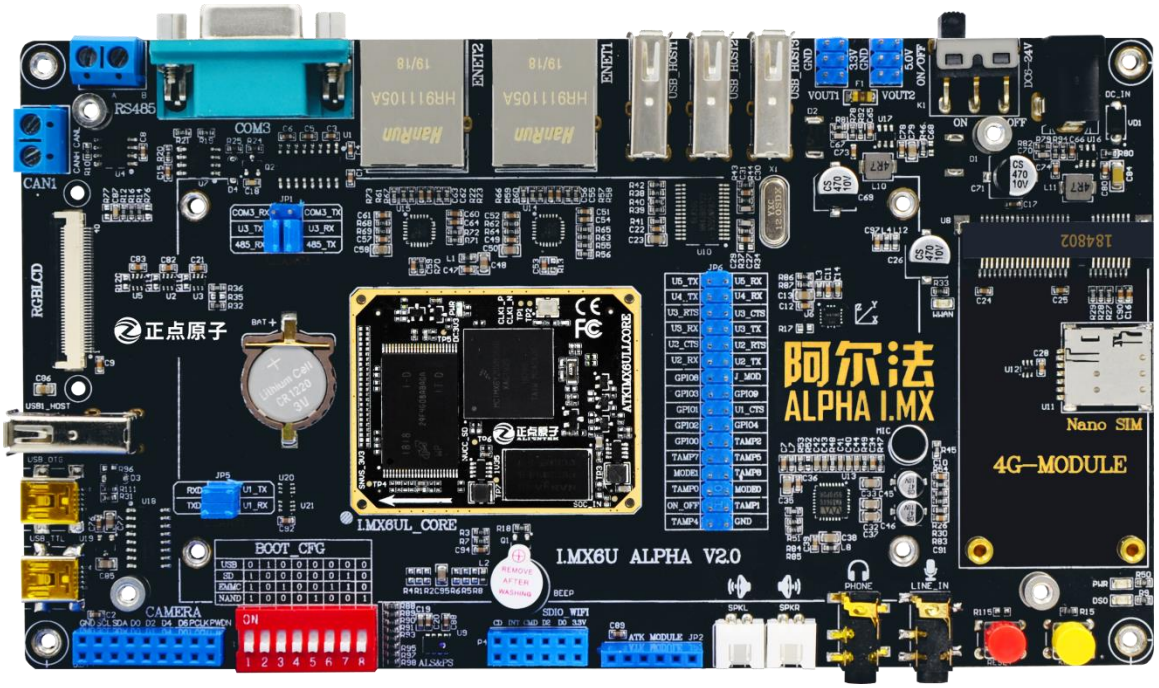
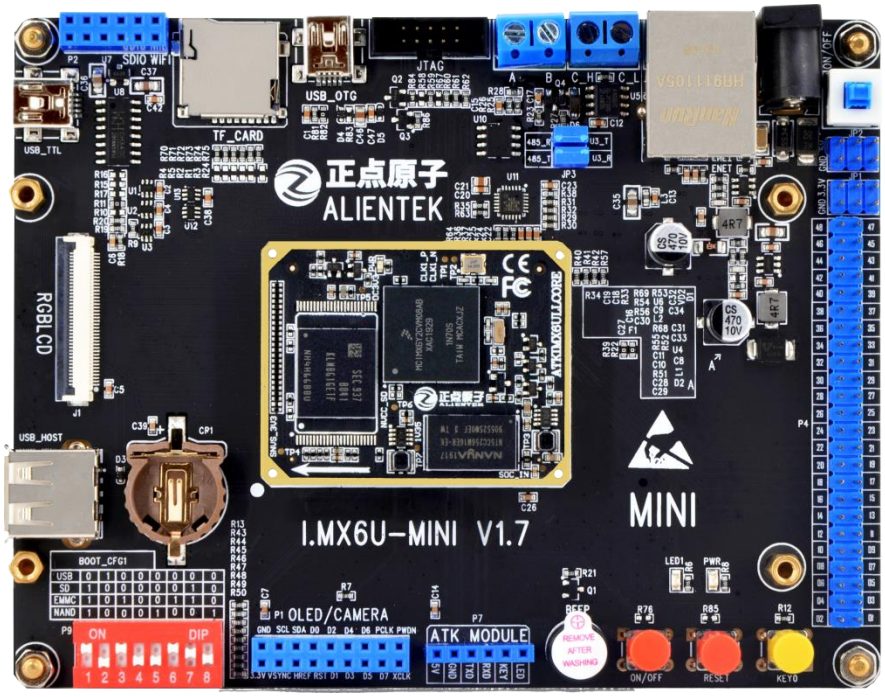
【正点原子】I.MX6U

移植 Qt4.8.4 V1.1





正点原子

正点原子公司名称 ：广州市星翼电子科技有限公司

原子哥在线教学平台：www.yuanzige.com

开源电子网 / 论坛 ：http://www.openedv.com/forum.php

正点原子淘宝店铺 ：https://openedv.taobao.com

正点原子官方网站 ：www.alientek.com

正点原子 B 站视频 ：https://space.bilibili.com/394620890

电话：020-38271790 传真：020-36773971

请关注正点原子公众号，资料发布更新我们会通知。

请下载原子哥 APP，数千讲视频免费学习，更快更流畅。

扫码关注正点原子公众号

扫码下载“原子哥”APP

文档更新说明

版本	版本更新说明	负责人	校审	发布日期
V1.0	初稿：	正点原子 Linux 团队	正点原子 Linux 团队	2020.11.21
V1.1	1.修改文档的格式	正点原子 Linux 团队	正点原子 Linux 团队	2021.06.19

