# Platformfüggetlen szoftverfejlesztés – A Lazarus

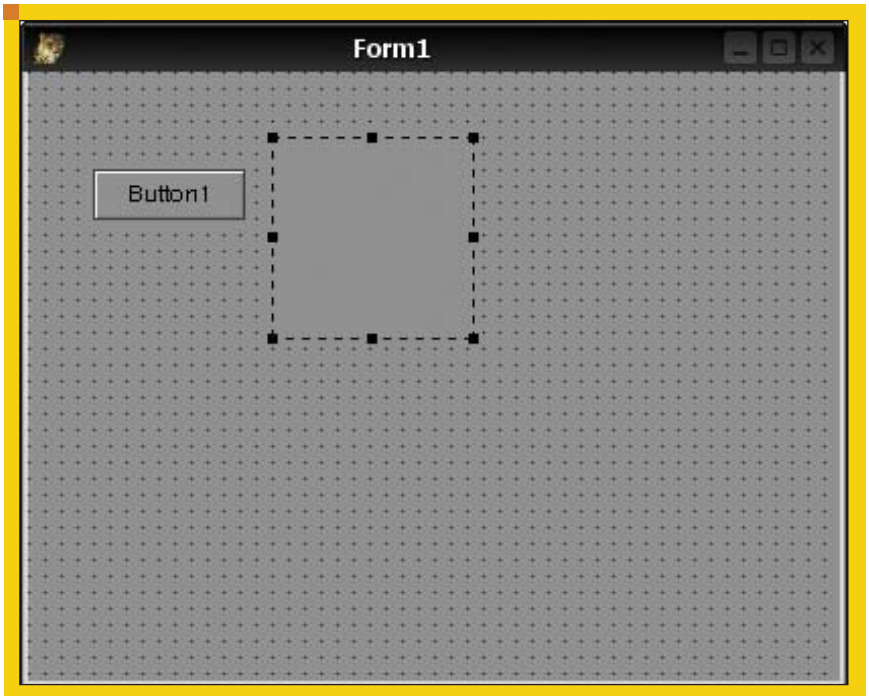
Az utóbbi években a Linux egyre inkább kezdi kinőni azokat a „gyermekbetegségeit” amelyek miatt nem volt alkalmas otthoni felhasználásra. Most, hogy egyre több helyen jelenik meg a Linux és már nem csak kiszolgáló szerepét tölti be elérkezett az ideje, hogy a programozók célplatformnak tekintsék és a 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?

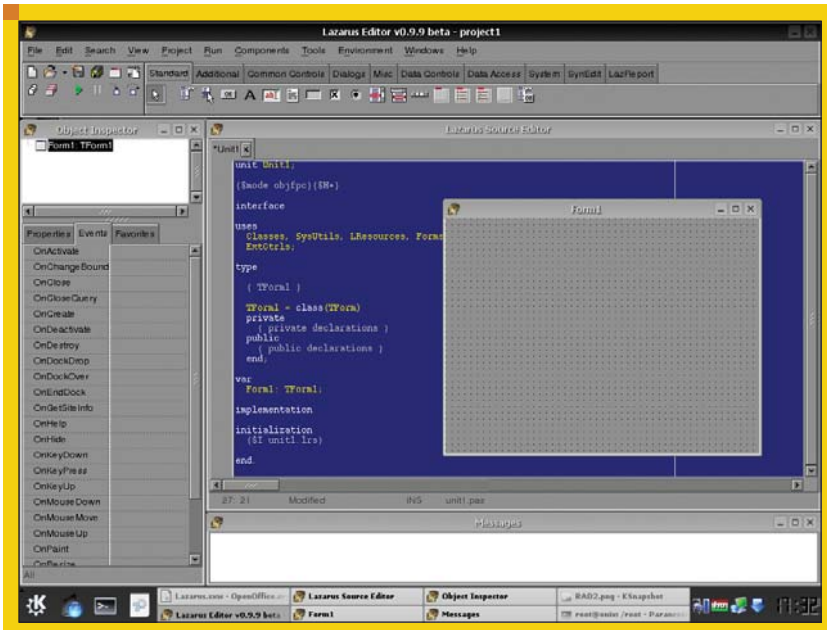
A *3G* nyelvek napjaink egyre gyorsabban terjedő programozási rendszerei melyek objektum-orientált szemléletük és eseményvezérelt programozásuk miatt napról-napra nagyobb teret hódítanak maguknak. Az eseményvezérlés nagyon megkönnyíti a programozó munkáját és rengetek monoton, felesleges programozással 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

