© Kiskapu Kft. Minden jog fenntartva

Kylix

Hatékony alkalmazásfejlesztés Linuxon

A Borland cég Kylix rendszere-a nagy sikerű Delphi Linuxra átírt változata. Anyanyelve az *object pascal*, amely a Pascal objektumközpontú továbbfejlesztett változata. A nyelv formai követelményeinek köszönhetően nagyon jól használható összetett objektumszerkezetek kezelésére (például objektumok többszintű egymásba ágyazására), és ugyanezen okból könnyű ezen jól áttekinthető programkódot írni. A rendszer a QT grafikus könyvtár alapjaira épült. Mind a fejlesztőeszköz, mind az általa fejlesztett programok szervesen illeszkednek a KDE ablakkezelő-rendszer felületébe, de természetesen nem csak KDE alatt futtathatók. A rendszernek három változata kapható. Az ügyféloldali alkalmazásfejlesztőknek szánt Desktop Developer, a kiszolgálófejlesztők számára kibocsátott Server Developer, és a nyílt forráskódra fejlesztők számára kiadott ingyenesen elérhető Open Edition. Ez utóbbi megjelenése e cikk írásának pillanatától számítva heteken belül várható.

Telepítés

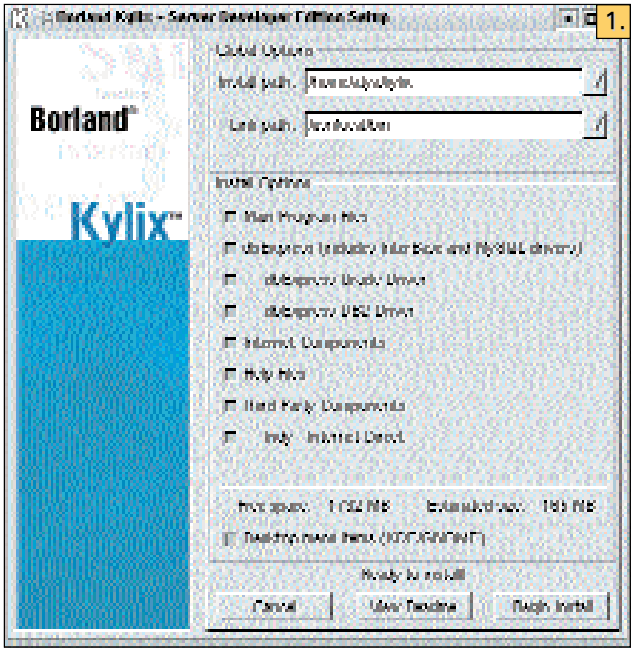
A Kylix viszonylag szerény erőforrásigényű. Futtatásához elegendő egy Pentium II 400 MHz-es processzorral rendelkező számítógép, 128 MB memóriával. Elvileg ennél gyengébb rendszeren is fut, de tapasztalatom szerint a hatékony fejlesztéshez a megadott kiépítés szükséges. A fejlesztők által ajánlott operációs rendszerek:

- SuSE 7.0 vagy későbbi kiadás,
- RedHat 6.2 vagy frissebb változat,
- Mandrake 7.2 vagy nagyobb változatszámú terjesztés.

Ez a gyakorlatban azt jelenti, hogy minden terjesztés használható, amely 2.2-es vagy újabb libc-t tartalmaz és rendelkezik grafikus felülettel. Én Debian Woodyra telepítettem, de kisebb hibákat tapasztaltam. Gyakorlatlan Linux-felhasználóknak az ajánlott terjesztések egyikét javaslom. A Kylix Server Developer dobozos termék tartalmazza a SuSE Linux 7.0-s változatát, így szükség esetén az operációs rendszer is telepíthető. A telepítőrendszer nagyon kényelmes grafikus felületet ad (*1. kép*), ahol ki lehet választani, hogy a rendszer mely részeit szeretnénk telepíteni. Egyetlen dolgot nem értek. Miért nem használja a Linux által adott csomagkezelőt, holott annak éppen ez lenne a feladata? A telepítésnél meg kell adnunk egy könyvtárat, amelybe a Kylix kerül (teljes telepítés esetén nagyjából kétszáz megabájtnyi hely szükséges). Ebben a könyvtárban a Kylix futási környezetet alakít ki magának saját függvénykönyvtárakkal.
Ez a módszer linuxos szemmel furcsának tűnhet, de megvan a maga haszna. A Kylix túléli a nagyobb programkönyvtárcseréket is, mivel a saját könyvtárait használja, viszont a vele fordított programoknak a valós rendszeren kisebb-nagyobb gondjai lesznek. Kellemetlen, hogy az üzembiztos Debian-rendszerben jelenleg nincs olyan csomag, amely tartalmazza a libqtinf.so-t. Mivel a fejlesztett program futtathatóságához erre a függvénykönyvtárra szükség van, kénytelen voltam a Kylix könyvtárából kimásolni a /usr/local/lib könyvtárba.

