



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

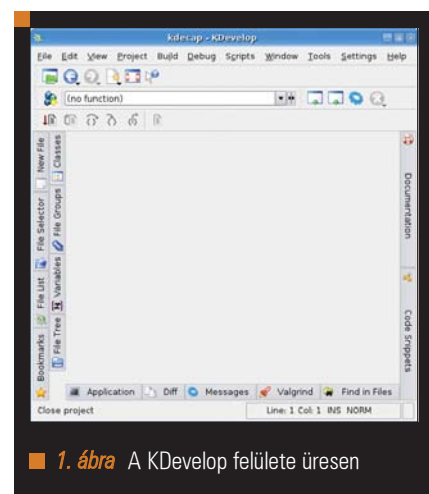
Más operációs rendszerek alatt hosszú évek óta számos alkalmazásfejlesztést segítő eszköz érhető el, ma már inkább hiányuk volna szokatlan. *Linux* alatt, noha legalább ugyanolyan régen elérhetők fejlesztést segítő eszközök és környezetek, kiforrott, magas funkcionalitású, valódi integrált fejlesztői környezetre viszonylag sokáig kellett várunk. A *KDevelop* egy ilyen környezet, amely több programozási nyelven történő fejlesztést támogat egy rengeteg funkcióval ellátott alkalmazásban. Fő erőssége *KDE/QT* alkalmazások fejlesztése, amely segítségével integrált grafikus felület-szerkesztőt 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 hármastól kezdve 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

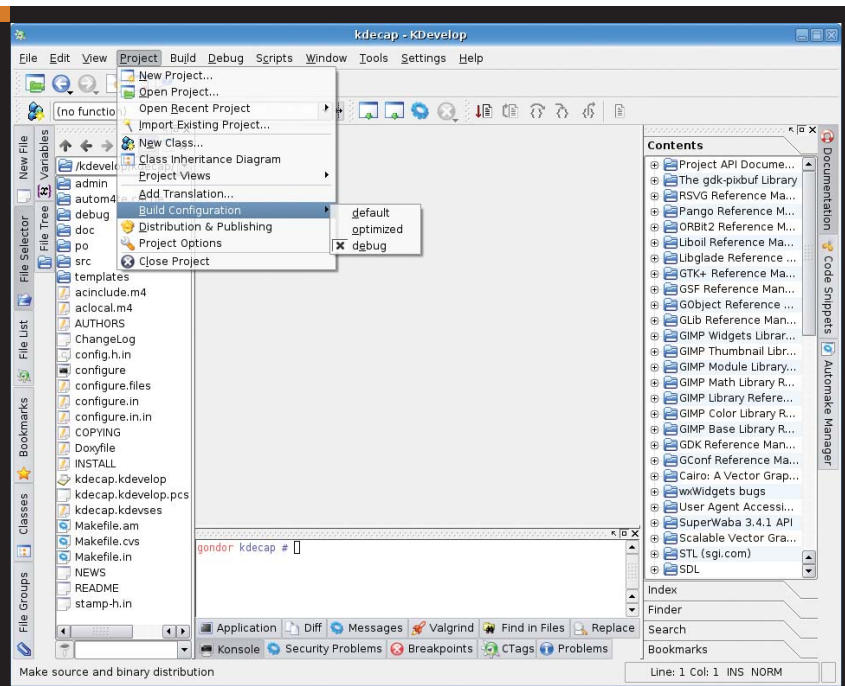
a belső eszközök és funkciók leírásánál szót fogunk ejteni. Az írást nem teljesen kezdő programozóknak szánjuk, sokkal inkább azon olvasóknak, akiknek van már némi programozási tapasztalatuk ill. használtak már fejlesztői környezeteket így rendelkeznek némi összehasonlítási alappal.

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 található az ún. *Tool Dock*-ok. Ezek helye



1. ábra A KDevelop felülete üresen



2. ábra File Selector, Documentation és Konsole view-k megnyitása után

```

183
184 void kdecapWidget::v4l_device_textChanged(const QString&)
185 {
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

3. ábra Elágazások összevonva (fent) és kibontva (lent)

szabaddan felcserélhető. Az egyes dock-okhoz ún. View-k tartoznak, amelyek címkeire 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 elérhető 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 httdig 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 koncentrálnunk.