目录

前言

Qt5 已经在战场上驰骋快 10 年了，在迎接 Qt6 时代的到来时，Qt4 玩家却忘不了以前的美好时光。早些年有些软件是 Qt4 开发的，在某些 Qt4 版本上稳定运行。Qt4 写的软件要在 Qt5 运行可能需要修改比较多内容。对于一个偌大的软件来说，改起来实在是令人头痛！由于正点原子出厂 I.MX6U 出厂系统的 Qt 版本是 5.6.2 版本的（2020.11.21 日后升级到 Qt5.12.9 LTS 版本），所以我们为正点原子 I.MX6U 移植 Qt4.8.4 给 Qt4 的忠实用户，如有错漏，请到正点原子论坛指正，或者联系本文档编写作者 QQ1252699831 指正错误。

本文档所使用的环境：

- Windows 7 64bits，也适用于 Windows 8-10。不建议用 Windows 32 位来开发，Windows 32 位支持的内存大小有限，系统性能有限。
- Ubuntu12.04，Ubuntu 建议使用 12.04 下载地址为（https://releases.ubuntu.com/precise/ubuntu-12.04.5-desktop-amd64.iso），因为 Qt4.8.4 是 2012 年发布的，所以那时 Ubuntu 版本只有 Ubuntu12。否则安装及编译环境不一样导致出错，请自行解决！
- 要求读者会使用 FileZilla、WinSCP 及 Windows Git 进行 Ubuntu 与 Windows 间互传文件的方法。

第一章 下载安装通用交叉编译器

编译 ARM 平台的 Qt 源码，那么我们需要使用 ARM 类型的的交叉编译器。

1.1 下载通用交叉编译器

我们要编译 Qt 源码，需要使用通用的交叉编译器，这里下载 Linaro 出品的交叉编译器，也就是正点原子 I.MX6U 嵌入式 Linux 驱动开发指南第 4.3 小节里推荐的 4.9 版本的编译器，这里重复写下载的方法。如果已经知道怎么安装或者已经安装可跳过第一章。下面是下载地址。https://releases.linaro.org/components/toolchain/binaries/4.9-2017.01/arm-linux-gnueabihf/。

图 1.1 1 选择对应系统版本的交叉编译器下载

请根据个人 Ubuntu 系统的位数，选择对应版本下载即可。也可以在我们正点原子 I.MX6U 开发板光盘 A-基础资料->5、开发工具->1、交叉编译器下找到下载好的交叉编译器。如下图。

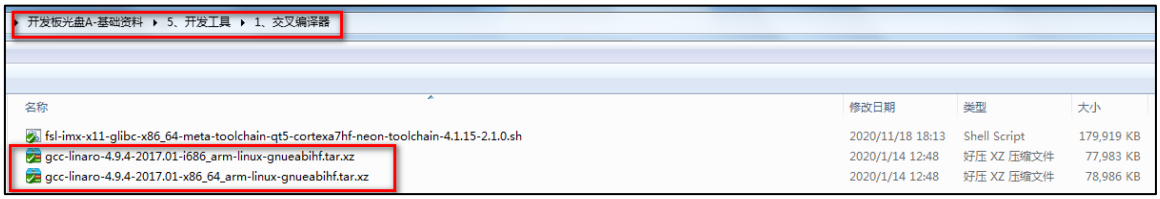


图 1.1 2 资料盘处的交叉编译器

1.2 安装通用交叉编译器

将上面 1.1 小节下载好的通用交叉编译器压缩包拷贝到 Ubuntu 虚拟机，解压进行安装。编者是 64 位的 Ubuntu。所以通用 FileZilla 或者 WinSCP 拷贝 gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf.tar.xz 到 Ubuntu 虚拟机。如下图，编者已经拷贝到“家”目录下。

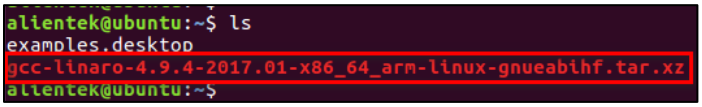


图 1.2 1 拷贝交叉编译器到家目录

在 Ubuntu 目录下创建/usr/local/arm 文件夹，为下面安装到/usr/local/arm 这个文件夹做准备。

```
sudo mkdir /usr/local/arm
```

解压交叉编译器压缩包至/usr/local/arm 目录下，稍等片刻，解压完成如下。

```
sudo tar xf gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf.tar.xz -C /usr/local/arm/
```

```
alientek@ubuntu:~$ sudo mkdir /usr/local/arm
[sudo] alientek 的密码:
alientek@ubuntu:~$ sudo tar xf gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf.tar.xz -C /usr/local/arm/
alientek@ubuntu:~$
```

图 1.2 2 解压交叉编译器

使用 vi 指令编辑/etc/profile 这个文件。

```
sudo vi /etc/profile
```

打开/etc/profile 以后，在末尾添加如下所示内容。

```
export PATH=$PATH:/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin
```

添加完成如下图，保存退出，重启系统。

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ "$PS1" ]; then
  if [ "$BASH" ] && [ "$BASH" != "/bin/sh" ]; then
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi
export PATH=$PATH:/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin
~
```

图 1.2 3 在/etc/profile 下添加全局环境变量

1.3 验证通用交叉编译器

要使用此编译器，还要在 Ubuntu 上安装一些库。

```
sudo apt-get install lsb-core lib32stdc++6
```

在 1.2 小节修改环境变量重启系统后，在终端输入 arm-linux-gnueabihf-gcc -v 来查看安装的交叉编译器版本号。看到如下结果，表明成功！

