Кросскомпиляция Qt 5.9 и Qt 5.12 для Raspberry Pi 1, Raspberry Pi 3 (B+)

Пока в разгаре новогодние праздники захотелось попробовать в действии Raspberry Pi (Здесь для модели 3 B+, но так же подходит и для 1,2), а именно чего-нибудь для неё написать, хотя бы Hello World с помощью Qt. Ставить весь Qt на саму малинку как-то долго, да и пока на ней компилируется простейшая программа можно упиться в усмерть, поэтому будем настраивать кроссс-компиляцию, что бы всю разработку вести на своём пк (Ubuntu 16.10 x64).

Приступаем.

Обновлено для Qt 5.9.1 и Raspbian 2017-09-08 Stretch для Raspberry PI3 Model B
Обновлено для Qt 5.12.2 и Raspbian 2018-11-13 Stretch для Raspberry PI3 Model B+

1. Первым делом необходимо скачать образ ОС для малинки, это будет raspbian (можно и Lite (без X), тогда получится сделать «kiosk mode») и залить её на SD карточку, которая уже должна быть воткнута.

```
mkdir ~/Projects/RaspberryPI
cd ~/Projects/RaspberryPI
wget https://downloads.raspberrypi.org/raspbian_latest # Обычный
#Если нужен облёгчённый, бек иксов: wget https://downloads.raspberrypi.org/raspbian_lite_latest
unzip raspbian_latest
#unzip raspbian_lite_latest
sudo dd if=2018-11-13-raspbian-stretch.img of=/dev/mmcblk0 bs=4M
#sudo dd if=2018-11-13-raspbian-stretch-lite.img of=/dev/mmcblk0 bs=4M
sync
```

Что бы узнать адрес флэшки, наберите *lsblk*

Если так случалось, что нет картридера, то залить прошивку можно и на винде с помощью win32diskimager (запускать от рута)

2. Когда заливка завершиться, вытаскиваем СДшку и загружаем с неё малинку.

Теперь её немножко надо настроить, переходим *Menu -> Preferences -> Raspberry Pi configuration*.
Либо в консольке(*Menu->Accessories->Terminal*) :

```
sudo raspi-config
```

Расширяем систему на весь объём флэшки: Advanced options->expand filesystem,
включаем ssh в Interfacing options,
подрубаемся к wifi,
в Advanced options->GL Driver включаем GL (Full KMS) и перезагружаемся.

Вбиваем в консольке *ifconfig*, запоминаем айпишник, на этом пока что всё.

Кстати, о настройке WIFI raspbian из консоли (если без графического интерфейса, т.е. lite версия):

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
#Дописываем:
network={
    ssid="The_ESSID_from_earlier"
    psk="Your_wifi_password"
}

#Выходим с сохранением и перезагружаемся:
sudo reboot
```

3. Приконнектимся к малинке по ssh, раскомментим получение исходников пакетов, установим нужные пакеты и создадим нужные папки

```
ssh pi@192.168.2.101 (пароль по дефолту "raspberry")
sudo nano /etc/apt/sources.list  #расскомментить строчку с deb-src
sudo apt-get update
sudo apt-get upgrade
sudo rpi-update #Обновим ядро и прочее основное
reboot
sudo apt-get install -y libudev-dev libinput-dev libts-dev libxcb-xinerama0-dev libxcb-xinerama0
sudo apt-get install -y libfontconfig1-dev libfreetype6-dev libx11-dev libxext-dev libxfixes-dev libxi-dev li
sudo apt-get install -y libxcb1 libxcb1-dev libx11-xcb1 libx11-xcb-dev libxcb-keysyms1 libxcb-keysyms1-dev li
sudo apt-get install -y build-essential libgl1-mesa-dev freeglut3 freeglut3-dev mesa-common-dev libglapi-mesa
sudo apt-get install -y libwayland-egl1-mesa libraspberrypi-dev  freeglut3-dev freeglut3  libegl1-mesa libeg
sudo apt-get install -y libraspberrypi0 libva-egl1 libegl1-mesa libegl1-mesa-drivers gegl
sudo apt-get install -y libglfw3-dev libgles2-mesa-dev
sudo apt-get build-dep qt4-x11 #ради зависимостей
sudo apt-get build-dep libqt5gui5 #ради зависимостей
#Чувствую, большую половину из этого не надо, но я не знаю какую точно т.к. решал проблемы с OpenGL и прочичи
sudo mkdir /usr/local/qt5pi
```

