Qtopia – Az első lépések

A Trolltech egyik mérnöke beavat bennünket a Qtopiával történő fejlesztés rejtelmeibe.

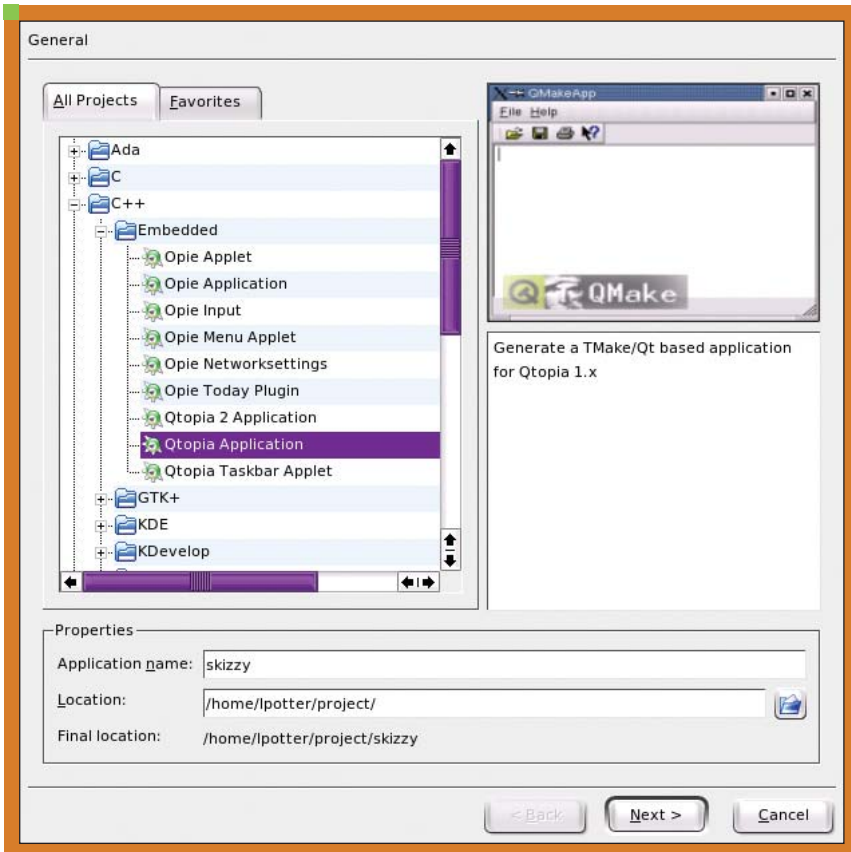
A *Trolltech* által fejlesztett *Qtopia* egy olyan beágyazott alkalmazások fejlesztésére alkalmas fejlesztői keretrendszer, amely teljes szabadságot ad a forgalmazóknak az alkalmazások testreszabásával kapcsolatban. Ezzel párhuzamosan a cég lehetővé teszi, hogy a *GPL* hatálya alá tartozó alkalmazásokhoz a *Qtopia* ingyenesen felhasználható legyen, míg a kereskedelmi szoftverek fejlesztői meg is vásárolhatják azt. Röviden tehát a *Qtopia* kétféle licensszel kerül forgalomba. A rendszerrel történő fejlesztés első közelítésben ugyanolyan, mintha a *Qt/KDE* párost használnánk, eltekintve persze néhány extra lépéstől és eszköztől. Jómagam egy *Gutenbrowser* névre keresztelt alkalmazást kezdtem fejleszteni, ami nem egyéb, mint egy olvasó- illetve letöltőprogram a *Project Gutenberg* által kínált megszámlálhatatlan szöveges dokumentumhoz.
A *Gutenbrowser* először kizárólag *Linuxon* elérhető programként kezdte a pályafutását, de aztán viszonylag gyorsan megszületett a *Windows*, *Qtopia* illetve *Opie* (*Open Palmtop Integrated Environment*) változata is. Az *Opie* amúgy egy a *Qtopia GPL*-es változatán alapuló környezet, amit egy teljes közösség fejleszt.
A *Qtopia* alapját a *Qt Embedded* fejlesztőkörnyezet képezi, így minden, a *Qt* segítségével megírt program könnyen átalakítható úgy, hogy a *Qtopia* környezetben és képes legyen futni. Nekem körülbelül két hetembe telt, amíg „részmunkaidős" nyílt forrású fejlesztőként működésre bírtam a *Gutebrowsert* egy *Sharp Zauruson*. Ebben pedig benne volt magának a *Qtopia API*-nak a megtanulása, valamit a többplatformos fordítás elsajátítása is.
Aki most akarja elkezdeni a *Qtopia* megtanulását, annak azt ajánlom, hogy várja meg a *Qtopia 4* megjelenését, mivel jelentős különbségek vannak a *Qt Embedded 2.3*-as és *4*-es változata között. Ami az átjárhatóságot illeti, *Qt3*, *KDE* sőt még *GTK+* alkalmazásokat is sikeresen portoltak már a *Qtopia 1*-es és *2*-es változataira, de ehhez szükség volt némi extra munkára. Helyettesíteni kellett bizonyos osztályokat, a *KDE* programokhoz pedig a *microkde* forrása is szükséges volt.