```
arm-linux-gnueabihf-gcc -v
```

```
alientek@ubuntu:~$ arm-linux-gnueabihf-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabihf-gcc
COLLECT_LTO_WRAPPER=/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin/../libexec/gcc/arm-linux-gnueabihf/4.9.4/lto-wrapper
Target: arm-linux-gnueabihf
Configured with: /home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/snapshots/gcc-linaro-4.9-2017.01/configure SHELL=/bin/bash --with-mpc=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu --with-mpfr=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu
--with-gmp=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu --with-gnu-as --with-gnu-ld --disable-libmudflap --enable-lto --enable-objc-gc --enable-shared --without-included-gettext --enable-nls --disable-sjlj-exceptions --enable-gnu-unique-object --enable-linker-build-id --disable-libstdcxx-pch --enable-c99 --enable-clocale=gnu --enable-libstdcxx-debug --enable-long-long --with-cloog=no --with-ppl=no --with-isl=no --disable-multilib --with-float=hard --with-mode=thumb --with-tune=cortex-a9 --with-arch=armv7-a --with-fpu=vfpv3-d16 --enable-threads=posix --enable-multiarch --enable-libstdcxx-time=yes --with-build-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/sysroots/arm-linux-gnueabihf --with-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu/arm-linux-gnueabihf/libc --enable-checking=release --disable-bootstrap --enable-languages=c,c++,fortran,lto --build=x86_64-unknown-linux-gnu --host=x86_64-unknown-linux-gnu --target=arm-linux-gnueabihf --prefix=/home/tcwg-buildslave/workspace/tcwg-make-release/label/docker-trusty-amd64-tcwg-build/target/arm-linux-gnueabihf/_build/builds/destdir/x86_64-unknown-linux-gnu
Thread model: posix
gcc version 4.9.4 (Linaro GCC 4.9-2017.01)
alientek@ubuntu:~$
```

图 1.3 1 验证交叉编译器

第二章 获取和编译 tslib

要想 Qt 支持触摸还需要编译 tslib，以生成触摸相关插件。

2.1 获取 tslib

获取 tslib 的源码，git 地址为 https://github.com/kergoth/tslib。tslib 源码已经放到开发板光盘中路径为：开发板光盘 A-基础资料->1、例程源码->7、第三方库源码->tslib-1.21.tar.bz2。将压缩包发送到 Ubuntu 中并解压，得到名为“tslib-1.21”的目录，此目录下就是 tslib 源码。注意 github 有可能被更新源码导致出现 bug。请使用资料盘里的 tslib-1.21 源码。

拷贝至 Ubuntu 虚拟机目录下，将其解压，得到 tslib-1.21 文件夹。

```
tar xf tslib-1.21.tar.bz2
```

进入 tslib-1.21 文件夹下。

```
cd tslib-1.21
```

```
alientek@ubuntu:~$ cd tslib-1.21/
alientek@ubuntu:~/tslib-1.21$ ls
acinclude.m4  cmake           config.sub    doc          m4           plugins     THANKS
aclocal.m4    CMakeLists.txt  configure     etc          Makefile.am  README      tools
AUTHORS       compile         configure.ac  INSTALL      Makefile.in  README.md   tslib.pc.in
autogen.sh    config.guess    COPYING       install-sh   missing      src
ChangeLog     config.h.in     depcomp       ltmain.sh    NEWS         tests
alientek@ubuntu:~/tslib-1.21$
```

图 2.1 1 进入解压后的 tslib 源码目录下

2.2 编译 tslib

生成 Makefile，还需要安装以下软件。

```
sudo apt-get update
sudo apt-get install autoconf automake libtool
```

执行 autogen.sh 生成 Makefile，以编译源码。

```
alientek@ubuntu:~/tslib-1.21$ ./autogen.sh
libtoolize: putting auxiliary files in '.'.
libtoolize: copying file './ltmain.sh'
libtoolize: putting macros in AC_CONFIG_MACRO_DIRS, 'm4/internal'.
libtoolize: copying file 'm4/internal/libtool.m4'
libtoolize: copying file 'm4/internal/ltoptions.m4'
libtoolize: copying file 'm4/internal/ltsugar.m4'
libtoolize: copying file 'm4/internal/ltversion.m4'
libtoolize: copying file 'm4/internal/lt~obsolete.m4'
configure.ac:58: installing './compile'
configure.ac:7: installing './missing'
plugins/Makefile.am: installing './depcomp'
alientek@ubuntu:~/tslib-1.21$
```

图 2.2 1 生成 Makefile

执行下面的指令指定我们上面第一章安装交叉编译工具与 tslib 编译输出路径/home/alientek/tslib-1.21/arm-tslib。复制时请注意，下面是一条指令不分行，建议分段复制。注意不要多写空格。指令会检查编译器等是否会工作，如下图。

```
./configure --host=arm-linux-gnueabihf ac_cv_func_malloc_0_nonnull=yes --cache-file=arm-linux.cache -prefix=/home/alientek/tslib-1.21/arm-tslib
```

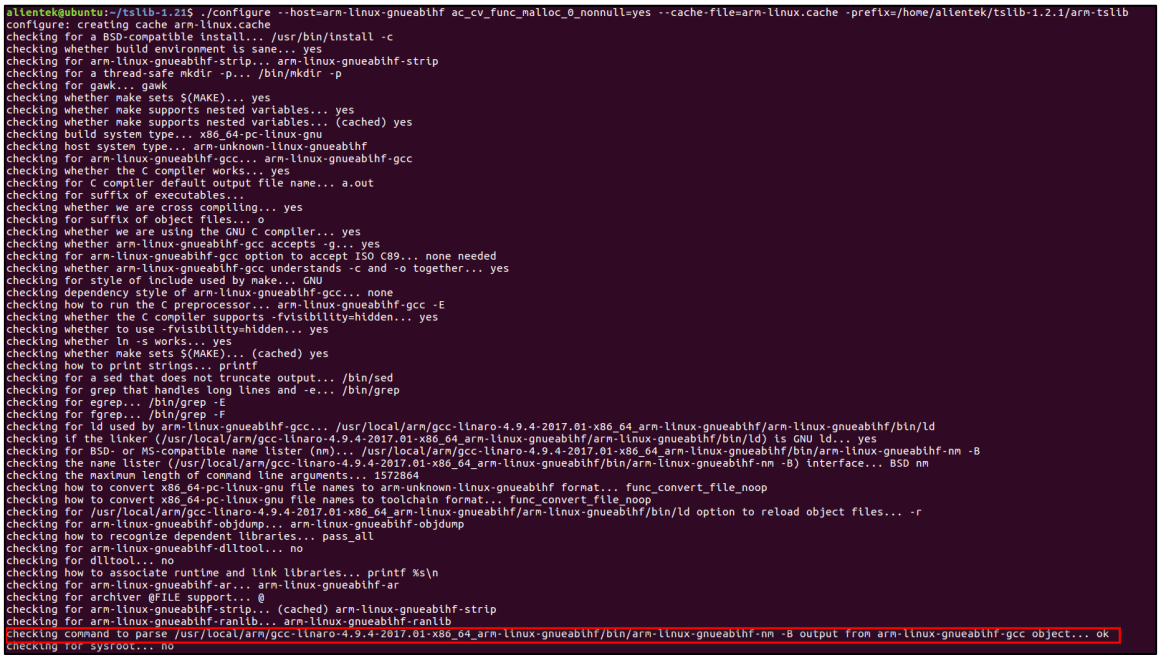


图 2.2 2 配置 tslib

直接执行 make 编译及 make install 安装。

