Grafikus felület programozása Qt-ban (1. rész)

Írásunkkal a grafikus felhasználói felülettel rendelkező programok készítésének rejtelmeibe kívánunk betekintést nyújtani. Készítsünk egyszerű szövegszerkesztő programot együtt! A cikk második részében bővíteni fogjuk a program képességeit, valamint elkezdjük a honosítását is

Már régen elmúltak azok a napok, amikor a grafikus felhasználói felület készítése egyet jelentett kedvenc karakteres szerkesztőnk és a `make` órákon keresztüli nyuvasztásával. Ma már a bonyolult fejlesztői felülettel (amilyen például a KDevelop) nem rendelkező fejlesztői rendszerek is a súgójukban azt ígérik, hogy a grafikus felületek fejlesztését a lehető legfájdalommentesebbé teszik.

Aki azt hiszi, hogy a KDE által is használt Qt fejlesztőcsomag mindössze egyetlen programkönyvtárból áll, nagyon meglepődik, amikor egy 14 MB-ot meghaladó méretű tárállományt tölt le a ➲ http://ftp.trolltech.com/pub/qt/source címről. Annak ellenére, hogy a Qt korábbi változatai híresek voltak megbízhatóságukról, a Qt 3.0-nak 2001. október közepi első megjelenése óta annyi hibája vált ismertté, hogy már a negyedik javítócsomag jelent meg. Sőt, további csomagok megjelenése várható. Emiatt minden esetben a legfrissebb csomag használatát javasoljuk.

A `qt-x11-free-3.0.x.tar.gz` tárállományban – mi magunk a Qt 3.0.4-et használtuk – nem csupán az osztálykönyvtárat találjuk meg, hanem annak HTML formátumú leírását is, valamint több olyan eszközt, amelyek életünk megkönnyítését ígérik a grafikus alkalmazásfejlesztés területén.

Egy egyszerű szövegszerkesztő-alkalmazás grafikus felhasználói felületének elkészítéséhez szerezzük be a Qt Designer programot. Az említett szövegszerkesztő megtalálható a 40. CD Magazin/Qt könyvtárában. A programmal együtt birtokunkba kerül egy felhasználóifelület-fordító eszköz (User Interface Compiler – UIC), amely a Designer program XML formátumú kimenetét C++ nyelvű programmá alakítja. Írásunk második részében további C++-kóddal egészítjük ki kedvenc szövegszerkesztőnket, hogy életet vigyünk grafikus felületébe.

A jövő hónapban kitérünk a Qt fejlesztőkörnyezet újabb tagjára, a Qt Linguistre is, mellyel elkezdjük a program honosítását. A Qt Designerhez hasonlóan ez a segédeszköz is grafikus felülettel rendelkezik, valamint XML formátumot használ ki- és bemenetként. Az XML formátumú fordítást igénylő kifejezéslista elkészítéséhez a programcsomagban egy parancssori eszközt kapunk, amit az `lupdate` névre kereszteltek. Amint a fordítás elkészült, egy másik parancssori eszköz, az `lrelease` az XML-állományt az alkalmazás futása idején a szükséges bináris formátumúvá alakítja.

`Makefile`-okat készíteni Qt-alkalmazásokhoz távolról sem gyerekjáték. A Qt régebbi változatai nem tartalmaztak ehhez a feladathoz segédeszközt; a Qt gyártója, a Trolltech a `tmake` nevű eszközt kínálta külön letöltésre. A Qt 3.0-s tárállományban szerepel a `qmake` nevű új segédprogram, ami nagy segítségünkre lehet a `Makefile` létrehozásához. Erre is a cikk második részében térünk ki.

Tervek szövögetése

A g++ fordítókörnyezettel az összes szükséges fejlesztőeszköz a birtokunkban van; most már csak az elkészítendő szövegszerkesztő-alkalmazásra kell összpontosítanunk. De vajon milyen igényeket is támasszunk programunkkal szemben?

Mivel egyszerű szövegszerkesztőt szeretnénk csak készíteni, csupán az alábbi feladatokat várjuk el: új szerkesztőablak megnyitása és bezárása, dokumentum mentése más néven és kilépés az alkalmazásból. Természetesen tudjon szöveget másolni, kivágni, beilleszteni. Szeretnénk a visszavonás-műveletet is elérni, valamint megköveteljük a programtól, hogy a betűjellemzőknél a dőlt, félkövér és aláhúzott formázást lehessen alkalmazni, illetve ezek tetszőleges kombinációit is. A szerkesztő ***Névjegy*** ablakában jelenjen meg pár sor, a neve pedig legyen mondjuk, `ljedit`.

