). Képekkel, magyarázatokkal minden részletre választ kaphatunk, ha problémába ütközünk.

Összegzés

Mi, akik *Linux*ot használunk, eleve hívei vagyunk a tiszta dolgoknak. Valamennyien tudjuk, hogy mennyi munkába kerül egy program megírása, meg ha az apró is. Egy teljes játék, vagy komolyabb szoftver fejlesztése évekbe kerül. Ehhez az idő/munka befektetéshez képest szinte elenyésző az a pár ezer forint, amiért legálisan megvehetjük. Vagy, hogy egy másik példát említsek: mivel magam is zenélek, tudom, hogy mivel jár egy zeneszám megalkotása. Érzések, idő, tudás – ez mind közrejátszik benne.
Mindenki tudja, hogy drágák a CD-k manapság, de azért én mindenkit buzdítanék arra, hogy ezután is boltban vegye meg a zenei és informatikai kiadványokat. Ennyivel mindenki tartozik azoknak az embereknek, akik ezeket létrehozták.
A fájlcserélő programokat használjuk legális dolgokra, mint például szabad szoftverek terjesztésére, *Linux* disztribúciók letöltésére.
Amennyiben valaki mégis az illegalitást választja, legalább annyit tegyen meg, hogy a letöltött adatot megtartja magának és nem terjeszti tovább, azaz nem húz hasznot belőle.

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

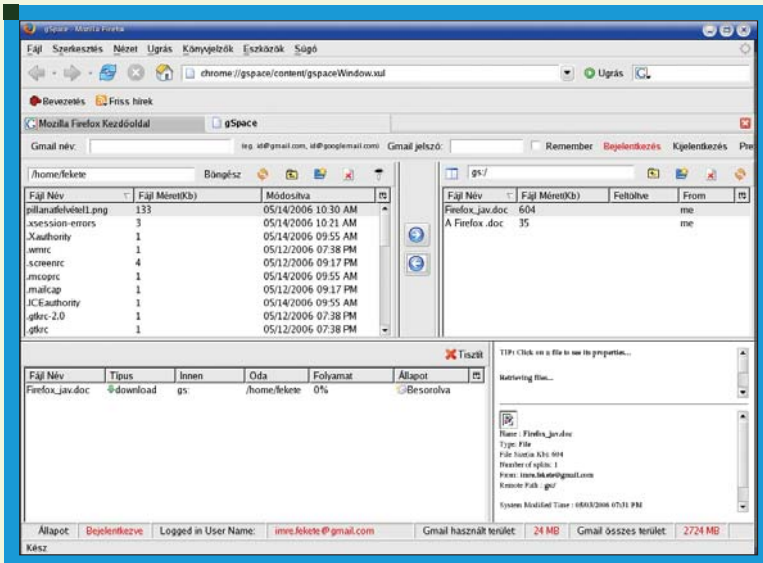
24 éves, jelenleg az ELTE programozó matematikus szakán másodéves hallgató. Hobbija a zene (gitározás), az olvasás (Stephen King) és a számítástechnika (Linux, Unix, VMS).

A Firefox kiterjesztései

Sok lelkes programozó nem elégedett meg a Firefox eddigi tudásával, hanem úgynevezett kiterjesztések fejlesztésébe fogott, melyek segítségével az élet különböző területeiről származó igények kielégítésére alkalmas funkcióval bővíthetjük a böngészőt. A következőkben ezekből mutatok be néhányat.

A *Firefox* egy ingyenes, nyílt forráskódú, *Mozilla* alapú webböngésző *Windows*, *Linux* és *Mac OS X* rendszerekre. Jól használható kiterjesztései nélkül is, de teljes értékűvé csak ezekkel válik. Már most számos elérhető ezekből, és számuk folyamatosan növekszik. Újakat telepíteni a következőképpen lehet: a *Firefox* menüjében *Eszközök/Kiterjesztések*-re kattintunk, és a felugró ablak jobb alsó sarkában kiválasztjuk „További kiterjesztések letöltése” lehetőséget.
Ekkor a *Mozilla* oldalára jutunk, és itt megtalálhatjuk az egyes kategóriákban a nekünk megfelelőt. A választható lehetőségekből mutatnék be párat:

1. ábra Ezen a képen a már telepített kiterjesztéseket összefoglaló ablak látható.



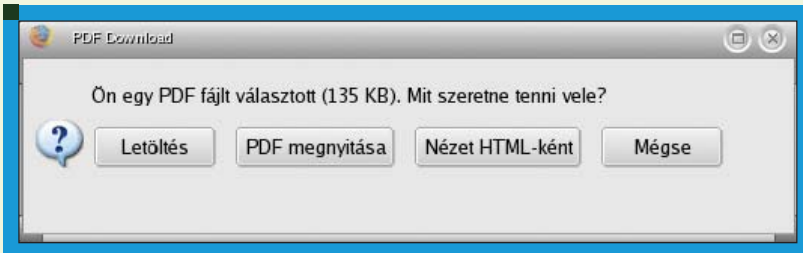
2. ábra Ezen a képen a Gmail Space látható munka közben

Gmail Space

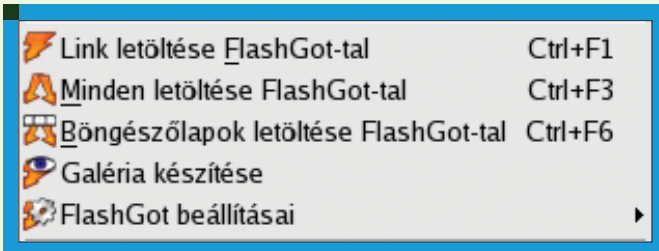
A *Gmail Space* használatához szükség van egy gmail-es e-mail címre. A segítségével az e-mail tárhelyünket fájlok tárolására használhatjuk fel. Feltelepülése után az *Eszközök/GSpace*-re kattintva indítható el, valamint megjelenik egy nyomógomb a böngésző jobb felső sarkában, ami a *Gmail Space* beállításainál kikapcsolható. A megjelenő új fül bal felső panelében jelenek meg az aktuális gép fájljai, bejelentkezés után pedig a jobb felsőn a Gmail-en tároltak. A húzd és ejtsd módszer nem működik a középen található nyilakkal lehet a fel/letöltés elindítani. A program üzenetei a jobb alsó sarokban lévő négyzetben jelenek meg angolul.

PDF Download

Ha egy olyan linkre kattintunk rá, mely egy *.pdf* kiterjesztésű fájlt nyitna meg, akkor ezzel a kiterjesztéssel már nem a *Firefox* alapmegjelenítője indul el, mely sok esetben lassú, hanem egy menü jelenik meg. Itt kiválaszthatjuk, hogy mi történjen az állománnyal.



3. ábra Ezen a képen a Pdf Download felugrómenüje látható



4. ábra Ezen a képen a Flashgot jobb egér kattintással elérhető funkciói láthatók

(Letöltés, megnyitás, nézet HTML-ként)
A ***Pdf Download*** beállításainál megadhatjuk, hogy milyen program nyissa meg a pdf állományt.

Flashgot

A legnépszerűbb kiterjesztés, mely leegyszerűsíti a kommunikációt a kedvenc letöltést vezérlő programunk és a ***Firefox*** között. Több tucatnyi vezérlővel működik együtt, melyek listája megtalálható a angolul: ➲ www.flashgot.net/features címen. Jobb egér kattintással elérhető funkciói:

– „Minden letöltése ***Flashgot***-tal" az oldal teljes tartalmat letölthetjük.
– „Böngészőlapok letöltése ***Flashgot***-tal" az előzményekben megtalálható honlapok közül választhatjuk ki, hogy melyiket akarjuk letölteni.
– „Link letöltés", ha egy linkre kattintunk, letölti annak tartalmát.
– „Kijelölés letöltése ***Flashgot***-tal" A felhasználó által kijelölt tartalom letöltése.
– „Galéria készítése" A kiterjesztés által nyújtott bónusz funkció, mely segítségével összeállíthatunk egy oldalt a különböző helyekre elszórt hasonló nevű fájlokból. Így könnyebben és gyorsabban tölthetjük le azokat, mint a „Minden letöltése Flashgot-tal" funkcióval.

Galéria készítése

Miután a galéria készítésre kattintottunk megjelenik egy ablak ahol beállíthatjuk, hogy milyen minta szerint ismétlődik a tartalma. Ismétlődő mintát []-ben kell megadni

1. Ha számokban térnek csak el, akkor :

```
[elsőszám-utolsószám;lépésköz]
```

Például:
`www.valami.com/konyvtar`
`➥[01-10;1]/fájl[01-10;1].jpg`

Melynek jelentése a `konyvtar01`-gyel kezdve 1 lépésközzel készítsen galériát a `fájl01`-től a `fájl10`-ig egy lépésközzel az összes ilyen nevű fájlból

2. Ha az ábécé betűiben különböznek akkor: `[kezdőbetű-utolsóbetű]` kell megadni, ebben az esetben nincs lépésköz. Egyesével végig megy az ábécé megadott részén.

3. ***JavaScript*** függvényt is megadhatunk [] között, de akkor ezt előtte definiálni kell a ***„Javascript függvény"*** fülön.

Tabbrowser Preferences

Feltelepülése után bővül a ***Firefox*** fülkezelési beállításainak a lehetőségei. Lássunk néhányat ezek közül.
Hivatkozások/Külső hivatkozások megnyitása: Új ablakban (Ez a ***Firefox*** alapértelmezése), új böngészőlapon vagy a jelenlegi böngésző lapon.
Megjelenítésen belül megadható, hogy a fülek alulra kerüljenek, megjelenjen egy gomb, melynek segítségével új fület nyithatunk (Ez gyorsabban megoldható a `Ctrl+T` lenyomásával)
A ***Beállításoknál*** megadhatjuk, hogy egy fület a középső egérgombbal bezárhassunk, és legyen-e figyelmeztetés, ha több böngésző lapot tartalmazó ablakot akarunk bezárni. Ez utóbbi igen hasznos, ha mondjuk valamelyik lapon be vagyunk jelentkezve egy oldalra, és elfelejtettünk kijelentkezni.
Böngészőlapoknál megadható többek között, középső egérgombbal nyithassunk meg egy hivatkozást. Böngészőfüleket aktiválódása, ha az egér föléjük ér.

Ambiance

Ha valaki már unja a szürke színű böngészőt és szeretné, ha a program, mint a kaméleon változtassa a szintén, akkor érdemes feltelepíteni az ***Ambiance***-t, mely megpróbálja kitalálni, hogy milyen színű az éppen meglátogatott oldal, és a böngésző kinézetét ehhez alakítja.
Zárásképpen megemlíteném, hogy ugyanarra a problémára már többféle megoldás is születhetett. Az itt bemutatott bővítmények csak kiragadott lehetőségek, melyeket gondolatébresztőnek szántam. Érdemes havonta ellátogatni a Mozilla oldalára, majdnem biztos, hogy egy újabb érdekes funkcióval bővülhet a böngészőnk.

Fekete Imre
(imre.fekete@gmail.com)

Programtervező-matematikusként végeztem a Debreceni Egyetemen. Jelenleg rendszergazdaként dolgozom. Kedvenc böngészőm a Firefox.

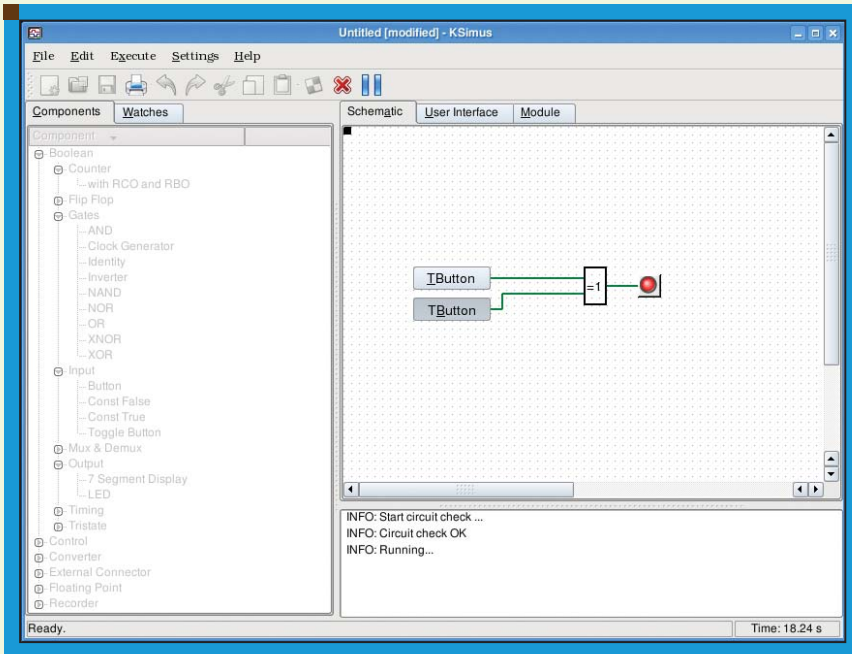
Bedrótozva – Logikai áramkörök szimulációja Linux alatt

Azok a Linux felhasználók, akik nemcsak egyszerű hobbiként űzik a számítástechnikát, hanem intézményesített keretek között is van lehetőségük tanulni róla, előbb-utóbb találkoznak a logikai függvények és áramkörök, bővebben a digitális technika varázslatos világával. Kereskedelmi szimulációs eszközök sokasága lelhető fel, melyek jó része platformfüggetlen, de mitévő legyen az a lelkes halandó, aki csupán egyszerű logikai áramköröket szeretne szimulálni?

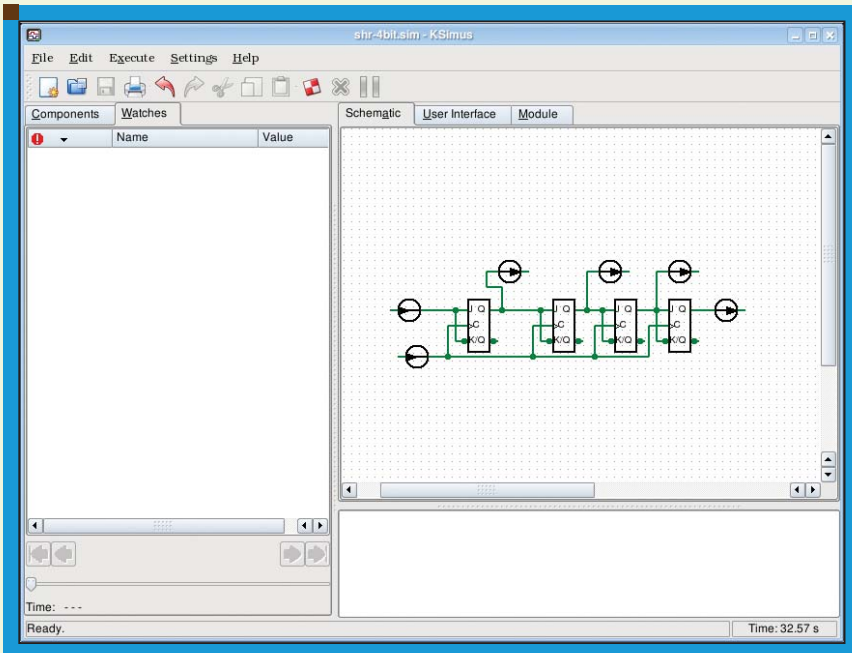
Ez a cikk bevallottan szűk réteghez szól: ha a nyájas olvasó a flip-flop szóról a jin-jangra, a regiszterről pedig határidőnaplóra asszociál, akkor érdemes lapozni: a többi cikk talán több izgalmat tartogat!
No de mi ez a mogorva kiszólás? Csupán tudni kell, hogy a most bemutatandó programok használatában kizárólag a digitális technika elemi ismerete segítségével tudunk egyről a kettőre jutni.
Ha ebben a témakörben kicsit is elmélyedünk, óhatatlanul szükségünk lesz arra, hogy az elméletben, papíron szépen mutató függvényeket és kapcsolásokat kipróbáljuk. Két út kínálkozik: a helyi elektronikai boltból beszerezni a kívánt alkatrészeket és a szó szoros értelmében megépíteni a kapcsolást vagy egy szimulációs-tervező környezetben dolgozva néhány egérkattintás után megcsodálni a működő rendszert. Az első verzió is tartogat érdekességeket, de az már végképp a magazin profilja által felállított kereteket feszegetné, így most egeret és billentyűzetet ragadva lássunk munkához!

Egy egyszerű megoldás

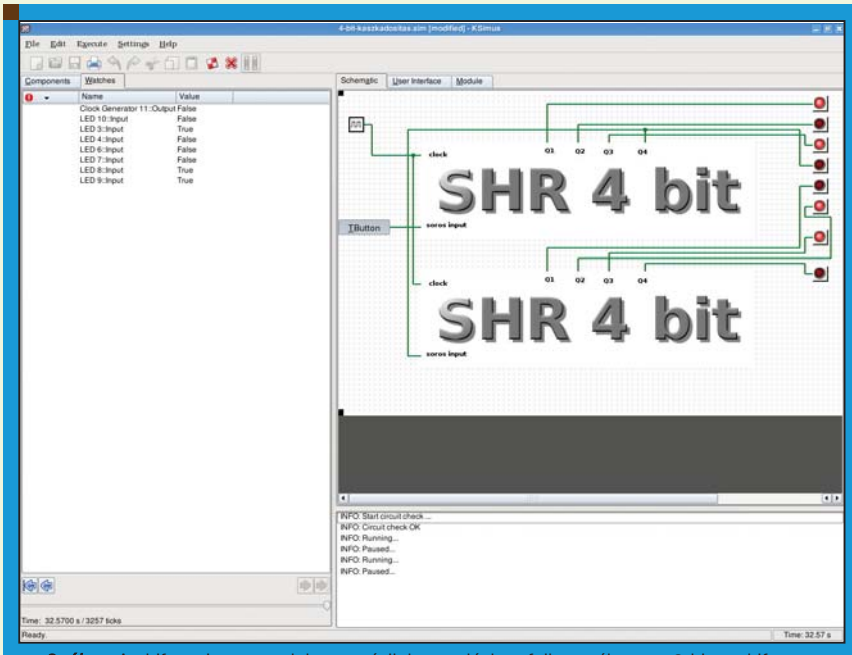
Elsőként a *KSimus* szoftverrel fogunk megismerkedni, amely *Rasmus Diekenbrock* munkája. Ezzel a szoftverrel pontosan azt kapjuk, amire a témával most ismerkedő felhasználónak szüksége lehet: könnyen és gyorsan össze tudunk rakni egy egyszerűbb kapcsolást, de akkor sem hagy cserben minket, amikor már összetettebb feladatok felé kacsingatunk. A 0.3.6-os verziószámot viselő csomagot, ha nem része a disztribúciónknak, a ➲ *http://ksimus.berlios.de/* címről tudjuk letölteni.
Kicsit jártasabb olvasóink biztos sejtik, hogy a K betűvel kezdődő név a *KDE* könyvtáraktól való függőséggel jár együtt és ez így is van. *Gtk* alapú logikai szimulátor létezik (*tlogsim*), de sokkal kevésbé funkciókban gazdag, mint *KDE*-s társai. A *KSimus* indulása után szembeötlik, hogy habár a rendszeren a magyar locale be volt állítva, angol nyelvű maradt a felület. Ezen persze bárki könnyedén segíthet, hiszen a szabad programok lefordításában akárki részt vehet. Még az is előfordulhat, hogy mire megjelenik a cikk, elkészül a fordítása a programnak.



1. ábra Első összeállított egyszerű kapcsolásunk



2. ábra Shift regiszter kapcsolási rajzon, felkészítve modulként való felhasználásra



3. ábra A shift regiszter modul egy másik kapcsolásban felhasználva egy 8-bites shift regiszer céljára

A felhasználói felület nem bonyolult: a komponensek listája és a kapcsolási rajz tervezőfelülete tűnik elő indításkor. A komponensek listájában a ***Boolean*** csoport az, ami az első szárnypróbálgatásokhoz kell. A tankönyvek, jegyzetek és diák lapjairól oly jól ismert alkatrészek sorakoznak itt: különféle kapuk, flip-flopok, számlálók, multiplexerek és így tovább. A komponensek gyűjteménye egyáltalán nem olyan teljes, mint némely kereskedelmi szoftveré, de ez két okból nem baj. Egyfelől az kezdeti lépések megtételéhez több mint elegendő a választék, másrészt ami hiányzik, elkészíthető. A program nagy erőssége, hogy új komponenseket (ezeket moduloknak nevezi a program), tetszőleges belső kapcsolással magunk is összeállíthatunk. Így hát nem baj, hogy shift regiszter mondjuk nincs a felsorolásban, kis erőfeszítéssel csinálhatunk egyet, amit aztán akárhányszor felhasználhatunk. Most már tényleg lássunk munkához! A komponensek ***Boolean/Gates***, ***Boolean/Input*** és ***Boolean/Output*** csoportjaiból van szükségünk elemekre egy elő példára. A komponens nevére történő kattintással az adott elemet beszúráshoz kijelöltük, utána ha a ***Schematic*** fülön lévő lapon kattintunk egyet, létrejön az elem most már ténylegesen a kapcsolási rajzunkon. Tegyünk egy próbát az ***XOR***-ral, a kizáró vagy logikai kapcsolat kapujával! Egy kapu önmagában mit sem ér, most a ***Toggle Button***-ból helyezzünk el két darabot, aztán pedig a ***LED***-ből egyet. Néhány programban az alkatrészek összedrótozása precíz egérmozgatást igényel, itt a két ki illetve bemenetet (természetesen kimenetet bemenettel és viszont) kell kijelölnünk és a program elkészít egy megfelelő vezetékezést. Nehezen bírható rá a program szerencsére, hogy átláthatatlan és kaotikus vonalakkal borítsa el a kapcsolást huzal címén, hacsak eleve az alkatrészeket logikátlanul helyeztük el. Ezután az ***Execute/Start*** menüpontot kiválasztva életre kel a kapcsolás, a gombok által adott logikai értékek függvényében a ***LED*** hol világít, hol nem. Most a példánkban csak akkor világít a ***LED***, ha pontosan az egyik kapcsoló van benyomott állapotban.

Modulok, ameddig a szem ellát

Vegyünk egy shift regisztert! Azaz vennénk, de a komponensek között hiába keressük. Sebaj, interneten shift regiszter kapcsolást lehet találni. Egy saját modult fogunk belőle készíteni. Tervezzünk tehát négy bites shift regisztert! Kell hozzá négy *D* flip-flop néhány ki és bemenet és összeköttetések. A ➲ ***http://www.eelab.usyd.edu.au/digital_tutorial/part2/register02.html*** címen található kapcsolást készítjük el. Modult pontosan ugyanúgy kell csinálni, mint bármilyen más kapcsolási rajzot: elhelyezzük az alkatrészeket, ki- és bemeneteket, huzalozunk, tesztelünk. A program egyik hátránya máris szembetűnik: *D* flip-flop nincs, a ***J-K*** flip-flopot lehet átalakítani. Ezt úgy tehetjük meg, hogy a ***J-K*** flip-flop *K* lábára a *J*-re adott jel negáltját kötjük. Ha követjük a kapcsolási rajzot a megadott weblapon, úgy ezzel el is lehet készíteni a shift regisztert.

Hogy legyen belőle modul? A gombok és lámpák helyére tegyünk ***Extrernal Connectorokat***, ***Bool Inputot*** a gombok, ***Bool Outputot*** a lámpák helyére. Ennyi az egész. Természetesen a belső, csak tesztcéllal elhelyezett ledek helyére nem kell kivezetést tenni. Ezután a ***Module*** fülön eldönthetjük, hogyan is nézzen ki újonnan létrejött alkatrészünk. Akár saját képet is rakhatunk az alkatrész felületére. Ami ennél azért lényegesebb: itt rendelhetjük hozzá az egyes ***Input/Output***-okat az alkatrész dobozán kivezetésekhez. Miután elmentettük a fájlt, máris felhasználható. Hozzunk létre egy új fájlt, aztán válasszuk az ***Edit/Insert module***-t és szúrjuk be a tervbe a saját alkatrészünket, akár több példányban. Én a példánál maradva méretnöveléssel 4-bites shift regiszterből készítettem egy 8-biteset. A példában szereplő megtervezett alkatrész nem teljes, egy tisztességes regisztert le kellene tudni törölni, amit itt nem lehet.

A lehetőségek innen már csak a tervező fantáziájának szabta korlátokba ütközhetnek! Biztos ez? Akár ***KDE*** alá is találhatunk a fentinél nagyobb tudású rendszert, a ***KTechLabot*** (➲ ***http://ktechlab.org/***), amit szintén mindenkinek ajánlok kipróbálásra, mert logikai áramkörökön túl még nagyon sok másra is alkalmas.

Mi az oka, hogy mégis a ***KSimus*** került terítékre? Egyszerűen az, hogy ezzel a szoftverrel gyorsan lehet látványos eredményeket elérni és nagyon gyorsan meg lehet tanulni a használatát. Persze akik nem csak játékszernek vagy tanulásuk eszközének tekintik a digitális tervezést, azok megmosolyogják ezeket a primitív eszközöket. Igazuk is van. Messzire menni mégsem kell, az ***Icarus Verilog*** majd minden disztribúcióban helyet kapott. Sőt mi több, ***Linuxvilág*** negyedik lapszámában ***Michael Baxterrel***, az ***Icarus Verilog*** fordító készítőjével is olvashatunk egy érdekes interjút. Létezik, hogy nincs meg az összes lapszám kezdetektől fogva? Nagy hiba, de a ➲ ***www.linuxvilag.hu*** minden regisztrált felhasználónak elérhetővé tette a cikket (nemcsak ezt, hanem számtalan régebbi lapszám cikkét is).

Akár kedvtelésből, akár szakmai alapokkal közeledünk a digitális tervezés felé, tapasztalhatjuk a szabad szoftveres megoldások életképességét. Az egyszerűbb igényeknek a grafikus felületű, könnyen megtanulható kapcsolási rajz alapú tervezők állnak, a tényleges tervezői munkához pedig a ***Verilog*** hardverleíró nyelv fordítója, az ***Icarus*** nyújt segítséget. A Veriloghoz egy rövid leírást találunk a ➲ ***http://www.asic-world.com/verilog/veritut.html*** webcímen. Míg a ***KSimus*** és a ***KTechLab*** digitális technikában járatosaknak néhány kattintás után kiismerhető, a ***Verilog*** olyan, mint egy programozási nyelv, hosszabb ideig tart, mire ténylegesen értelmes dologra tudjuk használni.

Novák Áron
(aaron@szentimre.hu)

BME-VIK-es hallgató, műkedvelő rendszergazda. Jelenleg leginkább a NetBeans-szel és mindenféle hordozható eszközzel foglalkozik, legalábbis mindazokkal, amelyeket meg lehet szólaltatni Linux alatt.

Digitális nosztalgia – Spectemu

A sokak által zseninek tartott Sir Clive Sinclairnek minden kétséget kizáróan a ZX Spectrum volt a főműve. A ZX80 és ZX81 utódja kora legnépszerűbb géptípusa, jelentőségét sokan a C64-eséhez hasonlítják. Közkedveltségét többek között a nyolcvanas évek elején ritkaságnak számító színes megjelenítésnek, és egyszerű programozhatóságának köszönhette. Programozási nyelve a Sinclair Basic mára legendává vált.

Történetünk a már távolinak tűnő 1982-ben kezdődik. Az angol ***Sinclair Research Ltd.*** ekkor rukkolt elő legújabb termékével a ***ZX Spectrum*** személyi számítógéppel. A géptípust a 3.54 MHz órajelű ***Zilog Z80A*** processzor hajtotta. (Egy mai közönséges zsebszámológép 5-6 MHz-es processzorral rendelkezik.) Ehhez 16 kB ROM és 16 kB vagy 48 kB-os memória tartozott (két kiszerelésben volt kapható). A 16 kB-os változathoz 125 Font-ért lehetett hozzájutni, 48 kB-os testvérét pedig 175 Font-ért árulták. A dobozban volt egy tápkábel, egy monitorkábel, egy vezeték a gép és a magnó összekötéséhez, egy rövidebb és egy hosszabb leírás, na és természetesen maga a számítógép volt *(1. ábra)*.

A ***Spectrum*** fizikai méretei alig haladták meg egy 400 oldalas A5-ös formátumú könyvét. Súlya körülbelül fél kiló lehetett. Hátoldalán csupán öt csatlakozó található ide a tápkábelt és monitort kellet csatlakoztatni és máris hadra fogható állapotba került a gépezet. Be/ki kapcsoló illetve reset gomb volt rajta ezeket funkcionálisan az áramforrás kihúzásával és bedugásával kellet helyettesíteni.

A maradék három csatlakozóra egy magnót és egy külső kiegészítőt (memóriakártya, Joystick …) lehetett kapcsolni. Mivel a ***Spectrum*** háttértárolóval illetve mágneslemez meghajtóval nem rendelkezett, a programokat magnókazettán terjesztették. „Beviteli egységként" gyakorlatilag bármilyen egy átlagos háztartásban megtalálható magnó alkalmas volt. (A szerző egy ***Philips*** kazettás magnót használt erre a célra). A kábelen átvezetett hangot a ***Spectrum*** bináris adatokká alakította, majd miután az egész programot betöltötte, elkezdte futtatni. (A betöltés percekig tartott, feltéve hogy a felhasználónak mindent elsőre sikerült eltalálnia...) A felhasználó által készített programot ennek az eljárásnak a megfordításával lehetett kazettára kimenteni, majd későbbi használat előtt visszatölteni.

1. ábra A Spectrum gyári kiszerelésben

A mai számítógépektől eltérően a billentyűzet a „gépház" tetején kapott helyet így azt nem külső eszközként kellet csatlakoztatni. A billentyűzet *(2. ábra)* az angol abc-t, számokat nullától kilencig és néhány speciális billentyűt tartalmazott. A használat gyorsítása érdekében

■ *2. ábra* A Spectrum billentyűzete

a számítógép parancsait és – beépített programozási nyelve – a ***Sinclair Basic*** kulcsszavait egy-egy billentyű mint második illetve harmadik funkciót tartalmazta. A kulcsszavakat tehát nem kellet begépelni, csupán a megfelelő gombot leütni.
„Monitorként" bármilyen ***PAL*** rendszerű ***TV*** szolgálhatott. Ha a ***TV*** készülék képes volt színes megjelenítésre akkor a ***Spectrum*** 8 színben pompázott (a géptípus neve is a színes megjelenítésre utal). Grafikus üzemmódban 256 x 192 pixeles felbontásra, míg szöveges módban 32 x 24 karakter megjelenítésére volt képes.
Egy beépített egy csatornás „hangszóró" segítségével a ***Spectrum*** hangkeltésre is alkalmas volt. Ez a gyakorlatban annyit jelentett, hogy különböző hangtartományokban sípolt.
(A ***Spectemu*** a ***Spectrum*** grafikus és hangkeltő képességét egyaránt emulálja ezért ezt később kipróbáljuk.)
A ***ZX Spectrum*** hátoldalán helyett kapott egy bővítőkapu, ahova a géptípushoz elérhető különböző külső eszközöket lehetett csatlakoztatni. Néhány ilyet a ***Sinclair Ltd***. is készített (például nyomtatót, memóriakártyát), de szép számmal akadtak más gyártótól származó kiegészítők is, sőt házi készítésűek is felbukkantak.