Milyen eszközökre lesz szükségünk

Ahhoz, hogy megkezdhessük a munkát a *Qtopia*-val először is szükségünk lesz néhány segédeszközre. A *Qtopia* jelenleg csak *Linuxon* használható tehát szükségünk lesz egy ilyen operációs rendszert futtató gépre. Nyilván kell majd egy olyan képességekkel rendelkező szövegszerkesztő, mint amilyen az *emacs* vagy a *vi*. Én a saját projektemben történetesen a *KDevelopot* használtam, amely rendelkezik egy igen egyszerű *Qtopia* sablonnal.
Ha valamilyen speciális eszközre fejlesztünk alkalmazást, akkor szükségünk lesz egy keresztfordítóra is. Esetünkben az eszköz legyen az *Archos PMA430*, ami a *Qtopia*-hoz az *arm-linux-gcc 2.95*-ös változatát használja. Bár az igaz, hogy a *GCC 3*-as jobban optimalizál, alkalmazásunknak kompatibilisnek kell maradnia az eszközön futó egyéb szoftverekkel, tehát öregecske ugyan a *2.95*-ös, most mégis megteszi. Az *ARM* rendszerekre történő fordításhoz szükséges eszközök az interneten számos helyen fellelhetők. Esetünkben az *Archos* eszközkészletet a ➲ *www.archos.com/products/overview/pma_400_sdk.html* címen találjuk, de számos más link is fellelhető a ➲ *qtopia.net* webhelyen.

Forráskód vagy kész SDK?

Természetesen a *Qtopia*-t magát sem nélkülözhetjük, de itt is van két választási lehetőségünk: letölthetjük a forráskódot, vagy használhatunk



1. ábra Új Qtopia projekt létrehozása

egy lefordított *Qtopia SDK* csomagot is. Ami a *PMA430* platformot illeti a fejlesztőkészlet *GPL* és kereskedelmi változatban egyaránt fellelhető, akárcsak a Qtopia maga. A kereskedelmi csomagot a ➲ *www.trolltech.com/products/qtopia/pricing.html* webhelyen szerezhetjük be teljesen elfogadható összegért, míg a szabad, vagyis a *GPL* hatálya alá tartozó csomagot a ➲ *ftp.trolltech.com/qtopia/sdk* helyről tölthetjük le. Mindkettő a */opt/Qtopia* könyvtárba kerül telepítés után. Ha nincs */opt/Qtopia/arm* könyvtárunk, akkor telepítés után adjuk ki a következő parancsot:

```
# ln -s /opt/Qtopia/sharp
➥/opt/Qtopia/arm
```

A KDevelop projekt

Indítsuk el a *KDevelop* fejlesztőkörnyezetet, majd a *Project* menüből válasszuk a *New Project* pontot. Nyissuk meg az *Embedded* könyvtár alatt található C++ könyvtár ikont. Kattintsunk a *Qtopia Application* nevű fájlra, amellyel egy új *Qtopia* projektet nyitunk meg. Ennek elvileg bármilyen nevet adhatunk, hívhatjuk akár víziónak is. A projekt létrehozására szolgáló dialógusablak az *1. ábrán* látható. Ha megnyitottuk az új projektet, elkezdhetjük azt saját igényeinknek megfelelően átszerkeszteni. Ha egy *Qtopia* projekthez szerkesztünk *.ui* (*user interface*) fájlokat, ehhez mindenképpen a *Qt 2*-ben található Designert használjuk, mivel a későbbi változatokkal készített fájlok nem kompatibilisek a *Qtopia*-val. Én ezt úgy oldottam meg, hogy felvettem egy egyedi külső eszközt, és az *.ui* fájlokat a *Designer 2*-ből nyitottam meg. Ezen a ponton érdemes megjegyezni, hogy nem szabad a *.ui* fájlokat dupla kattintással megnyitni, mivel ennek hatására a *Designer 3* fog elindulni, mivel a *KDevelopban* alapértelmezésként ez van hozzárendelve ehhez a fájltípushoz. Ha ezt el akarjuk kerülni, akkor a *KDevelop* rendszert parancssorból kell futtatnunk úgy, hogy előtte megadjuk a következő környezeti változókat:

```
export PATH=/opt/Qtopia/
➥bin:$PATH
export LD_LIBRARY_PATH=/opt/
➥Qtopia/lib:$LD_LIBRARY_PATH
```