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 szóképítő 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.

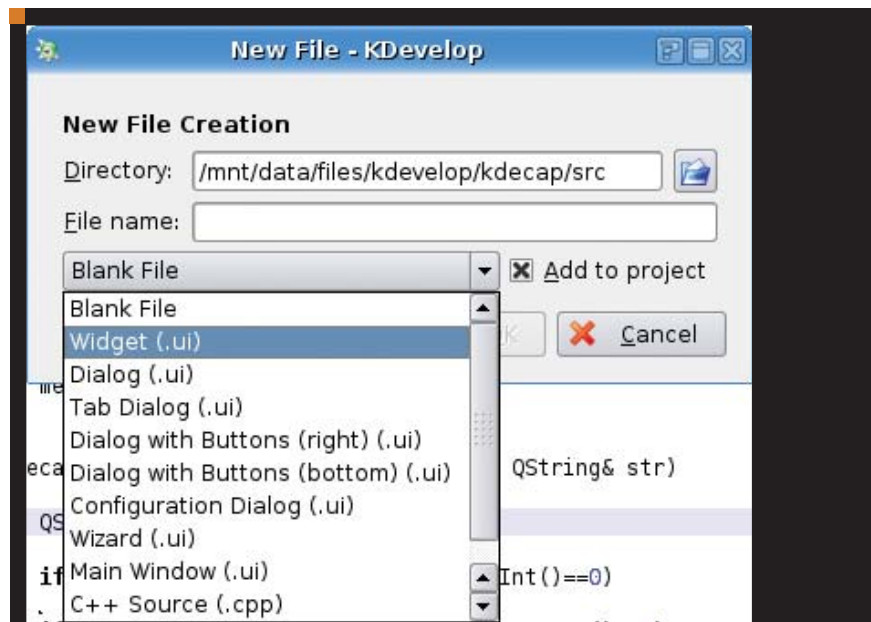
Ez utóbbit a *File->New* menüpontban tehetjük (4. ábra).

A felületeket tartalmazó fájlok *.ui* kiterjesztéssel jelennek meg a projektünkben. A baloldali *dock->File Groups view*-jában a *User Interface* alatt található. Az *.ui* fájlra kattintva megjelenik a szerkesztő, a *KDevelop* szerves részeként (5. ábra).

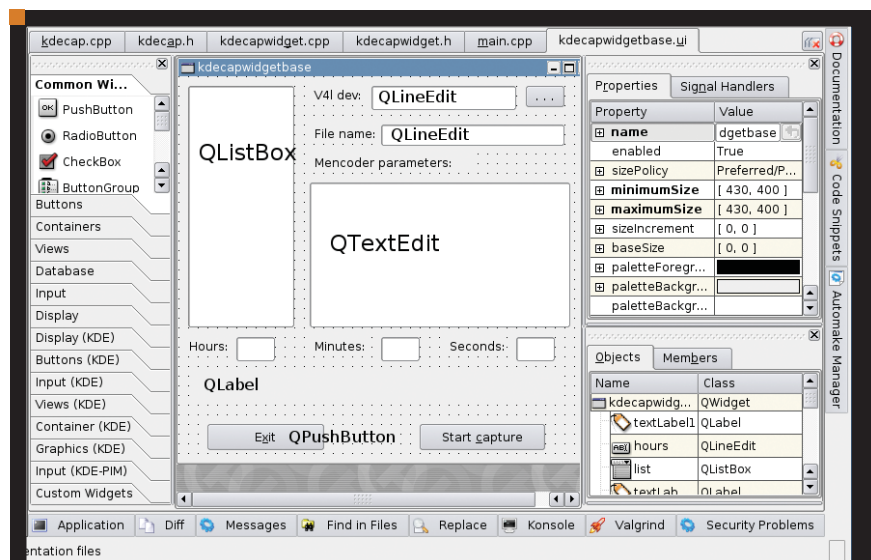
A felületszerkesztőben bal oldalon elérhetjük a felhasználható számos *widget*-et, amelyeket elhelyezhetünk a készülő felületen, egy-egy lehelyezett *widget*-re kattintva