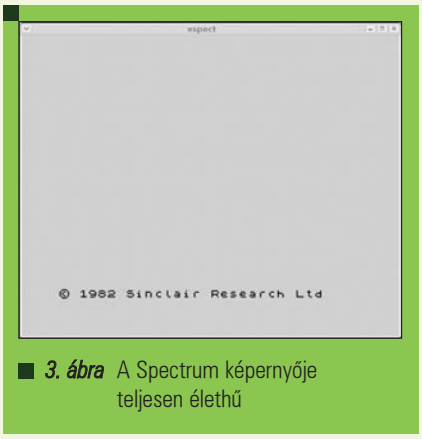
A Spectemu

Manapság már nehezen beszerezhető ez a géptípus, ráadásul a javítása is elég nehézkes vagy lehetetlen, (a ***Spectrum*** sajnos tartalmaz egy konstrukciós hibát, a billentyűzet alatt található fólia nagyon hamar tönkremegy) ezért a nosztalgiára vágyók kénytelenek valamilyen ***Spectrum*** emulátort használni.
Ilyen például a ***Spectemu,*** amely emulálja a teljes ***Zilog Z80*** utasításkészletet, fut ***GNU/Linux***-on és természetesen nyílt forráskódú. Ráadásul mivel bizonyos részei assemblyben vannak megírva, igen gyors is: egy 25 MHz-s 486-os számítógépen is tökéletesen fut.
Előnye, hogy grafikus és karakteres terminálon egyaránt képes futni, tud pillanatképet készíteni az emulált gép állapotáról majd szükség szerint azt visszatölteni. Képes kazettafájlokból programokat betölteni és azokba programokat menteni, valamint a ***Spectrum*** hangkeltő eszközét is emulálja.
Hátrányként említhető hogy a program nem rendelkezik grafikus konfiguráló eszközzel, sem grafikus felhasználói felülettel (a ***Spectrum*** billentyűzetét leszámítva) ezért a programot különböző billentyűkombinációk lenyomásával kell vezérelni.
A cikkben csak a grafikus változatot tárgyalom, a leírtak pedig a 0.94a-3-as Debian csomagból telepített változatra vonatkoznak.
A telepítés nem különösebben nehéz. Debian felhasználóknak nem kell mást tenni, mint a ***spectemu-x11*** és a ***spectemu-common*** csomagokat telepíteni. (A csomagok a ***contriban*** foglalnak helyet ezért a ***sources.list***-ban ezt is engedélyezni kell ezt is.)
A ***Spectemu*** amúgy ***rpm*** csomag formájában is letölthető a program hivatalos oldaláról (➲ ***http://www.inf.bme.hu/~mszeredi/spectemu)***.
Ha ezek a formátumok nem felelnének meg, a fenti oldalról megszerezhetjük a program forráskódját is, így magunk is lefordíthatjuk.
Miután valamilyen módon telepítettük a programot, indítsuk el az `xspect` paranccsal. A felbukkanó ablakban megjelenik a © ***1982 Sinclair Research Ltd*** felirat (*3. ábra*), akárcsak a valódi ***Spectrum*** esetében. A mai számítógépektől eltérően nem kell rendszerindításra várni, rögtön meg lehet kezdeni a munkát.

Játsszunk!

Legelőször játsszunk valamit! Ehhez meg kell szerezni néhány kazettát amin ***Spectrumhoz*** íródott játékok vannak. Arra is lehetőség, hogy valódi magnókazettát digitalizáljunk be, és azt használjuk, de most az egyszerűbb megoldást választjuk: valamilyen már digitalizált kazettáról fogunk betölteni egy játékot.
A digitalizált kazettákat külön erre a célra kitalált fájlformátumokban tárolják. Maga a ***Spectemu*** a ***tap*** és a ***tzx*** formátumot képes kezelni. Ilyen fájlokat többek közt a ➲ ***http://www.worldofspectrum.org/games/index.html*** oldalról lehet letölteni. Az oldalon található listából csak

olyan játékokat töltsük le aminek neve mellet a ***Memory*** oszlopban 48 szerepel, a ***Spectemu*** ugyanis a ***Spectrum*** 48K memóriával rendelkező változatát emulálja. (Egyes kazettafájlok tömörítve vannak!)
A ***Spectemuban*** nyomjuk meg a ***Ctrl-k*** billentyűkombinációt, amire megjelenik a ***Spectrum*** billentyűzetét emuláló ablak. A virtuális billentyűzeten nyomjuk meg a ***J*** gombot, mire az emulált gép képernyőjén `LOAD` szó jelenik meg. Amint említettem, a ***Spectrum*** billentyűzetén minden gombhoz egy vagy több parancs tartozik, és mivel még nem adtunk meg semmilyen parancsot ezért automatikusan ez a funkció lép működésbe.
Most gépeljünk be két idézőjelet (a rendes billentyűzeten tartsuk nyomva a jobb ***Shift*** gombot majd a virtuális billentyűzeten üssük le kétszer a ***P***-t) és nyomjunk ***Enter***t. Üssük le a ***Ctrl-p*** billentyűkombinációt és hozzuk előtérbe azt a terminálablakot, amiben kiadtuk az ***xspect*** parancsot.
A terminálablakban most már megjelent a `Enter tape file path` szöveg. Gépeljük be a betölteni kívánt kazetta fájl elérési útját a ***.tap*** vagy ***.tzx*** kiterjesztéssel együtt és üssünk ***Enter***-t (a PC billentyűzeten). Az emulált gép képernyője ettől elkezd villogni, és különböző színekben pompázni. Ez jó jel, ugyanis azt mutatja, hogy a kazetta elkezdett betöltődni. (Ez a folyamat egyébként teljesen élethű, a valódi ***Spectrumnál*** is így ment a dolog.)
Miután a program a kazettáról teljesen betöltődött, a terminálon megjelenik az `End of Tape` felirat, és máris elkezdhetjük használni.
A kazettáról való betöltés persze így meglehetősen lassú folyamat, ám aki türelmetlen, használhatja a ***gyors betöltés*** funkciót (a ***Spectemu*** szerzője ***Quick load***-nak nevezi). A valódi ***Spectrumon*** ilyesmi természetesen nem volt. Ez a módszer azoknál a programoknál nem használható amelyek saját maguk gondoskodnak a kazetta betöltéséről. Az ilyen program onnan ismerhető fel, hogy az első blokk kazettáról való betöltése után megáll. Ekkor mindenképp a normál betöltést kell alkalmazni.
A gyors betöltés funkció bekapcsolásához a ***Ctrl-y*** billentyűkombinációt kell lenyomni a ***Spectemu*** főablakában. Ezután kell a `LOAD` ***„”*** parancsot kiadni, majd a terminál ablakba begépelni a kazettafájl teljes elérési útját és ***Enter***-t ütni.



3. ábra A Spectrum képernyője teljesen élethű

Programozzunk!

Miután eleget „játszottunk” evezzünk komolyabb vizekre. Írjunk néhány egyszerűbb programot. A ***ZX Spectrum*** beépített programozási nyelve a ***Sinclair Basic***. A billentyűzet ezen nyelv kulcsszavait is tartalmazza ezért azokat nem kell begépelni csupán a megfelelő billentyűkombinációt lenyomni.
A ***Spectrum*** sem ***reset*** sem ***ki/be*** kapcsoló gombot nem tartalmazott ezeket a tápkábel kihúzásával és csatlakoztatásával kellet helyettesíteni.
A ***Spectemu*** (hogy ne keljen minden alkalommal kilépni majd újra elindítani a programot) ezt a műveletsort a ***Ctrl-q*** billentyűkombinációval helyettesíti.
Miután újraindítottuk a virtuális gépet, gépeljük be az *1. listában* látható programot. A ***Basic*** nyelv a ma használt legtöbb programozási nyelvtől eltérően megköveteli, hogy a programsorokat beszámozza a felhasználó. Bizonyára feltűnt az is, hogy a programsorok számai nem egymást követő számok. Ez azért van mert ha később a felhasználó be akar szúrni egy sort két már meglévő sor közé, azt úgy teheti meg hogy a meglévő sorok száma közé eső számot ad az új sornak.
Ezekből most már világosan látszik hogy az *„a”* és *„b”* program között semmi különbség.
Ez az egyszerű kis program a ***Spectrum*** színes képességeinek szemléltetésére való. A ***Spectrum*** 8 szín megjelenítésére képes. A színekhez számok vannak rendelve a nullától (fekete) hétig (fehér). A képernyő két részre van osztva egy külsőre ami-

1. Lista Programozzunk!

```
"a" változat

10 FOR x=0 TO 7
20 BORDER x
30 PAPER 7-x: CLS
40 PAUSE 50
50 NEXT x

"b" változat

10 FOR x=0 TO 7
30 PAPER 7-x: CLS
40 PAUSE 50
50 NEXT x
20 BORDER x
```

nek a neve `BORDER` és egy belsőre aminek a neve `PAPER`. Ezeknek a területeknek a színét a `BORDER` és a ***PAPER*** parancsokkal lehet állítani oly módon hogy a parancs után megadjuk a kívánt színt.
A példaprogram pontosan ezt teszi a `PAPER` és a `BORDER` színét változtatja nullától hétig. A `FOR` egy ciklus, ami az `x` változótól – aminek kezdeti értékét egyben nullára állítjuk – függ, a ciklustörzs addig ismétlődik amíg `x` értéke nem nagyobb mint hét. A `30`-as számú sorban lévő `CLS` parancs kikényszeríti a képernyő újrarajzolását, így az új színbeállítások érvényesülnek. A `PAUSE` paranccsal időlegesen meg lehet állítani a program futását, jelenesetben egy másodpercet fog várni mielőtt a következő sorra ugrik. A `NEXT` parancs az `x` változó értékét növeli egy megadott értékkel, a példánkba egyel. (ez az alapértelmezett érték ezért ez sehol nincsen megadva a példaprogramban)
Nem szükséges semmilyen szövegszerkesztő elindítás, a program begépeléséhez egyszerűen hívjuk elő a virtuális ***Spectrum*** billentyűzetet a ***Ctrl-k*** billentyűkombinációval és kezdjük el gépelni. Először a numerikus billentyűk segítségével gépeljük be a kódsor számát majd üssünk egy szóközt.
Most nyomjuk le az ***F*** gombot, és – szóköz kihagyása után – gépeljük be hogy `x=0` (a ***Symbol Shift*** gomb a rendes billentyűzeten a jobb ***Shiftnek*** felel meg). A `TO` is beépített kulcsszó ez az ***F*** betűn van, a ***Symbol Shift*** nyomva

A Spectrum billentyűzetének használata

A *Spectemu* emulálja a *Spectrum* billentyűzetét, ezt a *Ctrl-k* billentyűkombinációval lehet előcsalogatni. A *Spectrum* billentyűzeten a gombokat az egér mutatójának segítségével kell lenyomni.

Az egyszerűség kedvéért a *Spectrum* billentyűzet összes gombja megfelel a *PC* billentyűzet egy-egy gombjának. A betűk, számok, az *Enter*, a *Space* (szóköz), a *Backspace* és a nyilak ugyan azok mindkét billentyűzeten. Ha az *ALT* gombot lenyomva tartjuk akkor a különböző szimbólumokat (,./;'-=<>?:"_+[]{}\|~) a *PC* billentyűzet kiosztásának megfelelően kell gépelni.

Alapértelmezetten a *PC* billentyűzet egyéb billentyűi az alábbiak szerint felelnek meg a *Spectrum* billentyűinek:

Left Shift -> *Caps Shift*
Right Shift -> *Symbol Shift*
Back Space -> *Caps Shift + '0'*
Escape -> *Caps Shift + '1'*

tartása közben nyomjunk meg az *F*-et. Ezután gépeljük be a hetet és üssünk *Enter*t. A tizedik sor – feltéve hogy mindent jól ütöttünk be – a képernyő tetejére kerül, elkezdhetjük gépelni a következő kódsort. (A gombok alá írt utasításokat úgy kell aktiválni hogy nyomva tartjuk a *Symbol Shiftet* és leütjük a *Caps Shift*-et majd a parancsot tartalmazó gombot.)
Ha valamit netán elszúrunk például a tízes számú kódsorban akkor az úgy javítható hogy egy újabb kódsort ütünk be tízes számmal, ekkor a *Spectrum* lecseréli azt az újabbra. Miután mind az öt sort a fentihez hasonló módon begépeltük, nyomjuk meg az R gombot majd az *Entert*. Az *R* billentyűhöz rendelt parancs a RUN a program futását eredményezi.
A következő példaprogram a *Spectrum* hangkeltő képességeinek bemutatására készült:

```
10 FOR x=0 TO 24
20 BEEP 2, x
30 NEXT x
```

Futása során az összes hang halható amit a *Spectrum* képes kiadni.
Ami új az előző példaprogramunkhoz képest az a BEEP parancs. Ennek segítségével utasíthatjuk a *Spectrumot* hang kiadására. A BEEP-nek két szám a paramétere, ezeket vesszővel elválasztva kell írni. Az első szám egy időintervallumot jelent, ennyi ideig szól egy hang, a második szám a hangamagasságot jelenti.

Miután a felhasználó elkészít egy programot felmerül az igény, hogy azt elmentse későbbi használatra. Erre a SAVE parancs nyújt lehetőséget, segítségével kazettára lehet menteni a felhasználó által készített programokat. A SAVE kulcsszó után (*S* billentyűn van) idézőjelek között kell megadni hogy milyen néven szeretnénk menteni a programot, például SAVE „prog1”. Ezután nyomjuk le a *Ctrl-r* billentyűkombinációt hogy elinduljon a felvétel, majd a terminálablakban amiből elindítottuk a *Spectemut* adjuk meg a kazettafájl nevét amibe az adatokat menteni kívánjuk.
Ha ez a fájl már létezik akkor az új információ a már már meglévők után írja föl a program. Ilyen esetben a LOAD parancsnál nem elég paraméterként a dupla idézőjel hanem a programunk nevét, jelen esetben prog1-et kell megadni. Eddig azért volt elég megadni csupán két idézőjelet a LOAD-nak mivel feltételeztük hogy a kazettán csupán egy program van, és a *Spectrum* alapértelmezetten az első programot tölti be.
A *Spectrum* billentyűzetén nyomjuk meg egy gombot hogy elinduljon az adatok kimentése, majd várjunk amíg ez teljesen végbemegy. Ha az összes adat ki lett mentve akkor állítsuk meg a felvételt a *Ctrl-s* billentyűkombinációval.
A *Spectremu* egy igen hasznos funkciója a pillanatkép (*snapshot*) készítés. Ezen funkció segítségével egy fájlba menthetjük a virtuális gép állapotát, majd igény szerint vissza állíthatjuk ebből a fájlból. Pillanatkép készítéséhez csupán a *Ctrl-t* kombinációt kell lenyomni, majd abba a terminálablakba amiből a *Spectemut* elindítottuk begépelni a pillanatképfájl nevét. A *Spectemu* a *.z80* és a *.sna* formátumokat támogatja. Ha a felhasználó nem ad meg kiterjesztést akkor a program feltételezi hogy *.z80* formátumba szándékozik menteni.
A pillanatképek visszaállítása roppant egyszerű., csupán a *Ctrl-l* kombinációt kell lenyomni majd a abba a terminálablakba amiből a *Spectemut* elindítottuk begépelni a pillanatképfájl nevét.
Remélem ez a rövid bevezetés sok olvasó érdeklődését fölkeltette a *Spectrum* és ha hozzá kapcsolódó programok iránt. A *Spectrum billentyűzetének használata* keretes írás egy kis segítséget nyújt a gépeléshez. A *Kapcsolódó címek* keretes írásban olyan oldalak címei találhatók melyek hasznosak lehetnek azok számára akik kicsit jobban meg akarnak ismerkedni ezzel a kitűnő géptípussal. Ezek közül is had emeljem ki a *Personal Computer World* 1982-es júniusi számában megjelent *Sinclair lays a golden egg* című cikket.

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

A Spectemu honlapja:
➲ http://www.inf.bme.hu/~mszeredi/spectemu/spectemu.html

Egy cikk a Spectrumról 1982-ből:
➲ http://www.nvg.ntnu.no/sinclair/computers/zxspectrum/spec_pcw0682.htm

Egy Spectrummal foglalkozó oldal:
➲ http://www.worldofspectrum.org/

Egy másik spectrumos oldal:
➲ http://www.nvg.ntnu.no/sinclair/planet/

Solaris 10 (2. rész)
Rendszereszközök dióhéjban

A tapasztalt linuxos rendszergazda el tud ugyan boldogulni alapvető esetekben, ha egy Solaris konzoljához viszi a sors szeszélye vagy a saját kíváncsisága, de igazándiból a rendszeradminisztrációhoz használatos eszközök kezelésében több a különbség, mint a hasonlóság. Így aztán annyi felfedezni valót tartogat a rendszerek őrének a Sun operációs rendszere, hogy nem is pazarlom tovább a karaktereket, íme a felhozatal!

Vezérlőpult, nem amatőröknek

A ***Sun Management Console*** (röviden ***smc***) használatát fogjuk most áttekinteni. A vezérlőpult szóhoz talán olyan pejoratív mellékzöngék társultak az idők során, hogy a hozzáértést pótolja egy csillogó-villogó felület.

Bár a hasonló szemlélet miatt egy alcím alá vettem ezt a két hasonló nevű ketyerét, egészen más célt szolgálnak. Az ***smc***-vel elemi rendszeradminisztrátori teendőket tudunk gyorsan és kényelmesen elvégezni, a ***Java Web Console*** a ***Solaris 10***-re telepített különféle alkalmazások, szolgáltatások (már amelyik támogatja ezt) finomhangolását végzi el.

Lépjünk be bármilyen felhasználóval a grafikus felületre, majd indítsuk el az `/usr/sbin/smc` parancsot.

Ez a program ***Java*** alapú, ezért a ***Gnome*** rendszer többi összetevőjéhez mérten kissé lomha. Ezt viszont érdemes elnézni neki, mert az a kis veszteség, ami a program sebességében jelentkezik, többszörösen megtérül a funkcionalitás miatt. Egy rendszert egy úgynevezett ***toolbox*** testesít meg. A ***toolboxok*** lehetnek helyben, vagy más szervereken. Így kényelmesen, egy képernyőről navigálhatunk több szerver beállításai között.

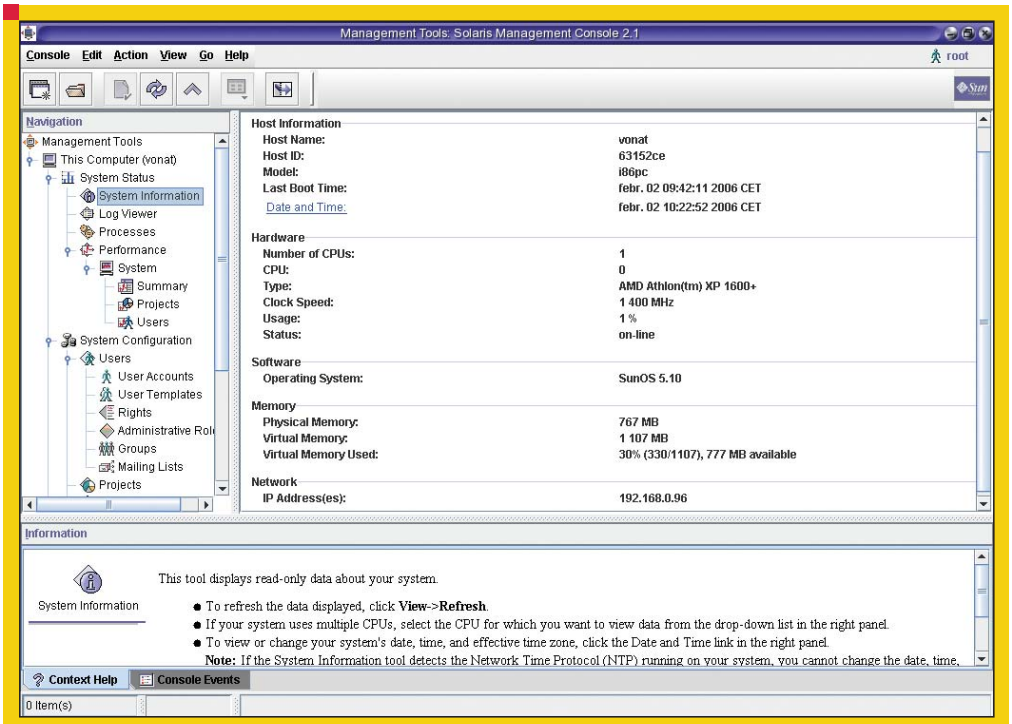
Ahhoz, hogy hozzáférést kaphassunk, be kell jelentkeznünk, hiszen az smc programot bármelyik felhasználó elindíthatja. Mik is azok a beállítások, amiket így elérhetünk? Egyfelől a ***System Status*** alatt az életjeleket figyelhetjük, kicsit kifejezőbben szólva rajta tarthatjuk az ujjunkat a rendszerek ütőerén. A top parancshoz hasonló felületet és a naplókat láthatjuk itt, valamint kapunk egy rövid összefoglalót a rendszer hardverkörnyezetéről a ***System Information*** alatt.

A ***Processes***-t érdemes jól tanulmányozni, a folyamatokat nemcsak listába rendezhetjük itt, hanem egészen mélyre is áshatunk: a környezeti változókat, a fájlleírókat, a használt könyvtárakat (library) és még sok minden mást láthatunk folyamataink magánéletéből.

A ***System Configuration*** választva már mi magunk is aktívan bekapcsolódhatunk: a teljes felhasználókezelést elvégezhetjük, felhasználók, csoportok, jogok (***rights***) és szabályok (***roles***) fölött rendelkezünk itt. A ***Services*** csak az ütemezett feladatokat rejti, pár kattintással összerakhatunk bonyolult `cron` feladatokat. A ***Storage*** az ***NFS***-sel kapcsolatos megosztási munkálatokat egyszerűsíti le, valamint a felcsatolt fájlrendszerek és azok kihasználtsága is megjelenik. A ***RAID*** tömböket is menedzselhetjük egy füst alatt.

A ***Devices and Hardware*** eléggé nagyképű menünév, mivel csak a soros portokat kezelhetjük.

A ***Java Web Console*** ötlete briliáns: az egyes ***Java***-ban írt alkalmazások saját vezérlőpultot készíthetnek, amelyeket aztán egy központi felületről gyorsan elérhetünk, akárhol is vagyunk.



1. ábra Információk a rendszerről

Sajnos még a ***Solaris 10***-be nincsenek beépítve olyan szoftverek, melyek kihasználnák ezt a lehetőséget.

Nyomkövetés vagy új programozási nyelv?

A ***Solaris*** egyik érdekes eszköze a ***DTrace***, mely mind a fejlesztők, mind a rendszergazdák körében nagy sikerre számíthat. Mire vonatkozik tehát az alcím kérdése? Ahhoz, hogy a ***DTrace***-t értelmesen használni tudjuk, alapszinten el kell sajátítanunk a ***D*** programozási nyelvet, melyet a ***Sun*** kimondottan a nyomkövetés céljából hozott létre. Még szerencse, hogy a ***D*** gyakorlatilag a ***C*** nyelv egyszerűsített, módosított változata. A szintaktika például ismerős lesz a ***C*** programozóknak. Szerteágazó eseményeket kérhetünk, hogy figyeljen a `dtrace`, a tesztrendszeren különféle telepítgetések után bő 42 ezer elemet számláló listából válogathatunk. Rendszergazdai jogosultságok kellenek a továbbiak használatához. A `dtrace -l` listázza ki a figyelhető akciókat. Vegyük elő a jól bevált szövegszerkesztőt és hozzuk létre az első ***D*** nyelvű programot:

```
proba.d
proc::exec_common:exec
{
    printf("%d - %s - %d\n",pid,
    ➥execname, uid);
}
syscall::kill:entry
{
    printf("%d - %s - %d\n",pid,
    ➥execname, uid);
}
```

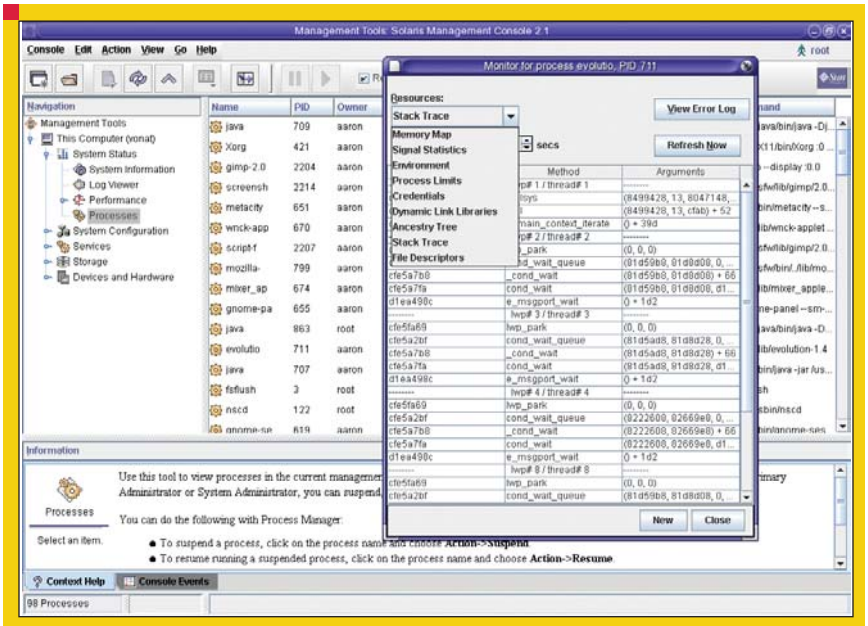
A ***D*** nyelvben is éppúgy létre lehetett volna hozni a klasszikus Helló, világ! példát, de ettől inkább eltekintek, hiszen a ***D*** nem általános célú programozási nyelv. Hogy is lehet megszólaltatni a fenti sorokat? A `dtrace -s proba.d` utasítással, ami nálam egy kis ténykedéssel ezt eredményezte: *1. lista*.

A `dtrace` addig fut, amíg meg nem szakítjuk és gondosan figyel. A ***proba.d*** első blokkja arra ad utasítást, hogy a `proc::exec common:exec` hívásokat kell figyelni és ha egy ilyen érkezett, akkor hajtsa végre a szabványos ***C*** nyelvből jól ismert printf függvényt a megadott formában. A folyamat azonosítója (***process id***), a folyamat neve és a tulajdonos azonosítója (***user id***) kerül ki a képernyőre. A második blokk szinte azonos, csak itt a `syscall::kill:entry`-ket figyeli. Most lássuk ezek után, mi történt, miközben én futtattam a `dtrace`-t. Elindítottam én (100-as ***UID***-em van) a ***Bash***-t, ahol megpróbáltam kilőni a `kill` paranccsal egy folyamatot. Ezután a `su` paranccsal átlényegültem rendszergazdává (0-ás ***UID***), aztán a `df` és az `uptime` utasításokat használtam. Ezután leállítottam a `dtrace`-t, eddig tartott az előadás.

Az, hogy a figyelés lehetőségét és a kapott adatokat mire használjuk fel, már csak a kreativitáson múlik. Egy tipp ***C*** fejlesztőknek: a memóriakezeléshez segítség lehet, ha a lefoglalásokat és felszabadításokat követjük nyomon. Rendszergazdáknak: a ***TCP*** események figyelésével látható, hogy mely folyamatok akarnak hálózati kapcsolatot létesíteni.

1. lista

```
dtrace: script 'proba.d' matched 2 probes
CPU     ID                    FUNCTION:NAME
  0    558           exec_common:exec 4349 - bash - 100
  0     78                kill:entry 3996 - bash - 100
  0    558           exec_common:exec 4350 - bash - 100
  0    558           exec_common:exec 4350 - su - 0
  0    558           exec_common:exec 4351 - bash - 0
  0    558           exec_common:exec 4352 - bash - 0
  0    558           exec_common:exec 4352 - uptime - 0
  0    558           exec_common:exec 4353 - bash - 0
```



2. ábra A folyamatok lelkivilága

Svájci bicska vagy szakbarbár?

Az újonnan telepített rendszerünk az alábbaikban áll a külvilág szolgálatára:

```
Starting nmap 3.81
➥( http://www.insecure.org/
➥nmap/ ) at 2006-02-09
➥19:38 CET
Interesting ports on
➥192.168.0.96:
(The 1642 ports scanned but not
➥shown below are in state:
➥closed)
PORT       STATE SERVICE
21/tcp     open  ftp
22/tcp     open  ssh
23/tcp     open  telnet
25/tcp     open  smtp
79/tcp     open  finger
111/tcp    open  rpcbind
513/tcp    open  login
514/tcp    open  shell
587/tcp    open  submission
898/tcp    open  sun-
➥manageconsole
4045/tcp   open  lockd
6000/tcp   open  X11
7100/tcp   open  font-service
32771/tcp open   sometimes-rpc5
32772/tcp open   sometimes-rpc7
32773/tcp open   sometimes-rpc9
32774/tcp open   sometimes-rpc11
32775/tcp open   sometimes-rpc13
32776/tcp open   sometimes-rpc15
32777/tcp open   sometimes-rpc17
32780/tcp open   sometimes-rpc23
32786/tcp open   sometimes-rpc25
MAC Address: 00:01:02:AA:1B:88
➥(3com)
Device type: general purpose
Running: Sun Solaris 9
OS details: Sun Solaris 9
Uptime 0.345 days (since Thu
➥Feb  9 11:22:38 2006)
```

Az `nmap` ezen verziója elvéti ugyan az operációs rendszer verziószámát, de az kiderül, hogy a rendszer túl sok porton figyel a külvilág felé. A telepítés során számtalan szolgáltatás kerül fel és ezek automatikusan be vannak üzemelve. Ez jó vagy rossz? Attól függ, hogy azt tekintjük kisebb munkának, hogy egy futó szolgáltatást kell kikapcsolni vagy pedig nem beüzemelt szolgáltatásokat bekonfigurálni. Én az előzőre szavazok, pláne, hogy a Solaris oly kényelmes eszközét kínálja a szolgáltatások kezelésére. Az alcím analógiájával élve: a svájci bicskát pillanatok alatt szakbarbárrá tehetjük, hogy kizárólag azt csinálja, amit mi megengedünk neki. Az

```
svcs -a
```

parancs kilistázza a telepített szolgáltatásokat, utána az `svcadm`-mal kezelésbe vehetjük ezeket. Öt perc alatt ki lehet gyomlálni a nem kívánt, külvilág felé portot nyitó fölösleget, ami jóval kevesebb idő, mint a meghagyott szoftverek érintőleges konfigurálása. Az `svcadm` használata sem bonyolult, utasításokat kell használnunk az svcs által kilistázott szolgáltatás-*URL*-ekre. Általános formában így néz ki: `svcadm` parancs szolgáltatás. A parancs lehet `enable`, `disable`, `restart` és `refresh`. Sorrendben: engedélyezi és elindítja, leállítja és letiltja, újraindítja és újra beolvassa a konfigurációját az adott szolgáltatásnak. Gyors munkát teszt lehetővé. Többre vágyunk? A rendszer portjait tűzfal is védheti!

Szögesdróton innen

Mint minden tisztességes és korszerű operációs rendszerhez, ehhez is tartozik tűzfal. A választás a szabadon elérhető ***IP Filter*** megoldásra esett. 2001 májusáig szolgálta például az ***OpenBSD*** felhasználókat ez a tűzfal. A köztudottan a biztonságot szem előtt tartó *BSD*-variáns csupán licensz okok miatt vált meg aztán az IP Filtertől. A Solaris 10-ben nem is olyan egyszerű engedélyezni a csomagok szűrését. Első lépésként a /etc/ipf/pfil.ap fájlban a hálózati kártyánknak megfelelő sorból távolítsuk el a megjegyzés(#) jelét. A hálózati kártyánkról az

```
ifconfig -a
```

parancs ad felvilágosítást. Ezután indítsuk újra a kapcsolódó szolgáltatást:

```
svcadm restart network/pfil
```

Most jön a neheze, állítsuk össze a tűzfal szabályait. A tesztgépen az */etc/ipf/ipf.conf* tartalma ez volt:

```
block out all
block in  all
pass out quick on elxl0 proto
➥tcp from any to any port = 80
➥keep state
pass out quick on elxl0 proto
➥tcp from any to
➥levelezoszerver_ipje port
➥= 995 keep state
pass out quick on elxl0 proto
➥tcp from any to smtp_ipje
➥port = 25 keep state
pass out quick on elxl0 proto
➥udp from any to dns_ipje port
➥= 53 keep state
```

A `to` utáni magyar szavak helyére a szolgáltatónk adatait helyettesítsük be, ha az olvasó is megelégszik azzal, hogy webezni és levelezni szabad, semmi mást nem. A 995-ös portot attól függően, hogy titkosított, illetve titkosítatlan és *POP3* vagy *IMAP* a protokoll, valószínűleg módosítani kell(súgó: */etc/services*). A tűzfal, mint ebből a kis példából kiderül, állapottartó, a kifele menő új kapcsolatok visszafelé jövő ágát engedélyezi, ha utasítjuk erre (`keep state`).
Az *IP Filter* tudásának ez csak egy egészen piciny szelete, érdemes megismerkedni vele.
A ➲ *http://coombs.anu.edu.au/~avalon/* oldalon részletes dokumentációt találunk. De még mindig nem működik a szűrés! A következő lépcső:

```
svcadm enable network/ipfilter
```

Elvileg a *SUN* útmutatása alapján ilyenkor nem kell újraindítani a rendszert, de talán az a legegyszerűbb, hogy végezzünk a munkával. Akik pedig inkább a nehezebb utat választják, ezzel a parancshalmazzal tudják beüzemelni a virtuális szögesdrótot:

```
ifconfig elxl0 unplumb
ifconfig elxl0 plumb ip_cim
➥netmask ide_a_netmask_jon  up
```

Persze az `elxl0`-t mindenki a saját hálózati kártyája szerint módosítsa. Ennyi volt, működik. Az ip címek fordítását (*NAT*) is elvégzi a *Solaris 10*, ha akarjuk, ehhez az *IP Filter* oldalán találunk segítséget.