A fejlesztőrendszer

A Borland szokásához híven most is elegendő leírással látta el fejlesztőit. A doboz az alábbi három angol nyelvű könyvet tartalmazza: Developers Guide, Quick Start és Language Guide, amelyek a CD-n PDF formátumban is megtalálhatók. Véleményem szerint ezeknek, valamint a Súgórendszernek (*2. kép)* és az elemek forráskódjának birtokában bármely fejlesztés közben felmerülő gond orvosolható. Nálam a rendszer első indításakor egy *Generating font matrix. Please wait…* feliratú párbeszédablak jelent meg és valami megette a teljes processzoridőt. Egy darabig bírtam cérnával, aztán mivel úgyis fontos söröznivalóm volt, otthagytam. Mivel visszatértemkor még mindig dolgozott, kezdtem gyanút fogni… Az strace segítségével hamar kiderült, hogy végtelen ciklusban kering, így lezártam az ablakot.
A hiba okát feltehetőleg a saját beszerzésű betűkészletek jelentették. Az ablak lezárása után a Kylix hibátlanul elindult. Azóta telepítettem

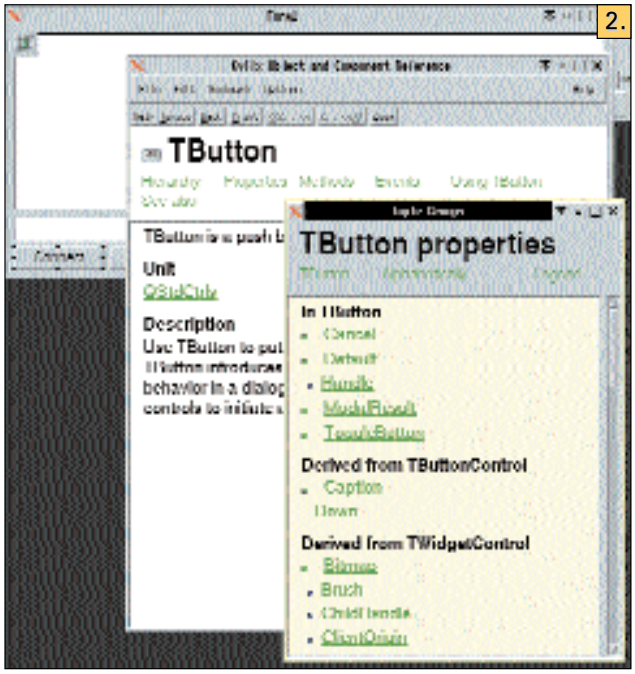


már SuSE 7.1-re és Debian Potato alaprendszerre is. Egyik alatt sem jelentkezett az előbbi hiba, futásuk között pedig látszólag semmi különbség sincs. Egy dologra viszont a fenti gond felhívta a figyelmet. A Kylix ugyan valódi Linux alatt futó program, de bizonyos Win32-höz kapcsolódó szolgáltatások Windowsról való átemelésére a Borland a Wine-t használta. A Kylix általános sebessége megfelelő (egyedül a Súgórendszer lassú kissé), az általa fordított programoknak pedig egyáltalán nincs szükségük a Wine-ra.

