KDevelop

Integrált fejlesztői környezet KDE alapokon

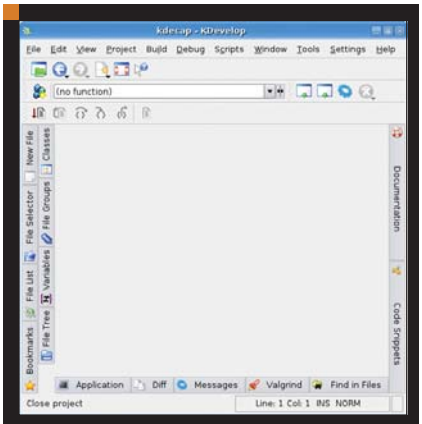
Sokaktól lehetett és néha még mostanság is lehet hallani, hogy a Linux egyik nagy hátránya a kiforrott, magas szintű, könnyen kezelhető integrált fejlesztői környezet(ek) hiánya. Ebben az írásban megpróbáljuk megmutatni, hogy ez a kijelentés ma már egyáltalán nem állja meg a helyét. A Linux alatt ma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

Ismerkedés

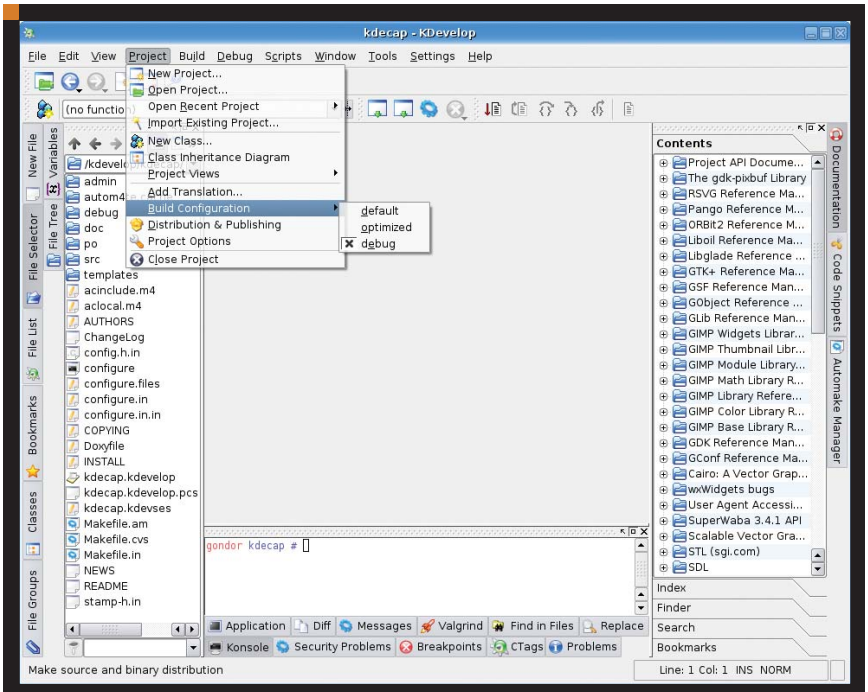
Más operációs rendszerek alatt hosszú évek óta számos alkalmazásfejlesztést segítő eszköz érhető el, ma már inkább hiányuk volna szokatlan. *Linux* alatt, noha legalább ugyanolyan régen elérhetők fejlesztést segítő eszközök és környezetek, kiforrott, magas funkcionalitású, valódi integrált fejlesztői 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*).