Rendszerek, amerre a szem ellát

Az, hogy egy szem árva rendszerünk lett a telepítés során, az szép és jó, de mi van, ha éppenséggel tíz rendszert akarok? No miért is kéne egy rendszernél több? Mondjuk azért, hogy külön dobozba tudjuk zárni a sebezhető szoftvereket vagy esetleg az egyes szolgáltatások felügyelete több rendszergazda feladata, így a saját rendszerén mindenki egyedüliként lehet a 0-ás azonosítójú felhasználó. Ezt a képességet itt zónáknak hívják. Megkülönböztetünk globális és nem globális zónákat. A globális zóna tulajdonképpen az a rendszer, ami telepítéskor felkerült. Egyedül a globális zóna tartalmaz indítható rendszermagot (bootable kernel). A nem globális zónák a virtuális rendszereink sokasága. 8192 zóna lehet egy rendszeren elméletileg. Gyakorlatilag pedig annyi, amitől a gép még nem kezd csigalassúsággal vánszorogni. Persze az elvi határt akármilyen erős gép esetén sem léphetjük át. Úgy gondolom, hogy ez a szám nem fog senkit akadályozni a feladatai ellátásában. Hozzunk létre egy új zónát!
Elindítjuk a `proba_zona` nevű zóna beállítását:

```
# zonecfg -z proba_zona
proba_zona: No such zone
➥configured
Use 'create' to begin
➥configuring a new zone.
```

Létrehozzuk:

```
zonecfg:proba_zona> create
```

Beállítjuk a könyvtárát:

```
zonecfg:proba_zona> set
➥zonepath=/zones/proba_zona
```

Ha el akarjuk indítani automatikusan indításakor:

```
zonecfg:proba_zona> set
➥autoboot=true
```

A `tmp` fájlrendszer beállítása:

```
zonecfg:proba_zona> add fs
zonecfg:proba_zona:fs> set
➥dir=/shared/tmp
zonecfg:proba_zona:fs> set
➥special=/tmp
zonecfg:proba_zona:fs> set
➥type=lofs
zonecfg:proba_zona:fs> end
```

A hálózat beállítása:

```
zonecfg:proba_zona> add net
```

Egy külön *IP* címet kap a zóna:

```
zonecfg:proba_zona:net> set
address=192.168.0.5
```

Ide a saját hálózati kártya neve kerül:

```
zonecfg:proba_zona:net> set
➥physical=elxl0
zonecfg:proba_zona:net> end
zonecfg:proba_zona> verify
zonecfg:proba_zona> commit
zonecfg:proba_zona> exit
```

A munka oroszlánrészével már végeztünk is. A `zoneadm -z proba_zona install` elvégzi a fenti utasításokat és telepíti a teljes zónabeli környezetet. Mint egy valódi rendszert, a zónát is el kell indítani a `zoneadm -z proba_zona boot` paranccsal. A bejelentkezés sem marad el, a `zlogin proba_zona` beütésével már be is kerülhetünk az első létrehozott nem globális zónába. Itt aztán szájunk íze szerint konfigurálhatjuk a különféle szolgáltatásokat és az égvilágon mindent, anélkül, hogy ez hatással lenne a globális és a többi általunk létrehozott zónára. Most ugyan külön *IP* címet kapott a létrehozott zóna, de semmi akadálya annak, hogy a legkülönfélébb szolgáltatásokat (*SMTP*, web, *IRC*, stb) szétszedjük mondjuk három zónára három külön ip címmel, aztán a globális zónában az *IP Filtert* úgy állítjuk be, hogy úgy tűnjön, hogy egy rendszer dolgozik összesen. A zónák nagyon hasonlóak a *FreeBSD jail*-jéhez vagy a *chroot*-olt környezetekhez *Linux* alatt.
Az itt leírtak csupán ízelítőül szolgálnak arra, hogy felfedezzük és megismerjük azt, hogy mire képes a Solaris 10 rendszergazdai kezekben. A *Sun* meglepően bőséges és jó minőségű dokumentációt kínál térítés nélkül a ➲ *http://docs.sun.com* weboldalon, valamint létrehozott egy külön weboldalt *BigAdmin*(➲ *http://www.sun.com/bigadmin/home/index.html*) címmel, ahol szintén sok érdekes olvasnivalót találunk.
További élvezetes felfedezést kívánok mindenkinek!

Novák Áron
(aaron@szentimre.hu)

BME-VIK-es hallgató, műkedvelő rendszergazda. Jelenleg leginkább a NetBeans-szel és mindenféle hordozható eszközzel foglalkozik, legalábbis mindazokkal, amelyeket meg lehet szólaltatni Linux alatt.

Linuxos mindenes – Ultimate Boot CD

Nemrégiben egy floppymeghajtó nélküli számítógépre kezdtem telepíteni Linuxot. Az egész merevlemezen NTFS fájlrendszer volt, de átméretező programomat, a floppyról induló BootIt-NG-t nem tudtam használni. Ekkor segített egy zseniális eszköz, amire az egyik UHU-Linuxos DVD indítómenüje is épül: az Ultimate Boot CD, amely többek között tud floppy-emulációt is.

Miért használjuk?

Az ***Ultimate Boot CD*** (➲ ***www.ultimatebootcd.com***, a továbbiakban ***UBCD***) kiválóan használható, ha floppy-alapú diagnosztikai eszközöket vagy telepítőprogramokat szeretne valaki futtatni *CD*-ről. A floppylemez tartalma a memóriába töltődik be, virtuális meghajtóként. Ha van is floppymeghajtó a gépben, szempont lehet a sebesség: sokkal gyorsabban betöltődhet egy nagyobb program *CD*-ről. Az is meggondolandó, hogy ily módon az összes diagnosztikai eszközünket egyetlen, jól kezelhető *CD*-re tudjuk összegyűjteni, nem kell porosodó lemezek közt turkálni az éppen szükséges program előkereséséhez.

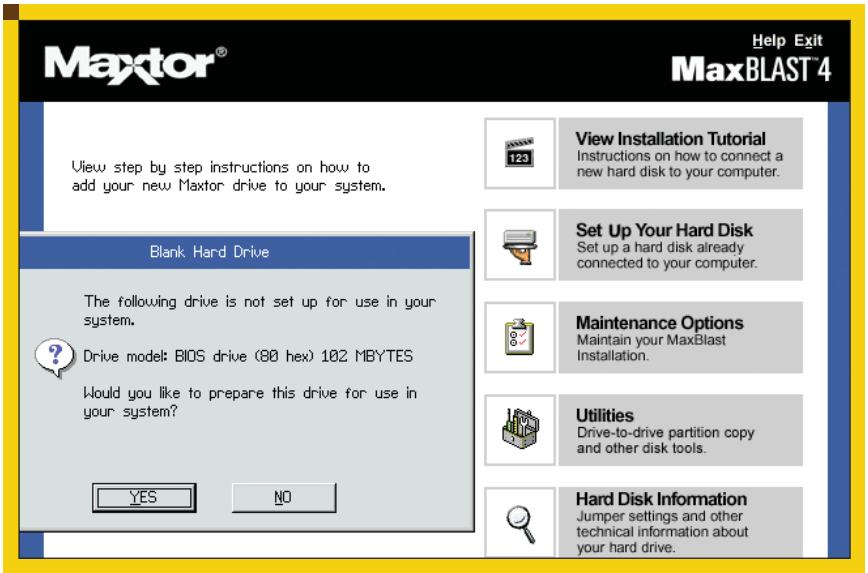
Eddig a hivatalos buzdítás, ami az ***UBCD*** webhelyén is olvasható, ahol több olyan neves „tekintély”, mint például ***PC World*** vagy a ***Softpedia*** hirdeti a projekt elismertségét. Önmagában egy bootfloppyból bootCD készítése egyszerűbb módon is megoldható:

```
mkisofs -b floppiképmás -o
➥kimeneti.iso konyvtar_ahol_
➥van_a_floppiképmás
```

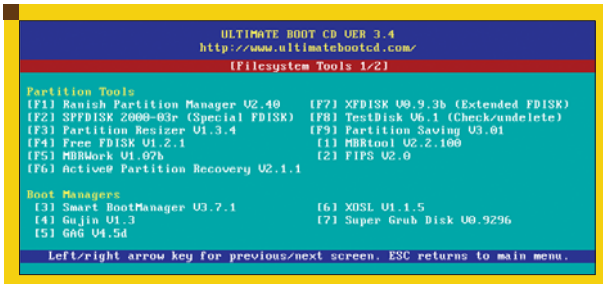
Igazából más okok miatt tetszett meg ez a projekt. Általában azért szoktak az emberek a ***Linuxhoz*** vonzódni, mert valami érthetetlen okból érdekli őket a számítógép, mint olyan. Tehát nemcsak az lebeg előttük, hogy minél fájdalommentesebben szeretnék használni ezt a jószágot, hanem érdekli őket, hogy mi is zajlik a háttérben, s hogy mik a „legmenőbb” programok, gondok és irányvonalak, amik az informatika világát tűzbe hozzák. (Például melyik a legnagyobb tömörítési arányú linuxos tömörítő?) Esetleg szeretnénk valamit le is tenni a *„Linux* asztalára”, hiszen ez „ajándékozó társadalom”, amiben annál nagyobb valaki, minél többet ad (nem pedig: minél többet elvesz). Ha mást nem, legalább annyit mindenki meg tud tenni – főleg egy ilyen remek programkészlettel a zsebében – , hogy egyegy fellelkesített barátja gépére ***Linuxot*** telepít, vagy helyrehozza a hibákat, amik ilyen veszedelmes „alapműveletek” (particionálás, átméretezés stb.) közben adódhattak.

Néhány kivételesen szerencsés embertársunkon kívül, akiknek ez egyben a munkája is, ritkán adatik meg, hogy ezeket a vágyakat célirányosan meg tudjuk élni. Ez a gyűjtemény azonban, amit kezünkben tartunk, remek ugródeszka lehet a tájékozódáshoz, különösen is az *„INSERT”* révén, amely része a teljes verziónak (lásd később). Ilyen irányból nézve talán nem lesznek olyan szárazak az alábbi felsorolások, hiszen ha valamelyik név semmit sem mond, akkor ott érdemes megállni, és kicsit utánanézni. Megéri a fáradságot.

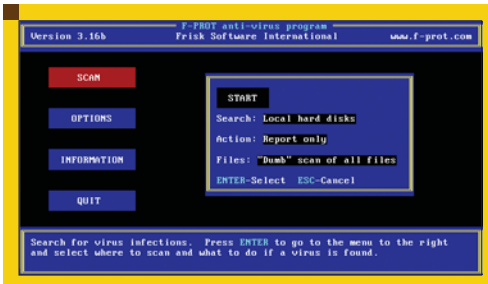
Mondhatná valaki, hogy „hiszen erre a célra bármely ***Linux*** disztribúció megfelelő”. Ez részben igaz, de itt nem kell annyira megküzdeni a bőség zavarával,



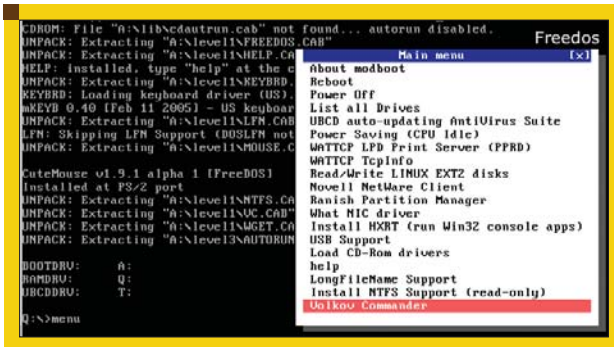
1. ábra Egy merevlemez-varázsló



■ *2. ábra* A fájlrendszerkezelő eszközök első menüje

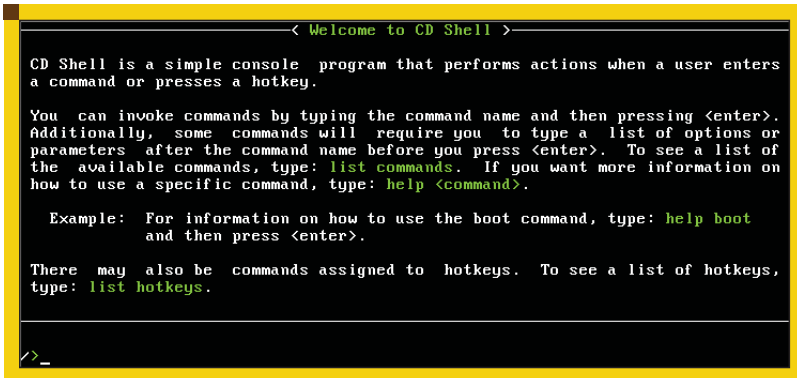


■ *3. ábra* F-PROT víruskereső futás közben



■ *4. ábra* Betöltött FreeDOS menüje – lehetséges IP(X) hálózatot is felállítani

■ *5. ábra* Trinux



■ *6. ábra* Az UBCD parancssora

másrészt pedig lehet tudni, hogy az ilyen mentőlemezeket az élvonalbeli szakemberek szokták használni, tehát feltehetőleg az egyik legprofibb készlettel van dolgunk. (Nem véletlen, hogy a ➲ *www.professionalsecuritytesters.org* weboldalon is említik: – rákeresve az *insert* szóra, láthatjuk a cikket.)

Letöltés, változtatások

Az *UBCD* összeállítója *Victor Chew*. Különböző tükörszerverekről le lehet tölteni a 150 MB körüli „teljes verziót”, ami még kiegészítésekkel együtt is ráfér egy 185 MB-os vagy 210 MB-os, hitelkártya méretű *CD* lemezre. Az alapverzió csak feleekkora, de az nem tartalmazza a *Knoppix*-alapú *INSERT* Linux disztribúciót, amit pedig kár lenne kihagyni. (*Charles Appel* és *Matthias Mikuletz,* ➲ *www.insert.cd* (mókás egy *URL*); *„Inside Security Rescue Toolkit”*, azaz *„Belső Biztonsági és Mentőlemez Eszköztár”*, másrészt viszont a szójáték a *„betesz, beszúr”*-re is utal, mármint a *CD*-t a meghajtóba, s innentől nyerésre állunk...).

2006. február 14-én érkezett meg a 3.4-es verziójú *UBCD*. Célszerű ránézni a *„Bugs and workarounds” (Hibák és gyorsfoltozások)* menüpontra, mert előfordul, hogy egy-egy javítás (mint patch) letölthető. Magam is találtam egyet a 3.4-es verzióhoz: egy *.zip* fájlt, ami kicsomagolva fölülírja a javítandó fájlokat. Maga az eredetileg letöltendő *UBCD* anyag egy *.iso* fájl (hiszen végül is *CD*-nek szánták), de ezt loopback eszközön keresztül becsatolva – vagy egyes disztribúciókban egyszerűen entert ütve a fájl nevére *Midnight Commander*ben – már átmásolható a tartalma egy könyvtárba. Ezt célszerű egy `chmod u+w -R` paranccsal írhatóvá tenni, így már megvalósítható a testreszabás és mindenféle módosítás (mint például a javítófájl kibontása).

A *CD* előállításához ezt a parancsot használtam (amit magának az eredeti *.iso* fájlnak az elejéből is ki lehet lesni):

```
mkisofs -N -J -joliet-long -D
➥-V UBCDFULL -o sajat.iso -b
➥boot/loader.bin -no-emul-boot
➥-boot-load-size 4
➥gyokerkonyvtar
```

Lehetséges, hogy vannak olyan alkönyvtárak a gyökérkönyvtárban, amelyeket nem akarunk kiírni: ezeket `-m` kapcsolóval ellátva kihagyja az `mkisofs`.

Az eredeti vagy a módosított *.iso* fájlt ki lehet írni újraírható *CD*-re például a

```
cdrecord -blank=fast sajat.iso
```

paranccsal, vagy *k3b*-vel az *Eszközök/CD/Írás CD-képmásból*

menüponttal. Igen javasolt az újraírható *CD* használata, hiszen nagy valószínűséggel nem az először kipróbált változat lesz a végső. Meggyorsítja a kísérletezést, ha *CD*-re írás előtt *qemu*-val kipróbáljuk. A mellékelt képeket *qemu*-ból való futtatás során készítettem, hiszen más úton elég körülményes lenne egy éppen induló rendszer képeit elmenteni. (➲ *hup.hu/wiki/index.php/QEMU*, illetve a *Linuxvilág* 56. számának 39-42. oldala.)

Az *UBCD* menürendszere igen egyszerű és áttekinthető.

A *boot/menus* könyvtárban van minden *.scn* fájl (a „screen”-re utalva – egy fájl egy képernyő). A *main.scn* a főmenü, ennek tartalma ilyesféle:

```
...
print " [F6] DOS/Linux Boot
➥Disks "
if file[\INSERT\INSERT]; else
➥set textColor = color
➥[brightgrey on cyan]
print " [F7] INSERT for UBCD  "
...
if ($lastKey == key[f6]); then
➥script bootdsk.scn
if file[\INSERT\INSERT]
then if ($lastKey == key[f7]);
➥then script insert.scn
...
```

Semmi extra trükk nincs benne, kezdők számára is érthetőek az alternatívák. (Aminek csak a lehetősége van meg, de nincs jelen az aktuális *CD*-n, az halványabb szürkével látható.)

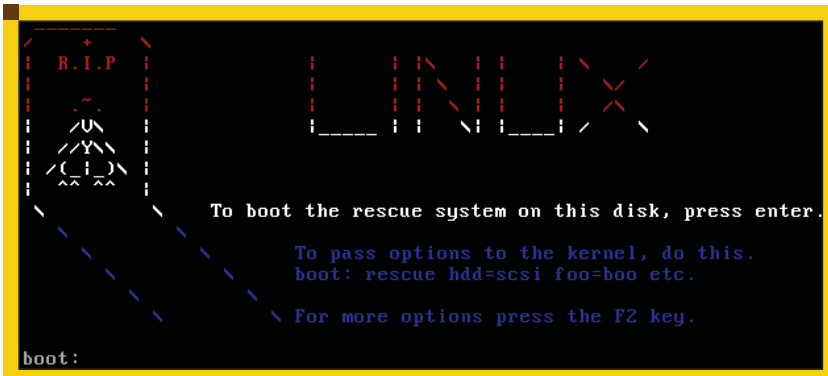
Amiatt is érdemes lehet végigfutni a menüfájlt, mert például az *„Ins”* gombbal előidézhető *„expert”* mód csak innen látható, nincs kiírva a képernyőre. Csak az „írástudók” jönnek rá.

A *custom.scn* fájl szolgál arra, hogy a saját alkalmazásainkat, hajlékonylemezeinket stb. ellássuk menürendszerrel, a könyvtárrendszerben pedig a */custom* a javasolt helye a fájloknak, hogy a frissítéskor ne kelljen sokat keresgetni őket. De természetesen máshova is betehetők saját fájljaink.

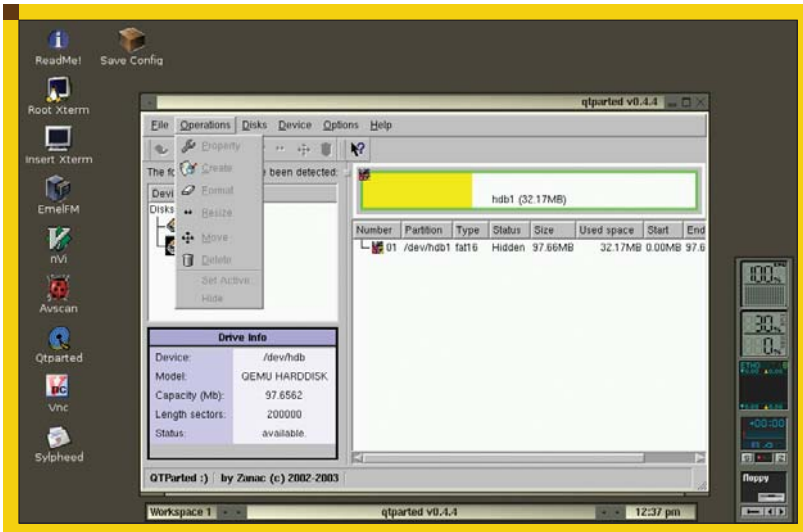
A CD-n található alkalmazások

A teljes alkalmazás-lista megtekinthető az *UBCD* honlapján (vagy magyarul a ➲ *www.osb.hu/z/ubcd_lista.html* címen.)

A főmenüben a következők közül választhatunk:



7. ábra R.I.P. Linux

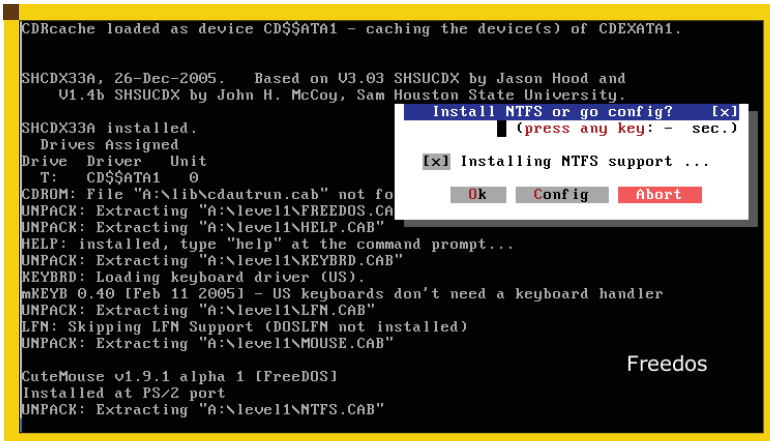


8. ábra Qtparted az INSERT-ben

- Alaplapi eszközök (ilyen például a magyar fejlesztésű Aida ➲ www.aida32.hu)
- Merevlemezkezelők
- Fájlrendszerkezelő eszközök
- Egyéb eszközök, azaz vírusirtók és hálózati eszközök
- Felhasználó által definiált eszközök
- *DOS/Linux* betöltőlemezek
- *INSERT for UBCD*
- Első/második merevlemezről történő bootolás
- Konzol (nem *Linux*, sem nem *DOS*, hanem az *UBCD* saját, egyszerű parancsnyelve néhány utasítással).

A menürendszer eléggé szerteágazó; nem mindig könnyű kitalálni, hogy mely programot melyik „bugyorba” szánta a szerkesztő. Ilyenkor a fent említett *gyökérkönyvtár*ban levő *boot/menus* alkönyvtárban érdemes rákeresni a kívánt szóra. Például a *QwikTest* megkeresésére a `grep -il qwik *` parancs megadja, hogy ő bizony az *mboard2.scn*-ből hívható meg. Ekkor vagy egyből tudjuk, hogy ez az alaplapi menüpont 2. része, vagy rákeresünk az ugyanitt levő *main.scn*-ben, hogy vajon mit is takar ez a menüpont (vagy ha nincs meg, akkor a „szám nélküli” részére kereshetünk: `grep mboard main.scn` – s látjuk, hogy az *F1*-gyel jön elő az alaplapi menü, de innen még lapozni kell a következő oldalra, hogy láthatóvá váljon a *QwikTest*-indító gyorsbillentyű).

Számomra először nem volt nyilvánvaló, mit takar a *„System Burn-In Test”* fogalma. A stressztesztet vagy beégetési teszt arra való, hogy a számítógépet a lehetőségekhez képest sokáig maximális terhelés alatt tartsa, hogy a normális terhelés többszörösének produkálásával elősegítse az igen ritkán vagy kizárólag az egységek teljes terhelésének kihasználása során (például túlmelegedés miatt) jelentkező hibák, működési rendellenességek felderítését.



9. ábra FreeDOS betöltés közben

1. táblázat *Mazsolák...*

Program(ok) neve	Változat	Leírás
fbdesk	1.1.5	Ikonkezelő program *Fluxbox* ablakkezelőhöz
nessus	2.0.10a-6	Hálózatbiztonság
nmap, nmapfe	3.50-1	Portpásztázó
gpg	1.2.6	GNU Privacy Guard
Smartmontools	5.32	Merevlemezek meghibásodása előrejelezhető [*SMART=Self-(smartctl, smartd) Monitoring, Analysis and Reporting Technology System*].
7za (7-Zip)	.91	A legnagyobb tömörítési arányú linuxos tömörítő
rdesktop	1.3.1-1	Kliensprogram távoli *Linux* vagy *Windows* elérésére
xvncviewer	3.3.7-1	*„Virtual Network Computing"* kliensprogram
sylpheed	.9.12-1	Levelezőkliens
hping2	2.rc3-3	Hálózati szkenner, TCP/IP csomag előállító/analizátor
netcat	1.10-23	Hálózati kapcsolatokat író/olvasó háttérprogram
netsed	.01c-2	Hálózati csomagok valós idejű megváltoztatására képes
ngrep	1.40.1-3	net-grep, hálózaton áthaladó adatokban tud keresni szabályos kifejezésekre
cpio	2.5-1.1	Fájlokat másol archívumokba és archívumokból
samba és smbd	3.0.5	*SMB/CIFS* (*Common Internet FileSystem*) fájlszerver
gRun	.92	GTK alapú futtatóablak
Qtparted	0.4.4	Partíciószerkesztő, tud *NTFS* átméretezést is
Monkey web server	0.8.4 F2	Webkiszolgáló
Knoppix wireless drivers	3.4	Drótnélküli eszközök meghajtói

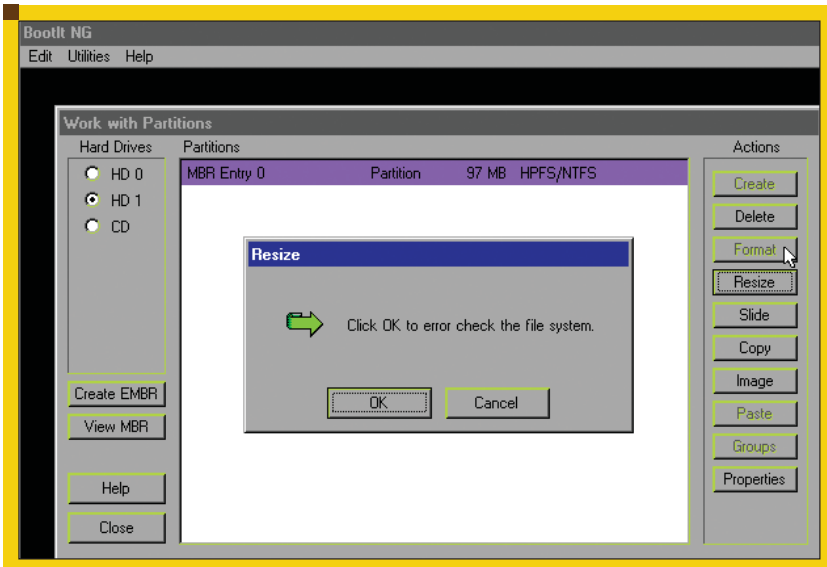
Örömmel és némi meglepetéssel vettem észre, hogy az ***UBCD*** egyik processzor-tesztjeként a ***Mersenne Prím-teszt***et hívja segítségül (***F1***, ***F3***).
A ***Linux*** betöltőlemezek közül nekem különösen tetszett a ***R.I.P***, ami a *7. ábrán* is látható sírhant-feliratra utal, hogy *„**Nyugodjék békében**"* (latinul) – ennek a szójátéknak a biztatóbb feloldása a *„**Recovery is Possible**"* (*„**Lehetséges a helyreállítás**"*). A ***help*** paranccsal részletes, kezdők számára is megérthető eligazítást kapunk a helyreállítás miként jéről, lehetőségeiről.