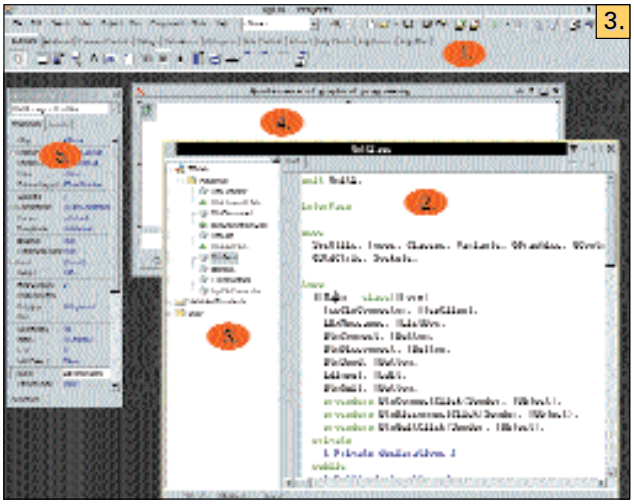
A Kylix felülete

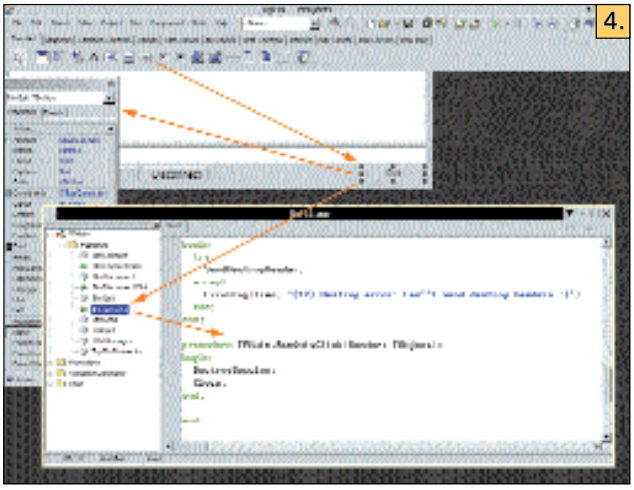
A rendszert futtatva a *3. képen* látható kép tárul a szemünk elé. Öt alapvető rész különíthető el, melyeket az ábrán számokkal jelöltem. Az öt rész a következő:

1. a menüt, az eszköztárat és az elempalettát tartalmazó ablak;
2. a forráskód módosítására használható szerkesztőablak;
3. a Code Explorer, amely gyors tájékozódást tesz lehetővé a szerkesztett alkalmazás forráskódjában;



2.


3.


4.

4. az alkalmazás felületének megtervezésére használható Form Designer;
5. az alkalmazás elemeinek finomhangolását lehetővé tevő Object Inspector.

A Kylix a Linuxon jelenleg rendelkezésre álló eszközökkel ellentétben nemcsak a rendszer felületének megtervezését teszi lehetővé, hanem az eseményekhez tartozó forráskód is azonnal szerkeszthető. A fejlesztés folyamatát mutatja be a *4. kép*, mely a különböző részek összefonódását szemlélteti. A fejlesztő az elempalettáról összeválogatja a szükséges elemeket, és a Form Designer segítségével kialakítja a leendő ablakok tervét. Az ablakra kerülő felületelemek jellemzőit az Object Inspector segítségével lehet beállítani. A szükséges eljárások és függvények megírására a szerkesztőablakban van lehetőség, majd a felület objektumaihoz köthető eseménykezelőket kell megírni. Az eseménykezelők szerkesztését szintén az Object Inspectorból a legegyszerűbb kezdeményezni, mert itt az adott felületelemhez tartozó összes fontosabb eseménykezelő megadható. Az általánosan használt ablaktervek és egyéb, többször használt elemek elmenthetők egy mintakatalógusba (Object repository), ahonnan újból felhasználhatók. A Form Designer szokatlan jellemzője, hogy az X Window System alatt eddig megszokott felülettervezőkkel ellentétben kizárólag állandó méretű objektumokkal dolgozik (mint tudjuk, X alatt a grafikus objektumok általában az ablakkal együtt önműködően méreteződnek), így az ablak átméretezése esetén a felület részeinek átméretezéséről szükség esetén magunknak kell gondoskodnunk.