```
make
make install
```

编译出现警告不用理会以，安装完成后在当前目录下生成一个 arm-tslib 目录。也是我们上面指令的编译输出目录路径。可以查看生成目录下有以下内容。

图 2.2 3 编译完成，arm-tslib 目录下的产物

第三章 编译 ARM 平台 Qt4.8.4 源码

本章介绍如何编译 ARM 平台的 Qt4.8.4 源码。

3.1 下载 Qt4.8.4 源码

我们可以在 Qt 下载地址 https://download.qt.io/找到 https://download.qt.io/archive/qt/4.8/4.8.4/，进入下载页面如下。

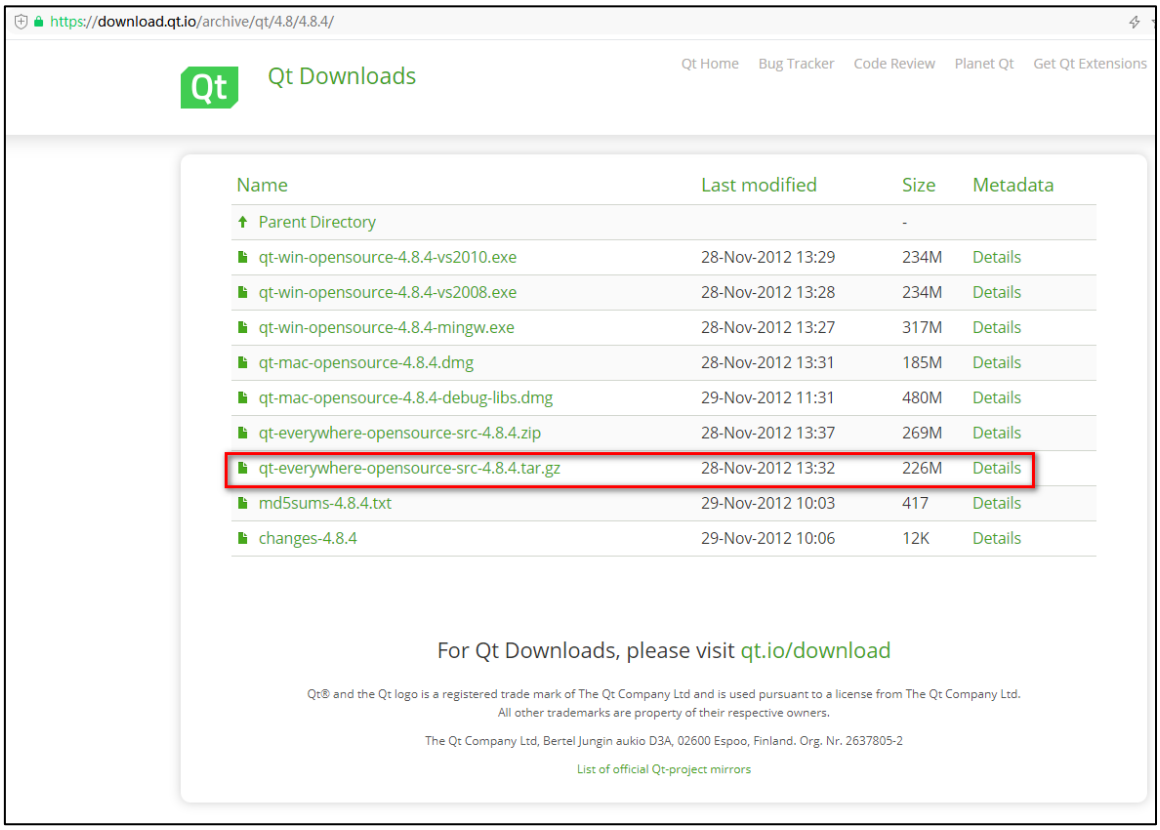


图 3.1 1 下载 Qt4.8.4 源码

在 Windows 上下载好上面 qt-everywhere-opensource-src-4.8.4.tar.gz 源码拷贝到 Ubuntu 虚拟机下，或者直接右键上面的下载项，出现复制下载地址链接，在 Ubuntu 下直接使用 wget 指令下载。

在 Ubuntu 使用 wget 指令下载 qt-everywhere-opensource-src-4.8.4.tar.gz 源码如下。如果觉得慢，就在 Windows 下复制下载链接地址使用迅雷下载再拷贝到 Ubuntu 虚拟机吧！

```
wget https://download.qt.io/archive/qt/4.8/4.8.4/qt-everywhere-opensource-src-4.8.4.tar.gz // >=ubuntu1404
wget http://download.qt.io/archive/qt/4.8/4.8.4/qt-everywhere-opensource-src-4.8.4.tar.gz // ubuntu12.04
```

解压下载好的 qt-everywhere-opensource-src-4.8.4.tar.gz 源码压缩包。

```
tar xf qt-everywhere-opensource-src-4.8.4.tar.gz
```

在当前目录出现 qt-everywhere-opensource-src-4.8.4 目录夹，此文件夹就是我们解压后的目录。进入此目录。