pedig jobb oldalon szerkeszthetjük a hozzá tartozó paramétereket és tulajdonságokat. Az egyes *widget*-ek eseményeit (például gomb 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

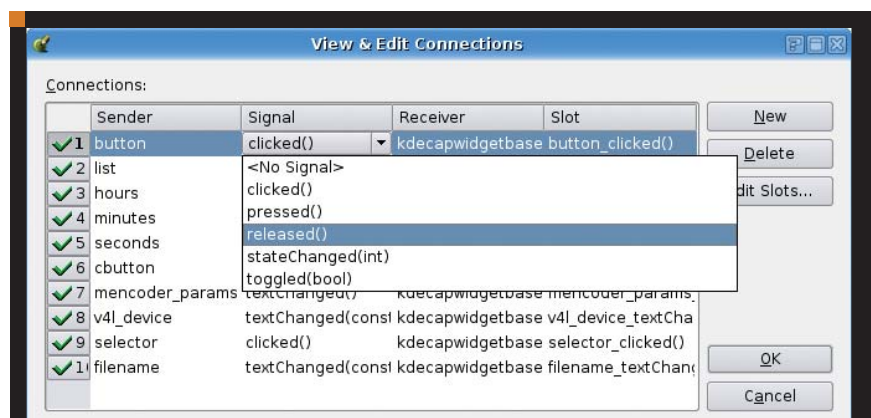
Szabályos kifejezések (*regexp*) kezelésekor (jellemzően szövegfeldolgozási feladatok során) gyakran még haladó programozók is elővesznek egy



4. ábra Új felület-elem hozzáadása a projekthez

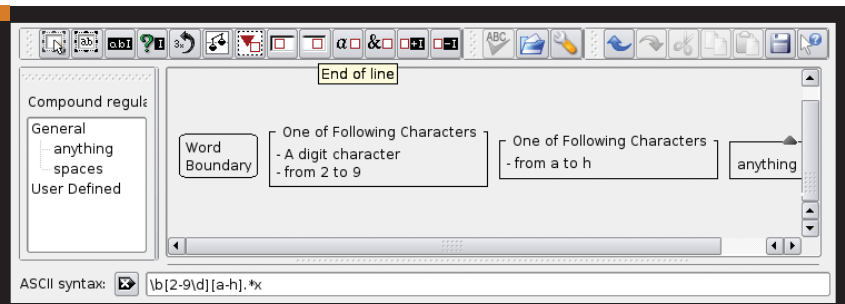


5. ábra A KDevelop beépített felület-tervezője

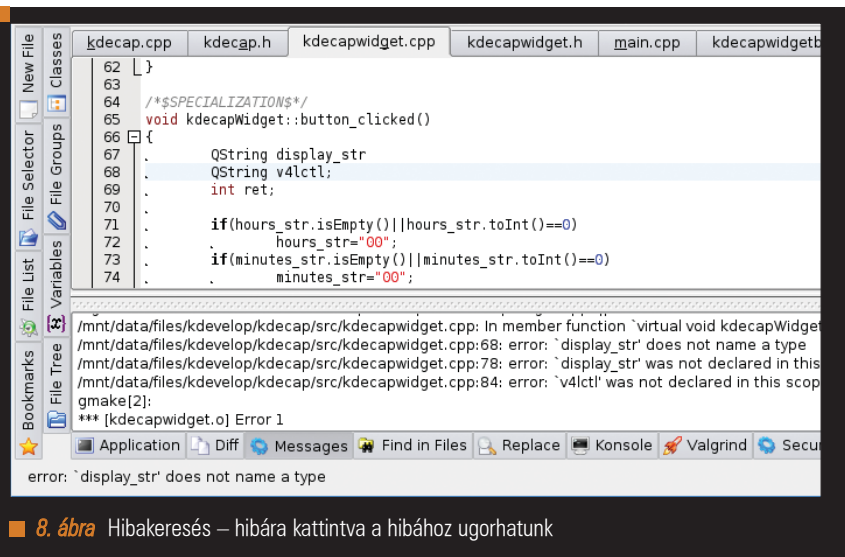


6. ábra Lehelyezett widget eseményéhez kezelőfüggvény rendelése

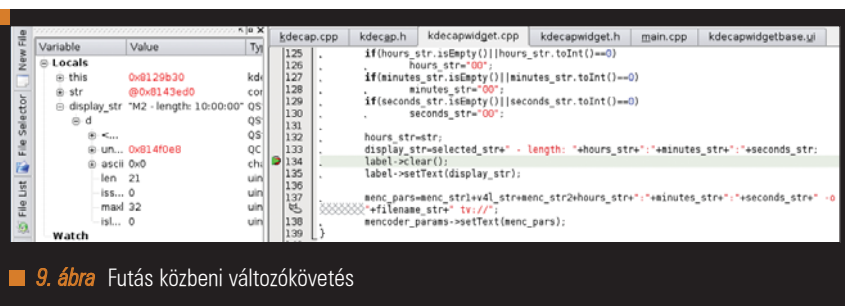
© Kiskapu Kft. Minden jog fenntartva