Mazsolák a mentőlemez száraz kalácsában

Érdemes egy pillantást vetni, milyen programokat gyűjtöttek egybe az ***INSERT*** szerkesztői. Ezekből ad ízelítőt az *1. táblázat* és a *8. ábra*. Részletesebb lista – és az is, hogy melyik program mire való – itt olvasható: ➲ ***www.inside-security.de/applicationlist.html***. Minimális a rendszer mérete ahhoz képest, hogy teljes ***Linux*** disztribúció áll rendelkezésünkre, 2.4.26-os kernellel.
A webböngészést és a részletes dokumentáció elolvasását a grafikus ***links*** teszi lehetővé, egyszerűen és szépen. Lehetőségünk van például az ***NTFS*** (és szinte minden más, manapság használatos) fájlrendszert írhatóan/olvashatóan csatolni a ***Captive*** csomag révén, s így hatékonyan futtatható a víruskereső. (a ***Captive*** használatáról a ***jobb klikk – doc – Captive*** alatt olvashatunk); ***ssh***-t és ***sshfs***-t tudunk használni, azaz biztonságosan csatolhatóak távoli fájlrendszerek, ***samba*** kliensként és szerverként tudunk üzemelni, van ***RAID*** és ***LVM***-kezelés, valamint különböző incidenskezelő, behatolásérzékelő (***IDS, Intrusion Detection Signature***), helyi (***chkrootkit***) és hálózati ellenőrzőeszközöket használhatunk. E témában hasznos olvasmány lehet: ➲ ***www.cert.hu/ismert/00tanulmany/Incid_FW.pdf.***

Kérdések, teendők, akadályok

Ha belenézünk a ***boot/menus*** alatti fájlokba, kiderül, hogy a *CD*-n található képfájlok `set method = "memdisk"`, vagy `set method = "diskemu"`, esetleg `set method = "bcdw"` „módszerrel" indíthatóak. Mi a különbség? Ez a kérdés saját képfájlok beüzemelése esetén merül fel. Amikor egy képfájlt megpróbálunk elindítani (ezt természetesen valamelyik ***.scn***



■ *10. ábra* BootIt-NG, a floppiról induló kereskedelmi partíciószerkesztő

fájlban kell tudatnunk a rendszerrel), akkor először a ***memdisk***-et érdemes próbálni, mert az a leggyorsabb (az egész képfájl a memóriába kerül). Ha ez nem sikerül, próbálható a ***diskemu*** (ilyenkor a ***CD***-ről érjük el a képfájlt); a ***bcdw*** pedig a 2.88 MB-nál nagyobb fájlokhoz ajánlatos. Nyilván úgy célszerű, hogy egyszerre mindhárom sort beírjuk a menübe, és ha valamelyik nem sikerül, hívjuk a másikkal.
Mi a teendő a tömörített ***.igz*** képfájlokkal? Hogyan lehet hozzájuk nyúlni, ha nem tetszenek egy betöltőlemez alapértelmezett beállításai? Ilyen és hasonló kérdésekre is választ kapunk az ***UBCD*** weboldalán levő ***FAQ***-ból. Konkrétan az erre adott válasz: az ***.igz*** fájlokat ***GZIP***-pel tömörítették (ez sejthető a névből is, ám egy szimpla ***gunzip***-et vagy ***gzip -d***-t ráeresztve nem jutunk előbbre). Át kell nevezni (vagy másolni) a fájlt:

```
mv memtestp.igz memtestp.img.gz
gzip -d memtestp.img.gz
mkdir konyvtara
su -c 'mount memtestp.img
➥konyvara -o loop'
```

Különös módon bele lehet szerkeszteni az így felcsatolt fájlba – nem úgy, mint egy ***iso9660*** típusúba! Lecsatolás után is megmaradnak a változtatások például egy ***FreeDOS*** betöltőlemezen.

```
su -c 'umount konyvtara'
gzip -9 memtestp.img
mv memtestp.img.gz memtestp.igz
```

Intelligens módon úgy van megoldva néhány betöltőlemez konfigurációja, hogy néhány másodpercet vár (látjuk, ahogy visszaszámlál, míg hozzá nem nyúlunk – *9. ábra*), de aztán továbbmegy az alapértelmezett értékekkel. Szükségem volt arra, hogy egy ponton megváltoztassam az alapértelmezett értéket, hogy ne kelljen ezt mindig kézzel megtenni. Kerestem ugyan ***grep***-pel (az imént leírt módon) kicsomagolt és felcsatolt ***.img*** lemezen egy szövegrészt, de nem találtam. Ekkor belenéztem az indítófájlba. Kiderült, hogy ***.cab*** formátumban van eltárolva néhány nagyobb egység – ebbe pedig nyilván nem látott bele a ***grep***. Ezt ***cabextract***-tal tudtam kibontani. Szerencsémre a feladatot meg tudtam úgy oldani, hogy elegendő volt néhány felesleges fájlt kitörölni (a becsatolt (!) képmásfájlból), mert egyébként gondban lettem volna ***Linux*** alatt, hogy hogyan csomagoljak vissza bármit is ***.cab*** archívummá. (Persze lehet találni erre szabad szoftvereket például ***FreeDOS***-ra, de azért ez mégis hiányérzetet kelt az emberben.) Az ***UBCD***-n levő ***OpenDOS***-t nem sikerült elindítanom, de ez lehet gépfüggő is. Mindenesetre a ***FreeDOS*** szépen futott, bár a ***CuteMouse*** elakadása miatt néha kénytelen voltan az elején ***„No UMB”*** defenzív memóriakezelést kérni a megfelelő menüpontban.
Bár maga az ötlet remek, hogy a ***Microsoft Windows*** saját fájljait használjuk föl az ***NTFS*** fájlrendszer írására, nekem még sosem sikerült tökéletes munkára bírnom a ***captive-ntfs***-t. Amikor fel szeretném csatolni a fájlrendszert, a ➲ ***www.jankratochvil.net*** oldalra utal a program, hogy van újrabb verzió, és nem csinál semmit. Több számítógépen és több mentőlemezzel (például a ***SysRescCd***-vel) is így jártam. Futó ***INSERT***-ben nem lehet frissíteni, mert nem írható a ***/usr*** könyvtár, az ***UBCD***-n viszont annyira kompakt egységet képez, hogy nem teljesen triviális feladat megváltoztatni (például frissítés révén). Ha valakinek mégis kedve támad ehhez, segítséget talál a ➲ ***www.linuxdevcenter.com/pub/a/linux/2003/11/20/knoppix.html*** oldalon. Az ***NTFS***-fájlrendszer átméretezésére inkább a ***BootIt-NG***-t (➲ ***www.bootitng.com***, ➲ ***members.shaw.ca/bootitng***) tudom ajánlani (ami sajnos nem szabad szoftver; ***10. ábra***), vagy egy ***SuSE Linux*** telepítőlemez használatát, ami induláskor lehetővé teszi ezt a műveletet.
Egy másik feladat volt a betöltött ***INSERT*** billentyűkiosztásának magyarrá tétele. A ***loadkeys hu*** csak a virtuális terminálokon hat, az ***X*** terminál átállításához magát a ***/etc/X11/XF86Config-4***-et kellett átírni (***XkbLayout***: ***„us”*** helyett ***„hu”***), majd ***Alt+Ctrl+Backspace*** gombkombináció után újraindítani az ***X-Window***-t. Pedig van konfigurációbeállító menüpont (jobb klikk), de a billentyűkiosztás megváltoztatását nem találtam.
Ha valakinek kifejezettem ***Microsoft Windows***-alapú program ***CD***-ről történő elindítására van szüksége, akkor a ➲ ***www.ubcd4win.com*** weboldalról letöltheti az ***UBCD*** ilyen irányú leszármazottját is, de ennek tárgyalása már nem a ***Linuxvilág*** hasábjaira tartozik. Az érdeklődők további képernyőképeket találhatnak a ➲ ***mrbass.org/ubcd*** és a ➲ ***www.osb.hu/z/ubcd*** címen.

Szabó Zoltán
Három gyermekével és feleségével Pannonhalmán él. Tíz éve kísérletezik a Linuxszal. Matematikát és informatikát tanít, diákotthonban keseríti a rábízottak életét. Szívügye a PHP és a PostgreSQL.
(szz@freemail.hu)

Pörgésben – Fájlrendszerek sebességének mérése

Amikor egy számítógép sebességéről beszélünk, akkor általában mindenki a processzor frekvenciájára gondol, ugyanakkor a tényleges teljesítményt nem mindig ez határozza meg. Hiszen a processzor idejének egy részét azzal is tölti, hogy a háttértár ki-beviteli műveletére vár (wait). Ilyenkor a sebesség már sokkal inkább a merevlemezek, merevlemez vezérlők gyorsaságán múlik.

A merevlemez egységet nem célom különösebben bemutatni, hiszen akit érdekel annak működése az az Interneten rendkívül sok anyagot talál róla, ilyen célra például tudom ajánlani a ***Wikipediát***. A merevlemez sebességét lényegében három tényező befolyásolja, az egyik a csatoló felület/szabvány, a másik a lemez körbefordulási sebessége, a harmadik pedig a beépített gyorsító tár mérete.
A hagyományos ***EIDE/ATA-2*** merevlemezek 1996-ban még csak 16 Mbyte/s adatátvitelre voltak maximum képesek, az ***ATA*** szabvány utolsó leszármazottja 2003-ban jelent meg, ***ATAPI-7*** néven (***Ultra DMA-133***), mely 133 Mbyte/s átviteli sebességre volt képes, és a legtöbb számítógépben még ilyen merevlemezek vannak a mai napig.
Amennyiben új számítógépet vásároltunk a közelmúltban, vagy vásárolunk, akkor mára főleg ***SATA*** csatolóval ellátott lemezeket találunk benne. Vállalati környezetben a kiszolgálókra ugyanakkor kivétel nélkül csak ***SCSI*** lemezeket szoktak használni. A ***SATA*** szabvány elméleti legnagyobb átviteli sebessége 150 Mbyte/s (***8B/10B*** kódolást használva a fizikai rétegen), míg a ***SATA II*** szabvány is már 300 Mbyte/s átviteli sebességre képes. 2007-re várhatóan a ***SATA*** szabvány képes lesz a 600 Mbyte/s adatvitelre. Ugyanakkor a ***SCSI*** merevlemezek már jóval régebben támogatják az ***Ultra320***-as szabványt, mely 320 Mbyte/s adatátvitelre képes, persze ezek az adatok az elvileg elérhető maximum értéket jelentik. Az használat közben tapasztalt sebesség akár ennek a töredéke is lehet.
A legegyszerűbb eszköz a merevlemezeink sebességének mérésére a ***hdparm*** (de ez nem alkalmazható SCSI merevlemezekre), ugyanakkor hamar rá is jövünk, arra, hogy ezzel a programmal inkább csak szép számokat, sem mint használható értékeket kapunk. Vegyük például azt az esetet, amikor a a merevlemez gyorsítótárának sebességét mérjük a programmal, szinte biztos, hogy az elvileg elérhető értéknél (ami a kábelen adott idő alatt képes átmenni) nagyobb értéket kapunk.
Mivel minket a tények érdekelnek nem pedig a szép nagy számok, ezért hívjunk segítségül egy professzionális programot az adott feladatra.

IoZone

Az ***IoZone*** egy fájlrendszer sebességmérő alkalmazás, tehát nem merevlemezünk tényleges adatátviteli sebességét tudjuk vele megmérni, hanem az adott fájlrendszer teljesítményét. Ugyanakkor mivel fájlrendszer nélkül nem sok mindenre tudjuk használni a merevlemezt, ezért felesleges is volna fájlrendszer nélkül mérni a sebességet. Az ***IoZone*** sokféle fájlművelettel képes megmérni az adott fájlrendszer, illetve számítógép, fájlműveleti teljesítményét, a teszteket nem csak helyi gépeken végezhetjük el, hanem lehetőségünk van megmérni távoli fájlrendszerek (például ***NFS***) sebességét is.
A sebesség mérését célszerű nem csak akkor elvégezni, amikor telepítjük és összehangoljuk a rendszert, hanem később is, például amikor a kiszolgáló funkciója bővül, megváltozik. Ha a kiszolgálót eddig adatbázis kiszolgálásra használtuk, és ezentúl mondjuk mentési feladatokat fog ellátni, akkor célszerű lehet a fájlrendszert is megváltoztatni úgy, hogy a szekvenciális olvasást a fájlrendszer jobban támogassa mint a véletlenszerűt.

Első tesztek

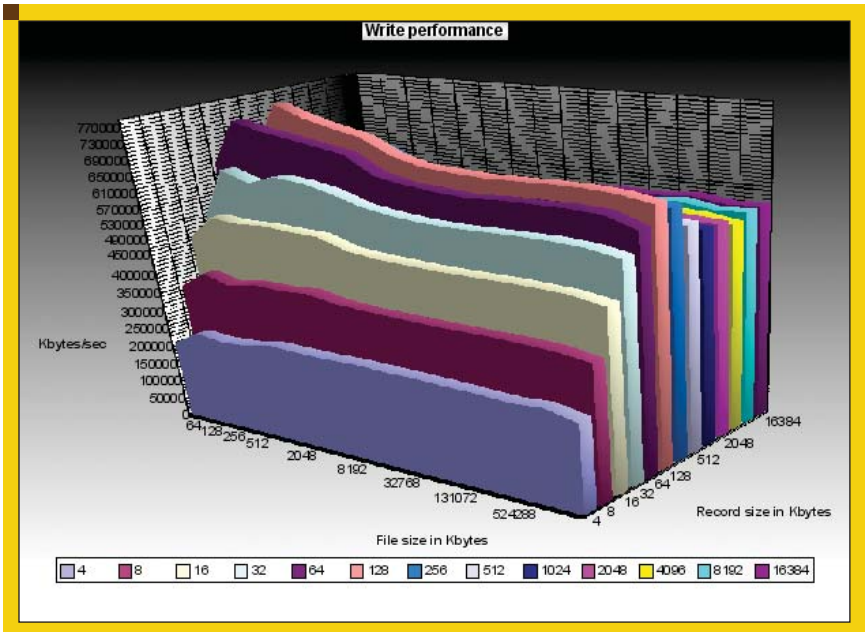
Az ***IoZone*** jól paraméterezhető, így magunk is definiálhatunk teszteket, de ha nem akarunk ezzel törődni akkor használjuk az ***IoZone***-t automatikus módban:

```
iozone -a
```

Amennyiben szeretnénk grafikonok formájában is megtekinteni a kimenetet, akkor használjuk az

```
iozone -Ra
```

parancsot, az elkészült kimenetből pedig ***OpenOffice*** segítségével készíthetünk grafikonokat. Egy ilyen grafikont láthatunk például az *1. ábrán*, melyen a ***ReiserFS*** fájlrendszer szekvenciális írási sebessége látható egy ***IBM x236***-os duál processzoros 3 Gb fizikai memóriával rendelkező kiszolgáló esetén.
A különböző színű grafikonok a különböző méretű kiírt rekordok méretét jelentik. Az x tengelyen pedig a kiírt fájl



1. ábra A ReiserFS fájlrendszer teljesítménye szekvenciális írás esetén

tényleges mérete látható, a 64 kByte méretűtől egészen 1 GByte méretű fájlokig. A grafikonról leolvasható például, hogy a jelen konfiguráció esetén a 128 kByte méretű rekord nyújtja a legjobb teljesítményt. A z tengely felosztása lineáris, 0 és 760000 kByte közötti értékek szerepelnek rajta. Lehetőségünk van egyből bináris állományba kimenteni a teszt eredményét, ehhez csak az

```
iozone -Rab eredmeny.wks
```

parancsot kell kiadni. Ahogyan már említettem, az ***IoZone*** remekül paraméterezhető, így például korlátot szabhatunk annak, hogy mekkora legyen a legnagyobb fájl, amin a teszteket elvégezzük, ha a legnagyobb fájlt 1Gb méretűnek engedjük meg, akkor a következő paraméterekkel indítsuk a teljesítménypróbát:

```
iozone -Ra -g 1G
```

Ha nem szeretnénk az összes tesztet lefuttatni, hanem csak az írás-olvasás eredményére vagyunk kíváncsiak, akkor a következőképpen futtassuk le az alkalmazást:

```
iozone  -Ra -g 1G -i0 -i1
```

Az *NFS* kliens sebességét a következő utasítások segítségével tudjuk megvizsgálni:

```
iozone -Rac
```

Itt a `c` paraméter azért kell, hogy kiküszöböljük vele az ***NFS 3*** esetén előforduló kliens oldali gyorsítótár hatását.

Mit tesztelünk

Az ***IoZone*** lehetőséget biztosít nekünk, hogy minél szélesebb skálán tudjuk megmérni a háttértárunk teljesítményét, viszont fontos tisztában lennünk azzal, hogy melyik teszt pontosan mit csinál, így fussunk át gyorsan az egyes teszteken.

- ***Írás (Write)*** – Ezzel egy új fájl létrehozásának az idejét tudjuk lemérni. Amikor egy új fájl létrejön, akkor nem csak a fájl tartalmát kell a merevlemezre írnunk, hanem azt is, hogy a fájl hol foglal helyet a lemezen, milyen jogosultságok tartoznak hozzá, stb. Ezeket az adatokat nevezzük metaadatoknak, melyek nem az eltárolni kívánt adat tényleges részei. Az első írás pontosan ezért valamivel lassabb, mint a fájl újra írása.
- ***Újraírás (Re-write)*** – Ez a teszt egy már létező fájlhoz ír hozzá adatot, mivel ilyenkor a metaadatok általában nem módosulnak, így az újra írás természetesen gyorsabb lesz.
- ***Olvasás (Read)*** – Egy már létező fájlból olvasunk ki adatot.
- ***Újraolvasás (Re-read)*** – Az újraolvasás sebessége sokkal nagyobb lesz az olvasás sebességénél, hiszen ilyenkor olyan adatokat olvasunk ki, melyeket a közelmúltban olvastunk. Az operációs rendszer (és esetleg a merevlemez belső) gyorsítótára miatt így nem a tényleges merevlemez sebességet mérjük meg.
- ***Véletlenszerű olvasás (Random Read)*** – Ennél a tesztnél is egy fájlból olvasunk ki adatokat, de a fájlt nem az elejétől a végéig olvassuk, hanem a fájl véletlenszerűen kiválasztott rekordjait olvassuk ki. Ezt az olvasási módot jellemzően az adatbázis kiszolgálók teljesítmény mérésénél célszerű figyelembe venni. Ennek a módnak a sebességét nagyon sok tényező befolyásolja, ilyen például a rendszer gyorsítótár mérete, a merevlemez fejmozgások ideje, a rendszer memória, stb.
- ***Véletlenszerű írás (Random Write)*** – A véletlenszerű olvasáshoz hasonló, csak most nem rekordokat olvasunk az adott fájlból, hanem írunk bele.
- ***Véletlenszerű írás-olvasás (Random Mixed)*** – Egy azon fájlt több folyamat ír és olvas egyszerre. Ez a teszt áll a leginkább közel egy tényleges adatbázis kiszolgáló háttértár használatához. Ez a teszt csak throughtput módban futtatható, és egy adott folyamat vagy csak írja, vagy csak olvassa a kérdéses fájlt. Az író-olvasó folyamatok közötti ütemezés a round robin algoritmust használja.
- ***Hátrafelé olvasás (Backward Reading)*** – Ez az olvasási mód elég extrémnek tűnhet, de vannak olyan a gyakorlatban is előforduló alkalmazások, melyek ilyen módon olvassák a fájlokat, mivel a fájlrendszerek általában az előre történő olvasásra vannak felkészítve, ezért így olvasva a fájlt sokkal rosszabb eredményeket kaphatunk.
- ***Rekord újraírás (Record rewrite)*** – Ennél a tesztnél a fájl egy adott darabját írjuk, majd ismételten írjuk. Itt többszintű értékeket is meg tudunk figyelni, attól függően, hogy az adott fájlrészlet elfér

a *CPU* gyorsítótárában, vagy csak a fizikai memóriában fér el. Ha a fájlrészlet fizikai memóriacíme megtalálható a ***Translation Lookaside Bufferben (TLB)*** akkor a gyorsabb virtuális fizikai címleképezés miatt szintén egy újabb értéket kapunk. Amennyiben a fizikai memóriacím nincsen benne a *TLB*-ben, de a fájl darabja megtalálható az operációs rendszer gyorsítótárában, akkor ez ad egy újabb értéket, és legutoljára kapjuk azt az értéket, amikor a fájl csak a merevlemezen található meg.

- ***Lépkedve olvasás (Strided Read)*** – Az olvasás egy bizonyos minta alapján történik a fájlból. Ilyenkor kiolvassuk a fájl egy darabját majd bizonyos adatokat átugorva egy ugyanakkora darabot olvasunk ki belőle, és ez így folytatódik a fájl végéig.
- ***Fwrite*** – A fájlt az ***fwrite()*** függvényen keresztül írjuk, mely bufferelt és blokkolt írási műveletet valósít meg. A buffer miatt a kisméretű rekordok írási sebessége jelentősen megnőhet. Mivel ennél a módnál is új fájlt hozunk létre, ezért itt is számolnunk kell a metaadatok hatásával.
- ***Frewrite*** – Az ***Fwrite*** művelethez hasonló, de egy már létező fájlt ír felül az ***fwrite()*** függvény segítségével.
- ***Fread*** – Olvasás a fájlból, az ***fread()*** függvény segítségével, tulajdonságai hasonlóak az fwrite() függvényhez.
- ***Freread*** – Az ***Fread*** olvasáshoz hasonló, de a nemrégen olvasott adatokat olvassuk ki újra.

Néhány naplózó fájlrendszer összehasonlítása

Ha már az ember összerak egy tesztkörnyezetet, akkor logikusnak tűnik, hogy amit csak lehet megmér vele. Én is így voltam ezzel, így egy ***IBM e-server x236 series*** és ***SCSI Ultra320***-as háttértárak segítségével a négy leggyakrabban használt fájlrendszer tesztjét mellékelem a cikkhez. Konkrétan az ***EXT3***, ***JFS***, ***ReiserFS*** és ***XFS*** naplózó fájlrendszereket vizsgáltam, a kiszolgálón a ***Novell Open Enterprise Server 9sp1*** kapott helyet. Mindegyik fájlrendszert alapbeállítással használtam, egy külön erre a célra fenntartott partíción. A teszteket azonos méretű partíción végeztem el, a partícióra előzőleg felmásoltam magát a ***IoZone*** programot, egyébként a partíció teljesen üres volt. A táblázatot beszúrni ide teljesen felesleges lenne, így inkább a cikkhez csatolom.

Azért néhány fontos dolgot megpróbálok kiemelni belőle. Mivel az első teszt alkalmával csak olyan fájlokon végeztem a műveletet, melyek elfértek az operációs rendszer gyorsítótárában, ezért a fájlrendszerek közötti különbségek itt nem a tényleges lemezműveleti sebességeket mutatják, ugyanakkor az egyes fájlrendszerek sebességviszonya nagyon jól látszik. Szembetűnik az is hogy az adott konfiguráció esetén a legnagyobb adatátvitelt a 128 Kb-os rekord esetén kapjuk (*1. ábra*).

A 4 vagy 8 Kb-os rekord esetén az átviteli sebesség az elérhető maximum harmada, illetve fele, ez valószínűleg azért van így, mert egy kisebb rekord be-kiviteli pluszköltsége fajlagosan nagyobb, mint egy nagyobb rekordé. Véletlenszerű olvasás esetén az ***XFS*** fájlrendszer alul marad a másik három fájlrendszerrel szemben, melyek sebessége körülbelül 13%-kal gyorsabb.

Véletlenszerű írás esetén az ***EXT3*** és a ***ReiserFS*** teljesítménye gyakorlatilag azonos, de a ***ReiserFS*** egy csöppet gyorsabbnak tűnik, míg az ***XFS*** szintén gyorsabb a másik kettőnél, de a ***JFS*** és az ***EXT3*** közötti különbség eléri a 36%-ot.

A nagy meglepetés a szekvenciális írás sebesség tesztjénél ért engem, amikor is az összes fájlrendszer sokkal rosszabbul teljesített, mint a véletlenszerű írás esetében.

128 kByte méretű rekordok esetében az ***EXT3*** fájlrendszer átlagos átviteli teljesítménye ~360 Mbyte/s volt, ugyanakkor az ***XFS*** fájlrendszer sebessége 828 Mbyte/s volt, ami gyakorlatilag a duplája az ***EXT3*** teljesítményének. Második helyezett lett a ***ReiserFS*** 676 Mbyte/s-cel és kizárásos alapon nem sokkal lemaradva tőle a ***JFS*** végzett 639 Mbyte/s-es teljesítményével.

A szekvenciális olvasás terén a négy fájlrendszer sebessége között gyakorlatilag nem volt semmilyen különbség, egységesen 2460 Mbyte/s-es teljesítményt nyújtottak. Természetesen ez a magas érték csak azért jöhetett létre, mert az operációs rendszer a memóriából olvasta vissza a fájlt.

További összehasonlítás

Az előző összehasonlításoknál a teljes fájl amin az ***IoZone*** végezte a műveletet elfért a rendszer memóriájában, ezért adódtak irreálisan magas értékek. Ugyanakkor azt is megpróbáltam megmérni, hogy mi a különbség az egyes fájlrendszerek teljesítménye között, amikor az aktuális fájl már nem fér el a rendelkezésre álló fizikai memóriában.

Ezeknél a teszteknél természetesen arra számítottam, hogy az elméleti 320 Mbyte/s-es értékeknél sokkal rosszabb eredmények fognak születni. Ugyanakkor ezek az értékek azok melyek valóban megmutatják fájlrendszerek közötti különbségeket.

A tesztalany gépen a rendszermemória 3GB volt, tehát egy 4 GB-os fájlon végzett műveletek már biztosan helyes eredményt adnak.

Az *1. Táblázat* alapján láthatjuk, hogy a gyakorlati adatátviteli sebesség közel a harmada annak amit a merevlemez, és a kábel elméletileg lehetővé

1. táblázat *Fájlrendszerek összehasonlítása*

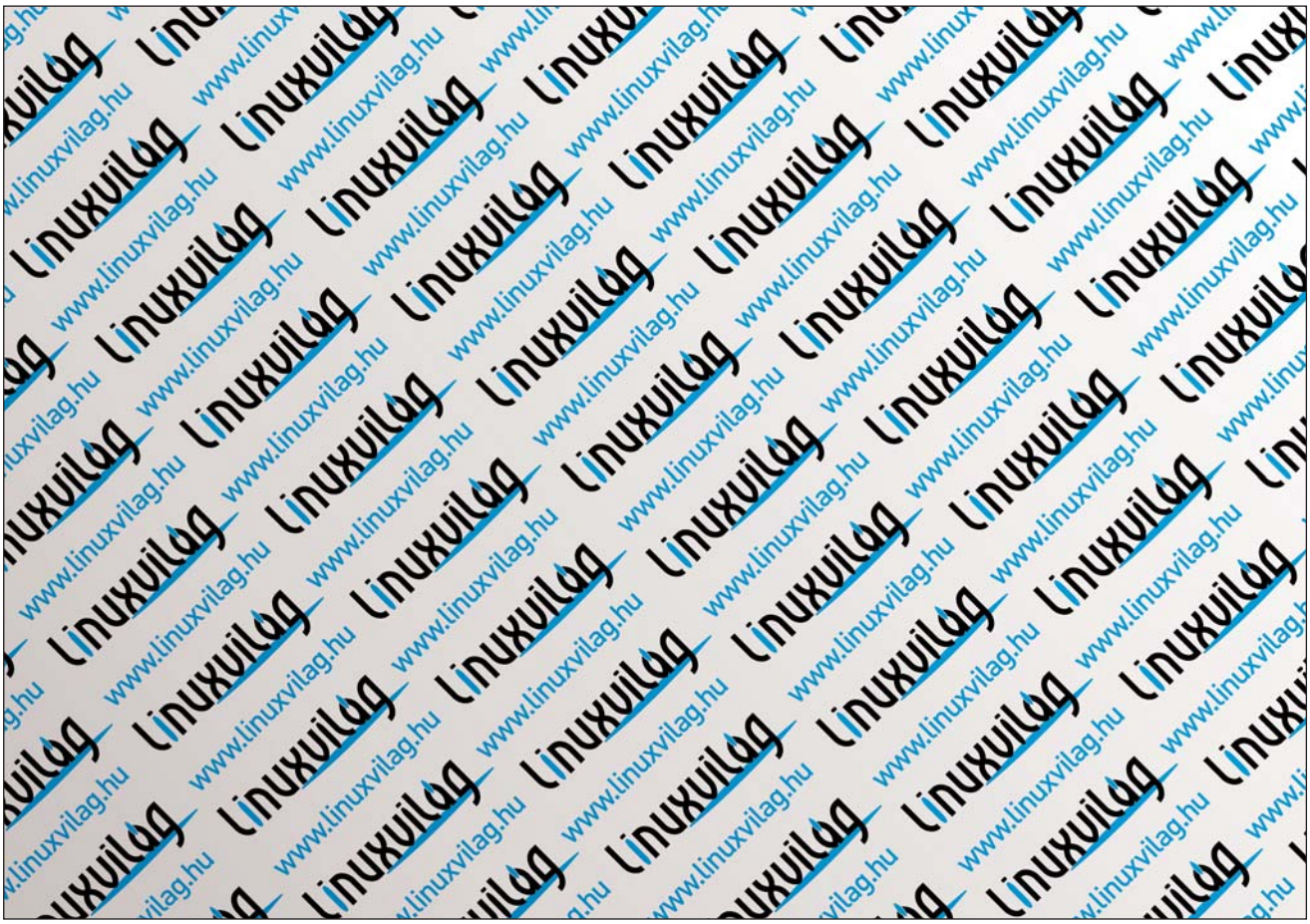
128 Kb rekord és 4GB fájlméret	ReiserFS MB/s	EXT3 MB/s	JFS MB/s	XFS MB/s
Szekvenciális írás	113	105,3	112,8	116,8
Újraírás	113,7	106,3	118,5	114
Olvasás	78,9	83,3	84,8	83,6
Újraolvasás	71,5	83,4	84,8	83,6
Véletlenszerű olvasás	65,8	66	68	64,9
Véletlenszerű írás	132,7	55,8	127,2	103,4

tesz. Furcsa anomáliát tapasztalunk ugyanakkor, ha megnézzük, azt, hogy a véletlenszerű olvasás sokkal lassabb, mint a véletlenszerű írás. Valószínűleg ez azért van, mert az írást az operációs rendszer nem írja ki rögtön a merevlemezre, hanem csak a memóriában tárolja ideiglenesen. Ugyanakkor az olvasásnál nem tud a gyorsítótárból olvasni, mert a fájl nincsen benne.

Konklúzió

Ahhoz, hogy megtaláljuk a számunkra leginkább megfelelő fájlrendszert mindenképpen ajánlom, hogy magunk végezzük el a tesztek a saját gépünkön, és vegyük figyelembe azt is mindig, hogy milyen alkalmazásokat akarunk használni. Ahogy telik az idő a fájlrendszerünk teljesítménye esetleg rosszabb lehet, mint kezdetben, mert időközben fellép a fájlok töredezettsége, mely teljesítményromláshoz vezet. Ez a jelenség ugyanakkor manapság nem olyan szembetűnő, mint régebben a nem linuxos *FAT* fájlrendszer esetén volt.
Akik idáig eljutottak a cikk olvasásában, azok joggal tehetik fel a kérdést, hogy miért mértem meg a fájlrendszerek teljesítményét akkor is, amikor a rendszer tulajdonképpen a memória gyorsítótárából használja az adott fájlt. A válaszom pedig erre igen egyszerűen az, hogy jó, ha ismerjük a fájlrendszerünk ezen tulajdonságát is, hiszen az operációs rendszerek mindig megpróbálnak lehetőleg arra törekedni, hogy az általunk leginkább használt fájlokat a memóriában tartsák. Ezért nagy esélyünk van arra, hogy rendszerünk főleg így üzemeljen, ekkor pedig nem hagyhatjuk figyelmen kívül ezeket az értékeket sem.
A fenti statisztikák alapján én azt javasolom, hogy ha lehet akkor a *ReiserFS* vagy a *JFS* fájlrendszert részesítsük előnyben, amikor naplózó fájlrendszert szeretnénk használni linuxos munkakörnyezetben.
Az *EXT3* fájlrendszert pedig semmiképpen se használjuk például adatbázis kiszolgálón, ahol véletlenszerű íráskor nyújtott teljesítménye igen gyenge.
Az egyes fájlrendszerek sebessége még tovább javítható, például ha úgy csatlakoztatjuk fel a fájlrendszert, hogy beállítjuk a *noatime* opciót, akkor a rendszer nem fogja naplózni, hogy mikor történt az utolsó hozzáférés az adott fájlhoz. Hiszen ha rögzítjük az utolsó hozzáférés idejét, akkor minden egyes olvasási műveletet szükségképpen egy írási művelet is követ. Ezt az opciót főleg flash memóriák esetén érdemes mindig bekapcsolni. De például a *ReiserFS* fájlrendszer esetében megadhatjuk azt is, hogy a naplót ne az adott partíción tárolja a rendszer, hanem egy másik lemezen.
Mindenkinek sikeres finomhangolást, és lemezfelpörgetés kívánok!

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



Mikor a Linux a Windows megmentője...

„A mentést a /dev/null-ra irányítom – így sokkal gazdaságosabb minden szempontból: nem kell szalagokat cserélgetni ötpercenként, ráadásul még gyorsítja is a mentési folyamatot, tehát csak jó lehet.” – A pokoli operátor naplója, első nap.

A *Pokoli Operátor* naplója egyike az informatikus humor legnépszerűbb gyöngyszemeinek. Egy végtelen történet, amelyben a főhős a felhasználók kínzásának újabbnál újabb módszerei kifejlesztésével próbál meg úrrá lenni a lehangoló semmittevésen. A kollégáival folytatott töretlen harcát figyelve apránként megismerhetjük személyiségét is, és kiderül, hogy nem egy átlagos szakiról van szó, hanem egy igazi zseniről.
Ne felejtsük el azonban, hogy a történet a 90-es évek elején született.
A háttértárak árának mélyrepülése már egy ideje elért arra a pontra, ahol adott esetben a szalagos egységnél kifizetődőbb megoldás a hálózati mentés. Jelen cikk ennek egy megvalósítását ismerteti. Természetesen ez nem azt jelenti, hogy a mai kor pokoli operátorainak lelke végleg megnyugodhat, mindenesetre a */dev/null* használatára úgy tűnik, egy ideig nem lesz szükség.
Tegyünk félre egy pillanat erejéig minden bennünk dübörgő vallási érzést, és képzeljünk el egy heterogén hálózatot, amelyben egy *Windows NT-* és egy *Linux*-alapú belső hálózati kiszolgáló békésen megfér egymás mellett. Mi több, feladatunk a két számítógép házasítása úgy, hogy bizonyos érzékeny adatokat tartalmazó könyvtárakról az éj leple alatt mentés készüljön. Az érzékeny adatokat az *NT*-n egy *Sybase* adatbázisszerver által felügyelt fájlokat jelentik, a *Linux* pedig egy *FTP* kiszolgálóval áll rendelkezésünkre. A frigyhez *Perlt* fogunk alkalmazni az *NT*-n, az alábbi hat felvonásban:

```
stop_service;
sleep 60;
create_archive;
start_service;
ftp_upload;
unlink $zip_name;
```

A szolgáltatás leállítása

Noha szkriptünk éjszaka fog futni, amikor feltehetőleg minden felhasználó alszik, és így nem jelentenek veszélyt, az adatbázisszervert jobb leállítani a mentés megkezdése előtt. Ehhez a `Win32::Service` modult használjuk. Íme:

```
sub stop_service {
    print "Stopping service.\n";
    my %status;
Win32::Service::GetStatus ('',
➥$service_key, \%status);
    {
        last if ($status
➥{CurrentState} == 1);
        die "Cannot stop service
➥'".$service_key."'.\n" unless

Win32::Service::StopService ('',
➥$service_key);
    }
    print "Service stopped.\n";
}
```

A kód a `$service_key` globális változó meglétét feltételezi, amely a *Windows* szolgáltatás egyértelmű azonosítója. Ha ezt tudjuk, egyszerűen vegyük fel a szkript legelejére. Ha nem, akkor szintén a fenti modulra támaszkodva kinyerhetjük a szolgáltatás leírásából, amire később láthatunk is példát.