```
cd qt-everywhere-opensource-src-4.8.4/
```

3.2 修改 qmake.conf

编辑 mkspecs/qws/linux-arm-gnueabi-g++/qmake.conf，配置相关编译器及路径。

```
vi mkspecs/qws/linux-arm-gnueabi-g++/qmake.conf
```

```
alientek@ubuntu:~/qt-everywhere-opensource-src-4.8.4$ vi mkspecs/qws/linux-arm-gnueabi-g++/qmake.conf
```

图 3.2 1 配置 qmake.conf

修改成如下，红色部分就是要修改的地方。

```
#
# qmake configuration for building with arm-none-linux-gnueabi-g++
#

include(../../common/linux.conf)
include(../../common/gcc-base-unix.conf)
include(../../common/g++-unix.conf)
include(../../common/qws.conf)

# modifications to g++.conf
QMAKE_CC                = arm-linux-gnueabihf-gcc  -lts
QMAKE_CXX               = arm-linux-gnueabihf-g++  -lts
QMAKE_LINK              = arm-linux-gnueabihf-g++  -lts
QMAKE_LINK_SHLIB        = arm-linux-gnueabihf-g++  -lts

# modifications to linux.conf
QMAKE_AR                = arm-linux-gnueabihf-ar  cqs
QMAKE_OBJCOPY           = arm-linux-gnueabihf-objcopy
QMAKE_STRIP             = arm-linux-gnueabihf-strip

load(qt_config)
```

3.3 配置编译选项

查看编译选项，输入./configure -help 指令，再输入 o，查看自由版本选项。

```
./configure -help
```

由于配置较长，编译总结了配置项，写了一个脚本。

```
vi autoconfigure.sh
```

在这个 autoconfigure.sh 添加以下内容

```
./configure   -prefix  /home/alientek/qt-everywhere-opensource-src-4.8.4/arm-qt  \
 -opensource  \
 -confirm-license  \
 -release  -shared  \
 -embedded  arm  \
 -xplatform  qws/linux-arm-gnueabi-g++  \
 -fast  \
 -optimized-qmake  \
 -no-pch  \
 -qt-libjpeg  \
```

```
-qt-zlib \
-qt-libpng \
-qt-freetype \
-little-endian \
-host-little-endian \
-no-qt3support \
-no-libtiff \
-no-libmng \
-no-opengl \
-no-mmx \
-no-sse \
-no-sse2 \
-no-3dnow \
-no-openssl \
-no-webkit \
-no-qvfb \
-no-phonon \
-no-nis \
-no-cups \
-no-glib \
-qt-mouse-tslib \
-nomake tools \
-nomake docs \
-qt-sql-sqlite \
-plugin-sql-sqlite \
-I/home/alientek/tslib-1.21/arm-tslib/include \
-L/home/alientek/tslib-1.21/arm-tslib/lib
```

其中红色部分为编者个人的路径，请修改成个人路径。

（1） /home/alientek/qt-everywhere-opensource-src-4.8.4/arm-qt 为编译输出路径。

（2） /home/alientek/tslib-1.21/arm-tslib/include 为第二章 tslib 头文件路径。

（3） /home/alientek/tslib-1.21/arm-tslib/lib 为第二章 tslib 相关库文件路径。

赋予脚本可执行权限。

```
chmod +x autoconfigure.sh
```

直接执行 autoconfigure.sh，配置完成如下。若未见此信息，请检查个人配置。

```
./autoconfigure.sh
```

```
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/addressbook-fr/part1/part1.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/addressbook-fr/part6/part6.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/addressbook-fr/part5/part5.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/addressbook-fr/part2/part2.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/modelview/6_treeview/6_treeview.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/modelview/5_edit/5_edit.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/modelview/4_headers/4_headers.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/modelview/1_readonly/1_readonly.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/modelview/2_formatting/2_formatting.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/modelview/7_selections/7_selections.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/modelview/3_changingmodel/3_changingmodel.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/addressbook/part4/part4.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/addressbook/part7/part7.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/addressbook/part3/part3.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/addressbook/part1/part1.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/addressbook/part6/part6.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/addressbook/part5/part5.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/tutorials/addressbook/part2/part2.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/script/calculator/calculator.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/script/context2d/context2d.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/script/qscript/qscript.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/script/marshal/marshal.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/script/qstetrix/qstetrix.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/script/customclass/customclass.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/script/defaultprototypes/defaultprototypes.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/script/qsdbg/qsdbg.pro (fast)
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/examples/script/helloscript/helloscript.pro (fast)

Qt is now configured for building. Just run 'make'.
Once everything is built, you must run 'make install'.
Qt will be installed into /home/alientek/qt-everywhere-opensource-src-4.8.4/arm-qt

To reconfigure, run 'make confclean' and 'configure'.

alientek@ubuntu:~/qt-everywhere-opensource-src-4.8.4$
```

图 3.3 1 配置成功

3.4 开始编译 Qt4.8.4 源码

直接执行 make 开始编译。

make -j 16 // -j 16 代表最多允许 16 条编译指令同时运行，参数 16，一般为个人分配给虚拟机的核心数的 2 倍

编译完成如下图，编译时长由个人计算机速度及分配给虚拟机核心数决定（编者编译了 7 分钟）。如按上面操作，遇到编译 Qt 源码出现错误，请重新配置！有可能是个人计算机分配给虚拟机内存太低所导致的，既然 Qt 源码能发布，除了配置错误，几乎 Qt 源码是不会有编译错误的！

编译完成如下图。