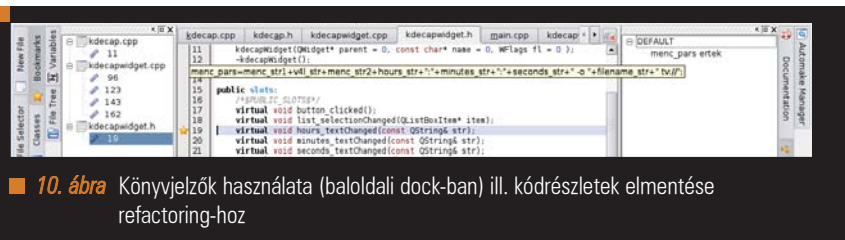
7. ábra Grafikus/vizuális reguláris kifejezés szerkesztő



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 megdöbbenően 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) segítő 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

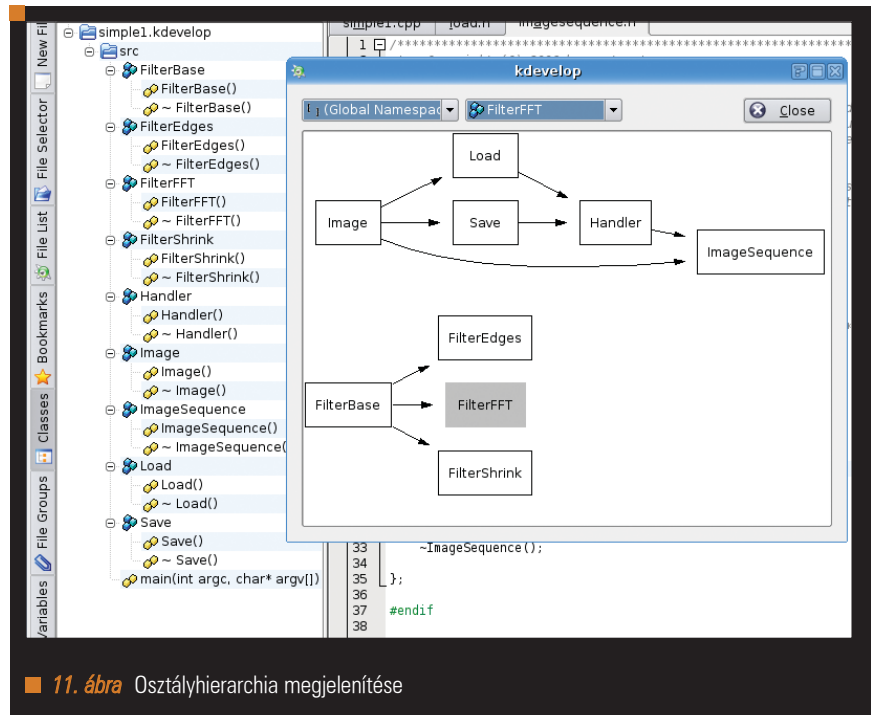
doxygen és graphviz alkalmazás használatával egy dialógusban megjeleníti készül alkalmazásunk osztályhierarchiáját (11. ábra). Új osztályokat a **Project->New Class** menüpontjával adhatunk hozzá a projekthez, amelyben az osztály minden tulajdonságát könnyen szerkeszthetjük. Az egyes osztályokat a baloldali **dock->Classes view**-jában érhetjük el.