```
sudo chmod -R 777 /usr/local/qt5pi
```

4. Так, возвращаемся к хосту, качаем тулчейн:

```
mkdir raspi
cd raspi
git clone https://github.com/raspberrypi/tools
```

5. Создаём sysroot и через rsync синхронизируем его с малиновым, что бы оттуда взять заголовочники, либы и компилятор

```
IP=192.168.2.101 # айпишник малинки
rsync -avz pi@$IP:/lib sysroot # опять идём пить кофе
rsync -avz pi@$IP:/usr/include sysroot/usr
rsync -avz pi@$IP:/usr/lib sysroot/usr # можно сходить просраться
rsync -avz pi@$IP:/opt/vc sysroot/opt
```

6. Дальше необходимо поправить символьные ссылки, что бы они были относительно нашей скопированной sysroot, для этого есть готовый скриптик, качаем и запускаем:

```
wget https://raw.githubusercontent.com/riscv/riscv-poky/master/scripts/sysroot-relativelinks.py
chmod +x sysroot-relativelinks.py
./sysroot-relativelinks.py sysroot
```

7. Теперь можно склонировать Qt, сконфигурировать и запустить на компиляцию

```
git clone git://code.qt.io/qt/qt5.git QtSources
cd QtSources
perl init-repository
git checkout 5.12 # Если нужно 5.9, то проверял с 5.9.1
git submodule update --recursive