```
alientek@ubuntu: ~/qt-everywhere-opensource-src-4.8.4
not used [-Wunused-local-typedefs]
     typedef typename Sequence::const_iterator Iterator;
                                               ^
In file included from ../../../include/QtCore/qtconcurrentfilter.h:1:0,
                 from ../../../include/QtCore/QtCore:53,
                 from ../../../include/QtGui/QtGui:3,
                 from fluidlauncher.h:45,
                 from main.cpp:45:
../../../include/QtCore/../../src/corelib/concurrent/qtconcurrentfilter.h: In function ‘QtConcurrent::ThreadEngineStarter<void> Qt
Concurrent::filterInternal(Sequence&, KeepFunctor, ReduceFunctor)’:
../../../include/QtCore/../../src/corelib/concurrent/qtconcurrentfilter.h:108:47: warning: typedef ‘Iterator’ locally defined but
not used [-Wunused-local-typedefs]
     typedef typename Sequence::const_iterator Iterator;
                                               ^
In file included from ../../../include/QtCore/qtconcurrentfilter.h:1:0,
                 from ../../../include/QtCore/QtCore:53,
                 from ../../../include/QtGui/QtGui:3,
                 from .moc/release-shared-emb-arm/../../fluidlauncher.h:45,
                 from .moc/release-shared-emb-arm/moc_fluidlauncher.cpp:10:
../../../include/QtCore/../../src/corelib/concurrent/qtconcurrentfilter.h: In function ‘QtConcurrent::ThreadEngineStarter<void> Qt
Concurrent::filterInternal(Sequence&, KeepFunctor, ReduceFunctor)’:
../../../include/QtCore/../../src/corelib/concurrent/qtconcurrentfilter.h:108:47: warning: typedef ‘Iterator’ locally defined but
not used [-Wunused-local-typedefs]
     typedef typename Sequence::const_iterator Iterator;
                                               ^
arm-linux-gnueabihf-g++ -lts -Wl,-rpath-link,/home/alientek/qt-everywhere-opensource-src-4.8.4/lib -Wl,-O1 -Wl,-rpath,/home/alient
ek/qt-everywhere-opensource-src-4.8.4/arm-qt/lib -Wl,-rpath,/home/alientek/qt-everywhere-opensource-src-4.8.4/arm-qt/lib -o fluidl
auncher .obj/release-shared-emb-arm/demoapplication.o .obj/release-shared-emb-arm/fluidlauncher.o .obj/release-shared-emb-arm/main
.o .obj/release-shared-emb-arm/pictureflow.o .obj/release-shared-emb-arm/slideshow.o .obj/release-shared-emb-arm/moc_demoapplicati
on.o .obj/release-shared-emb-arm/moc_fluidlauncher.o .obj/release-shared-emb-arm/moc_pictureflow.o .obj/release-shared-emb-arm/moc
_slideshow.o    -L/home/alientek/tslib-1.21/arm-tslib/lib -L/home/alientek/qt-everywhere-opensource-src-4.8.4/lib -lQtGui -L/home/
alientek/tslib-1.21/arm-tslib/lib -L/home/alientek/qt-everywhere-opensource-src-4.8.4/lib -lQtNetwork -lQtCore -lpthread
make[3]: Leaving directory '/home/alientek/qt-everywhere-opensource-src-4.8.4/demos/embedded/fluidlauncher'
make[2]: Leaving directory '/home/alientek/qt-everywhere-opensource-src-4.8.4/demos/embedded'
make[1]: Leaving directory '/home/alientek/qt-everywhere-opensource-src-4.8.4/demos'
alientek@ubuntu:~/qt-everywhere-opensource-src-4.8.4$
```

图 3.4 1 Qt4.8.4 源码编译完成

执行安装指令后，查看安装后的内容。

```
make install
ls arm-qt
```

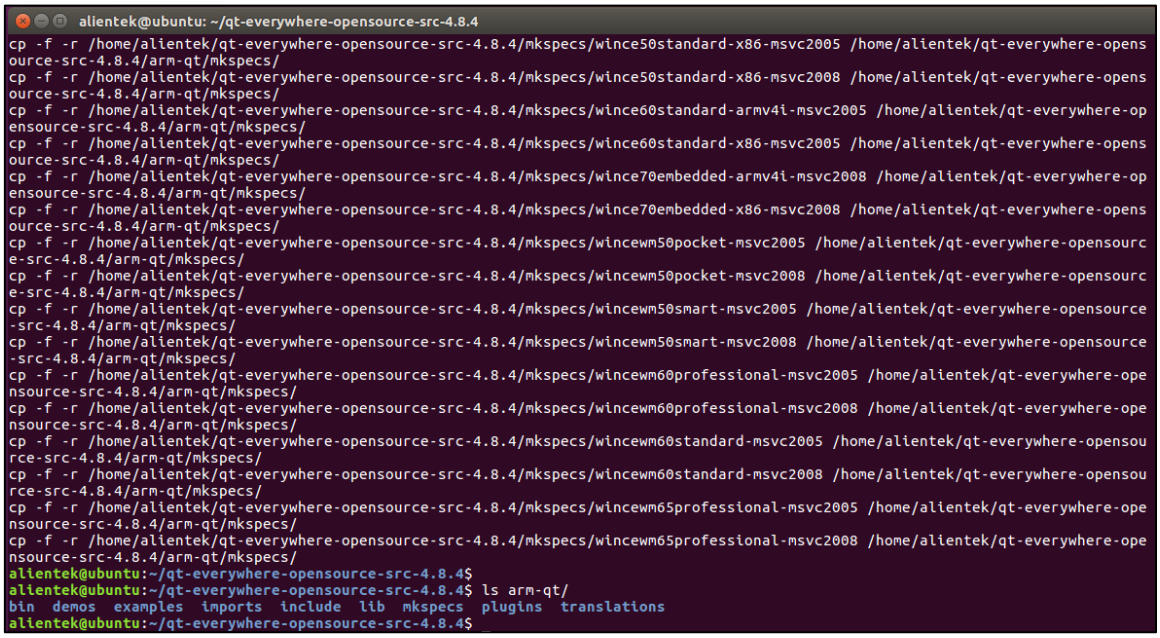


图 3.4 2 安装完成

第四章 移植 Qt 到文件系统

编译完成后，我们需要将编译的库移植到文件系统里。

4.1 烧写文件系统

这里使用【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.x.pdf 所构建好的 busybox 文件系统（不含 Qt 的文件系统），文件系统路径为开发板光盘 A-基础资料->8、系统镜像->2、教程系统镜像->3、文件系统->2、busybox-1.29.0 根文件系统下的 rootfs.tar.bz2。将此文件系统替换到开发板光盘 A-基础资料->5、开发工具->4、正点原子 MFG_TOOL 出厂固件烧录工具->mfgtool->Profiles->Linux->OS Firmware->files->filesystem 下的 rootfs.tar.bz2 (替换前请备份好原来的文件系统！)。

然后烧写文件系统到 eMMC 核心板或者 NandFlash 核心板。烧写方法请看【正点原子】I.MX6U 用户快速体验 V1.x.pdf 第 2.2 小节固化系统。不建议用 NFS 挂载来测试 Qt 的性能！

4.2 移植 tslib 到文件系统

使用下面的指令打包 2.2 小节编译安装好的 arm-tslib 文件夹打包成 tar.bz2 格式，打包以防止文件丢失。

