QtE5 – изучаем D и Qt-5 в комфортной графической среде

Немного лирики или с чего всё началось. Глядя на красивый KDE в Linux, мне хотелось научится писать программы и для него. Так я узнал о Qt. Всё хорошо, потихоньку изучаю, но «давит» гигантизм и запутанность C++. Хочется чего то компактного, быстрого и интелектуального. На глаза попадается D созданный Уолтером Брайтом. В своё время я начал изучение C++ с Zortech C++ (позже ставшего Symantec C++, а затем Digital Mars C++) и ни каких проблем с ним я не имел. Начал изучение D, который отлично работает в Windows/Linux, но вот проблема, хочется доступа к Qt. В инете есть QtD, но сколько я не пытался его реанимировать, ничего не получилось. Вот и решил я сделать что то для себя, благо ни куда не торопился.

QtE5 – это биндинг (обертка) функций Qt-5 в функции D. Сразу подчеркиваю, что Qt огромен, а QtE5 всего лишь очень маленькая его часть. Да, я постоянно дописываю новые функции, по мере изучения Qt и по мере необходимости в них, так как все свои проекты пишу на D + QtE5 + Qt. На момент написания статьи функций Qt в QtE5 около 400 штук.

QtE5 проверена и работает на Windows 32 (Win XP, 7, 10), Windows 64 (Win 7, 10), Linux 32/64 (Fedora 23), Mac Os X 64 (Хакинтош Mavericks 10.9).

Примеры ниже – это Win 32 (Win 7). Для работы нам понадобится D (dmd) и RunTime Qt-5. Устанавливать Qt из дистрибутива нет необходимости, если вы не собираетесь расширять QtE5. Скачать QtE5 + RunTime Qt-5 можно с репозитория гитхаба. Для понимания работы исходного кода желательно иметь представление о работе Qt. Имена методов и классов в QtE5 повторяют имена в Qt, что позволяет пользоваться справочной системой самой Qt.

Первое приложение – классика. Привет мир! Сохраним текст в файле ex2.d

```
1. // ex2.d
2. import qte5;               // Подключим QtE5
3. import core.runtime;       // Разбор аргументов ком.строки
4. int main(string[] args) {
5.         // Загрузим и инициализируем QtE5
6.         if (1 == LoadQt(dll.QtE5Widgets, true)) return 1;
7.         // Создадим объект приложения
8.         QApplication app = new QApplication(&Runtime.cArgs.argc,
   Runtime.cArgs.argv, 1);
9.         // Создадим виджет Label
10.        QLabel lb = new QLabel(null);
11.        // Добавим в него текст (поддерживается HTML) и обобразим
12.        lb.setText("<h1>Привет мир!</h1>").show();
13.        // Начнем цикл обработки графических событий
```

```
14.         app.exec();
15.         // Закончим программу
16.         return 0;
17. }
```

Компиляция и запуск:

Результат:

Тут даже комментировать нечего, все прозрачно и просто. Самое время перейти к более сложному примеру. Для себя я его называю:

> Нажми на кнопку и получишь результат!

Сразу несколько замечаний. Что мы хотим и должны получить. Будем изготавливать форму (простенькую) с двумя кнопками и надписью. При нажатии на кнопку будем выводить текст в консоль Windows русскими буквами и отображать окно сообщения.

Файл Ex5.d

```
1. // ex5.d
2. import qte5;            // Подключим QtE5
3. import asc1251; // Работа с русскими буквами в консоли Windows
4. import core.runtime;
5. import std.stdio;
6. // Обработчики событий
7. extern (C) {
8.         void on_acKn1() {
9.                 string s = "Нажата кнопка № 1";
10.                writeln(toCON(s)); msgbox(s);
11.        }
12.        void on_acKn2() {
13.                string s = "Нажата кнопка № 2";
14.                writeln(toCON(s)); msgbox(s);
15.        }
16. }
17. class Forma : QWidget {
18.        QVBoxLayout     vblAll;         // Общий вертикальный выравниватель
19.        QHBoxLayout     hblKn;          // Горизонтальный выравниватель для
    кнопок
20.        QLabel lb;                      // Надпись
21.        QPushButton kn1, kn2;           // Кнопки
22.        QAction acKn1, acKn2;           // События кнопок
23.        // _________________
```

```
24.         this(QWidget parent, QtE.WindowType fl) {
25.                 super(parent, fl);
26.                 resize(300, 100);
27.                 setWindowTitle("Пример работы кнопок");
28.                 // Горизонтальный и вертикальный выравниватели
29.                 vblAll = new QVBoxLayout();
30.                 hblKn  = new QHBoxLayout();
31.                 // Надпись
32.                 lb = new QLabel(this);
33.                 lb.setText("<h1>Работа с кнопками</h1>");
34.                 // Кнопки
35.                 kn1 = new QPushButton("Кнопка № 1", this);
36.                 kn2 = new QPushButton("Кнопка № 2", this);
37.                 // События
38.                 acKn1 = new QAction(this, &on_acKn1, null);
39.                 acKn2 = new QAction(this, &on_acKn2, null);
40.                 // Связываем нажатие кнопок с обработчиками событий
41.                 connects(kn1, "clicked()", acKn1, "Slot()");
42.                 connects(kn2, "clicked()", acKn2, "Slot()");
43.                 // Кнопки в горизонтальный выравниватель
44.                 hblKn.addWidget(kn1).addWidget(kn2);
45.                 // Всё в вертикальный выравниватель
46.                 vblAll.addWidget(lb).addLayout(hblKn);
47.                 // Вертикальный выравниватель в форму
48.                 setLayout(vblAll);
49.         }
50. }
51. int main(string[] args) {
52.         if (1 == LoadQt(dll.QtE5Widgets, true)) return 1;
53.         QApplication app = new QApplication(&Runtime.cArgs.argc,
    Runtime.cArgs.argv, 1);
54.         Forma f1 = new Forma(null, QtE.WindowType.Window);
55.         f1.show();
56.         app.exec();
57.         return 0;
58. }
```

Компиляция и запуск:

Результат:

В этом примере несколько интересных моментов. Первое, это обработчики события. Они обязательно должны быть объявляны как Extern (C) … Это связано с тем, что QtE5 может

работать с любым языком программтрования поддерживающим такие обратные вызовы. Второе, это

```
1. connects(kn1, "clicked()", acKn1, "Slot()");
```

Это стандартная связка сигналов и слотов в Qt. Очень большое достижение QtE5 в том, что с одной стороны она позволяет обойтись без метакомпилятора Qt, а с другой стороны сохраняет механизм сигналов/слотов, тем самым обеспечивая очень гибкую структуру взаимодействия различных частей программы друг с другом. И последний момент, на который хотелось бы обратить внимание. Это «сложность» компиляции приложения. Ни огромных Make файлов, ни длинных ключей компилятора, да и время компиляции – мгновенно. Чудес конечно не бывает, и за это приходится платить отсутствием гибкости в других моментах, но для обычных, рядовых программ это не существенно.

Считаю, что QtE5 хорошее дополнение к D, что бы сделать его изучение приятным и полезным.

От администрации блога: благодарим Геннадия Владимировича за прекрасную статью и очень важную для развития D библиотеку-связку с Qt5!

1