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

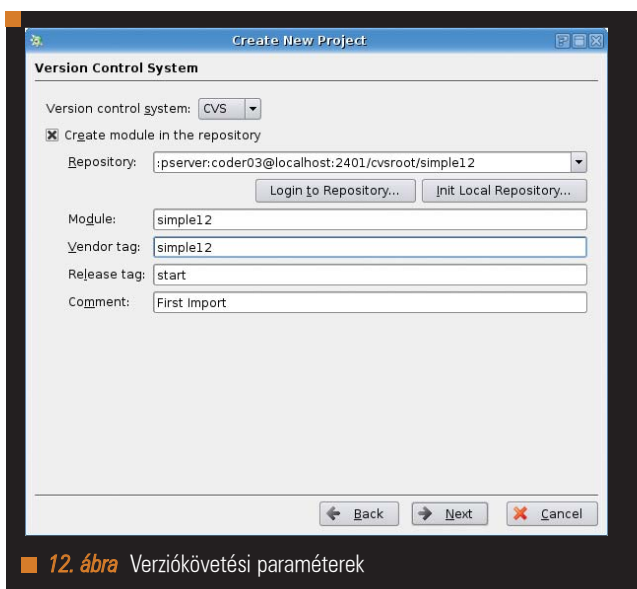
Mivel ez az írás nem a **KDE-s** alkalmazásfejlesztésről szól, hanem a **KDevelop** bemutatásáról, nem szándékozunk elmélyülni a **KDE/QT** könyvtárak is programozási interfészek (**API-k**) lelkivilágában. Tekintsük inkább át egy egyszerű



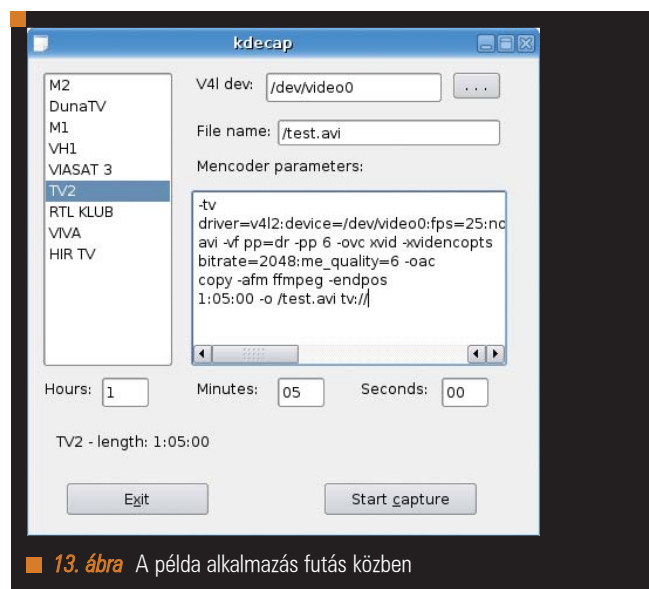
11. ábra Osztályhierarchia megjelenítése

dialógus-alapú **KDE-s** alkalmazás elkészítésének egyes lépéseit. Nevezük 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 lenyomunk egy gombot, ezekről mind tudnunk kel. Egy **widget**-re



12. ábra Verziókövetési paraméterek



13. ábra A példa alkalmazás futás közben

© Kiskapu Kft. Minden jog fenntartva

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.

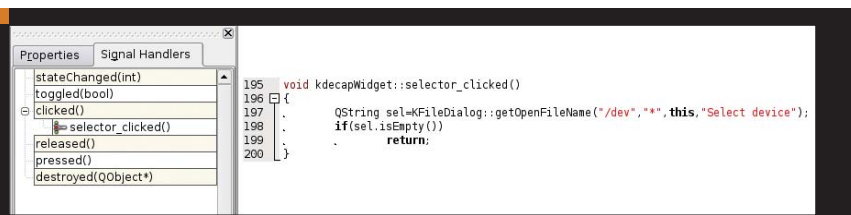
Például ha a *tuner* eszköz mellett „...” feliratú gombra kattintanak, akkor nyissunk egy dialógus ablakot amiben a felhasználó kiválaszthatja a *tuner* eszközt amiről rögzíteni szeretne. Az ezt kezelő függvény a 14. ábra jobb oldalán látható, a megnyitott dialógus pedig a 15. ábrán.