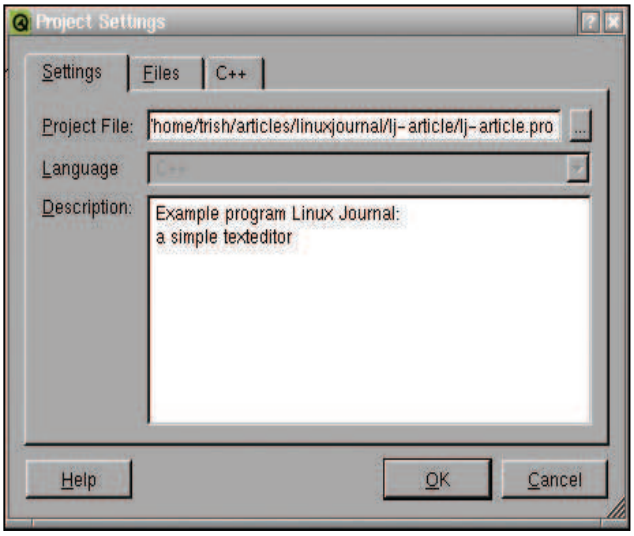
A betűjellemzőktől eltekintve minden feladat legyen elérhető a ***Fájl***, ***Szerkesztés*** és Súgó menün keresztül. A dőlt, félkövér és aláhúzott betűtulajdonságok kapcsolói az eszköztár almenüiként legyenek hozzáférhetők, és ugyanitt további ikonok szerepeljenek az állományok megnyitásához és mentéséhez, a műveletek visszavonásához és megismétléséhez, a szövegelemek kivágásához, másolásához és beillesztéséhez.

Minden ikonhoz tartozzon előugró súgó (buborék), ami jelenjen meg, ha a felhasználó az egérrel elidőz az egyes ikonok felett. A ***Mentés másként*** és a ***Kilépés*** szolgáltatásokat kivéve valamennyi lehetőség gyorsbillentyűkön keresztül is legyen elérhető.

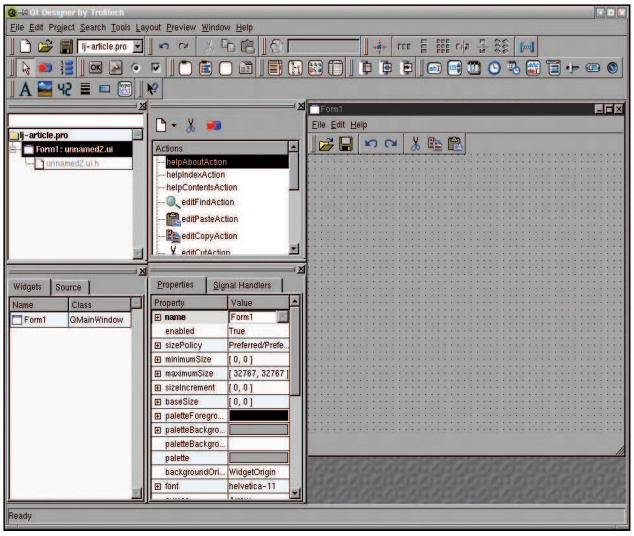
A felhasználókat a váratlan adatvesztéstől megóvandó az állományok megnyitásakor és bezárásakor, továbbá az alkalmazásból történő kilépéskor jelenjen meg a kérdés, hogy a felhasználó a régi állomány (megváltozott) tartalmát menteni szeretné-e, vagy veszni hagyja a módosításokat. A fentebb említett szolgáltatások többségének tervezése a Designerrel elvégezhető, az utóbbi feladat megvalósítására azonban ugyancsak következő számunkban térünk ki.