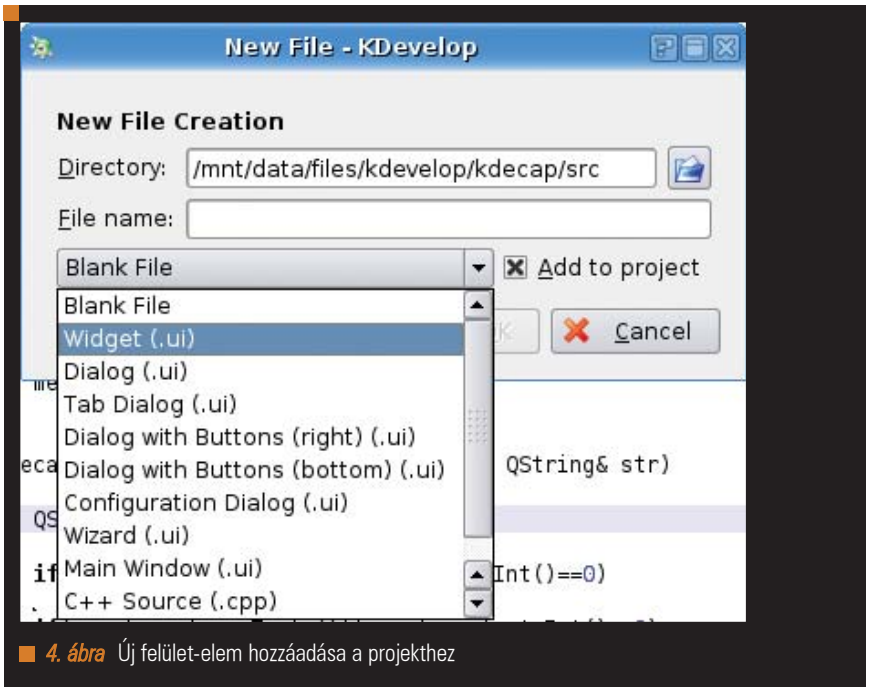
C/C++ kód esetén a ***KDevelop*** néhány karakter begépelése után lehetőségeket kínál a szó automatikus kipótlására (ú.n. ***autocomplete*** funkció), amellyel gyorsabbá válik a kód írása. Más nyelv esetén a szövegszerkesztő szókipótló funkcióira támaszkodhatunk, amelyet a ***Settings->Configure Editor->Plugins***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

Felület-tervezés

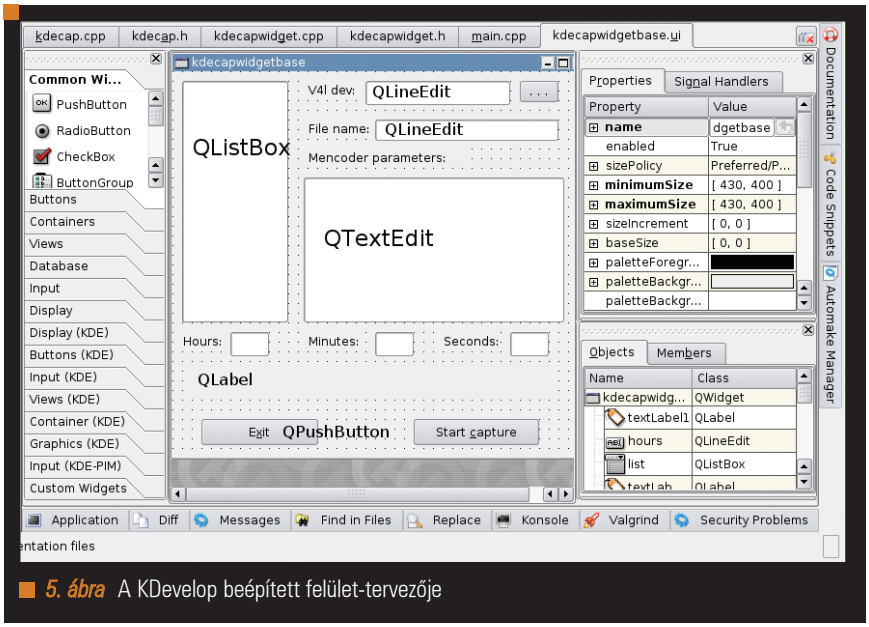
C++-ban készülő ***KDE***-s alkalmazás készítésekor lehetőségünk van vizuálisan szerkeszteni az alkalmazás felhasználói felületét. A 3.x-verziójú ***KDevelop*** beépített ***GUI*** szerkesztővel rendelkezik (korábbi verzióknál erre egy külső alkalmazás, a ***QT Designer*** szolgált). A felületszerkesztő 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