Szintén célszerű beállítani *QVFb* néven a *Qt Virtual framebuffert*, amely célszerűen a */opt/Qtopia/bin/qvfb* helyre kell mutasson. Ebben a virtuális környezetben fut majd az éppen fejlesztet alkalmazásunk a felhasználói felületen. A *Qtopia* közvetlenül a frambufferben jeleníti meg a kimenetét, vagyis nem terheli az *X11* kiszolgálót.

A környezet beállítása

Először is meg kell adnunk néhány környezeti változót a *KDevelop* projekthez. Indítsuk el a *KDevelop* rendszert, majd kattintsunk sorrendben a *Project->Project Options->Run Options* pontokra. Ezután hozzuk létre a következő változókat:

```
Name: QTDIR Value: /opt/Qtopia
Name: QPEDIR Value: /opt/Qtopia
Name PATH Value: /opt/Qtopia/
➥bin:$PATH
Name LD_LIBRARY_PATH Value: /opt/
➥Qtopia/lib:$LD_LIBRARY_PATH
```

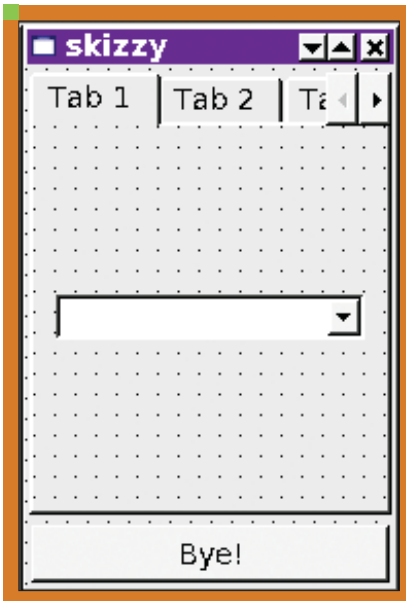
Hasonlóan a *Make Options* alatt is meg kell adnunk néhány értéket: A *skizzy.pro* projektfájlban a *LIBS* értékéhez adjuk hozzá a `-lqtopia` opciót, mivel a *Qtopia 1.7* rendelkezik egy saját könyvtárral. Ezen a ponton a *Makefile*-t kézzel kell létrehoznunk, mivel a *KDevelop* hibásan használja a `tmake` programot:

```
# export TMAKEPATH=/opt/Qtopia/
➥tmake/lib/qws/
➥linux-generic-g++
# tmake -o Makefile skizzy.pro
```

Ha mindezzel megvagyunk, az *F8* leütésével indíthatjuk a projekt fordítását. Ez az utóbbi probléma természetesen kiküszöbölhető, így a *Qtopia* későbbi változatai már a `qmake` programot fogják használni a *Makefile* felépítéséhez.

A beltartalom

Hozzáláthatunk az igazi fejlesztéshez, vagyis elkezdhetjük felruházni alkalmazásunkat néhány funkcióval.
Indítsuk el a *Designer 2* alkalmazást, és nyissuk meg vele a *skizzybase.ui* fájlt. Töröljük belőle a `Qlabel`-t. Hozzunk létre ugyanakkor egy `QTabWidget` objektumot egy



2. ábra A QLayouts használatával elérhetjük, hogy alkalmazásunk automatikusan a megjelenítő felbontásához igazodjon

`QComboBox`-szal egyetemben az első fülön, egy `QlistBox`-ot a másodikon, egy `QMultiLineEdit` elemet pedig a harmadikon (*2. ábra*).
Mentsük a módosított ***.ui*** fájlt. Nyissuk meg a ***skizzy.cpp*** fájlt a ***KDevelop***-ból. Látható, hogy az általunk fejlesztet alkalmazás a `skizzyBase`-ből van származtatva:

```
skizzy::skizzy( QWidget*
➥parent,  const char* name,
➥WFlags fl )
    : skizzyBase( parent,
➥name, fl )
```