A felhasználói felület megtervezése

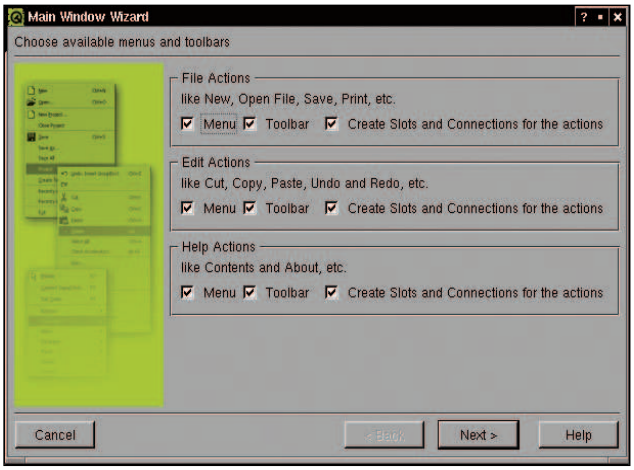
Amennyiben a rendszerre egynél több Qt rendszer van telepítve, a `QTDIR` változót a megfelelő Qt-változatot tartalmazó könyvtárra kell irányítani. Ezután egy terminálból a `designer &` paranccsal indítsuk el a fejlesztőkörnyezetet. Ha a `$QTDIR/bin` könyvtár nincs a keresési útvonalban, vagy több Qt-változat is telepítve van, a parancs kiadásánál érdemes használni a teljes útvonalat. Új feladatfájlt a ***File*** menü ***New*** pontjával hozhatunk létre. A megjelenő párbeszédablakban kattintsunk a ***C++ Project*** ikonon, majd döntésünk megerősítése végett kattintsunk az ***OK*** gombra. Most már létre tudunk hozni új qmake-feladatot, ha megadjuk a nevét, a helyét és



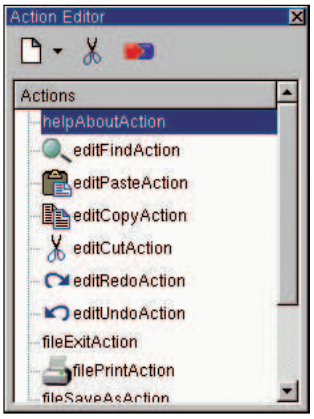
1. kép Az első lépés: a projekt létrehozása



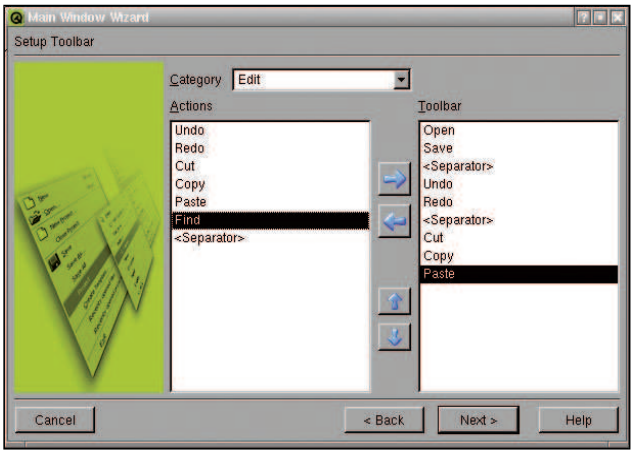
4. kép A Designer és az új form



2. kép A menük és az eszköztár létrehozása

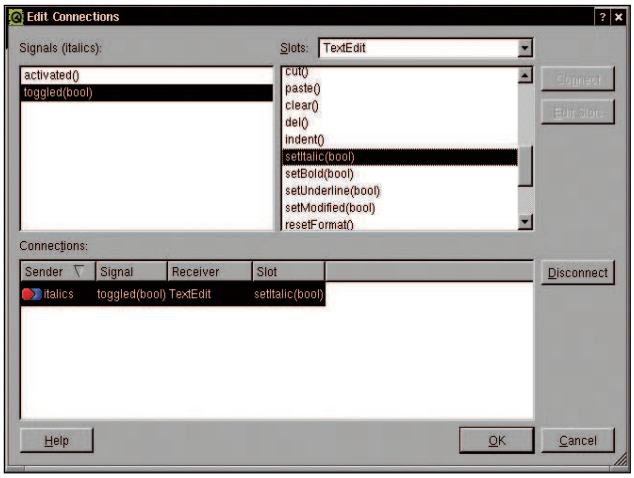


5. kép Az Action Editor

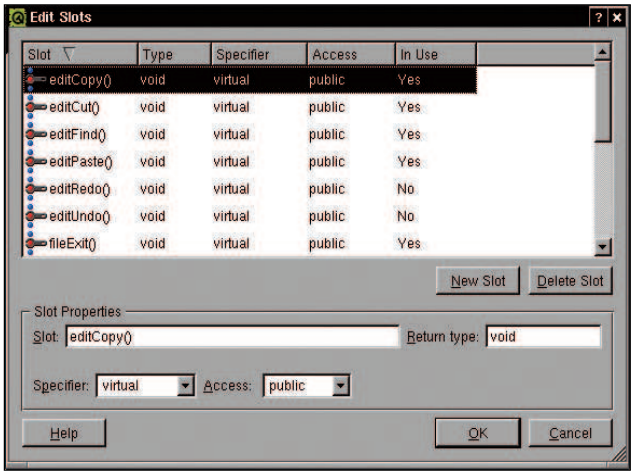


3. kép Töltsük fel az eszköztárat!