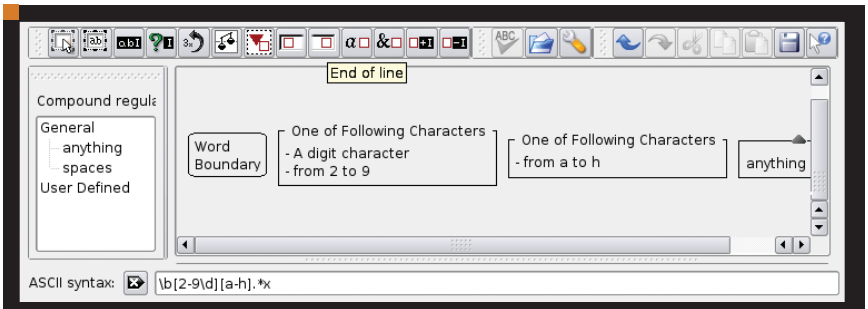


4. ábra Új felület-elem hozzáadása a projekthez

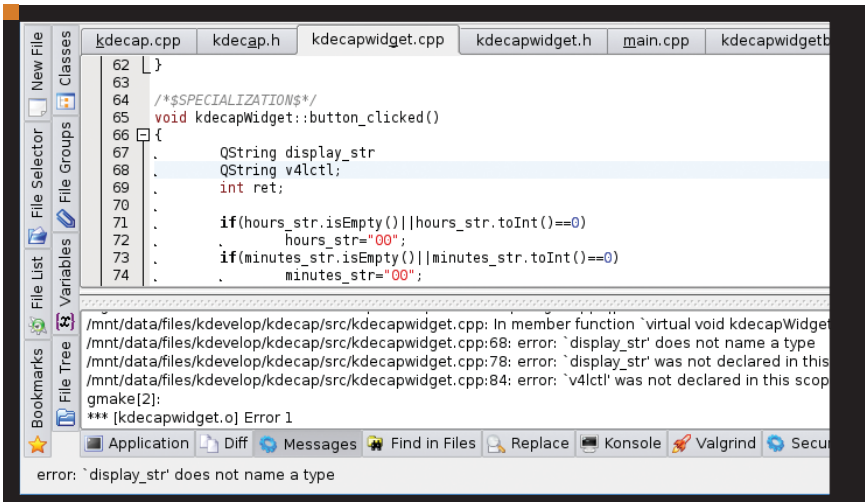


5. ábra A KDevelop beépített felület-tervezője

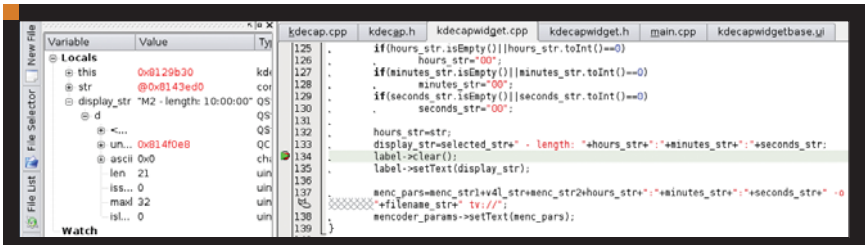
6. ábra Lehelyezett widget eseményéhez kezelőfüggvény rendelése



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*).