Az eljárás ellenőrzi, hogy fut-e a szolgáltatás, és ha igen, leállítja azt. Először a `Win32::Service` modul `GetStatus` metódusát használjuk. Ez három paramétert vár. Az első karakterlánc annak a számítógépnek a *NetBIOS* neve, amelyen a kérdéses szolgáltatás fut. Ha ez üres, akkor vizsgálódásunk tárgya a helyi számítógép. A második a szolgáltatás azonosítója, a harmadik asszociatív tömb pedig a státuszinformációt tároló változó referenciája, vagyis az eredmény.
A `GetStatus` hívás után egy blokkba lépünk, amit az első sorral azonnal el is hagyunk, ha a szolgáltatás nem fut (ezt jelzi az 1-es státusz). Ha a blokkot nem hagytuk el, akkor a `StopService` metódussal leállítjuk a szolgáltatást.

Egy kis pihenés

A szolgáltatás leállítása a legtöbb esetben minden mellékhatás nélkül megtörténik, viszont a metódus visszatérése után is időbe telhet egyes takarító folyamatok lefutása. A `sleep 60` segítségével 60 másodperc, azaz egy perc erejéig elaltatjuk a szkriptet. Ezalatt az adatbázis-kezelő biztosan leáll.
Természetesen egy jól irányzott ciklussal felkészülhetnénk arra az esetre is, ha ez nem történne meg, most viszont az egyszerűségre, és az ebből adódó kisebb hibalehetőségre hivatkozva rábízzuk magunkat az *NT* szolgáltatás-kezelőjére.
Vegyük észre, hogy a `StopService` meghiúsulása a szkript azonnali leállítását eredményezi, amiről, napi

mentésről lévén szó, a rendszergazda legkésőbb másnap értesül. Mindig az adott feladat határozza meg, hogy mennyire szigorúan követeljük meg a mentés sikerét. Ne feledjük, hogy a biztos mentés záloga egy időtúllépés esetén kikényszerített leállás lenne, ez viszont lehet, hogy több kárt okozna, mint hasznot.

Zip, zip, hurrá

Miután feltehetjük, hogy a szolgáltatás nem fut, az adatbázis-kezelő fájljait egyszerűen, az `Archive::Zip` modul segítségével tömörítjük.

```
sub create_archive {
    print "Creating archive.\n";
    my $zip_archive =
➥Archive::Zip -> new;
    foreach my $dir
➥(@directories) {
     die "'".$dir."' is not a
➥directory.\n" unless (-d $dir);
      print "Adding directory
➥tree '".$dir."' to
➥archive.\n";
   my $status = $zip_archive ->
➥addTree ($dir, basename($dir));
  die "Cannot add directory tree
➥'".$dir."' to archive.\n"
      unless ($status == AZ_OK);
    }
    print "Adding timestamp to
➥archive.\n";
    $zip_archive -> addString
➥(scalar localtime,
➥"timestamp.txt");
    print "Writing to file.\n";
    my $status = $zip_archive ->
➥writeToFileNamed ($zip_name);
    die "Cannot write to file '"
➥.$zip_name."'.\n"
    unless ($status == AZ_OK);
    print "Created archive.\n";
}
```

Ez a szubrutin elsőként létrehoz egy `Archive::Zip` objektumot, ami egy üres archívumot jelent. Ezután egyenként végiglépked a `@directories` globális tömb elemein, és minden elemről eldönti, hogy könyvtár-e. Ha nem, a szkript meghiúsul. Normális esetben rekurzív módon az elem teljes részfáját hozzáadja az archívumhoz. Természetesen ennél a lépésnél is történik hibaellenőrzés. Ebben a ciklusban a hangsúly az `addTree` metóduson van. Ennek első paramétere a könyvtár elérési útja, a második az a név, amelyen tárolni szeretnénk. Ennél segítségül hívjuk a `File::Basename` modul használatával elérhető `basename` függvényt, ami az azonos nevű ***UNIX*** paranccsal azonos viselkedésű.
A ciklus végeztével minden könyvtárat hozzáadtunk az archívumhoz. A teljesség kedvéért még egy sokszor meglepően hasznosnak bizonyuló metódussal időbélyeget helyezünk el az archívumban. Az `addString` metódus az átadott karakterláncot a hivatkozott fájlban helyezi el. Ehhez egyetlen függvényhívásra van szükségünk, és még csak feleslleges átmeneti állományokat sem igényel a művelet. Végezetül a `writeToFileNamed` metódussal lemezre írjuk a mentést. Az itt használt `$zip_name` globális változó hatékony meghatározására később visszatérünk. Fontos, hogy az eddig összegyűjtött könyvtárak tartalma valójában csak most kerül feldolgozásra, tehát nem kell tartanunk attól, hogy néhány óriási fájl felemésztené a memóriát.

A szolgáltatás elindítása

Mivel a tömörített mentést már csak fel kell töltenünk, ami várhatóan időigényes feladat, a szolgáltatást nyugodt szívvel elindíthatjuk.

```
sub start_service {
    print "Starting service.\n";
    my %status; Win32::Service:
➥:GetStatus ('', $service_key,
➥\%status);
    {
        last if ($status
➥{CurrentState} == 4);
       die "Cannot stop service
➥'".$service_key."'.\n" unless
            Win32:
➥:Service::StartService ('',
➥$service_key);
    }
    print "Service started.\n";
}
```

Ez az eljárás nem rejteget sok újdonságot a legelsőhöz képest. Az egyetlen nem magától értetődő pont az, ahol a szolgáltatás futásának ellenőrzése történik. A 4-es státusz jelenti azt, hogy a szolgáltatás már fut, ekkor nem indítjuk el.

Ez például akkor történhet meg, ha az archívum készítése alatt valaki elindította az adatbázis-kezelőt. Nagy valószínűséggel a szkript ez esetben már kilépett volna fájlzárolási problémák miatt, de azért nem árt az óvatosság.

FTP feltöltés

A szkript talán egyik legszebb része következik, az ***FTP***-n történő feltöltés. Ehhez a `Net::FTP` modult használjuk.

```
sub ftp_upload {
    print "Uploading to ftp
➥site.\n";
    die "Cannot connect to '"
➥.$ftp_site."': ".$@."\n"
➥unless
        my $ftp_session =
➥Net::FTP -> new ($ftp_site,
➥Debug => 0);
    die "Cannot login:
➥".($ftp_session -> message)
➥unless
        $ftp_session -> login
➥($ftp_user, $ftp_pass);
    die "Cannot set binary
➥mode: ".($ftp_session ->
➥message) unless
        $ftp_session -> binary;
    die "Cannot cwd to '"
➥.$ftp_dir."': ".($ftp_session
➥-> message) unless
        $ftp_session -> cwd
➥($ftp_dir);
    die "Cannot put file
➥'".$zip_name."':
➥".($ftp_session -> message)
➥unless
        $ftp_session -> put
➥($zip_name);
    $ftp_session -> quit;
    print "Uploading done.\n";
}
```

Ha valaki használt már parancssoros ***FTP***-klienst, a fenti kód sorról sorra történő magyarázata inkább fárasztó lenne, mint hasznos. Egyetlen lényeges, elsőre mégsem feltétlenül szembeötlő lépés a kapcsolat binárisra állítása. Mivel egy ***zip*** állománynak nem tesz jót a sorvége jel átalakítása, ez a sor elengedhetetlen.
A szubrutin az `$ftp_site`, `$ftp_user`, `$ftp_pass`, `$ftp_dir` globális változók meglétét feltételezi.

Takarítás

Ez az a része a mentési folyamatnak, ami általában annyira sok nyűggel jár, hogy sokan inkább a */dev/null*-t választják. Szerencsére ez jelen esetben egyetlen sor:

```
unlink $zip_name;
```

A nagy kép

Mindez úgy áll össze egy szkriptté, hogy a mindenkinek kötelező `use strict` után felsoroljuk a használt modulokat, kitöltjük a globális változókat, beszúrjuk a fenti rutinokat, végül pedig leírjuk az elején említett forgatókönyvet.
Lustaságunk mihamarabbi elhatalmasodása érdekében lássuk azt a két globális változót, melyet nem egyszerű értékadással határozunk meg:

```
my $service_key;
{
    my %services; Win32:
➥:Service::GetServices ('',
➥\%services);
    foreach my $key (sort keys
➥%services) {
        if ($key =~
➥$service_name) {
            $service_key =
➥$services{$key}; last;
        }
    }
}
my $zip_name = $zip_dir."/
➥backup-".(strftime "%Y%m%d",
➥localtime).".zip";
```

Szó volt róla, hogy a szolgáltatás leállítása és elindítása csak a szolgáltatás egyértelmű azonosítója ismeretében lehetséges. Ezt megállapíthatjuk a szolgáltatáslista kézi böngészésével, vagy ha ennél rugalmasabb megoldásra vágyunk, ki is találhatjuk. A szolgáltatás leírására történő mintaillesztéssel Csak nagyjából kell, hogy emlékezzünk a szolgáltatás nevére, ami nagyságrendekkel növeli a félálomban, mégis hasznos munkával töltött órák számát. Erre példa a `$service_key` beállítása.
A `$zip_name` egy *backup-20060228.zip* alakhoz hasonló nevet határoz meg. Ahhoz, hogy a dátum valóban a jelenlegi dátum legyen, igénybe vesszük a *POSIX* modul `strftime` függvényét. Ez a parancssori `date`-hez hasonló formátumot vár el, amivel könnyen elérhetjük a kívánt fájlnevet.

Kész?

A helyes válasz az, hogy attól függ. Ha valóban elviselhető, hogy szélsőséges körülmények között egy esti mentés kimarad a nem egy helyen elképzelhető kilépés miatt, akkor mindenképpen. Megjegyzem, erre az egy hónapos teszt üzem alatt nem volt példa. Ami ennél fontosabb, az a *Linux* kiszolgálón keletkezett mentések kezelése. Feltétlenül meg kell oldani, hogy ezek ne egyék fel a merevlemezt. Erre egy pár soros, `crontab`-ból futtatott szkript megfelel. A *.zip* fájlok elnevezése kiválóan alkalmas annak eldöntésére, hogy a mentés törölhető-e vagy sem, ezért ezzel már nem untatnék senkit.

Fülöp Balázs
(fulop.balazs@initon.hu)

Az Initon Kft. informatikai munkatársa.

Az élőszó ereje – TeamSpeak

A TeamSpeak (TS) egy Linux-Windows alatt is elérhető, kliens-szerver felépítésű, internetes hangkommunikációs szoftver. A kliens és szerverfunkciók jól elkülönítettek, a szerver dedikált szerverként működve szolgálja ki a klienseket, nem pedig P2P elven működik.

Képes több ezer felhasználót kiszolgálni egyidejűleg, a beszélgetőpartnerek szobákba, azon belül alszobákba rendeződhetnek. Tizenkétféle, különböző eljárásokkal és mintavételi frekvenciákkal dolgozó kodeket használhatunk, így a minőség/sávszél arány változtatható. Nem kereskedelmi felhasználásra a megoldás ingyenes, a bináris szabadon letölthető a szoftver honlapjáról. (➲ *http://www.goteamspeak.com*). Alkalmas konferenciabeszélgetések lefolytatására, a szereplők számának igazából csak a sávszélesség szab határt. Tipikus felhasználási területe az on-line játékok közbeni kommunikáció. Üzleti felhasználása is lehetséges, például távoli telephelyek közötti kommunikációra, sokszereplős konferenciabeszélgetésekre, telefonköltségek csökkentésére. Üzleti felhasználás esetén licenszelni kell a szoftvert, de szerverek és szobák bérelhetőek is. Előnye, hogy olcsón, alacsony sávszélen, egyszerű infrastruktúrával magas hangminőségű konferenciabeszélgetéseket lehet lefolytatni vele. Emellett egyszerű kezelőfelülete, webes szerver adminisztrációs felülete, jól átgondolt jogosultsági rendszere van. Hardverigénye a szervernek és a kliensnek is minimális (Pentium 1 166 64 MB RAM-mal, a szervernek még kevesebb, mivel nincs szüksége grafikus felületre), jól tűri a tűzfalat, a sávszélessége korlátozható. A program üzeneteit (például hogy új felhasználó csatlakozott) egy kellemes női hang mondja be angolul, de saját hangüzeneteket is állíthatunk be. Egyszerűen beállíthatók a gyorsbillentyűk is, és a szerver adminisztrációs funkciók elláthatók a kliensprogram segítségével is.

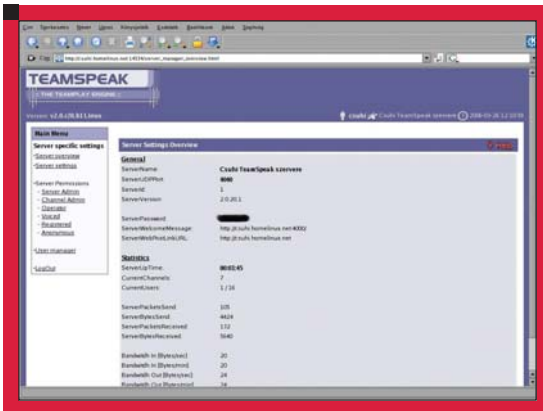
A *Skype*-pal, *MSN*-nel összehasonlítva a leglényegibb különbség az, hogy a *TS* elsősorban hangkommunikációra van kihegyezve, szöveges üzeneteket csak korlátozottan, fájlt egyáltalán nem lehet cserélni vele, és videokonferenciára sem alkalmas. A *TS* nem azért van, hogy arra figyeljünk, hanem azért, hogy tudjunk beszélgetni miközben valami mást (például on-line játék) is csinálunk.

Letöltés, telepítés, beállítás

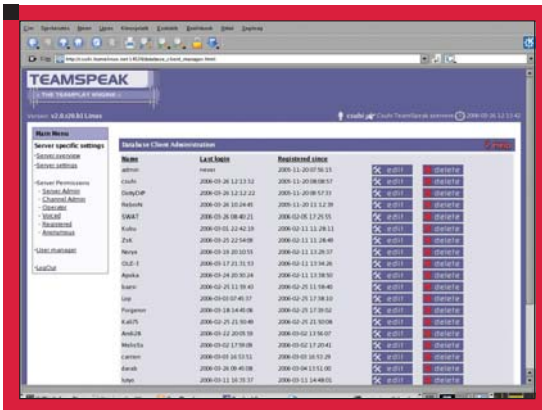
A program honlapjáról letölthetőek a *Linux* és *Windows* binárisok a szerverből és a kliensből egyaránt, illetve *Mac OS X*-re egy nem hivatalos kliens is. Az *EULA* elfogadása után már tölthetjük is, a *Linux* verziót *tar.bz2* formátumban. A szerver kitömörítés után a *tss2_rc2* könyvtárban foglal helyet. Lépjünk be a könyvtárba, majd a

```
./teamspeak2-server_startscript
➥start
```

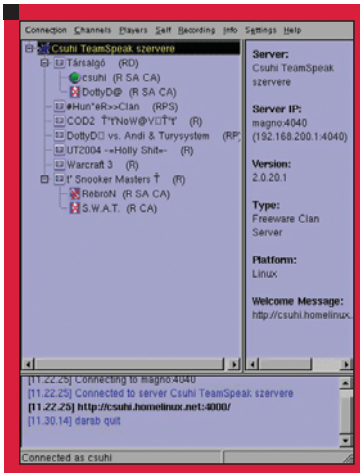
paranccsal indíthatjuk, `stop`-pal pedig leállíthatjuk a szervert. A szerver elindítása után a webes adminisztrációs felület alapértelmezésként a `http://<szerver>:14534/` útvonalon érhető el. A rendszer alapvető



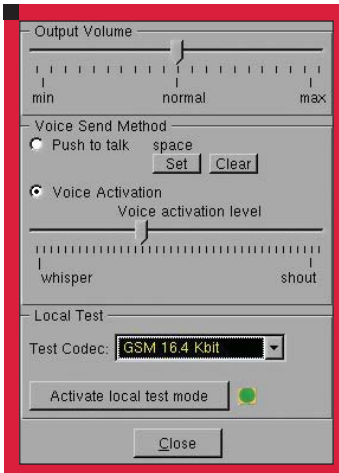
1. ábra WebAdmin felület, Server Settings



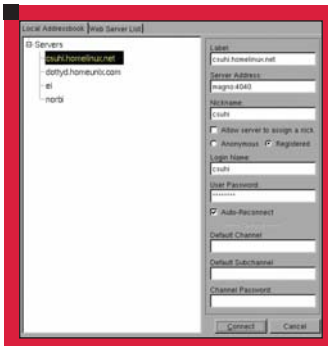
2. ábra WebAdmin felület, Felhasználók



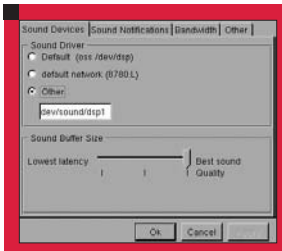
3. ábra A TeamSpeak főablaka



6. ábra Hangbeállítások



4. ábra Csatlakozás



5. ábra Beállítások

paramétereit indítás előtt a ***server.ini*** fájlban is megváltoztathatjuk, így a webes adminisztrációs felület által használt portot is, de a felület ki is kapcsolható itt. Kikapcsolni nem érdemes, ugyanis a webes felületen nyílik lehetőség a szerver fő beállításainak (név, port, MOTD, webpost, stb...) megadása mellet a felhasználók és a jogosultságok kezelésére is.
A következő jogosultsági szintek léteznek:

- Server Admin
- Channel Admin
- Operator
- Voiced
- Registered
- Anonymous

A szinteken belül egyenként nagyjából egyoldalnyi opció közül választhatjuk ki, hogy adott dolgot meg tehet-e a csoport vagy nem. Amennyiben szerverünk klán szerverként üzemel, fel kell venni a felhasználókat, publikus szerverként a nem regisztráltakat is enged(heti) csatlakozni a szerver. A jogosultsági rendszer nagyon rugalmas, új csoportokat ugyan nem enged felvinni, vagy meglévő nevét megváltoztatni, de ennek ellenére is könnyen finomhangolható a rendszer a meglévő lehetőségekkel is.
A kliens a ***setup.sh*** szkripttel installálható, alapértelmezésként a felhasználó könyvtárán belüli ***TeamSpeak2RC2*** könyvtárba. Innen indítható a ***./TeamSpeak*** paranccsal. Installálás után érdemes a ***Settings/Options***-ben kiválasztani a megfelelő ***DSP*** hangeszközt, amennyiben több hangkártyánk van. Itt lehet konfigurálni a program hangüzeneteit, illetve a sávszélesség használat lekorlátozását is. A szerverhez csatlakozni a szokásos módokon lehet, nyilvános szerverhez elég annak címe és a becenév. Klánszerverhez való csatlakozás esetén meg kell adni a leregisztrált név-jelszó párost is. A becenév eltérhet a regisztrált bejelentkezési névtől.
A hangbeállítások finomhangolása a ***Settings/Sound Input/Output Settings***-ben lehetséges. A mikrofon ki/bekapcsolására két mód van: kézzel, alapértelmezésként a szóköz billentyűvel, illetve megoldható automatikusan, attól függően, hogy a mikrofon által érzékelt hang ereje átlép-e egy küszöbszintet.
A küszöbszint a ***Voice Activation Level*** felirat alatti csúszkával állítható.
Az ***Activate Local Test Mode*** bekapcsolásával helyben, a többiek zavarása nélkül tesztelhetjük beállításunkat, így a saját hangunkat is visszaadja.
Az ***Options/Key Settings*** részben könnyen beállíthatóak a gyorsbillentyűk. A kliensprogram ezeken kívül rengeteg opcióval rendelkezik, jogosultságokat tudunk adni/elvenni, felhasználót regisztrálni, szobát létrehozni/kezelni, stb...
Rövid szöveges üzeneteket is képes küldeni, felhasználónak-szobának-mindenkinek, ezek a kliens alsó ablakában jelennek meg.

Tapasztalatok

Körülbelül fél éve használjuk barátaimmal ezt a megoldást internetes telefonálásra, konferenciabeszélgetésekre nagy megelégedéssel. A legnagyobb előnye a keresztplatformos jelleg, tehát hogy nem kell újraindítgatni ha beszélgetni akarok. Mindemellett kliens és a szerver is könnyen installálható és beállítható, nem igényel komoly hardvert és sávszélességet, és – kodektől függően – jó a hangminősége is.

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

KAPCSOLÓDÓ CÍMEK

TeamSpeak hivatalos nyilvános tesztszerverek:
➲ http://www.goteamspeak.com/index.php?page=getstarted

SLYTeam szerverlista:
➲ http://sly.hu/tss/listing.php?detail=195.56.111.225&detailport=8767&page=1&sort=server_name&direction=asc&showgroup=all

SLYTeam TS leírás, magyar nyelven:
➲ http://sly.hu/index.php?option=content&task=view&id=15&Itemid=25

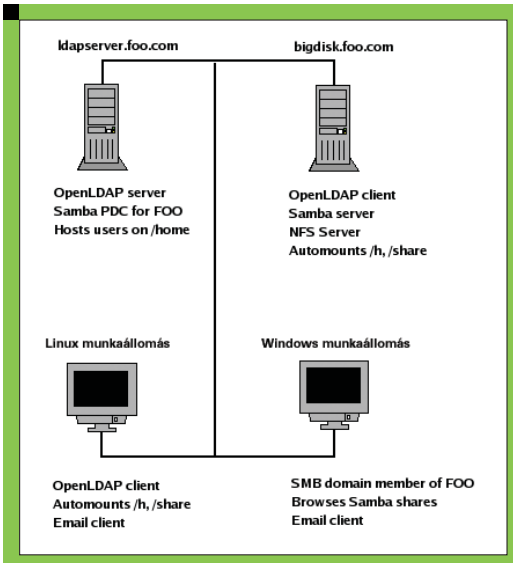
OpenLDAP mindenütt – újra

A Samba 3 új szolgáltatásként képes egységes címtárat biztosítani az összes ügyfél számára. Az új program közös alapot szolgáltat a levelezéshez, a fájlmegosztáshoz és a többi szolgáltatáshoz.

2003 januárjában megjelent, *„OpenLDAP mindenütt”* című írásunk alapján több olvasónk is sikeresen épített egységesített vállalati bejelentkezési rendszert. Azóta az *OpenLDAP* és a *Linux* egyaránt sokat fejlődött. Az alábbiakban szeretnénk bemutatni, hogyan használható az *OpenLDAP* vegyes környezet központi címtárszolgáltatásaként. Az *LDAP* kiszolgáló az összes ügyfél számára megosztott elektronikus levelező címtárat biztosít, valamint támogatja a *Linux* és a *Microsoft Windows* alapú ügyfelek bejelentkezését, a kezdőkönyvtárak önműködő befűzését és a fájlmegosztást. Az írásunk alapjául szolgáló, egyszerű, vegyes környezetet az *1. ábrán* szemléltettük.

LDAP-kiszolgáló telepítése és beállítása

Az írásunk tárgyát adó *LDAP* kiszolgálót bináris *RPM* csomagokból és az *openldap-2.2.13-2* segítségével telepítettük, *Fedora Core 3* rendszerre. Az *nss_ldap* csomagot szintén telepítenünk kellett. A legújabb források az *openldap.org* webhelyről érhetők el (lásd a forrásokat). Telepítés után az *1. kódrészletben* látható módon írjuk át a */etc/openldap/slapd.conf* beállító fájlt. Az üres karakterrel kezdődő sorok értelmezése az előző sor folytatásaként történik, vagyis a hosszabb sorok végére nem kell visszaperjelet írni.
Az *LDAP* séma a címtárbejegyzéseket alkotó objektumosztályokat és jellemzőket határozza meg. Az *RPM* csomagban is megtalálható *Red Hat*-féle *autofs* séma pont megfelel az igényeinknek. Ha hozzá szeretnénk adni a címtárhoz egy *objektumosztályt* (`objectClass`) vagy *jellemzőt* (`attribute`), tanulmányozzuk át az *OpenLDAP* felügyeleti útmutatóját.
Az adatbázis típusa az alapértelmezett *ldbm* lett. Példánkban az *LDAP* tartomány összetevőjét használjuk, így a `pelda.com` `dc=pelda,dc=com` formát nyert.
A *kezelő (manager)* teljes írási jogot kapott az *LDAP* bejegyzésekhez. A kezelő jelszavát a /usr/sbin/slappasswd segítségével hozhatjuk létre, majd a titkosított jelszót a *slapd.conf* `rootpw` bejegyzésébe kell bemásolnunk.
Az indexsorok révén gyorsítható a gyakrabban lekérdezett jellemzők elérése. A hozzáférés-vezérlés korlátozza a `userPassword` *(felhasználói jelszó)* bejegyzés elérését, ezt csak a felhasználó és a kezelő módosíthatja. Minden más bejegyzésre a kezelő írási, mások pedig olvasási jogot kaptak.



1. ábra Vegyes környezetben a Linux és a Windows alapú ügyfelek közös LDAP-infrastruktúrát használnak

A címtárszerkezet létrehozása

A címtár minden elemét egyértelműen azonosítja a *megkülönböztető neve (distinguished name,* `dn`*)*. A `pelda.com` megkülönböztető neve például: `dc=pelda, dc=com`.
Az `organizationalUnit` *(szervezeti egység,* `ou`*)* a bejegyzések csoportosítására biztosít lehetőséget. A címtár szerkezete a *2. kódrészlet* alapján követhető.
A legfelsőbb szintű bejegyzéseket *LDAP adatcsere formátumban (LDAP Interchange Format, LDIF)* hozzuk létre, majd a *3. kódrészletben* látható formában elmentjük őket a `top.ldif`-be.
A legfelsőbb szintű bejegyzéseket az `ldapadd` paranccsal adjuk hozzá a címtárhoz:

```
ldapadd -x -D 'cn=manager,dc=pelda,dc=com' \
-W -f top.ldif
```

Ezután egy az összes bejegyzést lekérdező `ldapsearch` paranccsal ellenőrizzük munkánkat:

```
ldapsearch -x -b 'dc=pelda,dc=com'
```

1. kódrészlet A slapd.conf fájl fontos, az LDAP biztonságos futtatásához szükséges beállításokat tartalmaz

```
# slapd.conf
# a használni kívánt sémák
include     /etc/openldap/schema/core.schema
include     /etc/openldap/schema/cosine.schema
include     /etc/openldap/schema/
            ➥inetorgperson.schema
include     /etc/openldap/schema/nis.schema
include     /etc/openldap/schema/samba.schema
include     /etc/openldap/schema/redhat/autofs.schema
# az adatbázis megadása
database    ldbm
suffix      "dc=pelda,dc=com"
rootdn      "cn=Manager,dc=pelda,dc=com"
# A nyílt jelszavakat, különösen a rootdn esetében
# kerülni kell. Használjunk erős hitelesítést.
# root jelszó
rootpw      {SSHA}xxxxxxxxxxxxxxxxxxxx
directory   /var/lib/ldap
# Az adatbázishoz fenntartott indexek
index       objectClass,uid,uidNumber,gidNumber,
            ➥memberUid eq
index       cn,mail,surname,givenname eq,subinitial
index       sambaSID eq
index       sambaPrimaryGroupSID eq
index       sambaDomainName eq
# A felhasználók hitelesítést végezhetnek, illetve
# megváltoztathatják jelszavukat.
access to attrs=userPassword,sambaNTPassword,
       sambaLMPassword
       by dn="cn=Manager,dc=pelda,dc=com" write
       by self write
       by anonymous auth
       by * none
# A többi jellemző mindenki számára olvasható.
access to *
       by self write
       by dn="cn=Manager,dc=pelda,dc=com" write
       by * read
```

Az e-mail címek megosztása

Ezen a ponton már elegendően bonyolult *LDAP*-szerkezetet alkottunk ahhoz, hogy valós használatba vehessük. Kezdjük az e-mail címek megosztásával. Az eljárás egyszerűbb, ha a névjegyalbumunkat ki tudjuk menteni *LDIF* formátumban, erre például a *Mozilla Thunderbird* az *Address Book (Címjegyzék)* ablak *Tools (Eszközök)* menüjéből biztosít lehetőséget. A kapott fájlt további műveletek végrehajtásával az alábbi példához hasonló formátumúra kell hozni, ez a legjobban talán *Perlben* oldható meg.

A névjegyeket a hozzájuk tartozó e-mail címek azonosítják egyedileg. Például egy névjegy megkülönböztető neve: `dn: uid=valaki@valahol.com,ou=contacts,?ou=people,dc=pelda,dc=com`.

2. kódrészlet Az LDAP megkülönböztető nevei a szervezeti egységek szerint faszerkezetbe rendeződnek

```
+ dc=pelda,dc=com
|- ou=People                  Személyek
|  |- ou=contacts,ou=people   E-mail címek
|- ou=Groups                  Rendszercsoportok
|- ou=auto.master             Automount mester térkép
|- ou=auto.home               Automount térkép
|- ou=auto.misc               Automount térkép
|- ou=Computers               Samba tartománytagok
|- cn=NextFreeUnixId          Következő szabad
                              Samba-azonosító
|- SambaDomainName            Samba tartományi
                              információs
                              objektumosztály
```

A névjegy teljes bejegyzése az összes adattal így alakul:

```
dn: uid=valaki@valahol.com,ou=contacts,
?ou=people,dc=pelda,dc=com
mail: valaki@valahol.com
uid: valaki@valahol.com
givenName: Valaki
sn: Kitudja
cn: Valaki Kitudja
objectClass: person
objectClass: top
objectClass: inetOrgPerson
```

Az egyes névjegyeket egy-egy üres sorral válasszuk el egymástól, majd mentsük el őket egy *contacts.ldif* nevű fájlba. A címtárhoz az `ldapadd` paranccsal adhatjuk hozzá a névjegyeket:

```
ldapadd -x -D 'cn=manager,dc=pelda,dc=com' \
-W -f contacts.ldif
```

Az utána következő ellenőrzést az előbbihez hasonlóan, az `ldapsearch` paranccsal végezhetjük el.

A levelezőügyfelek beállítása

A következő lépésünk a *Mozilla Thunderbird* beállítása az új *LDAP* kiszolgáló használatára (2. *ábra*). A *Thunderbird Tools (Eszközök)* menüjéből válasszuk az *Options (Beállítások)* parancsot. A *Composition (Összeállítás)* lapon válasszuk a *Directory Server (Címtárkiszolgáló)*, az *Edit Directories (Címtárak szerkesztése)*, majd az *Add (Hozzáadás)* parancsot. A *Directory Server Properties (Címtárkiszolgáló tulajdonságai)* panelen a következő adatokat kell beírnunk:

```
Name: PELDA
Server: ldapkiszolgalo.pelda.com
base DN: ou=people,dc=pelda,dc=com
```

Az *Advanced (Speciális)* lapon a címtár méretének megfelelően növeljük meg a kapott találatok számát. A `pelda.com` esetében mi 1000 találatot állítottunk be.

A beállításokat úgy ellenőrizhetjük, hogy írunk egy üzenetet egy az *LDAP* címtár névjegyalbumában szereplő személynek. Gépelés közben a programnak magától ki kell egészítenie a címet. Szintén alkalmas az ellenőrzésre, ha a ***Thunderbird Mail Address Bookjából (E-mail címjegyzék)*** indítunk egy *LDAP* keresést. A keresést a `PELDA` címjegyzékben végezzük, és keresési feltételként a ***Name or Email contains (Név vagy e-mail tartalmazza a következőt)*** mezőbe írjunk be egy csillagot; ekkor eredményként az összes névjegyet meg kell kapnunk.