Fő cél: a hatékonyság

A Borland sokéves tapasztalata jól tükröződik a Kylix felépítésében és működésében. Számos olyan, Linuxon eddig elérhetetlen eszközzel segíti a fejlesztőt, amelyek a hatékonyságot akár a többszörösére is növelhetik. Nézzünk meg – a teljesség igénye nélkül – néhány olyan eszközt, amelyek megkönnyítik a fejlesztést! A programozónak nagyobb projektek esetén nem szükséges fejben tartania akár több ezer objektum- és eljárásmeghatározást, az összevont fejlesztőrendszer ugyanis kérésre kiegészíti azokat. A nagy tudású beépített nyomkövető meggyorsítja és megkönnyíti a hibakeresést, illetve -elhárítást.

A helyzetérzékeny súgó segítségével a rendszerbe épített elemek használatát villámgyorsan el lehet sajátítani. Az alábbi forráskód szerkesztését segítő eszközök összefoglaló neve CodeInsight.

Code Completion

Ha beírunk egy osztálynevet ponttal kiegészítve, a rendszer felajánlja az adott osztály tulajdonságait (properties), tagfüggvényeit (methods) és eseményeit (events) (lásd *5. kép*). Egy eljárás, függvény vagy tagfüggvény nevét beírva a rendszer megadja annak értéklistáját.

Code Parameters

Egy tagfüggvény nevét és egy nyitó zárójelet beírva megkapjuk a tagfüggvény értékeinek formai követelményeit.

Code Templates

A CTRL+J lenyomására a rendszer felajánlja a mintatárában található gyakran használt programrészleteket. A mintatárat természetesen mi magunk is bővíthetjük.

Tooltip Expession Evaluation

A program nyomkövetése közben, amikor az éppen áll, bármely változóra mutatva megjelenik annak pillanatnyi értéke.

Tooltip Symbol Insight

A kód szerkesztése közben bármely azonosítójára mutatva megjelenik annak ismertetése.

© Kiskapu Kft. Minden jog fenntartva

© Kiskapu Kft. Minden jog fenntartva

A fejlesztőt segítik a fordítási hibáknál előbukkanó tippek, amelyek – láss csodát! – tapasztalataim szerint a leggyakrabban rá is mutatnak a valós hibára. A fordító nagyon gyors, hibaüzenetei jól érthetők. A forráskódban való gyors mozgás elősegítésére a rendszer kereszthivatkozásként tudja kezelni az azonosítókat, és böngészőhöz hasonló módon az adott hivatkozás meghatározásához ugrik. A visszajutás meggyorsítására mind a Forward, mind a Back gomb igénybe vehető.
Néhány további tulajdonság címszavakban:

- A fejlesztőket beépített teendőlista (To-Do list) segíti.
- A szerkesztőablakban a különböző nyelvi elemek könnyebb megkülönböztetésük végett eltérő színekkel jelöltek.
- Az eszközpaletták a kényelem érdekében teljesen átszerkeszthetők, illetve bármely ablakba beilleszthetők (docking system).
- A billentyűk kiosztása átszerkeszthető. A számos beépített kiosztás között az Emacsra jellemző is megtalálható.
- Függvénykönytárak és többszálú programok nyomkövetési lehetősége.
- Nyomkövetés közben képes naplót készíteni a fontosabb eseményekről, így lehetővé teszi a későbbi elemzést.