a leírását (lásd az 1. *képet*).
Ezután még egyszer a ***Fájl*** menüben válasszuk a ***New*** pontot, hogy létrehozzuk programunk első (és egyben a cikkben felsorolt igények számára szükséges egyetlen) grafikus elemét (widget). Válasszuk tehát a ***Main Window*** ikont, létrehozva a program főablakát.
Ha telepítve van a varázsló, akkor most megjelenik a Main ***Window Wizard*** ablak első oldala (lásd a 2. *képet*). Egy pillanatig elgondolkodunk, majd igen, gondoljuk úgy mi is, hogy a varázsló hozza létre a menüt és az eszköztárat. Ezzel egy keretet (a szokásos függvényekkel és csatolásokkal) kapunk, amelyet remekül tudunk majd használni.
A ***Next>*** gomb megnyomása után az eszköztár létrehozásával kapcsolatos lapra kerülünk. Itt a ***File*** kategóriából választjuk ki az ***Open*** és a ***Save*** pontokat, majd adjuk hozzá az eszköztárhoz a jobbra mutató nyíl segítségével. Ezután ugyanígy adjuk hozzá az ***Edit*** kategóriából az ***Undo***, ***Redo***, ***Cut***, ***Copy*** és ***Paste*** pontokat, valamint rendezzük el az eszköztárat elválasztókkal (***Separators***), ahogy a 3. *képen* is látható. Az utolsó oldalon semmit nem kell tennünk, csupán bezárnunk azt. Ezután a Designer a 4. *képen* hasonló állapotot mutatja.
Ahhoz, hogy kívánt programunk elkészüljön, összesen két további dologra van szükségünk: egyrészt a főablak neve ne Form1, hanem valamilyen értelmes név legyen, másrészt pedig szükségünk van a szerkesztőmezőre, hisz anélkül bajosan tudunk szöveget szerkeszteni. Az első igény könnyen kielégíthető: a ***Property Editor*** ablakban a ***Properties*** lapon kell körbenéznünk. Ez a lap mindig az éppen használt elem tulajdonságait és azok értékeit kínálja fel szerkesztésre. Mivel utoljára a Form1 formmal dolgoztunk, nem kell külön rákattintanunk. A ***name*** tulajdonság értékét változtassuk meg `ljeditor`-ra, ezáltal nevet adunk az osztálynak, majd a ***caption*** mezőbe írjuk be az ablak fejlécében kívánt szöveget (a változatosság kedvéért legyen `ljeditor`).
Most hozzuk létre a szövegszerkesztő területet. Kattintsunk a ***Richtext Editor*** ikonra (a 4. képen a jobb szélétől számolva az ötödik ikon, „abc de” felirattal), majd kattintsunk az



