QtE5 - обёртка Qt-5 для D

1. Введение

Библиотека **QtE5** является результатом моих настойчивых попыток воспользоваться возможностями Qt с языка forth, а потом и с D. Она не является полным "зеркалом"реального Qt-5, но вполне работоспособна (проверена) на Qt-5.5 на Windows 32/64 (Win XP, 7, 10), Linux 32/64 (Fedora KDE 23), Mac OSX 64 (10.9 Maveric). Работая над ней хотелось избавится от гигантизма полностью инсталированного Qt (4 Gb) и C++ (3 Gb). Думаю, что цель достигнута. Имея dmd + QtE5 + RunTime Qt-5 вполне можно делать реальные приложения.

Особенно она интересна для начинающих программистов D. Лаконичность, простота сборки, скорость работы и колоссальные возможности :-)

Hello world - классика!

```d
// Файл ex2.d - пример Hello world!
// --=--= компиляция ---=---
// dmd ex2 qte5

import qte5;
import core.runtime;

int main(string[] args) {
        // Загрузить библиотеку
        if (1 == LoadQt(dll.QtE5Widgets, true)) return 1;
        // Создать приложение
        QApplication app = new QApplication(&Runtime.cArgs.argc, Runtime.cArgs.argv, 1);
        // Создать Label окошко
        QLabel lb = new QLabel(null);
        // Вписать в него текст (поддерживает HTML)
        lb.setText("<h1>Привет мир!</h1>").show();
        app.exec();
        return 0;
}
```

Компиляция: dmd ex2 qte5

Обратите внимание на “сложнось” сборки приложения. Ни каких километровых ключей, ни огромных Make файлов. Скорость компиляции - мгновенно. Ещё один плюс, это очень подробная документация по Qt.

2. Установка

Для работы с QtE5 нет необходимости устанавливать полную версию Qt-5. Достаточно установить RunTime версию, что есть просто набор нескольких *.DLL (*.so).

3. QtE5.QSlot() - замена метакомпилятора C++

Архитектура QtE5 значительно отличается от Qt. Нет метакомпилятора и соответственно нет возможности сосздавать новые слоты. По этому был введен новый (не существующий в Qt) класс. Это **QSlot()**. Оснавная задача этого класса, предостовлять нам готовые объекты (экземпляры) с уже имеющимся набором готовых слотов. Каждый экземпляр QSlot содержит несколько готовых слотов. **Слоты QtE5 предопределены и статичны.** Они определены в qtewidget.h.

Благодоря тому, что слоты эти предопределены, мы можем использовать их в качестве приёмщиков сигналов. Так как QtE5 всё время дописывается, то и набор готовых слотов может отличатся от описанного здесь. Главное, что бы была понятна идея и тогда разобраться в работе QSlot будет просто.

Имена слотов предопределены и могут быть использованы в аргументах connect(). Обратите внмание, что имена слотов в кавычках. Они передаются в аргументах connect именно как строки. На текущий момент есть слоты:

1) “Slot()” - Простой слот.

2) “SlotN()” - Слот имеет дополнительный параметр.

3) “Slot_Bool(bool)” - Слот перекидывает с сигнала bool параметр

4) “Slot_Int(int)” - Слот перекидывает int

Что есть слот в терминах Qt? Это функция, которую можно вызвать из самого Qt, используя сигналы или явно как обычную функцию. Фактически, это функция обратного вызова (CallBack).

Хорошо, функция вызвана, а что дальше? Как вызвать нашу функцию на D?

Что бы была возможность из предопределенного слота вызвать нашу функцию, мы передаём в слот её **адрес**. Каждый экземпляр QSlot хранит только один адрес вызываемой функции. Сам слот, устроен так, что он во время своего выполнения вызывает нашу функцию используя хранящийся адрес.

```
Qt ---call---> Слот{QSlot QtE5} ---call---> НашаФункция()
```

Значит, экземпляр QSlot должен хранить в себе адрес вызываемой функции. И он хранит в себе такой адрес. Посмотим на конструктор слота.

```
QSlot slotLe1Cr = new QSlot(&testLE);

где &testLE - есть ни что иное, как адрес нашей функции обработчика

extern (C) void testLE() {
..........
}
```

Обратите внимание на определение нашего обработчика.

- -

extern (C) void testLE() ... обязательны в определении!!!

- -

Зачем так сделано? Дело в том, что QtE5 может быть использована с любого языка программирования поддерживающих формат C обратного вызова. Например из C, C++ и forth.

Хорошо. А как быть с тем, что сигналы с Qt могут иметь параметры?

Например события? Они имеют параметр (указатель) на экземпляр QEvent. Вот тут можно посмотреть вверх на имена предопределённых слотов.

Например, если мы в аргументах connect() напишем “Slot_int(int)” то будет задействован (вызван) слот который возьмет из сигнала параметр

типа int и вызовет нашу функцию с этим параметром. Он как эстафету передаст параметр от сигнала к нашему обработчику. Конечно жаль, что нет возможности не пользуясь C++ определить слот с другим набором аргументов. Но, как показала практика, набора в десяток различных вариантов покрывают почти все сигналы.

```
.....

extern (C) void onKn1() {
      writeln(toCON("Нажата кнопка N1"));
}

.....

void test {
.....
      // Определим кнопку и подпишем её
      QPushButton kn1 = new QPushButton("Это я, кнопка N1");
      // Определим слот для обработки кнопки,
      // предав ему адрес обработчика
      QSlot slotKn1 = new QSlot(&onKn1);
      // Свяжем сигнал Qt кнопки с нашим слотом
      // Обратите внимание на строковые параметры!!!
      // Именно они имена сигналов и слотов
      kn1.connect(
            kn1.QtObj, MSS("clicked()", QSIGNAL),
            slotKn1.QtObj, MSS("Slot()", QSLOT)
      );
      // Теперь, при нажатии на кнопку, произойдет вызов нашего слота
      // который вызовет обработчик onKn1()
.....
}
```