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 rendeljük a gomb *clicked* eseményéhez egy függvényt és nevezük *button_clicked*-nek, ahogy az a 6. ábrán látható. Ekkor létrejön a kezelőfüggvény és oda is ugrik a kurzor (16. ábra).

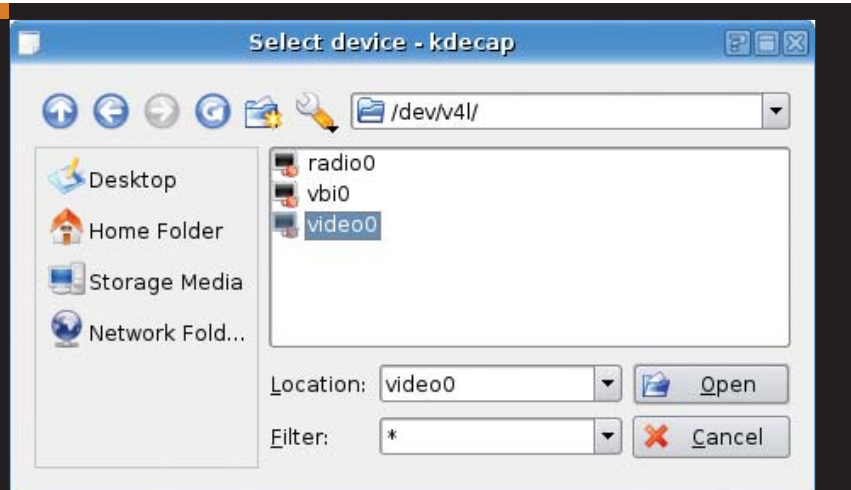
A KDE alapjait jelentő QT könyvtárakból használható osztályok, függvények, a KDE fejlesztői könyvtárainak függvényei a vonatkozó dokumentációkban mind elérhetők, ill. a cikk végi hivatkozásokban is megtalálhatók.

Összefoglalás

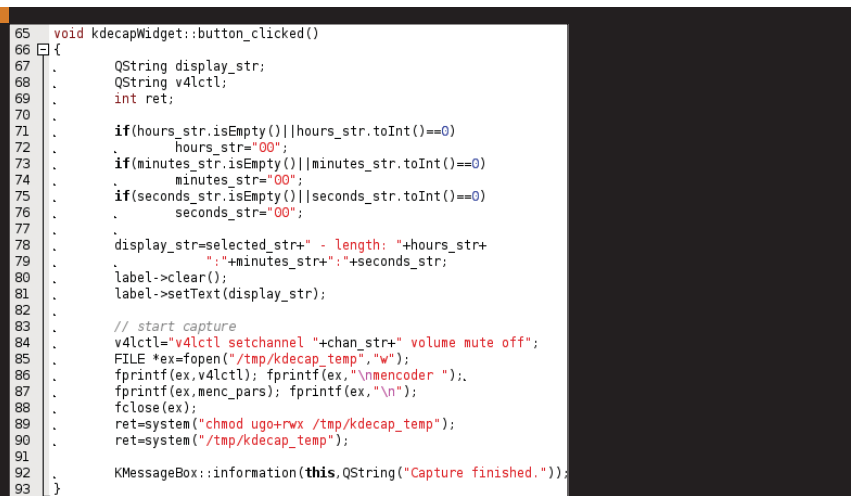
Röviden, de megpróbáltunk a *KDevelop* legfontosabb tulajdonságaiba betekintést adni, elsősorban C++ programozás esetén. Az utóbbi években, és leginkább a 3.x verziók megjelenése óta a *KDevelop* már kinőtt a hobbi-programozók köréből, és a megbízhatóan használható számos integrált eszköz és funkció 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



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