6. kép Az italics bekapcsolására az ljedit dőlt betűket használ majd

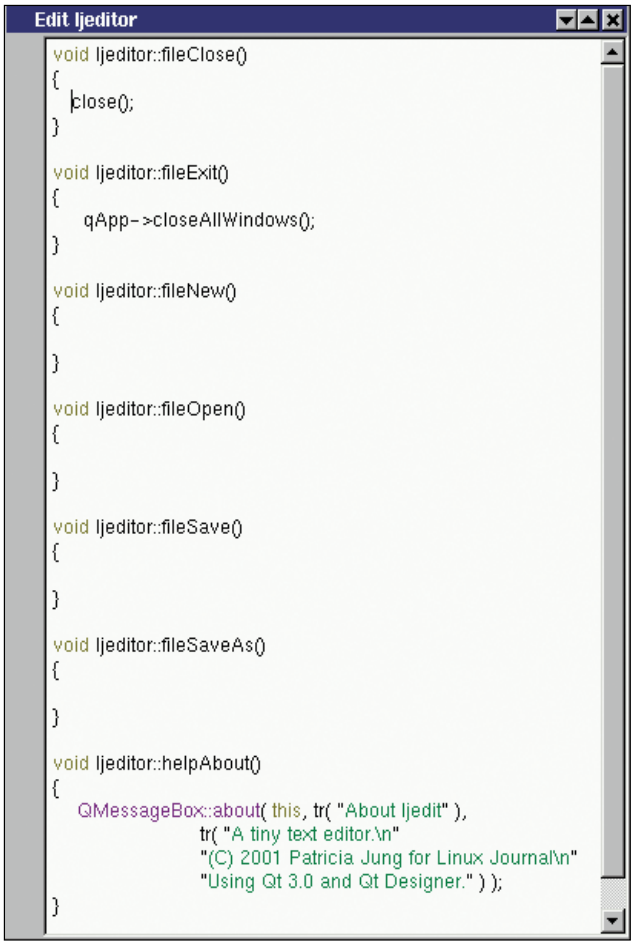


7. kép Új függvényváz létrehozása

ljeditor form pöttyözött hátterén. Az új szerkesztőmezőt a szokásos módon méretezhetjük. Kereszteljük el `TextEdit` névre. Már csak a program szolgáltatásait kell biztosítanunk. Először is távolítsuk el a nem használandó tevékenységeket (***Actions***) az ***Action Editor*** segítségével (*5. ábra*). Töröljük az ollót ábrázoló ikonnal (vagy a helyi menüből kiválasztott ***Delete*** ponttal) az alábbi elemeket: ***helpIndexAciton***, ***helpContentsAction***, ***editFindAction***.

Tevékenységek

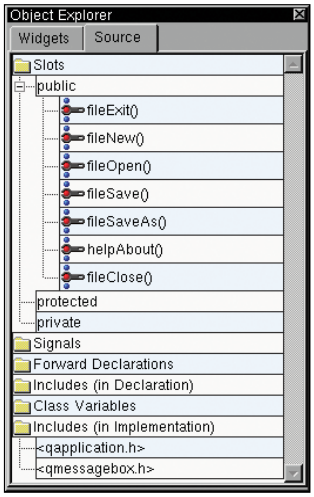
A szerkesztőnek szüksége van néhány további tevékenységre is, hogy kezelni tudjuk a szöveg formázását (félkövér, dőlt, aláhúzott). Az ***Action Editor*** ablakban lévő ***New*** lenyíló menüből (Egy üres lap) válasszuk a ***New Dropdown Action Group*** pontot, melybe majd a módosítók kerülnek, és állítsuk be fontosabb értékeit: a name tulajdonsághoz írjuk be a ***fontCharacter*** nevet, majd az ***iconSet*** tulajdonságra kattintva a megjelenő „...” gomb segítségével válasszuk ki a megfelelő ikont. Az ***Add...*** gombbal külső fájlt is adhatunk hozzá.
A ***menuText***-hez és a súgószövegekhez írjuk be a „Font Characteristics” szöveget. A ***tooltip***-hez és a ***statustip***-hez írjuk be: „Choose font characteristics”, valamint fontos, hogy az ***exclusive*** tulajdonság értékét ***False***-ra állítsuk. Ezzel érjük el,



8. kép Egyszerű forráskódsorokkal ilyen könnyű a szolgáltatásvázat kiegészíteni

hogy a felhasználó egyszerre több tulajdonságot is ki tud választani (például egy szövegrészt félkövérre és dőltre formázhat). Az ***exclusive*** értékét ***True***-n hagyva egyszerre csak az egyik elemet lehet kiválasztani.
A ***fontCharacter*** csoporton jobb gombot nyomva, majd a ***New Action*** (új tevékenység) menüpontot választva a helyi menüből, adjunk hozzá egy ***italics*** nevű tevékenységet a CTRL+I gyorsbillentyűvel, ***Italics*** szöveggel. Mivel a felhasználó ezt a tulajdonságot be és ki tudja kapcsolni, fontos, hogy a ***toggleAction*** értékét ***True***-ra állítsuk. Alapértelmezésként nem szeretnénk dőlt szöveget, ezért az ***on*** tulajdonságot kapcsoljuk ki.
A másik két gyermektevékenységet is hozzuk létre, mind a félkövér (bold, CTRL+B), mind pedig az aláhúzott (***underline***, CTRL+U) számára.
Az elkészített ***fontCharacter*** csoportot egyszerűen úgy adjuk hozzá az eszköztárhoz, hogy megfogjuk az ***Action Editor***-ban, és áthúzzuk az eszköztárra. Ha elé vagy mögé elválasztót szeretnénk, a kívánt helyen kattintsunk a jobb gombbal, majd a helyi menüből válasszuk az ***Insert Separator*** pontot.
Ha most átlépünk előnézetbe (CTRL+T), láthatjuk, hogy az elkészített elemek egyelőre semmit sem csinálnak. Hogy életre keltsük őket, ismét visszatérünk az ***Action Editor*** ablakba, kiválasztjuk az italics elemet, majd a piros-kék ***Edit Connections*** (Kötések szerkesztése) gombra kattintva kiválasztjuk a `toggled(bool)` jelet a ***Signals*** listából. Ahelyett hogy az `ljeditor` egyik foglalatához (slot) kötnénk, a ***Slots*** lenyíló