Egységesített, LDAP alapú bejelentkezés Linux alatt

Ha a felhasználói fiókok adatait *LDAP* alatt tároljuk, akkor ugyanazt a felhasználónevet és jelszót tetszőleges linuxos konzolon használhatjuk. Először el kell döntenünk, hogy mely felhasználóneveket szeretnénk bevinni *LDAP* alá. Az *1. táblázat* a saját felhasználói sémánk ***UID/GID (felhasználóazonosító/csoportazonosító)*** kiosztását szemlélteti. A fenti felhasználói sémával 9000 egységesített, *LDAP* alatti bejelentkezési bejegyzést tudunk létrehozni, valamint az *LDAP* alatti *UID*-kkal és *GID*-kkel nem ütköző helyi felhasználókat és csoportokat is meg tudunk adni. A felhasználói séma szerint a *Samba* elsődleges tartományvezérlője számára szükséges fiókokat is el tudjuk különíteni.

A felhasználói bejelentkezési bejegyzések létrehozása LDAP alatt

A felhasználói bejelentkezési bejegyzéseket `uid`-ként a bejelentkezési név azonosítja. A bejelentkezési felhasználók az `ou=people` tagjai, vagyis a megkülönböztető név a következő lesz:

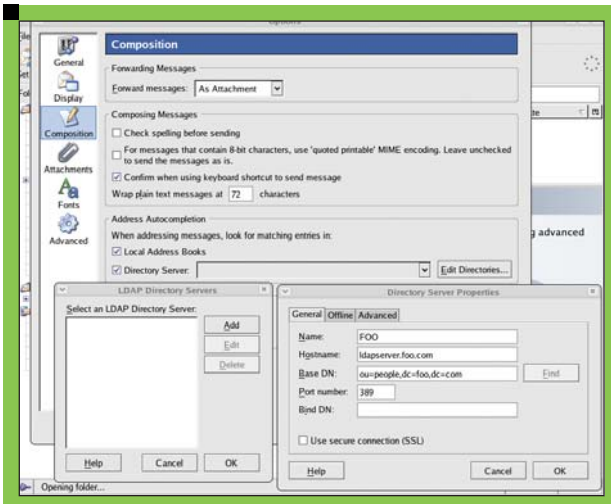
```
dn: uid=gomerp,ou=people,dc=pelda,dc=com
```

A teljes bejegyzés a fiók elérésének szabályozásához szükséges jellemzőket is tartalmazza, amint az a *4. kódrészletben* is látható, továbbá a *Samba* által igényelt beállításokat is felöleli; ezekről később lesz szó.
Az *OpenLDAP*-hoz áttérést segítő programok is tartoznak, ezekkel ki tudjuk nyerni a felhasználói fiókok adatait; minderről részletesebb tájékoztatást a */usr/share/openldap/migration* könyvtárban lehet találni. Ha meglévő */etc/passwd* fájlt szeretnénk *LDIF* formátumúvá alakítani, akkor először ismerkedjünk meg a ***migrate_common.ph*** fájllal. A fájlt át kell írnunk, megadva tartományunk nevét és az alapértelmezett alapsémát, illetve engedélyezve a kiterjesztett sémát:

```
# Alapértelmezett DNS-tartomány
$DEFAULT_MAIL_DOMAIN = "pelda.com";
# Alapértelmezett alap
$DEFAULT_BASE = "dc=pelda,dc=com";
# Bekapcsolásával általánosabb objektumosztályokat
# is támogathatunk,
# mint például a person (személy).
$EXTENDED_SCHEMA = 1;
```

Nyerjük ki a fiókok adatait a */etc/passwd* fájlból:

```
/usr/share/openldap/migration/migrate_passwd.pl \
/etc/passwd > people.ldif
```



2. ábra A vállalati címtár használatához adjuk meg a kiszolgáló adatait a Thunderbird Directory Server Properties (Címtárkiszolgáló tulajdonságai) párbeszédpanelén

1. táblázat *A felhasználói sémánk UID/GID-kiosztása*

Fiók típusa	UID
Rendszerfiókok	UID < 500
Különleges Samba fiókok	499 < UID < 1000
Egységesített fiókok	999 < UID < 10000
Helyi felhasználók, melyek nem szerepelnek az LDAP-ban	> 10000

Ellenőrizzük a kapott *LDIF* fájlt. A rendszerfiókokhoz – mint a root – és a helyi felhasználók saját, az *LDAP*-ban szükségtelen csoportjaihoz tartozó bejegyzéseket távolítsuk el. Adjuk hozzá a felhasználói bejegyzéseket a címtárhoz, majd az `ldapsearch` paranccsal a már ismert módon ellenőrizzük az eredményt:

```
ldapadd -x -D 'cn=manager,dc=pelda,dc=com' -W \
-f people.ldif
```

Mivel a bejelentkezési felhasználók az `ou=people` szervezet tagjai, ettől kezdve elektronikus levélcímüket a levelező-programunkból is elő tudjuk keresni.

Csoportbejegyzések létrehozása

Minden több linuxos számítógép között megosztani kívánt csoporthoz létre kell hoznunk egy csoportbejegyzést. Emellett minden felhasználót be kell sorolnunk egy saját csoportba. A csoportbejegyzéseket a cn azonosítja, és minden csoport az `ou=Groups` (csoportok) szervezethez tartozik. Például:

```
dn: cn=gomerp,ou=Groups,dc=pelda,dc=com
```

Egy felhasználó saját csoportja a következőképpen néz ki:

```
dn: cn=gomerp,ou=Groups,dc=pelda,dc=com
objectclass: posixGroup
objectclass: top
```

3. kódrészlet Az LDAP-fa csúcsát (top.ldif) kézzel hozzuk létre, egyszerű, kulcs: érték formátumban

```
dn: dc=pelda,dc=com
objectClass: dcObject
objectClass: organization
o: Példa Vállalat
dc: pelda
dn:ou=People,dc=pelda,dc=com
objectClass: organizationalUnit
ou: People
dn:ou=Groups,dc=pelda,dc=com
objectClass: organizationalUnit
ou: Groups
dn:ou=contacts,ou=people,dc=pelda,dc=com
associatedDomain: pelda.com
ou: contacts
ou: people
objectClass: organizationalUnit
objectClass: domainRelatedObject
```

```
cn: gomerp
userPassword: {crypt}x
gidNumber: 5223
```

Egy megosztott csoport pedig így:

```
dn: cn=web_dev,ou=Groups,dc=pelda,dc=com
objectclass: posixGroup
objectclass: top
cn: web_dev
gidNumber: 5019
memberUid: gomerp
memberUid: goober
memberUid: barneyf
```

Másoljuk ki a csoportadatokat a */etc/group* fájlból:

```
/usr/share/openldap/migration/migrate_passwd.pl \
/etc/group > group.ldif
```

Ellenőrizzük a kapott ***LDIF*** fájlt. A rendszercsoportokhoz és a helyi rendszerfelhasználókhoz tartozó, az ***LDAP***-ban szükségtelen adatokat töröljük belőle.
Adjuk hozzá a csoportbejegyzéseket a címtárhoz, majd az `ldapsearch` paranccsal végezzük el az ellenőrzést:

```
ldapadd -x -D 'cn=manager,dc=pelda,dc=com' -W \
-f group.ldif
```

Az ***automount*** beállítása a kezdőkönyvtárak megosztására és az ***NFS*** megosztások elérhetővé tételére.
Egységesített bejelentkezésnél a felhasználók központi, ***NFS***-en ***(Network File System, hálózati fájlrendszer)*** keresztül megosztott kezdőkönyvtárral rendelkeznek. Bár mi a kezdőkönyvtárakat az ***ldapkiszolgalo.pelda.com***-on helyeztük el, majd megosztottuk a */home* könyvtárat, a fájlkiszolgálónak és az ***OpenLDAP***-nak nem muszáj azonos gépen futnia. Az ***NFS*** tárgyalása túlmutatna témakörünkön, mégis idézzük be azt az egy sort a */etc/exports* fájlból, mely a kezdőkönyvtárak elérhetővé tételéért felelős:

4. kódrészlet Egy felhasználói bejelentkezési bejegyzés a bejelentkezéshez szükséges adatok mellett a Samba bizonyos beállításait is tartalmazza

```
dn:uid=gomerp,ou=People,dc=pelda,dc=com
uid: gomerp
cn: Gomer Pyle
sn: Pyle
givenname: Gomer
mail: gomer.pyle@pelda.com
objectClass: top
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: sambaSAMAccount
uidNumber: 5000
homeDirectory: /h/gomerp
loginShell: /bin/bash
description: Gomer Pyle
displayName: Gomer Pyle
gecos: Gomer Pyle
gidNumber: 513
userPassword: {SSHA}xxxxxxxxxxxxxxxxxxxxxxxx
sambaLogonTime: 0
sambaLogoffTime: 2147483647
sambaKickoffTime: 2147483647
sambaPwdCanChange: 0
sambaSID: S-1-5-21-1400792368-3813960858
➥-1703501993-11000
sambaPrimaryGroupSID: S-1-5-21-1400792368
➥-3813960858-1703501993-513
sambaLogonScript: gomerp.cmd
sambaHomeDrive: H:
sambaHomePath: \\LDAPKISZOLGALO\gomerp
sambaLMPassword: XXXXXXXXXX
sambaAcctFlags: [U]
sambaNTPassword: XXXXXXXXXX
sambaPwdLastSet: 1097240543
sambaPwdMustChange: 1105016543
```

```
/home *.pelda.com(rw)
```

A linuxos ***LDAP***-ügyfelek az ***automount*** és az ***NFS*** segítségével bejelentkezéskor befűzik a felhasználó kezdőkönyvtárát.
Az ***LDAP*** az ***automount*** szemszögéből a ***NIS (network information service, hálózati információs szolgáltatás) automount*** térképek helyettesítője. Az ***auto.master***, az ***auto.home*** és az ***auto.misc automount*** térképét kell helyettesítenünk, amihez új szervezeti egységet kell létrehoznunk az ***auto.master*** számára:

```
dn: ou=auto.master,dc=pelda,dc=com
objectClass: top
objectClass: automountMap
ou: auto.master
```

Az `auto.master` bejegyzést a `cn` azonosítja. Az `automountInformation` jellemző arra utasítja az automountot, hogy a térképet ***LDAP*** alatt keresse:

```
dn: cn=/h,ou=auto.master,dc=pelda,dc=com
objectClass: automount
automountInformation: ldap:ou=auto.home,
➥dc=pelda,dc=com
cn: /h
```

Ha már itt vagyunk, hozzuk létre a többi ***NFS***-en keresztül megosztott könyvtár `auto.master` bejegyzését is:

```
dn: cn=/share,ou=auto.master,dc=pelda,dc=com
objectClass: automount
automountInformation: ldap:ou=auto.misc,
➥dc=pelda,dc=com
cn: /share
```

Az `automount` bejegyzéseket ***LDIF*** formátumban készítsük elő, mentsük el őket ***auto.master.ldif*** névvel, majd adjuk hozzá a bejegyzéseket az ***LDAP***-hoz:

```
ldapadd -x -D 'cn=manager,dc=pelda,dc=com' -W -f
auto.master.ldif
```

Ez után hozzunk létre egy új szervezeti egységet az `sauto.home` számára:

```
dn:ou=auto.home,dc=pelda,dc=com
objectClass: top
objectClass: automountMap
ou: auto.home
```

A kezdőkönyvtár-bejegyzéseket a `cn` azonosítja:

```
dn: cn=gomerp,ou=auto.home,dc=pelda,dc=com
objectClass: automount
automountInformation:
➥ldapkiszolgalo.pelda.com:/home/gomerp
cn: gomerp
```

Hozzuk létre az összes felhasználó `auto.home` bejegyzését ***LDIF*** formátumban, mentsük el ***auto.home.ldif*** névvel, majd adjuk hozzá a bejegyzéseket az ***LDAP***-hoz:

```
ldapadd -x -D 'cn=manager,dc=pelda,dc=com' -W \
-f auto.home.ldif
```

Linux alapú ***LDAP*** ügyfélről automatikusan befűzve az `ldapkiszolgalo.pelda.com:/home/gomerp` kezdőkönyvtár a */h/gomerp* útvonalon lesz elérhető. Szükség esetén további ***NFS***-megosztásokat is megadhatunk ***LDAP*** alatt, illetve befűzhetünk automatikusan. Az `auto.misc` szervezeti egység ezeket az automatikus befűzést segítő térképeket tartalmazza, formátumuk `ou=auto.misc`.
A */share auto.master* bejegyzését a fentiek szerint már létrehoztuk, most következzen az `ou=auto.misc` bejegyzés:

5. kódrészlet Részletek az OpenLDAP címtárral való együttműködésre beállított Samba smb.conf fájljából

```
[global]
...
  obey pam restrictions = No
  ldap passwd sync = Yes
  ldap passwd sync = Yes
...
  passdb backend = ldapsam:ldap://
  ➥ldapkiszolgalo.pelda.com/
  ldap admin dn = cn=Manager,dc=pelda,dc=com
  ldap suffix = dc=pelda,dc=com
  ldap group suffix = ou=Groups
  ldap user suffix = ou=People
  ldap machine suffix = ou=Computers
  ldap idmap suffix = ou=People
  ldap ssl = no
  add user script = \
    /usr/local/sbin/smbldap-useradd -m "%u"
  ldap delete dn = Yes
  delete user script = \
    /usr/local/sbin/smbldap-userdel "%u"
  add machine script = \
    /usr/local/sbin/smbldap-useradd -w "%u"
  add group script = \
    /usr/local/sbin/smbldap-groupadd -p "%g"
  delete group script = \
    /usr/local/sbin/smbldap-groupdel "%g"
  add user to group script = \
    /usr/local/sbin/smbldap-groupmod -m "%u" "% g"
  delete user from group script = \
    /usr/local/sbin/smbldap-groupmod -x "% u" "%g"
  set primary group script = \
    /usr/local/sbin/smbldap-usermod -g "%g" "%u "
```

```
dn:ou=auto.misc,dc=pelda,dc=com
ou: auto.misc
objectClass: top
objectClass: automountMap
```

Az ***NFS***-megosztások bejegyzéseit az `ou=auto.misc` alatt kell létrehozni:

```
dn:cn=redhat,ou=auto.misc,dc=pelda,dc=com
objectClass: automount
automountInformation:
nagymeghajto.pelda.com:/pub/redhat
cn: redhat
dn:cn=engineering,ou=auto.misc,dc=pelda,dc=com
objectClass: automount
automountInformation:
nagymeghajto.pelda.com:/data/engineering
cn: engineering
```

A bejegyzéseket mentsük el ***auto.misc.ldif*** névvel majd adjuk hozzá őket az ***LDAP***-hoz:

```
ldapadd -x -D 'cn=manager,dc=pelda,dc=com' -W -f
➥auto.misc.ldif
```

Linux alapú *LDAP* ügyfélről automatikusan befűzve a `nagymeghajto.pelda.com:/data/engineering` megosztott könyvtár */share/engineering* névvel fog látszani.

A linuxos LDAP-ügyfél beállítása

A linuxos *LDAP* ügyfél beállításának megkezdése előtt telepíteni kell a ***névszolgáltatás-kapcsoló (name service switch)*** csomagot, az *nss_ldap*-t. A *Red Hat* */usr/bin/authconfig* segédeszköze kényelmesen alkalmazható az ügyfél beállítására. Válasszuk a ***Use LDAP (LDAP használata)*** beállítást, majd a ***Server (Kiszolgáló)*** mezőbe írjuk be a következőt: ***ldapkiszolgalo.pelda.com***, a ***Base DN (Alap DN)*** mezőbe pedig a következőt: `dc=pelda,dc=com`. Az *authconfig* a következő fájlokba írja az adatokat: */etc/ldap.conf*, */etc/openldap/ldap.conf* és */etc/nsswitch.conf*.

Ellenőrizzük, hogy a */etc/nsswitch.conf* fájlban szerepelnek-e a következő bejegyzések:

```
passwd:     files ldap
shadow:     files ldap
group:      files ldap
automount:  files ldap
```

Ellenőrizzük, hogy a */etc/ldap.conf* fájl tartalmazza-e a következő bejegyzéseket:

```
host ldapkiszolgalo.pelda.com
base dc=pelda,dc=com
```

Ellenőrizzük, hogy a */etc/openldap/ldap.conf* tartalmazza-e a következő sorokat:

```
HOST ldapkiszolgalo.pelda.com
BASE dc=pelda,dc=com
```

A linuxos kiszolgáló további beállításai

Azon az *NFS*-kiszolgálón, amelyen a kezdőkönyvtárak találhatók, el kell távolítani a felhasználók jelszó- és csoportbejegyzéseit a *password* és a *group* fájlból. Készítsünk biztonsági mentéseket, majd írjuk át a */etc/passwd*, a */etc/shadow*, a */etc/group* és a */etc/gshadow* fájlt, törölve az *LDAP* alatt is szereplő személyek bejegyzéseit. Esetünkben a */etc/passwd* fájlban nem maradhat 1000 és 9999 közötti *UID*. Ellenőrzésképpen jelentkezzünk be egy *Linux* alapú *LDAP* ügyfélre egy *LDAP* felhasználónév használatával. A felhasználó bejelentkezési héját és kezdőkönyvtárát kell látnunk. Az *auto.misc* megosztásokat a megosztási név alapján történő eléréssel ellenőrizhetjük, például:

```
cd /share/redhat
```

Az *automount* csak tényleges használatukkor fűzi be az *NFS* megosztásokat, tehát a */share/redhat* könyvtár csak használata után lesz látható.

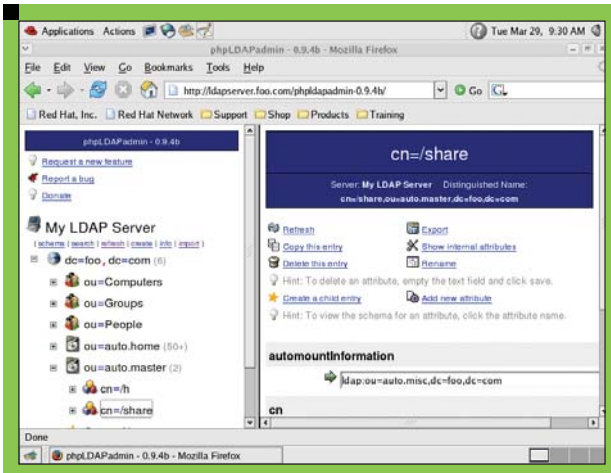
Egységesített bejelentkezés Samba és LDAP használatával

A *Samba* és az *LDAP* együttes használatának célja az, hogy egységes bejelentkezést tegyünk lehetővé a *Microsoft Windows* alapú ügyfelek számára. A gyakorlatban ez azt jelenti, hogy a felhasználók bármelyik munkaállomásról bejelentkezhetnek a hálózatra, illetve hozzáférést nyerhetnek a megosztott mappákhoz, fájlokhoz és nyomtatókhoz. Az egységesített bejelentkezés felé az első lépés a *Samba* beállítása ***elsődleges tartományvezérlőnek (primary domain controller, PDC)***. Ennek részletes tárgyalása túlmutatna írásunk keretein, ám a témáról kiváló *HOGYAN* található az *Idealx* webhelyén (lásd a forrásokat). Az *Idealx* munkatársai nagyszerű kiegészítéseket készítettek a *Samba Projecthez*, és aki komolyan akar foglalkozni a *Sambával*, annak érdemes megismerkednie segédeszközeikkel. Feltételezve, hogy már szereztünk tapasztalatot a *Samba* tartományvezérlők kezelésében, az *5. kódrészletben* részben megadott beállító fájl megfelelő alapot szolgáltathat a példánkban szereplőhöz hasonló környezet üzembe helyezéséhez. A teljes fájl a *Linux Journal FTP*-helyéről tölthető le. (Lásd a forrásokat.)

Mindezek után már csak annyi dolgunk maradt, hogy az *LDAP*-ot rávegyük a *Samba* az elmúlt évek során megjelent újdonságaiból fakadó lehetőségek kihasználására. Az eljárás hasonló a fentiekhez, de a *Samba* újabb szolgáltatásait is érdemes kihasználni. A *Samba 3*-as változatánál már lehetőség van arra, hogy az összes *Samba* fiókadatot az *LDAP* címtárban tároljuk, aminek az az előnye, hogy minden adat egyetlen központi helyre kerül.

6. kódrészlet Az smbldap-populate eszköz automatikusan hozzáadja az OpenLDAP-kiszolgáló és a Samba együttműködéséhez szükséges fiókokat

```
[root]# smbldap-populate
Using builtin directory structure
adding new entry: dc=pelda,dc=com
adding new entry: ou=Users,dc=pelda,dc=com
adding new entry: ou=Groups,dc=pelda,dc=com
adding new entry: ou=Computers,dc=pelda,dc=com
adding new entry: ou=Idmap,dc=pelda,dc=org
adding new entry:
➥cn=NextFreeUnixId,dc=pelda,dc=org
adding new entry:
➥uid=Administrator,ou=Users,dc=pelda,dc=com
adding new entry:
➥uid=nobody,ou=Users,dc=pelda,dc=com
adding new entry: cn=Domain
➥Admins,ou=Groups,dc=pelda,dc=com
adding new entry: cn=Domain
➥Users,ou=Groups,dc=pelda,dc=com
adding new entry: cn=Domain
➥Guests,ou=Groups,dc=pelda,dc=com
adding new entry: cn=Print
➥Operators,ou=Groups,dc=pelda,dc=com
adding new entry: cn=Backup
➥Operators,ou=Groups,dc=pelda,dc=com
adding new entry:
➥cn=Replicator,ou=Groups,dc=pelda,dc=com
adding new entry: cn=Domain
➥Computers,ou=Groups,dc=pelda,dc=com
```



3. ábra A címtár webes megjelenítése a phpLDAPadminnal

A Samba és az LDAP együttműködése

Az *LDAP* és a *Samba* együttes használatának fontos elemét képezik a tényleges együttműködéshez szükséges további fiókok és *LDAP* bejegyzések. Az egységesített bejelentkezési kiszolgáló működéséhez több jól ismert windowsos tartományi felhasználói és csoportfiókra is szükség van. A tartományi fiókok adatainak tárolásához különleges OU bejegyzéseket is létre kell hoznunk. Szerencsére létezik egy *smbldap-populate* nevű parancsfájl, mely az összes szükséges bejegyzést hozzáadja a rendszerhez. A parancsfájl része az *Idealx smbldap-tools* csomagjának, mely a *PDC* és a *Sambával* együttműködő *LDAP* címtár üzembe helyezésében egyaránt fontos segítséget jelent. A *6. kódrészlet* példa arra, hogy mit kell látnunk az *smbldap-populate* parancsfájl futtatásakor.

Ha megvizsgáljuk a parancsfájl kimenetét, láthatjuk, hogy jó néhány új felhasználót, csoportot és szervezeti egységet adott hozzá a címtárhoz, ilyen például a jól ismert *Domain Admins (Tartományi rendszergazdák)* és *Domain Users (Tartományi felhasználók)* csoport. A *Microsoft Windows NT* alapú változatai alapesetben is tartalmazzák ezeket a csoportbejegyzéseket. Mindegyikhez tartozik egy *viszonylagos azonosító (relative identifier, RID)* is, tehát az *LDAP* alatti felhasználói és csoportbejegyzésekhez a megfelelő windowsos felhasználók vagy csoportok *Windows* alatti *RID*-it kell rendelni. Az *smbldap-populate* parancsfájl ezekre a részletekre is ügyel. A szükséges jól ismert felhasználói és csoport *RID*-k a következők:

```
Név             RID
-----------------
Domain Admins 512
Domain Users  513
Domain Guests 514
```

A fenti felhasználói és csoportbejegyzések mellett további *OU* bejegyzésekkel egyéb tartományi szolgáltatásokat is elérhetővé tehetünk. Az első ilyen az `ou=Computers`, mely a tagkiszolgálók és a tartományi munkaállomások gépfiókjainak tárolására szolgál. A második az `ou=Idmap`, erre a *Samba Windows*-kiszolgáló által ellenőrzött tartomány tagkiszolgálójaként történő használatakor van szükség. Az utolsó új bejegyzés az `ou=NextFreeUnixId`, ez az új felhasználók és csoportok létrehozásakor következőként felhasználható *UID*-t és *GID*-t adja meg.

A címtár felügyelete

Az *LDAP* címtár kezdeti feltöltése és a *Samba* üzembe helyezése után készen állunk a felhasználók és a csoportok címtárhoz való hozzáadására. Az *Idealx* parancssori segédprogramjaival ezt a feladatot is könnyedén elintézhetjük, de léteznek *PHP* alapú címtárkezelők is, melyekkel szintén megkönnyíthetjük munkánkat, mi például a *phpLDAPadmint* és az *LDAP Account Managert (LAM)* ajánljuk. Mindkettő könnyen használható, a címtárat és a bejegyzéseket grafikus formában jeleníti meg, illetve módot ad az *LDAP* bejegyzések szerkesztésére is (*3. ábra*). A *Java* alapú *LDAP* böngésző egy további eszköz a címtár tartalmának megtekintésére és szerkesztésére.

A 2002 decemberében megjelent cikk óta a *Samba 3.x* kiadásai rengeteg fejlesztést hoztak. Az új változatokra áttérve komolyabb ellenőrzést valósíthatunk meg a fiókok felett, valamint továbbfejlesztett csoportmegfeleltetési szolgáltatásokat érhetünk el – mindent összefoglalva magasabb szintű felügyeletet valósíthatunk meg a tartomány felett.

Karbantartás

Mindenkinek javasoljuk, hogy új *LDAP* címtárát az *egyszerű hitelesítés és biztonsági réteg (simple authentication and security layer, SASL)* és a *szállítási rétegbeli biztonság (transport layer security, TLS)* alkalmazásával védje. Erről bővebben az internetes anyagokban lehet olvasni. Gratulálunk! *LDAP* kiszolgálónkat sikeresen üzembe helyeztük, és készen állunk az elektronikus levélcímek megosztására, az egységesített bejelentkezés, valamint a bármely ügyfélről elérhető, szintén egységesített fájltárolási szolgáltatások biztosítására.

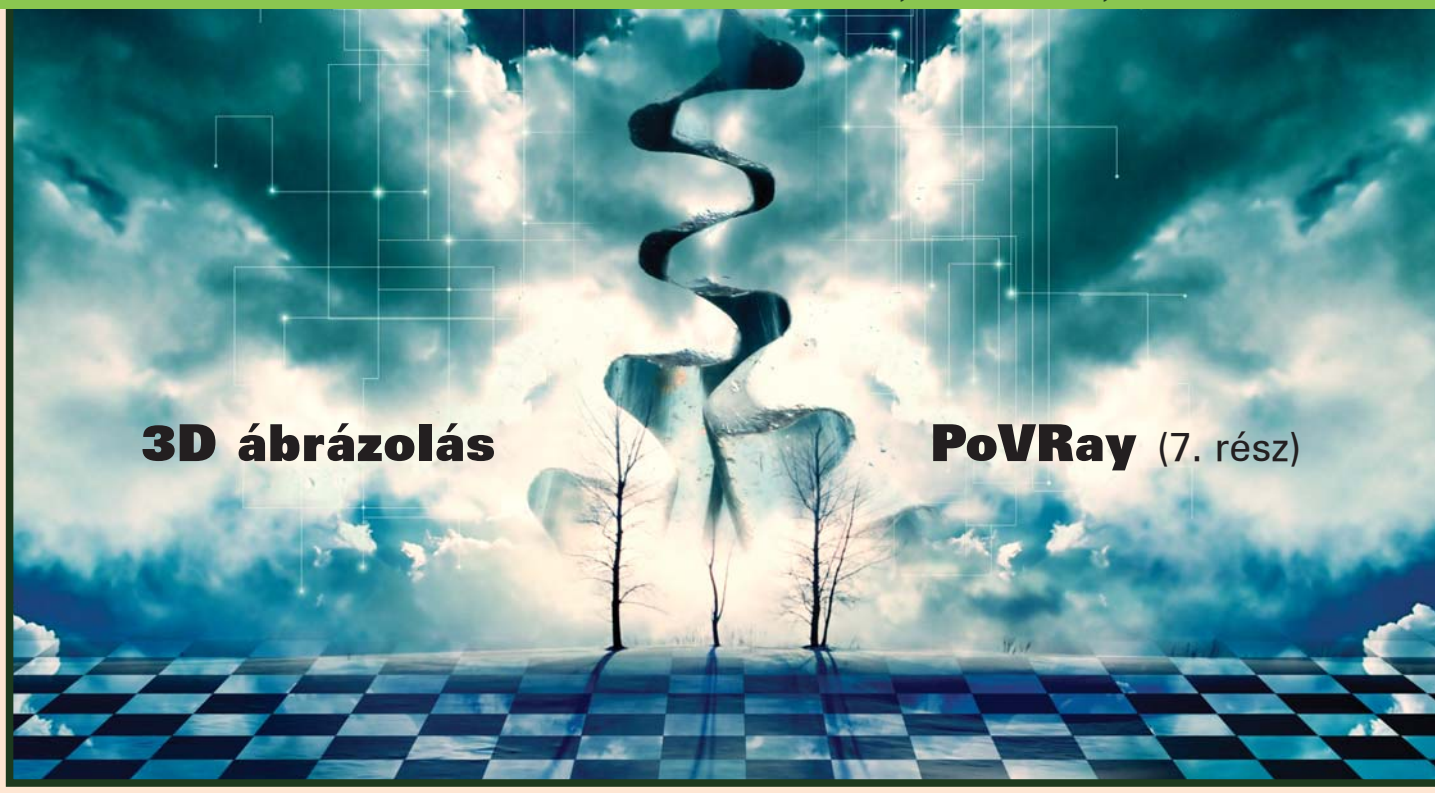
Linux Journal 2005. július, 135. szám

Craig Swanson
(craig.swanson@slssolutions.net) hálózatervezéssel és Linux tanácsadással foglalkozik az SLS Solutionsnél. A Midwest Tool & Die cégnél linuxos programok fejlesztésében is részt vesz. Craig 1993 óta használ Linuxot.

Matt Lung
(matt.lung@slssolutions.net) hálózatokkal és számítógéprendszerekkel kapcsolatos tanácsadást végez az SLS Solutionsnél, illetve a Midwest Tool & Die hálózati mérnöke.

KAPCSOLÓDÓ CÍMEK

A cikkhez tartozó források elérhetősége:
➲ www.linuxjournal.com/article/8267



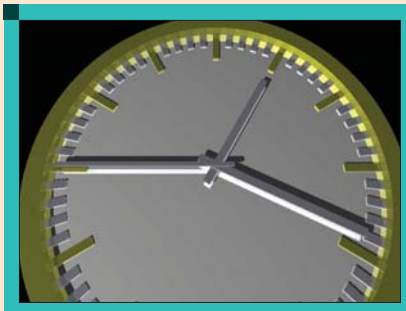
3D ábrázolás PoVRay (7. rész)

Szinte mindenki találkozott már az „animáció” szóval, amelyet általában rajzfilmekhez kötünk, holott a szó igazi jelentése a „mozgatás” vagy „haladás”, az „animátor” pedig az a személy, aki ezt a tevékenységet műveli, működteti a rá bízott „dolgokat”. Nekünk a szó közismert értelmében kell animációt készítenünk, mégpedig a 3D világunkban haladni az idővel és mozgatni vagy átalakítani a tárgyakat.

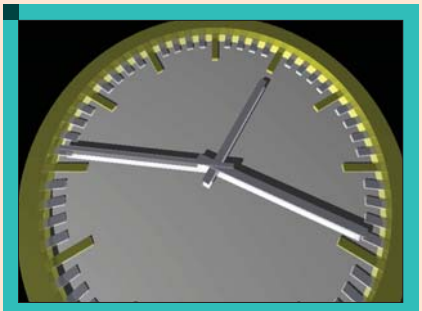
Kiindulás

Az animációk készítésénél látszik igazán, hogy a *PoVRay* egy háttérprogram (backend), amely alapvetően nem arra készült, hogy könnyedén kezelhető legyen, hanem egy felhasználói felület mögött hatékonyan és gyorsan képes legyen előállítani a kért *3D* világot. Ugyanis egyszerűen csak két változót kapunk a programtól, amelyekkel megtudhatjuk, hogy hányadik képkockát készítené el éppen; illetve egy „órajelet”, amely alapesetben 0.0 és 1.0 között változik, jobban mondva az értéke egyenletesen növekszik. Ennél több kényelemre sajnos nem számíthatunk.