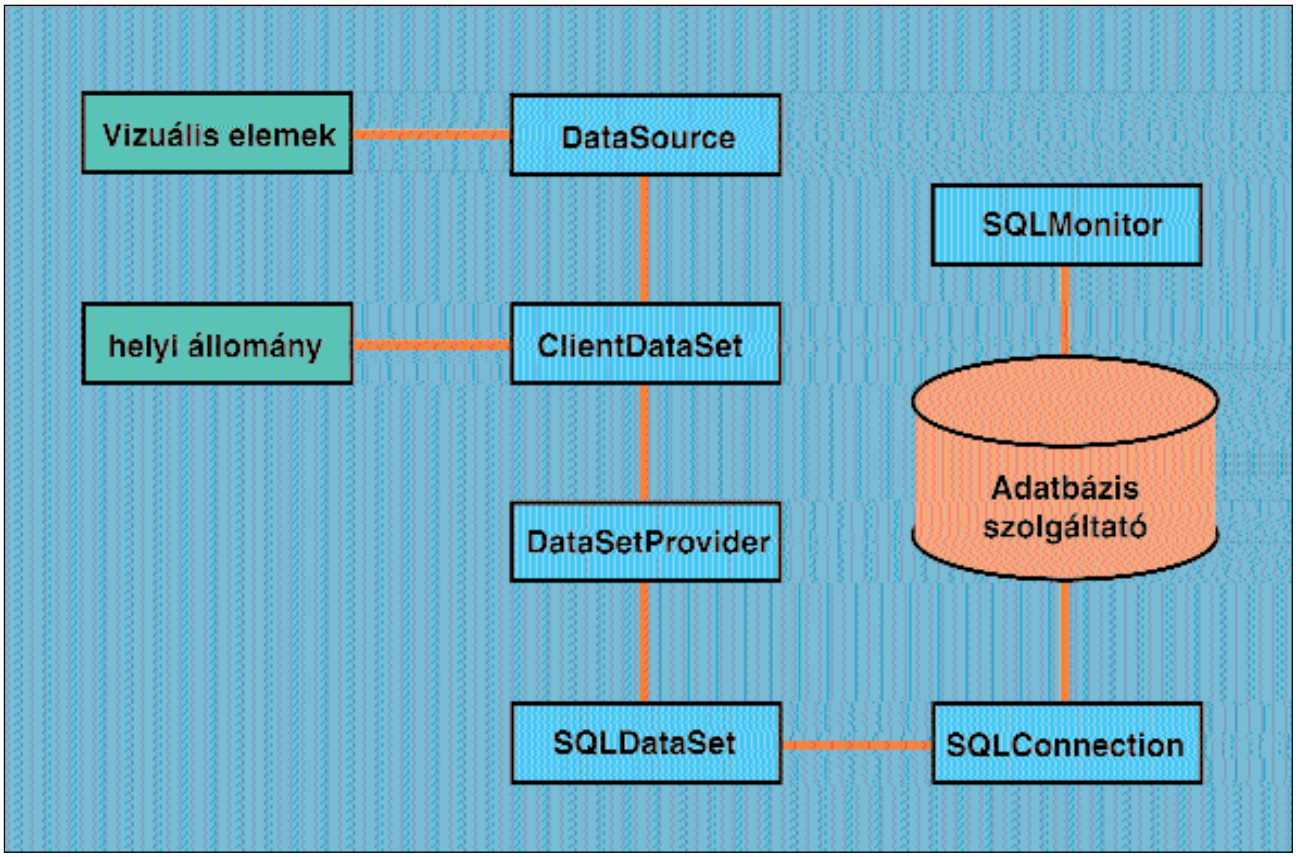

5.

- Nyomkövetés közben a töréspontok csoportokba foghatók össze, majd szükség esetén együtt élesíthetők vagy hatástalaníthatók.
- Mintaalkalmazásokat tartalmaz a programozási módszerek gyors elsajátításának elősegítésére.
- A Server Developer-változat olyan elemeket is magában foglal, amelyek segítségével az Apache webkiszolgálóhoz illeszthető internetes alkalmazások könnyen fejleszthetők.
- Az SQL Monitor az SQL-t használó alkalmazások kipróbálását, nyomkövetését és finomhangolását teszi lehetővé.
- A gyors hálózatos fejlesztéshez sok hálózati protokollra létezik előre elkészített elem.

A fenti felsorolás csak ízelítő abból a rengeteg lehetőségből, amit a Kylix kínál. Minden leendő fejlesztőnek javaslom a fejlesztőrendszer leírásának figyelmes tanulmányozását.