listából a `TextEdit`-et választjuk, majd az alatta lévő listából kiválasztjuk a `setItalics(bool)` elemet (ez egyébként a `QTextEdit` osztályhoz tartozik, a `TextEdit` is ennek az osztálynak a tagja). Itt semmi más dolgunk nincs, a kötés létrejött (ahogy az ablak alsó részében megjelenő új sor ezt jelzi is), zárjuk be az ablakot az ***Ok*** gombbal. Ugyanezzel az eljárással a ***bold*** tevékenységet kössük a `setBold(bool)` foglalathoz, valamint az ***underline*** tevékenységet a `setUnderline(bool)` foglalathoz. Ha ezzel megvagyunk, próbáljuk ki ismét az előnézetet. Igen! Programunk kezeli a CTRL+I, CTRL+B, CTRL+U gyorsbillentyűket!



9. kép Az Object Explorerrel könnyű új fejállományokat hozzáadni

Most pedig jöjjenek a beépített tevékenységek kapcsolatai! Ezeket nem lehet „váltani”, tehát a `toggled(bool)` jel nem jön szóba. Ehelyett el tudnak indítani egyszerű parancsokat, mint például „Mentsd az adatokat”. Ezért az `activated()` jelet kell a megfelelő foglalatba kötnünk.

Az `edit RedoAction` számára ez a `TextEdit::redo()`. Leválasztjuk tehát ezt a tevékenységet az `ljeditor::editRedo()`-ról. Ezt egyébiránt akkor használtuk volna, ha nem akarnánk a `QTextEdit Redo()`-jára támaszkodni. Ugyanígy, az `editUndoAction action()` jelét kössük a `TextEdit::Undo()`-hoz, és válasszuk le az `ljeditor`-ban lévő vázfüggvényről. Ugyanígy járjunk el az `editPasteAction` és a `TextEdit::Paste()` foglalat, az `editCopyAciton` és a `TextEdit:copy()` foglalat, valamint az `edit CutAction` és a `TextEdit::Cut` foglalat esetén.

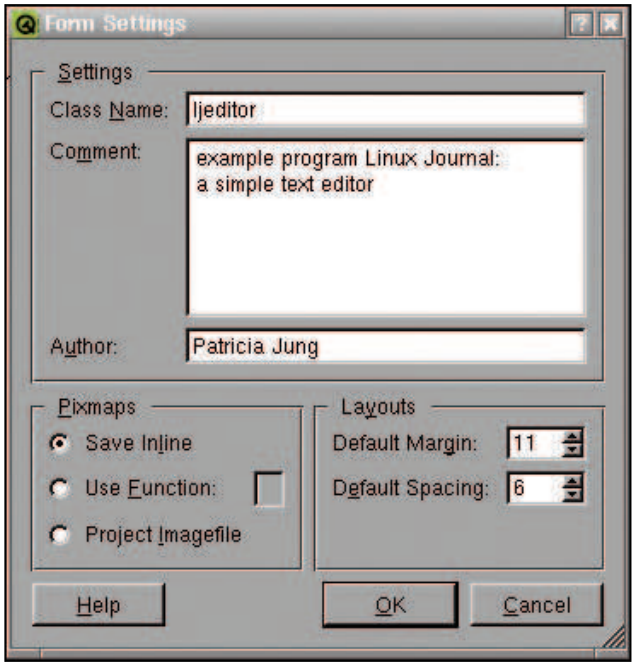
A további előre elkészített tevékenységek (`helpAboutAction`, `fileExitAction`, `fileSaveAction`, `fileSaveAsAction`, `fileOpenAction` és a `fileNewAction`) továbbra is az `ljeditor`-ban megadott vázhoz kapcsolódjanak, ezeket a későbbiek során még programkóddal egészítjük ki.

Egy fontos művelet azonban továbbra is hiányzik, nevezetesen az, amelyet a felhasználó akkor indít el, amikor az éppen használt szerkesztőablakot szeretné bezárni a `fileExitAction` művelethez hasonlóan (ez utóbbi a teljes alkalmazásból kilép).

A munkafolyamat immár ismerős: a tevékenységszerkesztőben kiválasztjuk a ***Create New Action*** menüpontot, majd a ***Property Editor***-ban a `fileCloseAction` nevet adjuk neki, begépeljük a „Close” szöveget; gyorsbillentyűnek pedig a CTRL-Z-t választjuk.

Mindazonáltal az `accelerated()` függvény kötési helye még mindig hiányzik. E hiányosság kijavításához a ***Designer Edit*** menüjében válasszuk a ***Slots*** pontot. A *7. képen* látható a megjelenő ablak.