***tulajdonság***okkal (***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*.

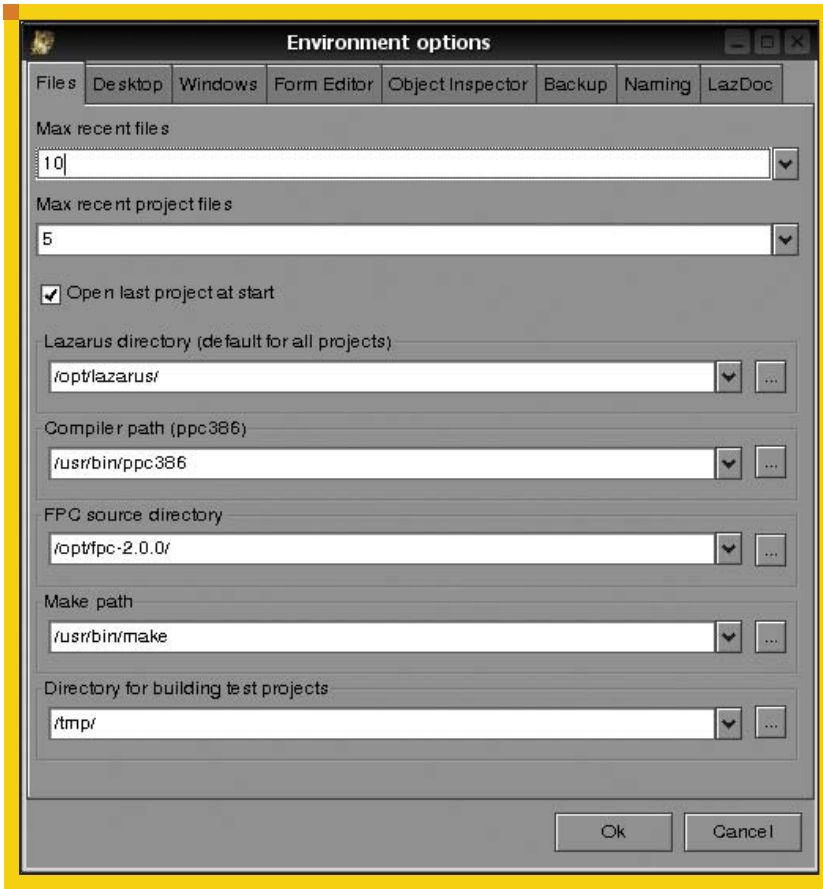
## Ismerkedés

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:

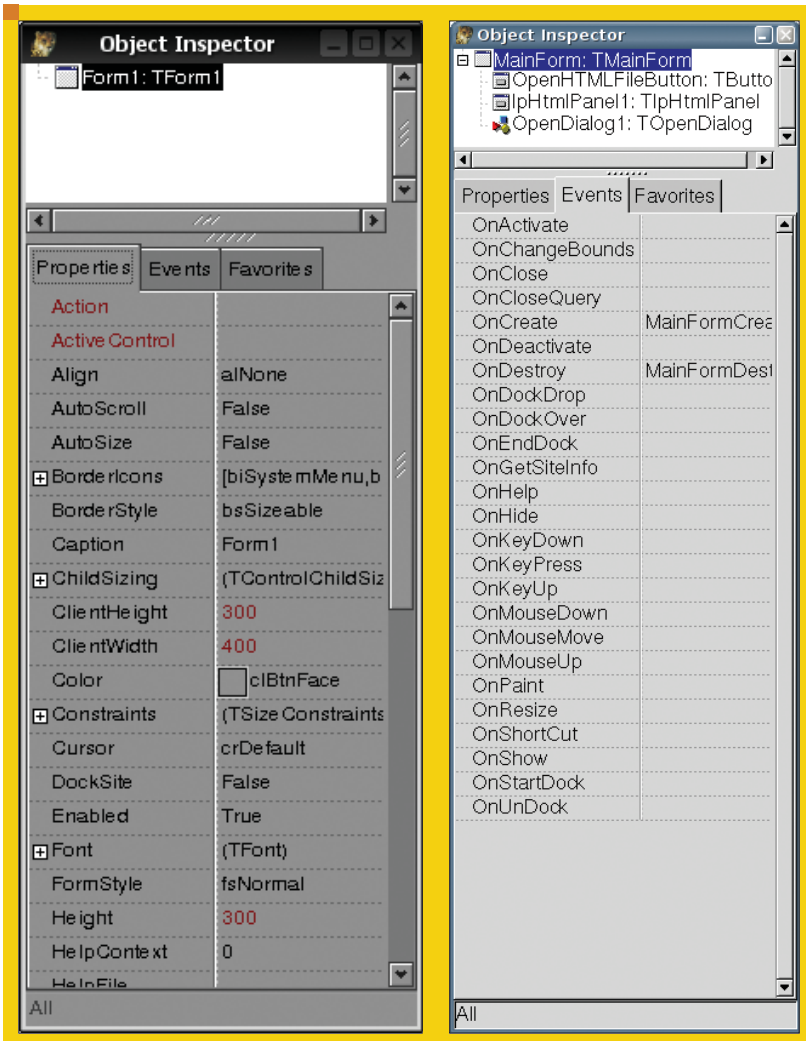
- *Standard*: Gomb, képek, jelölő négyzet, stb.
- *Additional*: Keretek, táblázatok, „rajztábla”, stb.
- *Common*: Fülekkel ellátott oldalak, fa megjelenítések, stb.
- *Dialogs*: Nyomtatási-, Fájl megnyitási és mentési ablakok.
- *Misc*: Naptár, színváltó.
- *Data Controls*: Adatbázisokhoz adatmegjelenítési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