Adatkezelő alkalmazások fejlesztése

A jelenleg fejlesztett alkalmazások jelentős részének valamilyen adatbázist is kezelnie kell. A nyilvántartó rendszerek után egyre gyakrabban bízzák mérőrendszerek adatait, internetes médiák közléseit adatbázis-keresőkre, jó tulajdonságaiknak köszönhetően (megbízhatóság, kereshetőség, összetett lekérdezések és sorolhatnám). Mivel egy rendszernek akár többféle adatforrást is tudnia kell kezelni, szükséges egy olyan eszköz, amely a különböző típusú adatforrások elérését valamilyen szinten szabványosítja. Erre a feladatra a Borland korábban a BDE-t alkalmazta, de a rendszer kellemetlen tulajdonságokkal is rendelkezett. A Borland a rendszer tulajdonságainak javítására a dbExpresst fejlesztette ki. A Kylix adatkezelése már a dbExpressre épül, ez a korábbi, Windowson fejlesztett alkalmazások átírása során gondot is okoz (lásd még később).
A dbExpress kifejlesztése során a következő szempontok voltak a legfontosabbak:



- a rendszer méretét és erőforrásigényét tekintve kicsiny legyen,
- legyen a lehető leggyorsabb,
- tegye lehetővé a könnyű fejlesztést,
- legyen operációsrendszer-független,
- könnyen lehessen új meghajtókat írni hozzá.

A rendszer meghajtóprogramjai kicsik és gyorsak, mert szolgáltatásaik szándékosan behatároltak. A dbExpress-szel a fejlesztő az SQL adatbázis-kezelők összes lehetőségét közvetlenül kihasználhatja. Az adatbázis-kezelő alrendszer néhány tulajdonságának megértéséhez szükséges betekintést nyernünk a rendszer elvi működésébe is. Az adatok a következő úton jutnak el az alkalmazás felületére (lásd az *ábrán*):

- Az SQLConnection-elem összekapcsolja az alkalmazást az adatbázishoz megfelelő dbExpress meghajtóval.
- A dbExpress valamely adatkezelő eleme szükség esetén az adatbázis-kiszolgálón lekérdezést vagy tárolt eljárást hajt végre.
- A DataSetProvider-elem a ClientDataSet kérésére a szükséges parancsokat az előző elemek valamelyikével végrehajtatja, és a kapott adatokat továbbítja a ClientDataSet-elemnek.
- A ClientDataSet a kapott rekordokat a memóriában tartja, melyek rajta keresztül megjeleníthetők vagy módosíthatók. A módosítások a memóriában tárolódnak és az ApplyUpdates tagfüggvény hívásával elküldhetők a DataSetProvidernek. Ez az elem ilyenkor indítja el a tranzakciót a kért módosítások véglegesítésére. Hiba esetén a tranzakció törlődik és a ClientDataSet értesítést kap a hibáról.

A fenti módszer előnyei:

- Lehetővé válik, hogy a tranzakciók idejét a lehető legkisebbre szorítsunki, így csökkenthetjük az adatkiszolgáló terhelését.
- Szükség esetén a rendszer eszközöket ad összetett lekérdezések (többtáblás) szerkesztésére is.
- Mivel a ClientDataSet a memóriában tárolja a lekérdezett adatokat, azok gyorsan újrarendezhetők és szűrhetők az SQL kiszolgáló felesleges terhelése nélkül.
- A ClientDataSet önmagában szolgál néhány SQL-hez hasonló szolgáltatással, ez szintén csökkenti a feldolgozási időt és az SQL kiszolgáló terhelését. A memóriában lévő sorokon összegzéseket lehet végezni, erre a következő függvények használhatók: Sum, Min, Max, Count, Avg. Az SQL WHERE feltételéhez hasonlóan lehet elvégezni a sorok szűrését.

Természetesen a memória használatának megvannak a határai. Ha a lekérdezés eredménye túl nagy mennyiségű adat, az SQL kiszolgáló és az ügyfél közötti hálózat erősen túlterhelődhet. Ha azonban egy lekérdezés ésszerűen felépített és adatait többször is fel lehet használni, a fenti lehetőségek nagyon hasznosak. Jelenleg a rendszer Desktop Developer változata a MySQL és az InterBase, a Server Developer-változat az Oracle és DB2 SQL adatbázis-kezelőket támogatja, mivel azonban a MySQL és InterBase-meghajtók forráskódja mintaként adott, könnyen fejleszthető a rendszerhez saját megbízható, hatékony meghajtómodul. Tudomásom szerint a PostgreSQL-hez ezt a meghajtót jelenleg is fejlesztik. Helyi gyorsítótáblák létrehozására a Borland a MyBase adatbázisrendszert alakította ki. Ez a memóriatáblák használata során nagy sebességet tesz lehetővé, és a könnyebb felhasználhatósághoz a bináris mellett képes XML formátumban is állományba menteni az adatokat.