Az animáció elkészítéséhez szükség van egy – már kész – *3D* világra, amelynek egyes elemeit meg akarjuk mozgatni vagy át akarjuk alakítani. Erre a célra tökéletes „alany” az előző részben elkészített óra, amelynek próbaképpen a másodperc mutatóját fogjuk mozgatni egy 60 kockából álló animáció keretében. Ehhez egyszerűen csak annyit kell tennünk,



1. ábra Az első kocka



2. ábra A második kocka

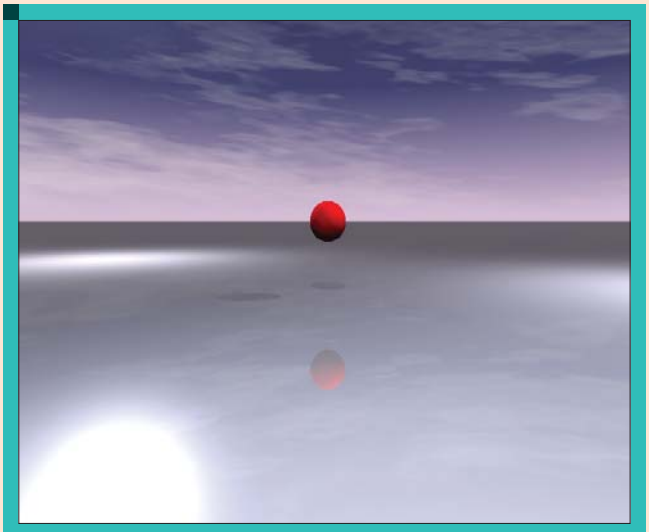
hogy az álló *3D* világba bevisszük a másodpercmutató mozgatását:

```
#declare OraMasodperc=clock*60;
```

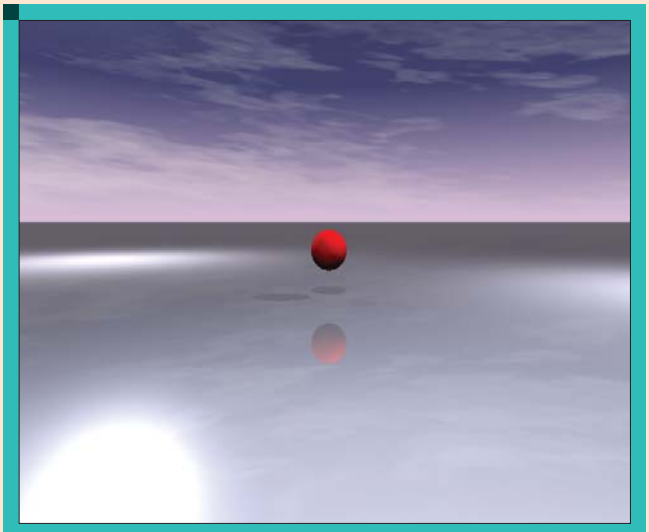
Ahhoz, hogy a *PoVRay* ne csak egy képet készítsen, meg kell adnunk a parancssorban, hogy animációt szeretnénk eredményül kapni:

```
$ povray +W800 +H600
➥+I/usr/share/povray-3.6/
➥include/ +KFI0 +KFF59
➥anim00.pov
```

A *PoVRay* a parancssornak megfelelően készíteni fog 60 darab állóképet, amelyek más-más fázisát tartalmazzák a mozgóképnek. A `clock` változó, mint „órajel”, a képkockák (`frame`) számától függő kis növekményekkel tart a 0.0 értéktől az 1.0 értékig: jelen esetben 1/60-ad ez a növekmény. A *PoVRay* egyszerűen csak annyit csinál, hogy hatvanszor meghívja a *renderelő* komponensét más-más „órajel” értékekkel. Az elkészült képeket pedig az `anim00.png` helyett az `anim0000.png` – `anim0059.png` nevű állományokba menti.



3. ábra Kis piros labda lebeg a padló felett



4. ábra Kis piros labda esik a padló felé (20. képkocka)

Ha megnézzük ezeket a képeket, akkor láthatjuk az egyes fázisokat, ahogy halad előre a másodpercmutató az óra számlapján. A *PoVRay* újabb hiányossága, hogy csak állóképek sorozatát képes elkészíteni, ebből összefüggő videót más programmal tudunk készíteni, például a `mencoder` tökéletes választás lehet:

```
$ mencoder "mf://anim00*.png"
➥-o anim00.avi -ovc lavc
```

Az óra animációját az anim00.avi állományban találhatjuk meg, természetesen lehet akár hangot is hozzáadni, de ezt már a `mencoder` leírásában (vagy Bokor Norbert cikkében) keressük meg...

Összetett mozgások

Sajnos csak a fenti eszközeink vannak animációk készítéséhez, amelyekkel igen szegényes módon tudunk „igazi” animációt készíteni, ahol a testek haladása nem vezethető vissza tisztán egyenes vonalú mozgássá vagy forgássá. Példaképpen egy labda pattogását nézhetjük, amely ideális esetben pont olyan magasra pattan vissza, mint ahonnan elejtettük, illetve esés közben folyamatosan gyorsul, majd felpattanva folyamatosan lassul. Egy kis fizika és matematika szükséges ahhoz, hogy a labda pattogjon a *3D* világunkban, ugyanis a valóságos térben a gravitáció gyorsítja esés közben, illetve a gravitáció lassítja, miután visszapattant.
A „kísérlethez” kell egy sima padló és egy piros labda (*3. ábra*, anim01.pov):

Célunk, hogy a labda közel élethű módon zuhanjon a padló felé, majd azt elérve pattanjon vissza. Ehhez az képletre lesz szükségünk, vagyis a labda aktuális sebességét és megtett útját a Földön megszokott gravitációs gyorsulás és az eltelt idő határozza meg. Ezeket egyszerűen át kell fordítani a *PoVRay* matematikai nyelvére:

```
#if (0.5>clock)
 #declare ido=clock*2;
 #declare ut=3-3*ido*ido;
#else
 #declare ido=(1.0-clock)*2;
 #declare ut=3-3*ido*ido;
#end
```

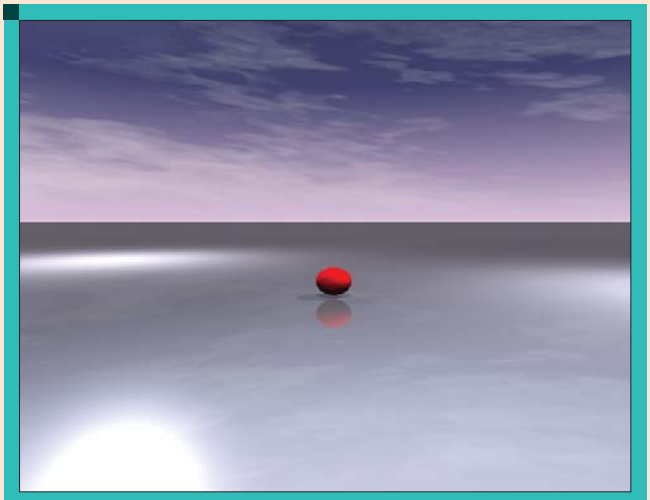
A *PoVRay* által biztosított `clock` értéket két részre bontjuk, az idő első felében a labda esni fog, a második felében pedig felpattan. Az út kiszámolásánál ezeket mind figyelembe vesszük, majd kivonjuk abból a magasságból, ahonnan a labda indulni fog. Már csak a labda elhelyezésénél kell ezt az utat felhasználni:

```
sphere{
 <0,0.8+ut,0>,0.8
 texture{
  pigment{
   Red}}}
```

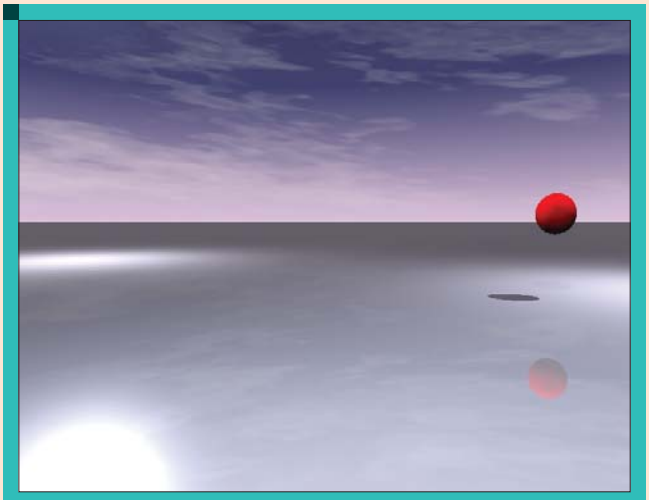
Az úthoz ne felejtsük el hozzáadni a labda sugarát, különben félig eltűnik a talajban, ami – valljuk be – nem túl realisztikus.
A valóságban a lepattanó labda egy picit benyomódik, amit nem látunk, a *3D* világunk azonban nem a valóság, a labdát a pattanás pillanatában össze tudjuk lapítani, hogy a rajzfilmekhez hasonlóan vizuálissá tegyük a pattanás tényét. Ez az összelapítás azonban nehéz ügy, hiszen akkor kell lapítanunk, amikor a labda kissé eltűnne a földben. A lapítás pedig pont annyival kell történjen, hogy a labda mindig érintse a talajt egy pontban:

```
#if (0.5>clock)
 #declare ido=clock*2;
 #declare ut=3.0-3.4*ido*ido;
#else
 #declare ido=(1.0-clock)*2;
 #declare ut=3.0-3.4*ido*ido;
#end

#if (ut<0)
 #declare labdaScale=-ut/0.8;
#else
 #declare labdaScale=1.0;
#end

sphere{
 <0,0,0>,0.8
 scale <0,labdaScale,0>
 translate <0,0.8+ut,0>
 texture{
  pigment{
   Red}}}
```

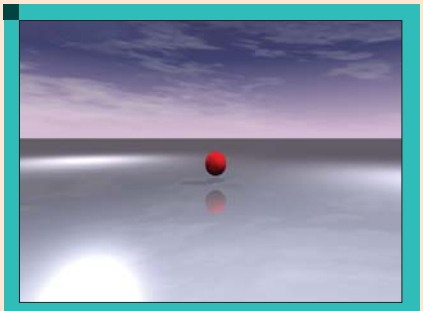
A fenti esetben a labda útját nem nulláig, hanem -0.4-ig vezetjük, s pont annyira nyomjuk össze, mint amennyit a talajba mélyedve töltene. Nem túl szép, a rajzfilmeken kicsit kevesebb matematikával és több tapasztalattal csinálják...



5. ábra Kis piros labda esik a padló felé, s odaérve lapul (38. képkocka)



6. ábra Kis piros labda érkezik jobbról (12. képkocka)



7. ábra A kis piros labda már alig pattog (50. képkocka)

Mozgások kombinálása

A kis piros labdánk érkezhetne jobbról, s folyamatosan balra haladva pattanhatna néhányat. Ehhez ki kell terjesztenünk az órajel tartományát és leválasztani a nekünk szükséges tartományokat. Ez a leválasztás annyit jelent, hogy az órajelet 0.0 és 10.0 között változtatjuk s ennek a tört részét használjuk fel a labda pattogtatásához, a többi részét pedig a jobbról balra történő mozgatáshoz:

```
#declare myClock=clock-floor
➥(clock);
#declare haladas=(clock-5.0)*3;

#if (0.5>myClock)
 #declare ido=myClock*2;
 #declare ut=3-3*ido*ido;
#else
 #declare ido=(1.0-myClock)*2;
 #declare ut=3-3*ido*ido;
#end

sphere{
 <haladas,0.8+ut,0>,0.8
 texture{
  pigment{
   Red}}}
```

Egy apró hiányérzet jelentkezik a videót nézve, mégpedig az, hogy a valóságban a labda egyre kisebb magasságra pattan vissza, a visszapattanás és a gurulás mozgási energiát von el:

```
#declare myClock=clock-floor
➥(clock);
#declare haladas=(clock-5.0)*3;
#declare magassag=3/(clock+1);

#if (0.5>myClock)
 #declare ido=myClock*2;
 #declare ut=magassag-
 ➥magassag*ido*ido;
#else
 #declare ido=(1.0-myClock)*2;
 #declare ut=magassag-
 ➥magassag*ido*ido;
#end

sphere{
 <haladas,0.8+ut,0>,0.8
 texture{
  pigment{
   Red}}}
```

Az utolsó – egyben lezáró – részben csokorba szedem a *PoVRay* apró tulajdonságait, amelyekkel jelentősen javíthatjuk a 3D világ minőségét. A cikk második felében pedig bemutatok röviden és tömören pár *PoVRay* előtét programot.

Auth Gábor
(auth.gabor@enaplo.hu)

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

KAPCSOLÓDÓ CÍMEK

A PovRay projekt honlapja
➲ http://www.povray.org

A cikkben említett fájlok
➲ http://user.enaplo.hu/~auth.gabor/pov/

Az első animáció
➲ http://user.enaplo.hu/~auth.gabor/pov/anim00.avi

A második animáció
➲ http://user.enaplo.hu/~auth.gabor/pov/anim01.avi

A harmadik animáció
➲ http://user.enaplo.hu/~auth.gabor/pov/anim02.avi

A negyedik animáció
➲ http://user.enaplo.hu/~auth.gabor/pov/anim03.avi

Az ötödik animáció
➲ http://user.enaplo.hu/~auth.gabor/pov/anim04.avi

Kiadványszerkesztés Linux alatt – Scribus (4. rész)
PDF Field a gyakorlatban

Kint, a végtelen mezőn sétálva néha az az érzésem támad, hogy minden más is kiszélesedhet. Kezdenek a határok eltűnni és mégis, a dolgok a maguk keretei között maradnak. Egyszerre összemosódnak és élesen el is vállnak. Ugyanúgy hátborzongató érzés ez, mint ahogy csodálkozásra nyitó is.

Az előző részben tárgyalt ***Scribus-PDF*** ismeretek gyakorlatban történő alkalmazásával folytatódik a sorozat. Számos eszköz áll rendelkezésre a ***Scribus***-ban arra, hogy a készítendő ***PDF*** dokumentumunkat hasznos megoldásokkal tegyük hatékonyabbá. A példában két elektronikai alkatrész, mérési jegyzőkönyvének elkészítésére kerül sor, egy dokumentumban. A jegyzőkönyv rajzait a ***Scribus***-ban található ***Shape*** eszközzel készítjük el, így megismerve annak működését. A szerkesztés előtt azonban nézzük át a ***PDF Fieldre*** (mező) vonatkozó tulajdonságokat, beállításokat. A ***PDF Fieldeket*** az eszköztáron találjuk meg, egy legördülő ikonlistában. Az *1-6. Táblázatokban* a tulajdonságok leírásait foglaltam össze.

1. táblázat *Properties/Tulajdonságok tartalma*

Type (Button, Text Field, Check Box, Combo Box, List Box)	PDF elem típusa (Nyomógomb, Szövegmező, Jelölődoboz, Kombinált doboz, Lista doboz)
Name	Az mező neve
Tool-Tip	Segédsúgó
Appearance	Megjelenés
Options	A típusra vonatkozó egyedi jellemzők
Action	Esemény hatására bekövetkező művelet
Format	Karakterlánc formátuma
Validate	Engedélyezés
Calculate	Számítási műveletek

Properties/Tulajdonságok

Type – A mező típusának meghatározása. Annak elkészítése után is lehetőségünk van a típus megváltoztatására. Arra figyelni kell, hogy az egyes típusok eltérően engedélyezik a további beállítási lehetőségek használatát. A Text Filed és a Combo Box mindent megenged, míg a Button, Check Box és a List Box csak az Appearance, Options és Action tulajdonságok használatát teszi lehetővé.
Name – A PDF Field neve. Automatikusan felajánl egy sorszámozott nevet, ami tetszés szerint módosítható.
Tool-Tip – A mezőhöz tartozó segédsúgó szövegét írhatjuk ide.

Appearance/Megjelenés

Border – A keret szélességét, színét és stílusát adhatjuk meg. Ha nincs szükségünk keretre, akkor a színt és a szélességet állítsuk ***None*** értékűre. A kiemelkedő és a süllyesztett stílusok térbeli megjelenést próbálnak kölcsönözni a mezőnek.
Other – A mezőbe kerülő tartalomra (karakterláncra) érvényes a felsorolt három tulajdonság.
Visibility – Befolyásolhatjuk a mező láthatóságát, nyomtathatóságát.

Options/Egyedi jellemzők a Button/Nyomógomb típusra

Az ikonok kiválasztása a megnyíló fájlkezelőn keresztül történik. A használható képformátumok a ***tif***, ***png***, ***jpg***, ***xpm***, valamint használható az ***eps Postscript*** formátum is. Méretükre vonatkozólag nincs megszorítás, e jellemzőt az ***Icon Placementben*** finomíthatjuk.
Layout – Az ikon és a gombban látható szöveg elhelyezkedését, egymáshoz való viszonyát rendezhetjük.
Scale – A képet a gomb méretéhez igazítja. A Never választásakor – a lenti vízszintes és függőleges csúszkákkal - a betöltött kép egy területét jelölhetjük meg, ami a gombba fog kerülni.
Scale How – Az aránytartás nélküli átméretezés teljesen kitölti a gombot, míg az aránytartónál előfordulhatnak üresen maradt sávok a széleken.

Options/Egyedi jellemzők a Text Field/Szövegmező típusra

Multi-Line – Többsoros bevitelt tesz lehetővé, használható benne a sortörés (Enter). Ha több a szöveg, mint ami a mezőbe látszólag belefér, akkor

2. táblázat *Appearance/Megjelenés tartalma*

Text - Font for use with PDF 1.3	Az elem által használt betűtípus
Border	Keret
Color	Szín
Width (None, Thin, Normal, Wide)	Szélesség (Nincs, Keskeny, Normál, Széles)
Style (Solid, Dashed, Underline, Beveled, Inset)	Stílus (Egyszerű, Szaggatott, Aláhúzott, Kiemelkedő, Süllyesztett)
Other	Egyéb, a kész PDF-re vonatkozó jellemzők
Read Only	Csak olvasható, a benne szereplő tartalom nem módosítható
Required	A tartalom megadása mindenképpen szükséges
Don't Export Value	Nem exportálható a tartalom
Visibility	Láthatóság
Visible	Látható
Hidden	Rejtett
No Print	Nem nyomtatható
No View	Nem látható, de nyomtatható

3. táblázat *Options/Egyedi jellemzők a Button/Nyomógomb típusra*

Text	Szöveg
Text for Button Down	A gomb lenyomásakor megjelenítendő szöveg
Text for Roll Over	A gomb fölé vitt egérmutatóra megjelenő szöveg
Highlight	A gomb lenyomását kísérő effektus
None	Nincs effektus
Invert	Inverz megjelenés
Outlined	Körvonalazás
Push	„Besüllyesztő”, a Text for... szövegeit ennél a beállításnál jeleníti meg
Icons	Ikonok használata
Normal	Alapesetben látható ikon
Pressed	A lenyomáskor megjelenő ikon
Roll Over	A gomb fölé vitt egérmutatóra megjelenő ikon
Icon Placement	Ikon igazítása
Layout	Elrendezés
Scale (Always, Too small, Too big, Never)	Méretezés (Mindig, Ha kicsi, Ha nagy, Soha)
Scale How (Proportional, Non Proportional)	Méretezés módja (Arányosan, Aránytartás nélkül)

a mező oldalán megjelenik egy gördítősáv. Ezt letilthatjuk a ***Do Not Scroll*** bejelölésével.
Password – A beírt karaktereket csillagozással helyettesíti.

Options/Egyedi jellemzők a Check Box/Jelölődoboz típusra

A jelölés stílusát választhatjuk ki, ami hat különböző jel lehet (***Check*** – Pipa, ***Cross*** – Kereszt, ***Diamond*** – Gyémánt,

4. táblázat *Options/Egyedi jellemzők a Text Field/Szövegmező típusra*

Multi-Line	Többsoros
Password	Jelszóbeviteli mező
Limit of x Characters	A beírható karakterek maximális száma
Do Not Scroll	Görgetés tiltása
Do Not Spell Check	Helyesírás ellenőrzés tiltása

Circle – Kör, ***Star*** – Csillag, ***Square*** – Négyzet). A kijelölést alapértelmezetté is tehetjük a ***Default is Checked*** beállításával.

Options/Egyedi jellemzők a Combo Box/Kombinált doboz és List Box/Lista Doboz típusokra

A dobozok szerkeszthetőségét engedélyezhetjük az ***Editable*** bejelölésével. A két doboz közötti egyik lényeges különbség, hogy a ***Combo Box*** a beírt sorokat egy legördülő menüben tárolja, míg a ***List Box*** a sorokat a mező méretétől függően írja ki, ha kell gördítő sáv segítségével. A másik eltérés, hogy a ***List Box*** csak az ***Appearance***, ***Options*** és ***Action*** tulajdonságok használatát engedélyezi.

Action/Esemény hatására bekövetkező művelet

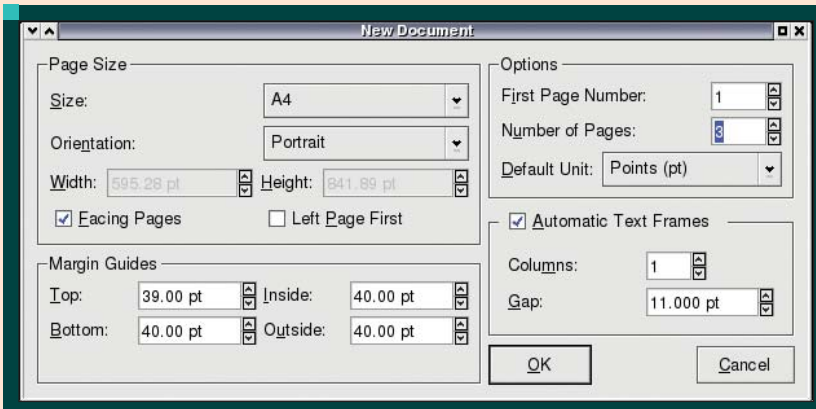
A mezőhöz valamilyen esemény hatására végrehajtandó műveletet kapcsolhatunk. Az események használatához a ***JavaScript*** ismerete szükséges.
A ***Type*** menüből kiválasztva egy újabb menü jelenik meg a választható eseményekkel (***Mouse Down***, ***Mouse Up***, ***Mouse Enter***, ***Mouse Exit***, ***On Focus***, ***On Blur***). Az ***Edit*** ikon egy egyszerű szkriptszerkesztőt rejt, itt lehetőség van külső, már meglévő szkriptek betöltésére is.
A ***Button*** típusnál további események is választhatók:
Go To – A dokumentum (vagy egy külső PDF dokumentum) egy meghatározott részére teszi át a nézetet.
Submit Form – A mezők tartalmát elküldi a megadott URL címre, akár HTML formátumban is.
Reset Form – Törli az összes mező tartalmát.
Import Data – automatikusan egy külső fájlból (***FDF – Forms Data Format***) vett

5. táblázat *Formátumok*

Plain	Szabadon kitölthető
Number	Szám
Percentage	Százalékérték
Date	Dátum
Time	Idő
Custom	Szkripttel vezérelhető

6. táblázat *A Calculate alapműveletei*

Sum	Összegzés
Product	Szorzat
Average	Átlagolás
Minimum	Legkisebb
Maximum	Legnagyobb



1. ábra Az új dokumentum

adatokkal tölti ki a mezőket. Ilyen fájlt a ***Reader Document->Forms->Export Data From Form*** menüjével hozhatunk létre, amit később vissza is tölthetünk a *PDF* fájlba a ***Document->Forms->Import Data To Form*** menüponttal.

Format/Karakterlánc Formátuma

A választható formátumokkal (Field is formatted as) a mezőbe kerülő szöveg beírását szabályozhatjuk.
Plain – szabadon kitölthető mezőt hozunk létre.
Number – Csak számot írhatunk be. Tovább fokozhatjuk a formátum kialakítását a ***Decimals – Tizedes*** rész és a ***Use Currency Symbol – Fizetőeszköz Szimbólum*** tulajdonságok megadásával. A ***Prepend Currency Symbol*** a szám elé helyezi a szimbólumot.
A szám, százalékérték, dátum és idő megadásakor előre beállított sablonokból választhatunk. A szkripttel vezérelhetőnél ***JavaScript*** megírásával „bonyolíthatjuk” a mező kitöltését.
Az alapformátumokat a *5. Táblázatban* foglaltam össze.

Validate/Engedélyezés

A mezőbe kerülő karakterláncot ellenőriztethetjük valamilyen előre meghatározott tartományon belül, vagy ***JavaScript*** segítségével.
Value is not validated – engedélyezés nélküli használat.
Value must be greater than or equal to – az értéktartomány alsó határa, ami még lehetséges érték.
And less or equal to – az értéktartomány felső határa, ami még lehetséges érték.
Custom validate script – szabadon megírt engedélyezési szkript.

Calculate/Számítási Műveletek

A mezőbe egy számítási művelet eredménye kerül, amit az alapműveletekkel vagy ***JavaScript*** használatával kaphatunk meg.
Value is not calculated – számítási művelet nélküli használat.
Value is the … of the following fileds – A mező a felsorolásban résztvevő mezők tartalmán végrehajtott művelet eredményét fogja tartalmazni. Itt a táblázatkezelőkből ismert egyszerűbb műveleteket használhatjuk, *6. Táblázat*. A számításban használatos mezőket a ***Pick*** gomb mögött megbújó ablakból válogathatjuk ki.
Custom calculation script – szabadon megírt számítási művelet.

Gyakorlat

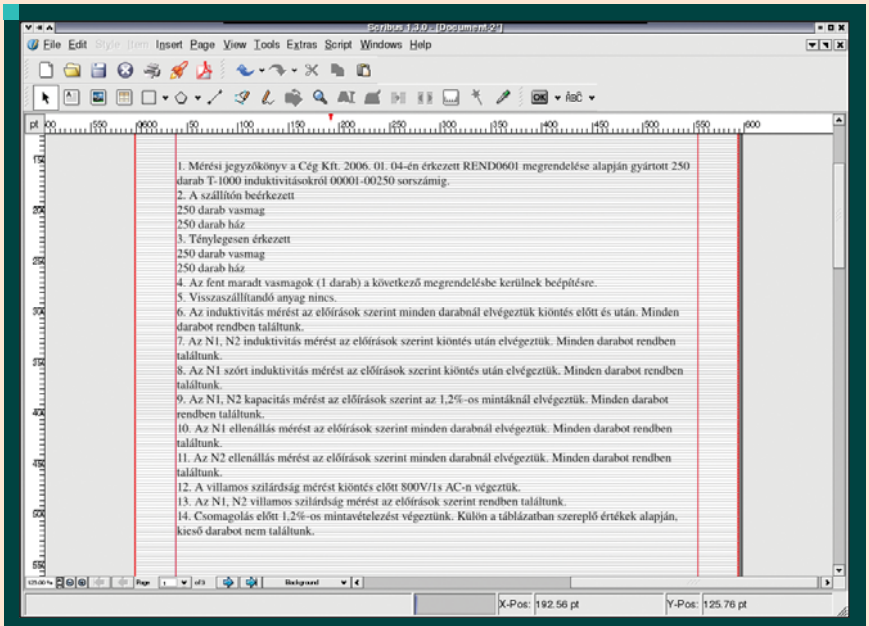
Első lépésként hozzunk létre egy háromoldalas új dokumentumot (***File->New***) egyhasábos tördelésre. A margókat állíthatjuk egységesre is, de célszerűbb a felsőt úgy beállítani, hogy ráessen az alapvonalra. Az *1. ábra* tartalmazza a dokumentum beállításait.
A dokumentumban egységesen a ***Nimbus Roman No9 L Regular*** betűkészletet használom.
Az első oldalra a jegyzőkönyv szövegét helyezzük el, a másodikra egy táblázatot készítünk a mérések eredményeivel, a harmadikra pedig az elkészített alkatrész(ek) rajzát. Az oldalszámozást és az egyéb díszítő elemeket (amennyiben ez utóbbinak szükségét érezzük) a mesterdokumentum elkészítésével (***Edit->Master Pages***), majd annak alkalmazásával illesszük be (***Page->Apply Master Page***).

Az első oldal

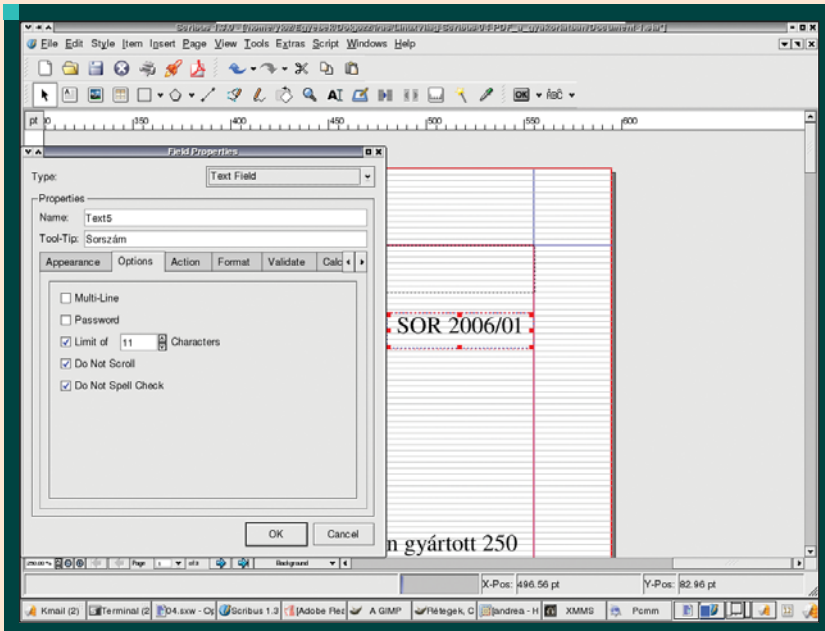
A befolyatott kenyérszövegből (*2. ábra*) a formázást követően minden olyan részt alakítsunk át ***PDF Text Fielddé***, ami jegyzőkönyvenként változik. Ilyen például a dátum, a jegyzőkönyv sorszáma, a megrendelés száma, a rendelt mennyiség, stb. Lesz egy olyan sor is, ahol a ***PDF Combo Box***-ot fogjuk használni, a teljes sor szövegének megváltoztatására.

A sorszám átalakítása

A ***PDF Text Field*** kiválasztása után egy négyszöget rajzolhatunk, ami a szövegmezőt jelképezi, ez a mező ugyanúgy formázható, mint a szövegdoboz, csak kiegészítették a fentebb említett *PDF* funkciókkal. A megrajzolt szövegmezőbe (érdemes valahol a dokumentum felső részén elhelyezni) írjuk be a sorszámot (az így beírt sorszám az ***Acrobat Readerben*** kitörölhető/átírható!) és kattintsunk kétszer a mezőre.
A felugró ablakban (*3. ábra*) a mezőre vonatkozó tulajdonságokat állíthatjuk be. Az ***Appearance*** lapon válasszuk a ***Times*** betűkészletet, mert ez hasonló a kenyérszövegnél használt betűkészlethez. A mezőt körbeölelő keret nem szükséges, ezért a Color és ***Width*** kapjon ***None*** értéket. A sorszám megadása egy ilyen dokumentumnál fontos, így a ***Required*** kapcsolót aktiváljuk. Az ***Options*** lapon korlátozzuk a beírható karakterek számát 11-re, tiltsuk le a görgetést és a helyesírás ellenőrzést.



2. ábra A formázatlan kenyérszöveg

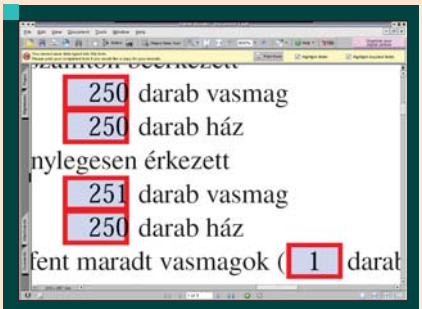


3. ábra Field Properties/Mező Tulajdonságok és a sorszám mező

A ***Format*** lapon maradhat az alapértelmezett ***Plain*** formátum, amennyiben alfanumerikus a sorszám felépítése (és még egyéb karaktereket is tartalmaz). Ugyanezzel az eljárással átalakíthatjuk a dokumentumban található dátumokat, a rendelési számot, a rendelt mennyiséget, típust.