Szeretném megváltoztatni a ***main.cpp***-t úgy, hogy az alkalmazás létrehozására egy olyan hatékonyabb metódust használjon, ami az után jelent meg, hogy létrehozták a ***KDevelop*** rendszer ***Qtopia*** sablonját.
Ehhez a szokásos main() függvényt fogjuk megváltoztatni, amely eredeti állapotában így fest:

```
int main( int argc, char **
➥argv )
{
    QPEApplication a( argc,
    ➥argv );
    skizzy mw;
    a.showMainWidget( &mw );
    return a.exec();
}
```

1. Lista Azon függvények megvalósítása, amelyekhez az alkalmazás által használt widget-ek jeleit kapcsoljuk

```
/*
Ez a függvény a Qtopis FontDatabase eszközét használja
arra, hogy egy combo dobozt feltöltsön fontok neveivel.
*/
void skizzy::fillCombo()
{
        QStringList families = fontdb.families();
        for ( QStringList::Iterator f = families.begin(); f
        ➥!= families.end();++f ) {
        QString family = *f;
                ComboBox1->insertItem( family);
        }
}

/*
Ez a függvény akkor fut le, amikor a felhasználó kiválasztja
a combo dobozt, Kitölti a második fülön található listadobozt
azoknak a fontoknak a nevével, stílusával és méretével,
amelyeket a felhasználó kiválasztott. */

void skizzy::comboSelected(const QString &selectedFont)
{
ListBox1->clear();
 QStringList styles = fdb.styles( selectedFont );
        for ( QStringList::Iterator s = styles.begin(); s !=
        ➥styles.end();++s ) {
              QString style = *s;
              QValueList<int> smoothies = fdb.smoothSizes
              ➥( selectedFont, style );
               for ( QValueList<int>::Iterator points =
               ➥smoothies.begin(); points != smoothies.end();
               ➥++points ) {
                   QString pointSize = selectedFont + " "+
                   ➥style +" "+QString::number( *points )
                   ➥+ " ";
                  ListBox1 ->insertItem( pointSize);
               }
           }
    TabWidget2->showPage(tab2);
}

/*
Ez a függvény egy mintaszöveget mutat meg a QMultiLineWidget
segítségével.
A szöveg a felhasználó által kiválasztott fonttal jelenik meg
a harmadik fülön. */

void skizzy::showFont( QListBoxItem *item)
{
        QStringList fontItemString = QStringList::split
        ➥(' ',item->text());
        QString family, style, point;
    family = fontItemString[0];
    style = fontItemString[1];
```

1. Lista folytatás

```
    point = fontItemString[2];
    bool ok;
    int i_size = point.toInt(&ok,10);

    if (!ok) {
        style += " "+fontItemString[2];
        point = fontItemString[3];
        i_size = point.toInt(&ok,10);
    }
    QFont selectedFont( family);
    selectedFont.setPointSize(i_size);

     if(style.find("Italic",0,TRUE) != -1) {
          selectedFont.setItalic(TRUE);
     }

     if(style.find("Bold",0,TRUE) != -1) {
        selectedFont.setWeight(QFont::Bold);
     }

     if(style.find("Light",0,TRUE) != -1) {
        selectedFont.setWeight(QFont::Light);
    }

    MultiLineEdit1->setFont( selectedFont);
    MultiLineEdit1->setText( tr( "The Quick Brown Fox Jumps
    ➥Over The Lazy Dog" ) );
    MultiLineEdit1->setWordWrap( QMultiLineEdit::WidgetWidth);
        TabWidget2->showPage(tab3);
}
```

A változás lényege, hogy a fenti függvényt lecseréljük a *Qtopia* alkalmazásmakrójára:

```
QTOPIA_ADD_APPLICATION("skizzy",
➥skizzy);
QTOPIA_MAIN
```

Ez a módosítás lehetővé teszi, hogy gyorsan indítható alkalmazást hozzunk létre, hiszen a lényeg itt annak a közös alkalmazás-konstruktornak a használata, amely már eleve a memóriában van.
Alkalmazásunk egyelőre nem csinál semmi értelmeset, tehát most már valóban álljunk neki az érdemi munkának a következőképpen:

```
#include <qpe/fontdatabase.h>
```

Hozzunk létre egy privát tagot:

```
FontDatabase fdb;
```

Aztán néhány függvény és privát *slot* következhet:

```
private slots:
void fillCombo();
void comboSelected(const
➥QString &);
void showFont( QListBoxItem *);
```