```
tar -jcf ./arm-tslib.tar.bz2 arm-tslib
```

然后使用 U 盘拷贝 arm-tslib.tar.bz2，在 USB 接口插上 U 盘，手动挂载后，在开发板根文件系统上解压到/usr/lib/目录下。如下图，编者已经解压到/usr/lib 目录下。

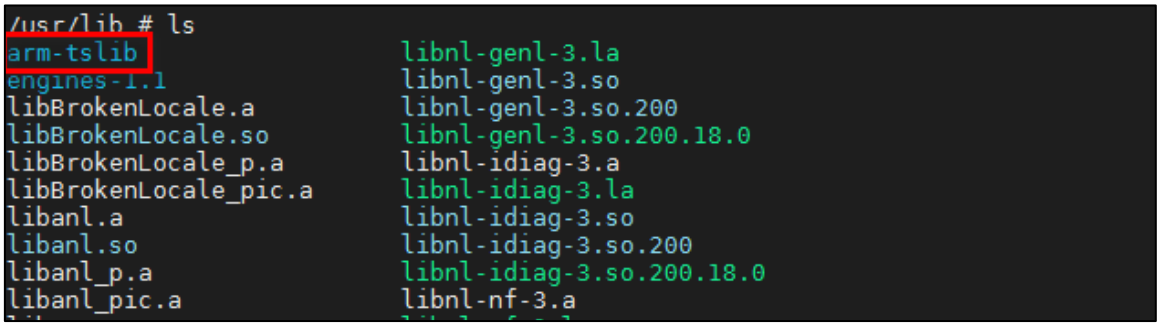


图 4.2 1 解压打包的 arm-tslib 到/usr/lib 目录下

➢ 配置 tslib 的环境变量

由于上面烧写的文件系统已经移植过 tslib，已经在/etc/profile 文件配置过 Qt5 的环境变量。所以我们的要重新写成 Qt4 的环境变量，及指定我们编译的 tslib 所在文件夹。避免和文件系统已经存在的 tslib 冲突。（那究竟能不能直接用文件系统已经存在的 tslib 呢？这个自行测试，怕用户在第二章移植 tslib 用了不同版本，所以我们还是自己指定我们移植的 tslib 吧）。

编辑/etc/profile 文件，修改成以下内容。如下图红色框部分。“export TSLIB_CALIBFILE=/etc/pointercal”这项是电阻屏专用，如果是电容屏，可不用加这项。电容加了这项，如果使用 ts_calibrate 校准后会生成/etc/pointercal 文件，请把它删除！否则可以触摸不准确，因为电容屏不需要校准。

```
export TSLIB_ROOT=/usr/lib/arm-tslib
export TSLIB_CONSOLEDEVICE=none
export TSLIB_FBDEVICE=/dev/fb0
export TSLIB_TSDEVICE=/dev/input/event1
export TSLIB_CONFFILE=$TSLIB_ROOT/etc/ts.conf
export TSLIB_PLUGINDIR=$TSLIB_ROOT/lib/ts
export TSLIB_CALIBFILE=/etc/pointercal
```

```
#!/bin/sh
LD_LIBRARY_PATH=/lib:/usr/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH

export TERM=vt100
export TERMINFO=/usr/share/terminfo

export TSLIB_ROOT=/usr/lib/arm-tslib
export TSLIB_CONSOLEDEVICE=none
export TSLIB_FBDEVICE=/dev/fb0
export TSLIB_TSDEVICE=/dev/input/event1
export TSLIB_CONFFILE=$TSLIB_ROOT/etc/ts.conf
export TSLIB_PLUGINDIR=$TSLIB_ROOT/lib/ts
export TSLIB_CALIBFILE=/etc/pointercal

export ALSA_CONFIG_PATH=/usr/share/arm-alsa/alsa.conf
~
```

图 4.2 2 配置 tslib 全局环境变量

➢ 测试 tslib

使能 tslib 的配置在/etc/profile 的环境变量，下次开机不用使能，开机会自动使能这个环境变量。

```
source /etc/profile                    // 使能环境变量
/usr/lib/arm-tslib/bin/ts_test         // 运行 ts_test 测试触摸是否正常，点击界面的 Draw 测试
```

触摸屏上点击 Draw 测试能够画出线条，且位置准确，说明 tslib 配置正常。按 Ctrl + c 结束 ts_test 指令。下图为 7 寸 800x480 分辨率的电容屏，使用 tslib 测试效果。

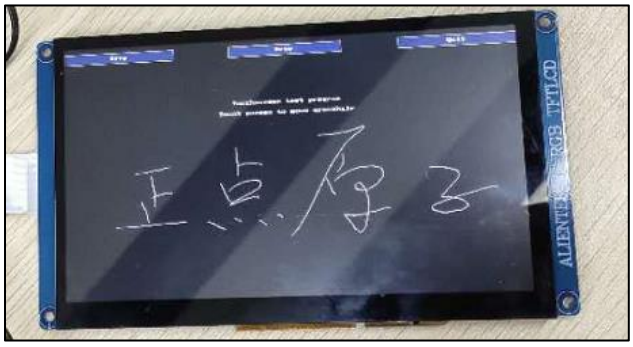


图 4.2 3 7 寸屏 800x480 的触摸效果

4.3 移植 Qt 到文件系统

使用下面的指令打包 3.4 小节编译安装好的 arm-qt 文件夹打包成 tar.bz2 格式，打包以防止文件丢失。