Szövegen belüli átalakítás

A ***PDF Fieldnek*** van egy olyan tulajdonsága, hogy az ***Acrobat Readerben*** beírt szöveget a mező közepére helyezi (vertikálisan). A tökéletes soregyen beállítása ezért lehetetlen, a nagy ékezetes magánhangzók azonnal „leültetik” a mező tartalmát. Amit tehetünk, az annyi, hogy a mező középtengelyét a kenyérszöveg betűinek középtengelyéhez igazítjuk. A *4. ábrán* az ***Acrobat Readerben*** megjelenített dokumentum látható, kiemelve a szövegmezőket (kék) és a kötelezően kitöltendő (***Required***) szövegmezőket (vörös). Ez a kiemelés a ***Reader*** sajátja, a dokumentum megnyitásakor azonnal felajánlja ezt a lehetőséget (***Highlight fields*** és ***Highlight required fields***)



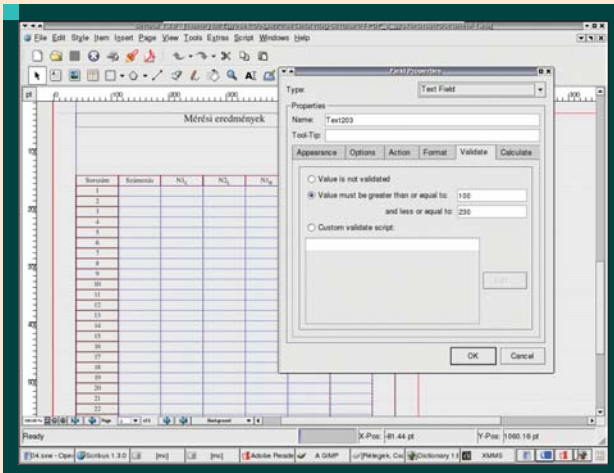
4. ábra Kiemelve és kitöltve az Acrobat Readerben

Combo Box egyszerűen

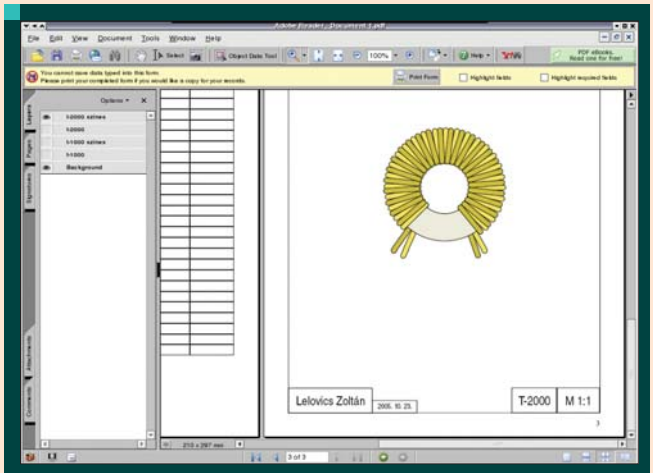
A kenyérszövegben található egy ilyen mondat „Visszaszállítandó anyag nincs”. Ez lenne mindig a legjobb, de néha keletkezik selejt is, amit szintén fel kell tüntetni a jegyzőkönyvben. Ezért ide behelyezünk egy ***Combo Box***-ot amit az esetenként előforduló mondatokkal töltünk ki. Minden egyes mondatot egymás alá írva helyezzük el a dobozban, célszerű a legtöbbet használtat írni az első sorba (1. sor Visszaszállítandó anyag nincs. 2. sor Visszaszállítandó anyag van, vasmag darab, ház darab.). Ezt követően a dobozt méretezzük át úgy, hogy csak egy sort foglaljon el. A ***Combo Box*** ilyen fajta alkalmazása esetén a ***Preflight Verifier***, ***Text overflow*** (szöveg túlfolyás) hibaüzenetet generál, amit nyugodtan hagyjunk figyelmen kívül (***Ignore Errors***).
A ***Field Properties***-ben engedélyezzük a szerkeszthetőséget (***Editable***), valamint a többi szükséges tulajdonságot állítsuk be.

A második oldal

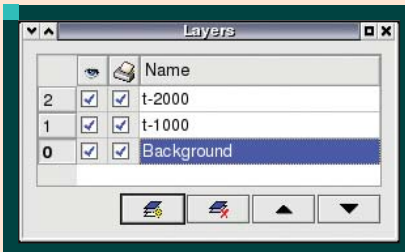
Az ide kerülő táblázat a termék mérési adatait fogja tartalmazni, minden cellát érdemes ellátni a dokumentum pontos kitöltését segítő engedélyezési (***Validate***) értékhatárokkal.
A táblázat címsorát és a sorszámokat tartalmazó oszlopot a ***Scribus*** táblázatkészítő eszközével, a kitöltendő cellákat pedig az ***Item->Multiple Duplicate***-tel sokszorosított ***PDF Text Fielddel*** hozzuk létre. A táblázat mérete: 7 (oszlop) x 31 (sor); az egyes cellák szélessége akkora legyen, mint a szövegtükör hetede, magassága pedig akkora, hogy a kenyérszöveggel azonos méretű karakterek elférjenek benne kényelmesen. A cella tulajdon-



5. ábra Engedélyezett táblázat



7. ábra A kész dokumentum egyik rajza



6. ábra Rétegelés

ságok megadását másolás előtt érdemes elvégezni. Az *5. ábrán* az elkészült táblázat, a megnyitott ablakban a ***Validate*** hangolása látható. Amennyiben a táblázat elemeit külön-külön csoportokba (***Item->Group***) rendezzük és hagyjuk, az így elkészített ***PDF*** fájl, a kitöltéskor e csoportokat figyelembe veszi. Érdemes kipróbálni egy és két oszlopos csoportokra.

A harmadik oldal

Két alkatrészt ábrázoló rajzot helyezünk el az oldalra, négy különálló rétegre (kettőn fekete-fehérben, kettőn pedig színesben). A réteget kezelő eszközt a ***Tools->Layers*** almenüben találjuk meg. A *6. ábrán* a már hozzáadott négy réteget (T-1000 és T-2000, valamint színes megfelelőik) és az alapréteget (***Background***) látjuk.
A felsorolás elején látható szem ikon a réteg láthatóságát, a nyomtató ikon a nyomtathatóságát kapcsolja ki-be. A szerkeszteni kívánt rétegre a sorszám vagy sor kijelölésével léphetünk. Az alsó ikonsoron (balról-jobbra) haladva:
Add a new layer – Új réteg létrehozása. Az új réteg mindig legfelülre kerül.
Delete layer – A kiválasztott réteg törlése.
Raise layer – Egy szinttel feljebb helyezi a réteget.
Lower layer – Egy szinttel lejjebb helyezi a réteget.
Elsőként a szövegmezőket készítsük el, majd rendezzük csoportba és készítsünk másolatokat, amiket a rétegek ugyanazon részére (például alulra) helyezzünk el.
A rajzok elkészítésénél a ***Shape*** eszköz beépített alakzatait használjuk. Az ilyen egyszerűbb rajzokat, mint ez a toroid tekercs, külső rajzolóprogram nélkül is könnyen elkészíthetjük. Az alakzatok között található egy gyűrű, ami már jó alapot nyújt a munkához. A tekercs drótozását, a négyszög alakzatból készítettem el.
A ***Properties*** ablak ***Shape*** lapján a ***Round Corners*** értékét állítva a négyszög sarkait a megfelelő mértékűre kerekíthetjük. A többi már csak egyszerű másolgatás (***Item->Duplicate***) és elforgatás (***Properties*** ablak, X,Y,Z lap, ***Rotation***). A tekercs kivezetéseit a drótozás megnyújtásával, a takarásba kerülését az X,Y,Z lap Level szintjének csökkentésével értem el.
Elkészült rajzunkat az utókor számára megőrizendő elmenthetjük a ***Scrapbookba (Item->Send to Scrapbook)***, így onnan bármikor elővehetjük újabb felhasználásra.
A *7. ábrán* az egyik – kiszínezett – toroid látható az ***Acrobat Readerben***. A kép baloldalán látható a rétegek listája, amik közül kiválaszthatjuk az éppen aktuálisat a nyomtatáshoz.

PDF írása

Az eddigiek és az előző részben leírtak után ez már nem okozhat gondot. Az írás megkezdése előtt a kötelező hibakeresés – esetünkben – talál egy hibát (***Text overflow***), ami szándékosan került a dokumentumba. Nem foglalkozva ezzel (***Ignore Errors***) jutunk el a ***Save as PDF*** ablakra. Amire fontos figyelni, hogy a rétegeket csak a ***PDF 1.5***-ös verzió támogatja. A rétegek tartalma bekerül az ettől eltérő verziójú ***PDF*** fájlba is, csak nem lehet válogatni közöttük, ki-be kapcsolgatni. Ezért, ha ***PDF 1.4***-es fájlt készítünk, akkor csak egy réteg láthatósága legyen bekapcsolva, különben az összeset egymásra pakolva készíti el a ***PDF***-et.
Nagyon sok lehetőség rejlik egy ***PDF*** fájlban, főleg, ha valaki ért a ***JavaScripthez*** is. Így a határok összemosódnak és a kiadványszerkesztőnek is lassan értenie kell a programozáshoz, szkriptíráshoz.
A következő részben visszatérünk a ***Preferences*** menühöz és annak bemutatásához, valamint belenézünk a ***Story Editorba***, ami a nagyobb munkák gyorsabb szerkesztését segíti elő.

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

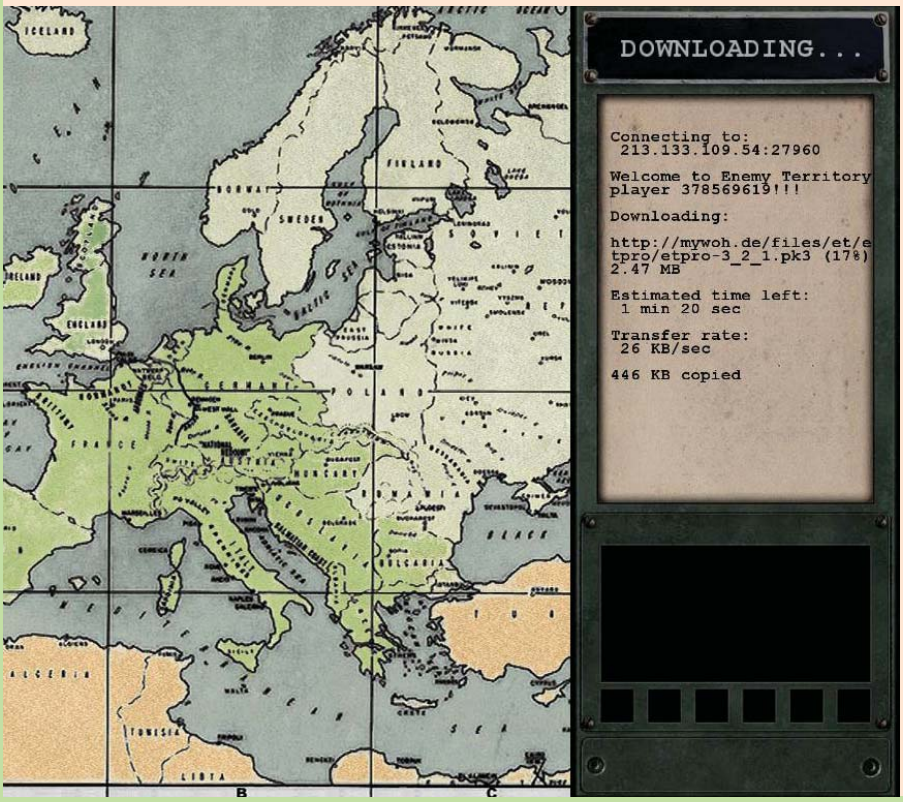
Enemy Territory

Ajánlom ezt a cikket azoknak akik egy hosszú munkanap után nem „gondolkodós” játékokkal, hanem inkább egy pergő cselekményű akció játékkal szeretnének kikapcsolódni, vagy csak egyszerűen szeretik az FPS-eket.

Az *Enemy Territory* a *Castle of Wolfenstein* című *FPS* játék ingyenesen letölthető verziója. A játéknak – azon kívül, hogy Linuxon is bámulatosan fut – az egyik legnagyobb előnye, hogy regisztrációs- és egyéb díjjak nélkül szabadon letölthető és játszható. Amikor a *Castle of Wolfenstein* megjelent, megnéztem a demó változatot és nekem nagyon nem nyerte el a tetszésemet. A megfelelő tűzerő birtokában egészen nyugodtan besétálhattunk bármelyik szobába és végezhettünk az ellennel úgy, hogy mindeközben bambán toporog és néha visszalő. Sajnos ez a stratégia a játék végéig működött, ahol a játék annyi változatosságot produkált, hogy nagyobb fegyver arzenálra volt szükségünk a főgonosz levadászásához. Az *Enemy Territory* fejlesztői is valószínűleg ráéreztek erre a problémára és a játékból kidobták az egyjátékos üzemmódot cserébe beletettek egy igen erős többjátékos lehetőséget. Egyszerre akár 100 ember is vadászhat egyszerre az ellenségre (akik szintén játékosok) a pályán.

Beszerzés és telepítés

Látogassunk el a ➲ *http://wolf.hcgamer.hu/* oldalra és szerezzük be a szükséges programokat. Először a *„Wolf: Enemy Territory 2.55 Linux”*-ra lesz szükségünk ami nagyjából 259 Mbyte terjedelmű. Ha ezt letöltöttük akkor keressük meg a 2.56-os és 2.60-as patch-et a játékhoz. Ez nagyon fontos, mert a legtöbb szerver nem fog beengedni minket ha nincs feltelepítve legalább a 2.56-os folt!
Ha sikeresen birtokunkba vettük a fent említett három állományt akkor telepítsük őket a gépünkre. Ehhez nem kell mást tennünk mint kitömöríteni a játékot tartalmazó 260 Mbyte-os fájlt egy számunkra szimpatikus könyvtárba, mellé másolni a két foltot majd konzolról kiadni az alábbi utasításokat:



Alapjáték telepítéséhez:

```
sh et-linux-2.55-x86.run
```

2.55-ös patch telepítéséhez:

```
sh et-linux-2.56.x86-update.run
```

2.60-as patch telepítéséhez:

```
sh et-linux-2.60.x86-update.run
```

A játék telepítője grafikus felületű, és azon kívül, hogy elfogadjuk-e az *idSoftware* szerződésének feltételeit valamint, hogy hova szeretnénk telepíteni a programot, nem fog mást kérdezni. Ez a telepítési metódus igaz a két foltra is.

Gépigény

A 2-3 ezer MHz -s gépek korában igen szerény gépigénnyel is elboldogul a program.

- Pentium III 600 Mhz vagy ennek megfelelő egyéb x86 architektúra

- 128 Mbyte RAM
- 32 Mbyte-os OpenGL-t támogató videokártya
- 56.6k modem

Ezek manapság nem hiszem, hogy teljesíthetetlen feltételek volnának. Fontos, hogy a videokártyánkhoz a megfelelő meghajtó program mindenképpen legyen telepítve, különben a program el sem fog indulni. Az ***ATI*** és ***nVidia*** kártyák telepítéséről kiváló cikk jelent meg egy előző számban!

Indulás a harcmezőre!

Ha az alapbeállításokat hagytuk a telepítőben akkor a */usr/share/games/enemy-territory* mappában bukkanhatunk rá a futtatható bináris állományra. Indítsuk el! Pár másodperces töltögetés után megjelenik az ***idSoftware*** logója, majd a program kér, hogy egy profilt hozzunk létre magunknak. A profil nevének a becenevünket adjuk meg és valahogy jelezzük, hogy melyik országból játszunk (például *[HUN]Andras007*). Természetesen ez nem kötelező, de ha vissza-vissza térünk egy szerverre ott előbb-utóbb úgyis meg fogják kérdezni, illetve néhány magyar szerverre csak magyarok léphetnek be. A profilban állítsuk be a képernyő felbontást (***resolution***) és a szín mélységet (***depth***), a sávszélességünket (***bandwidth***) és engedélyezzük a ***PunkBuster*** használatát (***enable PunkBuster***)! A ***PB*** egy olyan rendszer ami folyamatosan szűri a csalásokat a játékos részéről. Fair play. Mindenki használja, senki sem csal. Ha valaki mégis és ez kiderül akkor a legtöbb szerverről kitiltják. Van néhány dolog amiért kizárhatnak a szerverről, ezeket a „játékszabályokat" illik betartani:

- Kérdezzük meg játék előtt, hogy csatlakozhatunk-e. Egyes klán szervereken nem szeretik ha „külsősök" vannak.
- Ha kezdők vagyunk akkor legalább egy fél mondatot szóljunk arról, hogy most játszunk először. Ez néha megmenthet egy kirúgástól.
- Ne káromkodjunk chat-en.
- Ne lövöldözzük le a társainkat. Ha véletlen volt akkor azonnal kérjünk bocsánatot!
- Amire minden szerveren nagyon gyűlölnek az, ha az ellenfél „feltámadási pontján" (ahol elhalálozás esetén újrakezdünk) tábort verünk és az éppen újrakezdő játékosokat galád módon lelövöldözzük. Ez nem tisztességes, mert az ellenfélnek esélye sem lesz megmozdulni.

Most már tisztában vagyunk a szabályokkal, van profilunk is. Irány a harctér! Kattintsunk a ***Play Online*** gombra és várjunk türelemmel amíg a rendelkezésre álló szerverek listája betöltődik. A szerver kiválasztásánál vegyük figyelembe a szerver és a gépünk közti kommunikáció sebességét, (a ping minél kisebb annál jobb) hogy hány játékos tartózkodik a pályán és, hogy nincs-e lezárva a pálya. Ha le van zárva akkor egy kis lakat ikont láthatunk abban a sorban. Ebben az esetben csak jelszóval tudunk belépni. Válasszunk ki egy szimpatikus szervert és nyomjuk meg a ***Join Server*** gombot.
Ha mindent jól csináltunk akkor egy 1942 körüli ***Európa*** térképet kell látnunk, jobb oldalon pedig egy sávot ahol számunkra mindenféle fontos információkra lelhetünk. A játékhoz való csatlakozás sebessége sok mindentől függ. Sokszor megesik, hogy egy szerverre való belépéskor kiderül, hogy a játék egyik alap pályájának a módosított verzióját használják az adott szerveren. Ekkor jobb oldalon egy kis üzenet tájékoztat bennünket arról, hogy az új pálya letöltése folyamatban van, illetve arról, hogy még mennyi idő a letöltés végéig. Sokszor új hangokat, textúrákat is tölt le a program. Ebben az esetben több fájl is letöltődhet. A legrosszabb esetben is maximum 10 percet kell várnunk valami gigantikus méretű pályára és a hozzá tartozó kiegészítőkre.
Nem tudom, hogy melyik szerveren találkoztam a ***StarWarsMOD***-dal, de nagyon tetszett. A 6 részes kampány alatt különböző – a ***Csillagok Háborújából*** jól ismert – helyszíneken folyt a háború, ráadásul lépegetők is voltak...
Ha sikeresen letöltődött a kiegészítő akkor egy új képernyő tárul a szemünk elé. Egy térképet kell látnunk a képernyő nagy részén, rajta mindenféle zászlókkal és ikonokkal.
Ez az éppen aktuális csatatér térképe amelyen megnézhetjük, hogy melyik félnek hol vannak bázisai, elfoglalt épületei, vagy ha ***Objective*** típusú játékot játszunk (az adott pályán nem csak az ellenfél lepuffogtatása a cél, hanem egy adott objektum megvédése, vagy a másik szemszögből: elfoglalása) akkor a célpontok is láthatóak a térképen. A képernyő jobb oldalán két zászló ikont láthatunk, ezek közül kell kiválasztanunk, hogy melyik félhez kívánunk csatlakozni,

az alatta lévő ikonokkal pedig azt választhatjuk ki, hogy a csapatban milyen pozíciót akarunk betölteni (katona, orvos, mérnök, kém, mesterlövész). Mindegyik típusú karakternek más képességei és felszerelése van.
Aki a rohangálós, lövöldözős játékot szereti az válassza a katonát.
Aki inkább csendben meglapulva, perceken keresztül a sárban hasalva szeretné levadászni az ellent, az válassza a mesterlövészt.
A mérnök épületek és járművek felrobbantásában, illetve egy terület elaknásításában játszik nagy szerepet.
Az orvos és a kém kezdő játékosoknak nem ajánlott, mert ők teljesen más játékstílust kívánnak. Az orvos a csapatot kísérgeti és gyógyító csomagokat hajigálva segíti társait. Ha valaki meghal és a közelben van akkor adrenalin injekcióval visszahozhatja társát.
A kém egy levadászott ellenséges katona ruhájába búvja, csapdába csalhatja az ellenfeleit, a másik csapat játékosának adva ki magát.
Ha kiválasztottuk a katonánkat, belépnetünk az igazi játékba. Kezdő játékosok próbáljanak meg a játék elején együtt mozogni a csapattársaikkal. Ha az egész csapat egyszerre veti hasra magát akkor annak valószínűleg oka van. Például egy szőnyegbombázás.

True Combat

A ***True Combat*** egy kiegészítő amelyet az ***Enemy Territory***-hoz készítettek.
A játék teljesen más hangulatot ad a játéknak, mert az 1940-es évekből áthelyezi a játékot napjainkba. A ***TC***-ben nem világháborús katonákat, hanem terroristákkal harcoló kommandósokat személyesíthetünk meg. A beszerzés és a telepítés teljesen megegyezik az ***Enemy Territory***-val. Látogassunk el a ➲ ***http://www.truecombat.co.uk*** oldalra és töltsük le a legfrissebb telepítőt. Verziószámot nem írok, mert a játék nagyon gyorsan fejlődik és napról-napra adják ki az újabb verziószámú kiegészítőket.
A ***True Combat*** telepítéséhez szükségünk lesz egy 2.60 verziószámú ***Enemy Territory***-ra is!
A ***True Combat*** sokkal reálisabb mint a világháborús testvére. Itt egy lövéstől meghalhatunk, vagy egy rossz találattól pillanatok alatt elvérezhetünk, ráadásul itt nem lesz a segítségünkre orvos, mert a gyors utcai harcoknál nem kísérgeti a kommandósokat felcser.
A fegyverek kezelése is nagyon sokat fog változni. Nincs célkereszt, a fegyverekkel meg kell tanulnunk célozni, és harcosunk keze is remegni fog a hosszas célra tartás közben.
Mindenkinek sok sikert kívánok az utcai harcokhoz és a fronton egyaránt!

Tóth Péter
(thotacc@drotnet.hu)
A BMF hallgatója vagyok, mellette egy kis- és középvállalatok informatikai rendszereinek Linuxos átállításával és szoftverfejlesztéssel foglalkozó cég informatikai vezetőjeként tevékenykedem. Kevés szabadidőmet barátnőmmel és barátaimmal töltöm egy-két sör társaságában.

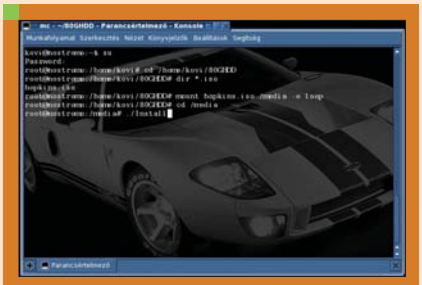
Hopkins FBI

Ebben a hónapban az eddig kevéssé favorizált „Point'n Click" játékstílus egy jelentős képviselőjét fogom bemutatni. Természetesen a kiszemelt program natívan fut Linux alatt.

A bevezetőben említett kategória igen nagy hagyományokkal rendelkezik. Ez a terület egykor elsősorban a ***LucasArts*** csapatának volt a specialitása: valószínűleg akad jó néhány olvasó, akinek (a példa kedvéért) ismerősen cseng a ***Day of Tenticle***, vagy éppen a ***Full Throttle*** nevek valamelyike. Ennek ellenére az idők folyamán a műfaj csaknem feledésbe merült, igazán komoly próbálkozások mára nagyon ritkán látnak napvilágot, arról nem is beszélve, hogy ennek a kevéske játéknak is csak apró töredéke az, amely „***Linux***-barát".
A ***„Point'n Click"*** kalandok világa rendkívül szűk rajongói tábort szolgál ki, mivel ezen a téren a játékosok rátermettsége és türelme egyaránt elvárt „erény", ezek hiányában itt senkinek sem terem babér. De miről is van szó tulajdonképpen?
Arról, hogy az aktuális terepen, adott karakterünkkel a megtalálható tárgyakat be kell gyűjtenünk, és ezek felhasználásával, a fejtörőkön keresztül, esetleg az adott helyzetben szükséges „jó irányú" párbeszéddel túl kell jutnunk az adott helyzetből. Mivel e stílus nem a csillogó küllemre helyezi a hangsúlyt, a grafikusok ezért igen sokszor mellőzik a modellező és animáló segédprogramok használatát játékuk elkészítéséhez. Ezen a ponton érdemes megfigyelni azt is, ahogyan a modern művészet iránti szeretetet feltétel nélkül hálálják meg a műfaj fejlesztői: sok program ugyanis szabadkézi rajzokból felépített, képregényhez hasonló képi világgal rendelkezik.

Hopkins FBI

A projekt az előbb említett stílusjegyek egyik iskolapéldája: ebben az esetben is színtiszta, kézzel rajzolt művészetről van szó, egyedi zenei aláfestéssel fűszerezve. A ***Hopkins FBI*** középszerű kerettörténettel rendelkezik: ***Bernie Berckson*** terroristavezér nyomára kell bukkanunk, akit nukleáris terrortámadása miatt ugyan halálra ítéltek, de kivégzése közben rejtélyes módon megszökött, így újabb fenyegetést jelent a világra. A történetet az animált bevezető természetesen bővebben kifejti, melyet mindenképpen érdemes végignézni, ha másért nem, a rajzfilmszerű videóbetét miatt, melyhez remekül passzol a szinkronhang ide vonatkozó elbeszélése is.
A bevezető után beléphetünk a játékba, ahol karakterünk irányítása a megszokott módon történhet: a helyszín



1. ábra A Hopkins FBI telepítése, lemeztükörről

szabad pozíciójába kattintva ***Hopkins*** ügynök a kívánt területre sétál, majd a célterületen eszközölt jobb egérgomb többszöri megnyomásával ki kell választanunk az elérhető feladatait (vizsgálat, keresés, mozdítás, a tárgy felvétele). A „kitallózott" tevékenységen bal gombot nyomva aktivizálódik az adott parancs. Bármely szobában,

2. ábra A bevezető egyik jelenete

3. ábra Az asztalon van egy csavarhúzó. Vegyük fel!

4. ábra A pisztolyunkra is szükség lehet...

bármely helyszínen lehetőségek egész tárháza adódik a felhasználó előtt, egyik megoldás pedig a következőnek igen sokszor a kulcsát jelenti: ebben az összetettségben rejlik a játék sava-borsa. Fontos lehet, hogy a ***Hopkins FBI*** világa nem szűkölködik a véres jelenetekben, így csak bizonyos korhatár betartásával érdemes foglalkozni vele.
A fejlesztői munkák az ***MP Entertainment*** csapatához fűződnek. Ezen a ponton meg kell említenem, hogy a program ***1998***-ban lett publikálva, így képességei valamivel szerényebbek, mint egy mai megvalósításé.
Telepítése szerencsére kellően egyszerű: a ***CD*** lemezt rendszerünkbe fűzve, a gyökerében található ***Install*** állományt el kell indítanunk rootként, egy terminálon. A folyamat szöveges felületen fog lezajlani, néhány kérdés kíséretében, melyekből kettő kitüntetett figyelmet érdemel: egyikük a telepítés módjára, másikuk az ***SVGA*** grafikák felmásolásának szükségére vonatkozik. Előbbire válaszoljunk a „***Full Install***" opcióval (ekkor nagyjából ***100MByte*** helyigénnyel kell számolnunk), utóbbira pedig igennel (további ***350MByte*** igény).
A program alapértelmezés szerint az ***/usr/local/hopkinsfbi*** útra települ, kötéssel az ***/usr/bin*** mappában. Figyelem! Az indító Hopkins_FBI link a médián található egyik állományra mutat, így mindaddig halott linkként kell majd számolnunk vele, míg a telepítő lemez tartalma nem látszik a szükséges elérési úton. A médiát (egyebek közt emiatt is) játék idejére mindig be kell fűznünk arra az útra, amelyről telepítettünk, éppen ezért a lemezemről készítettem egy ***iso*** lenyomatot, majd a lemezkép hurok rendszerű csatolása után erről indítottam már az installálást is (mindez könnyen nyomon követhető az első képen).
Ezek után a játékot indító szkriptet úgy módosítottam, hogy az ***iso*** állomány csatolásával kezdje a műveleteit. Így „megspóroltam" a lemez előbb említett állandó berakását / kivételét, és az optikai meghajtók szerény hozzáférési idejét is elcseréltem a merevlemezem kedvezőbb értékeire.
A ***Hopkins FBI*** sajnos csak a játékállás mentéseit tárolja el személyes mappánkban, így a képfelbontás beállítását a fő elérési úton található ***config.ini*** fájlban lehet megtenni (fontos, hogy tapasztalatom szerint teljes képernyőn nem használható kellő biztonsággal, így a mellékelt képek ablakos futtatásban készültek). A projekt ***Simple Directmedia Layer*** felületre támaszkodva fut ***Linuxon***, de a szükséges ***API*** állományait (***0.9.9*** verzióban) a játék megfelelő ***/SDLLIB*** mappája tartalmazza.
Hardverigénye meglehetősen szerény, tapasztalatom szerint már ***300MHz*** órajelű ***x86*** processzoron (mely akár egy gyenge ***Celeron*** is lehet) kompromisszumoktól mentesen fut. A teljes program beszerzése előtt érdemes lehet kipróbálni natív, linuxos demóját, mely a hivatalos honlapjáról, a ➲ ***http://www.hopkinsfbi.com*** címről elindulva szerezhető be. Ezen felül kifizetődő a játékkal foglalkozó ➲ ***http://www.littleigloo.org/hopkins.html*** oldalt is figyelemmel kísérni, ahol ez a letöltés szintén elérhető. A végigjátszáshoz szükséges segítség pedig a ➲ ***http://www.adventuregames.hu/modules.php?name=Content&pa=showpage&pid=142*** URL mögött található.

Összegzés

Őszintén leszek: nem vagyok megrögzött ***„Point'n Click"*** fanatikus. Ennek ellenére tisztán látom, hogy a kategória rendelkezik valamilyen „megfoghatatlan" hangulati elemmel, ami a régmúlt időkből táplálkozik. Így biztos vagyok benne, hogy akiben akár minimális elhivatottság is lakozik e kalandjátékhoz, az nagyon meg lesz elégedve vele.
Nem beszélve arról, hogy ***Hopkins*** ügynökünk kalandjai mellett más kihívást is választhatnak a műfaj szerelmesei. Célirányosan keresve érdekes, platformfüggetlen projektek is találhatóak a világhálón (például a ***Soviet Unterzögersdorf***) melyek jelenlegi állapotukban ugyan nem képesek konkurálni a „nagyokkal", de fejlődésük miatt kitüntetett figyelmet érdemelnek.
Azoknak pedig, akik ragaszkodnak a kidolgozott, effektekben dús ***3D*** képi világhoz, hadd ajánljam figyelmükbe az ***Agatha Christie*** egyik regényéből ötletet merítő ***And Then There Were None*** című alkotást. Utóbbi játék ***Cedega wrapper*** segítségével kelthető életre ***Linuxon*** (étvágya sajnos ***2GHz*** órajelű teljes értékű processzornál kezdődik).
Mindenkinek tartalmas kikapcsolódást kívánok.

Kovács Zsolt (kovi@linuxforum.hu)

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