BASEPATH=~/Projects/RaspberryPI/raspi # базовый путь, где все наши манипуляции происходят (без слэша на конце
./configure -skip wayland -skip script -skip webengine -no-pch -no-kms -no-xcb -no-use-gold-linker -nomake te
make -j4 # и поспим часик (4 - количество ядер проца, для ускорения)
make install
```

Чтиво по некоторым фичам: https://doc.qt.io/qt-5/embedded-linux.html, https://doc.qt.io/archives/qt-5.8/configure-options.html, https://doc.qt.io/qt-5/linux-requirements.html, https://doc.qt.io/archives/qt-4.8/configure-options.html

Если возникли ошибки при сборке и нужно переконфигурировать с другими параметрами, не забудьте удалить предыдущий результат, набрав «*git clean -dxf*» и «make clean» иначе будет ещё хуже =)
Для Qt < 5.9.1 нужен *-device linux-rpi3-g++*
При конфигурации отключён QtWebKit! Для Qt < 5.10 нужно указывать «-skip webkit» Кому нужен уберите «-skip webengine».
Возможно, нужно так же указать -qt-xcb (и убрать -no-xcb), что бы нормально работали QWidgets, у меня же все приложения только с QML. Так же нужно указать «-platform xcb» если у вас gui версия (обычная, а не lite)
При ошибке с JavaScriptCore/wtf/Platform.h:370:6: error: #error «Not supported ARM architecture» добавьте к конфигурации «-skip script» или добавьте к make флаг:»*make CFLAGS=»${CFLAGS}-D__ARM_ARCH_7M__»*» (взял из /3rdparty/javascriptcore/JavaScriptCore/wtf/Platform.h)
При ошибке с PCRE2: *PCRE2_CODE_UNIT_WIDTH, LINK_SIZE и прочие мессаджы с ней связанные* (отвечает за регулярные выражения) можно либо отключить (в конфиге «-no-pcre»), но почему-то у меня не было такого параметра,
либо прописать все дефайны вручную: *make CFLAGS=»${CFLAGS} -ldl -DPCRE2_CODE_UNIT_WIDTH=16 -DHAVE_INTTYPES_H=1 -DHAVE_MEMMOVE=1 -DHAVE_LIMITS_H=1 -DHAVE_MEMORY_H=1 -DHAVE_STDINT_H=1 -DHAVE_STDLIB_H=1 -DHAVE_STRING_H=1 -DLINK_SIZE=2 -DMATCH_LIMIT=10000000 -DMATCH_LIMIT_RECURSION=10000000 -DMAX_NAME_COUNT=10000 -DMAX_NAME_SIZE=32 -DNEWLINE_DEFAULT=2 -DPARENS_NEST_LIMIT=250 -DSUPPORT_UNICODE»* причина этому — почему-то не подключается файл config.h (Расположен qtbase/src/3rdparty/pcre2/src/config.h), где прописаны все эти дефайны.
При ошибке с zlib, можно её отключить -no-zlib (скорее всего каких-то либ на хосте нехватает, но т.к. мне было не важно, не стал разбираться, остальных проблем хватило)
При ошибке «ERROR: The OpenGL functionality tests failed!
You might need to modify the include and library search paths by editing
QMAKE_INCDIR_OPENGL[_ES2],
QMAKE_LIBDIR_OPENGL[_ES2] and QMAKE_LIBS_OPENGL[_ES2] in the mkspec for your platform.» ИЛИ «In function `QEGLPlatformContext::getProcAddress(char const*)':
qeglplatformcontext.cpp:(.text+0xa4): undefined reference to `dlsym'» для Raspbian Stretch нужно подредактировать *nano ./qtbase/mkspecs/devices/linux-rasp-pi3-g++/qmake.conf* и изменить названия либ «*-lEGL*» и «*-lGLESv2*» на «*-lbrcmEGL*» и «*-lbrcmGLESv2*» так как названия в */opt/vc/lib/* отличаются.

Возможно лучше будет использовать в -device «linux-rasp-pi3-**vc4**-g++»

Так же возможна проблема с тем, что EGLFS Raspberry Pi ..... no, в таком случае файлик нужно поправить, указав правильные пути.
Моя версия:
show

В случае ошибки «*error: invalid use of incomplete type 'X509 {aka struct x509_st}'*«, то это баг Qt: https://bugreports.qt.io/browse/QTBUG-52905. Исправления будут в версии 5.10, так что либо отключайте ssl при конфигурации: «-no-openssl» либо даунгрейдите openssl до 1.0

8. Теперь закинем на малинку скомпилированные библиотеки и заголовочники Qt:

```
cd ../
rsync -avz qt5pi pi@$IP:/usr/local
```

9. Ну и можно собрать пример и закинуть на малинку:

```
cd QtSources/qtbase/examples/opengl/qopenglwidget
$BASEPATH/qt5/bin/qmake
make
scp qopenglwidget pi@$IP:/home/pi
```

10. На девайсе необходимо дать знать линковщику о наших либах, а так же создать qt.conf в папке, откуда будем запускать все Qt приложения:

```
ssh pi@192.168.2.101 (пароль по дефолту "raspberry")
echo /usr/local/qt5pi/lib | sudo tee /etc/ld.so.conf.d/00-qt5pi.conf
echo QT_PLUGIN_PATH=/usr/local/qt5pi/plugins/ | sudo tee -a /etc/environment
echo QT_QPA_FONTDIR=/usr/share/fonts/truetype/dejavu | sudo tee -a /etc/environment
printf "[Paths]\nPlugins=/usr/local/qt5pi/plugins\nQml2Imports=/usr/local/qt5pi/qml" | sudo tee ~/qt.conf
cd /usr/local/qt5pi/lib
sudo ldconfig
```

При проблеме «*QFontDatabase: Cannot find font directory /home/pi/lib/fonts.
Note that Qt no longer ships fonts. Deploy some (from http://dejavu-fonts.org for example) or switch to fontconfig.*» нужно указать, где лежат шрифты, для этого добавим переменную окружения (выше уже добавлена) : «*echo QT_QPA_FONTDIR=/usr/share/fonts/truetype/dejavu | sudo tee -a /etc/environment*» видимо какую-то опцию забыл в конфиге добавить, наверное «*-fontconfig*«.

Перезагружаем малинку.

11. Но запускать ещё рано, у rasbian по дефолту грузиться mesa драйвер и opengl пахать не будет по нормальному, поэтому заставим её использовать нужный:

Не факт! Сначала всё таки лучше попробовать запустить =)