Hozzunk létre egy új foglalatot `fileClose()` függvénynévvel, visszatérési típusa (***return type***) legyen üres (***void***), elérése (***Acess*** type) pedig public. Az ***Edit Slots*** párbeszédablak nem képes csodákat művelni, csupán egy függvényvázat hoz létre az `ljeditor` osztálymeghatározásán belül. Ezek után hozzákötheti jük a `fileCloseAction`



10. kép Hadd tudja meg a világ, ki készítette ezt a programot!

`activated()` jelét az `ljeditor::flieClose()`-ához az ***Action Editor*** már ismert ***Edit Connections*** ablakában. Mivel ezt a szolgáltatást csak menün keresztül szeretnénk használni, egyszerűen fogjuk meg és húzzuk rá az `ljeditor` formon található ***File*** menüre.

További pár sor kód

A Qt alatt egy elem bezárása egyszerű: minden Qt-elem örökli a `close()` függvényt a Qt-elemek „ősatyjától”, a `QWidget`-től. Ez nem jelent komoly programozási feladatot, így elegendő mindössze egyetlen sort írni a `fileClose()` függvénybe. Miután a `ljeditor` formon jobb gombbal kattintottunk, megjelenik a helyi menü. Ennek Source pontja lesz a segítségünkre. A menüpont választáskor megjelenik a forráskód-szerkesztő ablak, itt már könnyedén beszúrhatjuk egysoros programunkat (*8. kép*).

```
void ljeditor::fileClose( )
{
      close( );
}
```

A `fileExit()` függvényt is egyből megírhatjuk. Az alkalmazásból való kilépéshez az alkalmazásobjektum `closeAllWindows()` függvényét kell meghívnunk:

```
void ljeditor::fileExit( )
{
      qApp->closeAllWindows( );
}
```

Minthogy a Designer magukkal a `QApplication` objektumokkal rendszerint nem foglalkozik, a `qapplication.h`-ra (mely tartalmazza a `qApp` proxyt az valódi alkalmazásobjektum eléréséhez) alapértelmezésben nincs hivatkozás, így az ***ui*** elkészítésekor létrejövő kód nem fordul le.

Tipptár

Tevékenységek
A mai grafikus programok használata közben gyakran megtörténik, hogy valamilyen tevékenységet el akarunk végeztetni, például a fájl mentését, egy ablak bezárását stb. Ezeket a tevékenységeket általában többféleképpen elérhetjük: menüből, gyorsbillentyűvel, az eszköztár egy gombjával. Hogy ne kelljen többször megírnunk ugyanazt a szolgáltatást, az egy-egy szolgáltatáshoz tartozó szükséges elemeket összegyűjtünk egy úgynevezett tevékenységben (action). Ahelyett, hogy a menü egyik pontjaként vagy egy eszköztárelemként írnánk meg ugyanazt, az elkészített tevékenységet egyszerűen felrakjuk az eszköztárra vagy a menübe, az pedig a környezet igényeinek megfelelően menüpontként, gombként vagy egyéb elemként viselkedik.

Jelek és foglalatok
Az összetevőkre bontható programozás érdekében Qt alatt az alábbi gondolatvilágot követjük: amikor egy objektumnak egy tulajdonsága megváltozik, egy jelet (signal) hoz létre (egy gomb például egy `clicked()` jelet hoz létre, amikor az állapota „üres”-ről a „kattintottak rajtam”-ra változik). Más objektum az erre felkészített saját függvényeit, úgynevezett foglalatait (slots) ráirányíthatja az őt érdeklő jelekre. Így például az eszköztár Megnyitás gombja által készített jelet hozzáköthetjük a `fileOpen()` foglalathoz. Mivel a jelek és a foglalatok nem részei a szabványos C++ nyelvnek, megfelelő formára alakításukhoz egy `moc` (meta object compiler) nevű előfeldolgozó szükséges. Ezért bonyolultak még az egyszerű Qt-alkalmazásokhoz készített `Makefile`-ok is.

Súgószövegek
Az állapotszöveg (statustip) és az eszköztipp (tooltip) két olyan szöveg, amelyekkel a felhasználót segíthetjük. Amikor a felhasználó elidőz egy elem fölött és az adott elemhez súgószövegeket adtunk meg, azok a megadott módon megjelennek (az állapotszöveg az állapotsorban, az eszköztipp az egér mellett, általában egy kis sárga ablakban).