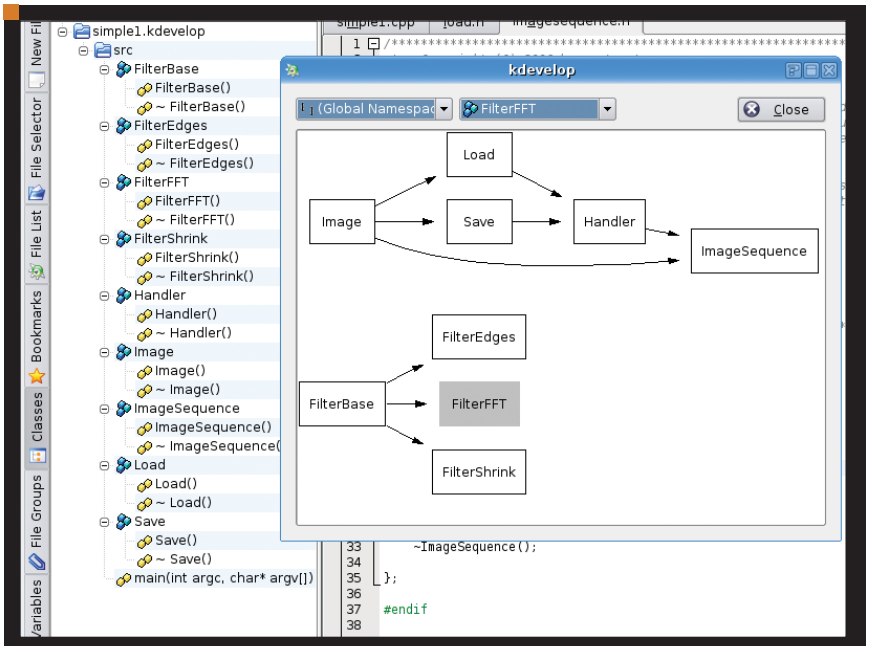
Kód-újrafelhasználás

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.



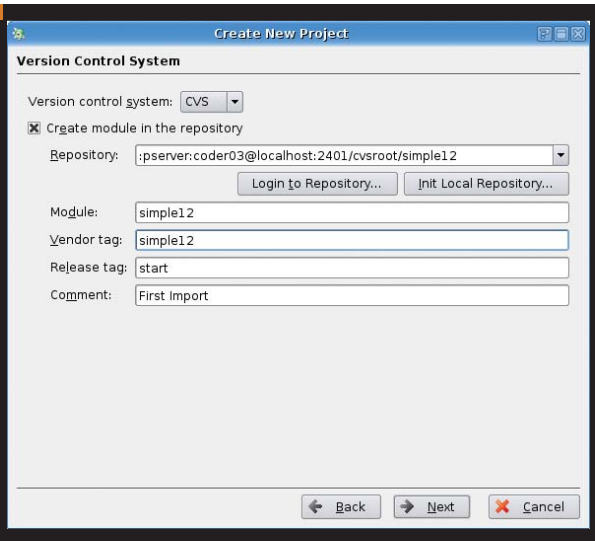
11. ábra Osztályhierarchia megjelenítése

Verziókövetés

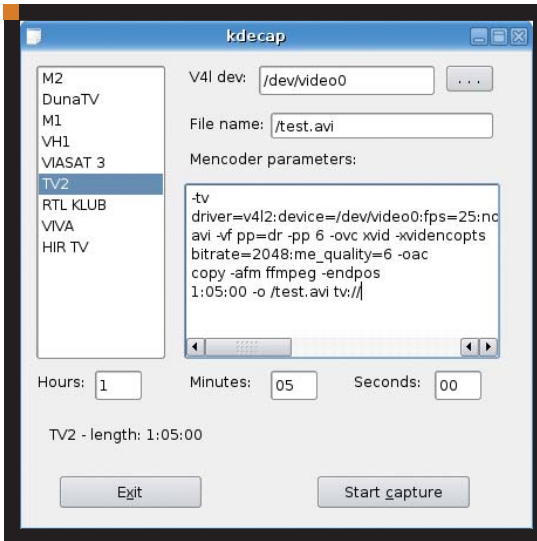
Nagyobb projektek, főleg ha többen részt vesznek programírásban, elképzelhetetlenek valamiféle verziókövetési rendszer használata nélkül. Ennek megfelelően a *KDevelop* is támogat verziókövetési szerverekhez történő csatlakozást és ezek használatát. Projekt létrehozásakor az első lépéseknél már lehetőségünk 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