A *skizzy.cpp* fájlban meg kell adnunk még néhány kiegészítő sort ahhoz, ami hamarosan következik:

```
#include <qstringlist.h>
#include <qcombobox.h>
#include <qtabwidget.h>
#include <qlistbox.h>
#include <qmultilineedit.h>
#include <qfont.h>
#include <qfontinfo.h>
```

3. ábra A kész program a néhány nem túlságosan hasznos funkcióval


```
#include <qpe/fontdatabase.h>
```

Aztán meg kell valósítanunk azokat a függvényeket, amelyekhez a widget-ek jeleit kapcsoljuk (*1. Lista*).
A *Qt* és a *Qtopia* egyaránt jeleket használ a widget-ek közti kommunikáció lebonyolítására. Minden widgetnek megvannak a kibocsátható jelei, amelyeket a kapcsolati makrók segítségével köthetünk a megfelelő függvényekhez.
Kapcsoljuk például a `ComboBox1` `activated` jelét, valamit a `ListBox1` `clicked` jelét a saját slotjainkhoz a következőképpen:

```
skizzy::skizzy( QWidget*
➥parent,  const char* name,
➥WFlags fl )
    : skizzyBase( parent, name,
    ➥fl )
{
    connect(bye, SIGNAL
    ➥(clicked()), this,
    ➥SLOT(goodBye()));
    connect(ComboBox1, SIGNAL
    ➥(activated(const QString
    ➥&)), this, SLOT
    ➥(comboSelected(const
    ➥QString &)));
    connect(ListBox1, SIGNAL
    ➥( clicked ( QListBoxItem
    ➥* )), this, SLOT(showFont
```

```
    ➥( QListBoxItem*)));
    fillCombo();
}
```

Figyeljük meg, hogy a slotot kezelő függvény pontosan ugyanolyan típusú argumentumot fogad, mint amilyen az őt vezérlő jel típusa.
A *KDeveleopban* nyomjuk meg az *F8*-at, vagy válasszuk a *Build* menüből a *Build Project* pontot. A program ennek hatására a natív fordítóprogram használatával lefordul. Ha futtatni akarjuk, indítsuk el a *QVFb*-t, majd egyszerűen válasszuk a *Build* menüből az *Execute Main Program* pontot. Programunk ennek hatására megjelenik a *QVFb* keretrendszerben.

Más platformra történő fordítás

Most tehát van egy úgy ahogy működő programunk, de egyelőre csak a fejlesztéshez használt gépen fut. Le kell tehát fordítanunk az *Archos* rendszerére is. A megfelelő könyvtárak használatához először is meg kell változtatnunk a projekt tulajdonságait. Mivel immár készen állunk a valódi eszközre való fordításhoz, töröljük is az eredeti projektet a *Build* menü *Clean Project* pontjával.
A *Project Options->Make Options* dialógus segítségével végezzük el a következő módosításokat:

```
Name: QTDIR Value: /opt/
➥Qtopia/arm
Name: QPEDIR Value: /opt/
➥Qtopia/arm
Name PATH Value: /usr/local/
➥arm/bin:/opt/Qtopia/tmake/
➥bin:$PATH
Name TMAKEPATH Value:/opt/
➥Qtopia/tmake/lib/qws/
➥linux-arm-g++
```

Töröljük a *Makefile*-t, majd futtassuk a parancssorból a következő parancsot annak a *Makefile*-nak a létrehozásához, ami már az *arm-linux* fordítót használja:

```
# export TMAKEPATH=/opt/Qtopia/
➥tmake/lib/qws/linux-arm-g++
# tmake -o Makefile skizzy.pro
```

A projekt lefordításához üssük le az *F8* billentyűt. A keletkező bináris állományt *USB* porton átmásolhatjuk az *Archos* eszközre, és ott futtathatjuk. Ha telepítőcsomagot akarunk létrehozni, akkor nem árt tudni, hogy a *Qtopia* a *handhelds.org* webhelyen található *Itsy Package Managemenet* (röviden *ipkg*) használja a programok telepítésére. Aki többet szeretne erről megtudni, az látogasson el a *Trolltech Qtopia.net* címen található webhelyére.

Linux Journal 2006., 142. szám

Lorn Potter
a Trolltechnél dolgozik mint a Qtopia közösség vezetője (Qtopia Community Manager). Amerikai, de jelenleg Ausztráliában, Brisbane-ben él ausztrál feleségével és fiával. A nyílt forrású programok fejlesztését saját erejéből tanulta meg. Jelenleg az Opie (Open Palmtop Integrated Environment) Projekt egyik vezető fejlesztője. Korábban zenészként, hangmérnökként és síőrültként is dolgozott.