Fejlesztés több operációs rendszerre

A korábban Delphiben fejlesztett programok átvitele Linux alá nem zökkenőmentes. Aki ismeri a Delphi-rendszert, az tudja, hogy úgynevezett VCL (Visual Component Library) elemekre épült. Ezek között sok olyan is akad, amelyek a Windows valamely egyedi lehetőségére épülnek (OLE, MTS stb.). Mivel ezek Linux alatt nem állnak rendelkezésre, így a velük készült alkalmazások átültetése nehézségekbe ütközhet, rossz esetben a program bizonyos részeit akár teljesen át kell írni. Míg Windows alatt a programok az alacsony szintű műveletek elvégzéséhez gyakran hívják segítségül a Win32 rendszerhívásokat, ezek Linuxon nem léteznek. Az ilyen hívásokat használó program Linux-rendszeren módosítás nélkül nem futtatható. A programok Windows alatt beállítási adataikat gyakran a Rendszerleíró adatbázisban (Registry) tartják, míg Linuxon nincs a Registryhez hasonló központi beállítási adattár. Az itt említett példák mutatják, hogy milyen nehézségek merülhetnek fel a több felületre történő fejlesztéskor. Sokfajta kisebb-nagyobb különbség miatt az eddig fejlesztett programok átvitele Linuxra gyakran igen költséges lehet. További gondok forrása lehet a Delphi 5-ig elterjedt adatelérési eljárás, a BDE (Borland Database Engine). Mivel ez a rendszer nem létezik Linuxra, így a vele fejlesztett programokat át kell írni.
A Kylixban a Delphi 5-ig használt elemek jó részét újraírták. Az új rendszer neve CLX (Component Library for X-Platform). A CLX objektumait úgy tervezték, hogy azok módosítás nélkül vagy a lehető legkisebb módosítással a különböző operációs rendszerek között átvihetők legyenek. A Delphi 6-os változatában is megtalálhatjuk ezeket a CLX-elemeket, így már lehetőség nyílik rá, hogy több operációs rendszeren futó programokat fejlesszünk, de csak akkor, ha az átvihetőséghez szükséges szabályokat betartjuk.

Összegzés

Amennyiben a fejlesztendő program grafikus vagy adatfelülettel érintkezik, mindenképpen javaslom a Kylix használatának megfontolását. Minden cégnek alapvető érdeke, hogy fejlesztései hatékonyak legyenek. A rendszer összevont fejlesztőfelülete gyors és kényelmes munkát tesz lehetővé, használatának elsajátítása akár önállóan is viszonylag egyszerű. A fejlesztők és a felhasználók közös érdeke, hogy a programba a lehető legkevesebb hiba kerüljön, és egy esetleges hiba felfedezése esetén gyorsan ki lehessen deríteni az okát, és el lehessen hárítani a bajt. Ez egy szerényebb tudású rendszerrel nehézkesen kivitelezhető. A Kylixnak megvannak ugyan a maga kisebb gyermekbetegségei, de hiszem, hogy ezeket ki fogja nőni. Végre a grafikus fejlesztőeszközök új nemzedéke indult útjára Linuxon is.

Mátó Péter (atya@andrews.hu) korábban több évig lelkes Delphi-fejlesztő volt. Miután főállású Linux-tanácsadó lett, kisebb grafikus fejlesztésekhez hosszasan keresett hatékony grafikus fejlesztőeszközt. Eddig sikertelenül.

Kapcsolódó címek

A Kylix-rendszer kifejlesztője a Borland Software Corporation
➲ http://www.borland.com
A Kylix-rendszer
➲ http://www.borland.com/kylix
Kiegészítő eszközök és elemek
➲ http://borland.com/kylix/resources/kylixtools.html
Nyitott forráskódú fejlesztőeszközök (Zeos library)
➲ http://www.zeoslib.org

© Kiskapu Kft. Minden jog fenntartva