Szerencsére egy ***Object Explorer*** (objektumtallózó) a segítségünkre siet a fejállományokra hivatkozás elkészítésekor is. Alapban a ***Widgets*** lap látszik rajta (lásd a 4. képen), de nekünk a ***Source*** lapra van most szükségünk. Kattintsunk ezen a lapon jobb gombbal az ***Includes (in Implementation)*** ponton, majd válasszuk a ***New*** pontot. Ne feledjük, hogy a fejállomány nevét <> jelek közé kell tennünk: `<qapplication.h>` (lásd a *9. képet*). Egy másik foglalat, amit anélkül tölthetünk meg tartalommal, hogy fordítási hibák tömegétől kelljen félnünk, a `helpAbout()`. Ez akkor kerül meghívásra, ha a felhasználó megnyitja a Névjegy (***About***) ablakot, és egy egyszerű párbeszédablakot jelenít meg „About ljedit” címmel és némi szöveggel. Ha a `tr()` függvényt használjuk a szövegek kiíratásakor, a honosítási kérdéseknél lényegesen kevesebb gonddal kell majd szembenéznünk (ezzel a témakörrel a következő számban foglalkozunk):

```
void ljeditor::helpAbout( )
{
    QMessageBox::about( this, tr( "About
        ➥ljedit" ),
        tr( "A tiny text editor.\n"
        "(C) 2002 Patricia Jung for Linux
        ➥Journal\n"
        "Using Qt 3.0.4 and Qt Designer." )
);
}
```

A `QMessageBox::about()` használatához a forráskódba be kell illesztenünk a `<qmessagebox.h>` fejállományt, ahogyan ezt a `<qapplications.h>` állománnyal is tettük. A maradék működésbeli kiegészítést alosztály formájában a következő alkalommal fogjuk elkészíteni.
Ezek után a Designerrel nincs más teendőnk, mint „letisztítani” az új felhasználói felületet.

Fontos csinosítások

Mielőtt bezárjuk a Designert, ellenőrizzük, hogy valamelyik gyorsbillentyűt nem használtuk-e fel kétszer. Ezt az ***Edit*** menü ***Check Accelerators*** pontjának kiválasztásával tehetjük meg. Továbbá eltávolítjuk az összes `ljeditor` foglalatot, melyet nem fogunk a későbbiekben sem használni, vagy kiváltottuk egy `TextEdit` foglalattal. Az ***Edit*** menü ***Slots*** pontját választva a már ismert ***Edit Slots*** ablakba jutunk. Jelöljük ki az `editUndo()`-t, majd a ***Delete Slots*** gombra kattintva töröljük (*7. kép*), majd teljesítsük be hasonlóképpen az alábbi foglalatok sorsát: `editRedo()`, `editCut()`, `editCopy()`, `editPaste()`, `editFind()`, `helpIndex()`, `helpContents()` és `filePrint()`.
Használjuk a GUI előnézetet, és távolítsuk el a felesleges elválasztókat is. Mivel az `editFindAction`-t nem használjuk, egy árva elválasztó található az `ljeditor` ***Edit*** menüjének alján. Valamikor régen volt még egy ***Find*** pont ez alatt. Az elválasztó törléséhez a jobb gombbal kattintsunk rá a formon, majd a helyi menüből válasszuk a ***Delete*** pontot. Ugyanígy járjunk el a ***File*** menüben is a felesleges elválasztókkal.
Minden elem a helyén van, most már a Qt dolga, hogy elhelyezze őket. Jelöljük ki az egész formot, majd válasszuk a ***Layout*** menü ***Lay Out Vertially*** pontját. Ha a felhasználó megváltoztatja az ablak méretét, a `TextEdit` mező is követni fogja az ablak változását. Ha további elemeket akarunk rakni a formra, előtte fel kell bontanunk az elrendezést.
És még egy dolog. Ezt a programot a sajátunkként szeretnénk terjeszteni. Töltsük tehát ki az ***Edit*** menü ***Form Settings*** hatására megjelenő ablak ***Author name*** és ***Description*** mezőit. Ezekben azt is megadhatjuk, hogy a programhoz tartozó képeket külön fájlban kívánjuk tárolni, vagy pedig befordíttatjuk őket közvetlenül a felhasználói felület forrásába.
Végül válasszuk a ***File*** menü ***Save All*** pontját, és már készen is vagyunk! Az utolsó lépéssel projektünköz hozzáadtuk a felületet leíró XML-fájlt (nevezzük, mondjuk ***ljeditor.ui***-nak), valamint az ***ljeditor.ui.hu***-t, mely az általunk begépelt kódokat tartalmazza.

Linux Journal 2002. szeptember, 101. szám

Patricia Jung
(trish@trish.de) háttérismereteit rendszergazdai, műszaki szakírói és újságszerkesztői tevékenységéből meríti, és ilyen minőségében jól érzi magát, hogy kiváltságosként a Linuxszal, illetve a Unixszal foglalkozhat.