```
sudo rm /usr/lib/arm-linux-gnueabihf/libEGL.so.1.0.0 /usr/lib/arm-linux-gnueabihf/libGLESv2.so.2.0.0
sudo ln -s /opt/vc/lib/libEGL.so /usr/lib/arm-linux-gnueabihf/libEGL.so.1.0.0
sudo ln -s /opt/vc/lib/libGLESv2.so /usr/lib/arm-linux-gnueabihf/libGLESv2.so.2.0.0
sudo ln -s /opt/vc/lib/libEGL.so /opt/vc/lib/libEGL.so.1
sudo ln -s /opt/vc/lib/libGLESv2.so /opt/vc/lib/libGLESv2.so.2
```

Вот теперь наконец-то можно запустить пример! Он на малинке */home/pi/qopenglwidget*

Если не запускается, в консольке прописываем

```
export QT_LOGGING_RULES=qt.qpa.*=true
./qopenglwidget
```

И гуглим ошибки.

Если совсем не запускается, то прописываем:

```
ldd ./qopenglwidget
```

И смотрим по правильному адресу ли лежат библиотеки и всех ли хватает. Потом гуглим на тему ldconfig.

Если будет в выводе:

```
* failed to add service - already in use?
```

То подправим файлик «sudo nano /boot/config.txt», убрав «dtoverlay=vc4-kms-v3d» (или закомментировать) и перезагрузиться.

Возможно, стоит при конфигурации указать «-device linux-rasp-pi3-vc4-g++» (при необходимости подправить названия библиотек).

12. С этим разобрались, теперь настроим QtCreator что бы можно было компилить и запускать на малинке в один клик:

12.1. *Параметры->Устройства->Добавить*
Обычное Linux-устройство
название на свой вкус и цвет

вводим айпишник, логин и пароль
завершить

12.2. *Параметры->Сборка и запуск->Компиляторы->Добавить*
GCC -> C++
Название: Raspberry PI3 GCC
Путь: /home/pavelk/Projects/RaspberryPI/raspi/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/arm-linux-gnueabihf-g++

12.3. *Параметры->Сборка и запуск->Отладчики->Добавить*
Название: Raspberry PI3 GDB
Путь: /home/pavelk/Projects/RaspberryPI/raspi/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/arm-linux-gnueabihf-gdb

12.4. Параметры->Сборка и запуск->Профили Qt->Добавить
Название: Qt 5.9.1 Raspberry PI3
Путь: /home/pavelk/Projects/RaspberryPI/qt5/bin/qmake

12.5. *Параметры->Сборка и запуск->Комплекты->Добавить*
Название: Raspberry PI3
Тип устройства: Обычное Linux-устройство
Устройство: выбираем добавленное из первого шага
Компилятор C++: Raspberry PI3 GCC
Отладчик: Raspberry PI3 GDB
Профиль Qt: Qt 5.9.1 Raspberry PI3

12.6. Нажимаем «Применить».

14. Создаём новый проект, в *.pro файл добавляем:

```
INSTALLS        = target
target.path     = /home/pi
```

Компилим и после завершения проект должен запуститься на малинке.

Вот как-то так в общем =)

P.S. Большая часть была взята с https://wiki.qt.io/RaspberryPi2EGLFS с моими небольшими правками — думал всё сложнее будет =)

Рекомендую:

1. **Кросскомпиляция Qt 5.12 для Raspberry Pi 1,2,3 B+ под Windows**
2. **Ubuntu Qt5 не задан компилятор, not found -lGL, not found GL.h**

Share

B+, cross compile, Pi, qt, Qt 5.12, Raspberry, Raspberry PI