```
tar -jcf ./arm-qt.tar.bz2 arm-qt
```

然后使用 U 盘拷贝 arm-qt.tar.bz2，在 USB 接口插上 U 盘，手动挂载后，在开发板根文件系统上解压到/usr/lib/目录下。如下图，编者已经解压到/usr/lib 目录下。

```
/usr/lib # ls
arm-qt                    libnl-genl-3.la
arm-tslib                 libnl-genl-3.so
engines-1.1               libnl-genl-3.so.200
libBrokenLocale.a         libnl-genl-3.so.200.18.0
libBrokenLocale.so        libnl-idiag-3.a
```

图 4.3 1 解压打包 arm-qt 到/usr/lib 下

➢ 配置 Qt4 的环境变量

Qt4 的环境变量与 Qt5 的配置不一样。编辑/etc/profile，在末尾添加以下内容。如下图红色框内。

```
QT_DIR=/usr/lib/arm-qt
export QT_QWS_FONTDIR=$QT_DIR/lib/fonts
export QWS_MOUSE_PROTO="tslib:/dev/input/event1 mouseman:/dev/input/mice"
export QT_QWS_DISPLAY="linuxFB:/dev/fb0"
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$QT_DIR/lib
```

```
#!/bin/sh
LD_LIBRARY_PATH=/lib:/usr/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH

export TERM=vt100
export TERMINFO=/usr/share/terminfo

export TSLIB_ROOT=/usr/lib/arm-tslib
export TSLIB_CONSOLEDEVICE=none
export TSLIB_FBDEVICE=/dev/fb0
export TSLIB_TSDEVICE=/dev/input/event1
export TSLIB_CONFFILE=$TSLIB_ROOT/etc/ts.conf
export TSLIB_PLUGINDIR=$TSLIB_ROOT/lib/ts
export TSLIB_CALIBFILE=/etc/pointercal

export ALSA_CONFIG_PATH=/usr/share/arm-alsa/alsa.conf

QT_DIR=/usr/lib/arm-qt
export QT_QWS_FONTDIR=$QT_DIR/lib/fonts
export QWS_MOUSE_PROTO="tslib:/dev/input/event1 mouseman:/dev/input/mice"
export QT_QWS_DISPLAY="linuxFB:/dev/fb0"
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$QT_DIR/lib
```

图 4.3 2 配置 Qt4 的全局环境变量

➢ 测试 Qt 运行

使能 Qt 的配置在/etc/profile 的环境变量，下次开机不用使能，开机会自动使能这个环境变量。

```
cd /usr/lib/arm-qt/demos/embedded/fluidlauncher // 进入 Demo 所在目录
./fluidlauncher -qws                            // 运行 Demo
```

7 寸屏 800x480 显示效果如下，同时点击图中的项，确认触摸正常。

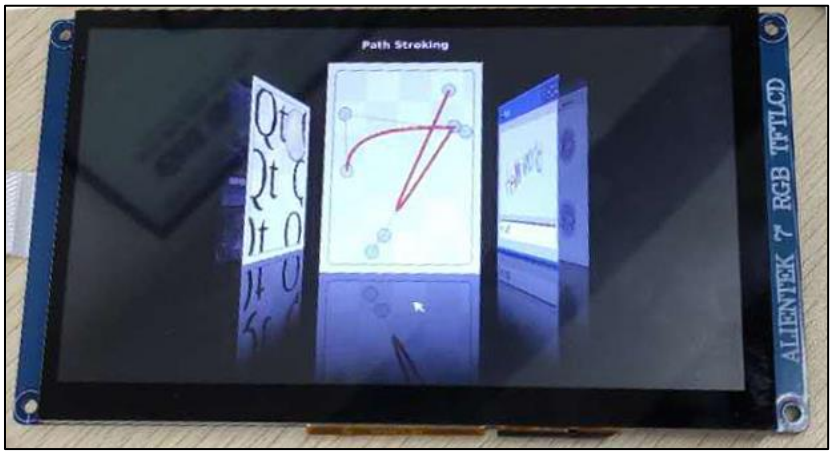


图 4.3 3 7 寸屏 800x480 Qt4 例程运行效果

串口打印如下，报了一个警告，因为我们用的是电容屏，可忽略这个警告。电阻屏才需要使用 ts_calibrate 校准。

```
/usr/lib/arm-qt/demos/embedded/fluidlauncher # ./fluidlauncher -qws
Could not read calibration: "/etc/pointercal"
```

图 4.3 4 运行例子时串口打印的信息

报 QWSLock::down(): Invalid argument 警告打印，请自行参考 https://blog.csdn.net/xikangsoon/article/details/101319384 博客，重新编译解决。

第五章 编译 X86 平台的 Qt4.8.4 源码

X86 平台的 Qt4.8.4 源码需要编译才能在 Ubuntu 上使用。编译 X86 平台的 Qt 源码方便我们在 Ubuntu 上仿真测试。

5.1 编译 X86 平台的 Qt4.8.4

此小节必须使用 Ubuntu12.04，因为 Qt4.8.4 是 2012 年发布的，所以那时 Ubuntu 版本只有 Ubuntu12。我们上面只是安装了 Qt Creator，因 Qt4 版本在 Linux 下还没有 Ubuntu 版本编译套件。我们需要自己编译，然后再安装在 Ubuntu，方便写程序及在 Ubuntu 上调试。

要在 Ubuntu 上编译 X86 平台上的 Qt4.8.4，需要安装 g++等库。

```
sudo apt-get install lsb-core lib32stdc++6
```

进入第三章的 Qt 源码目录，执行下面的指令清理工程。因为第三章是编译 ARM 平台的 Qt4.8.4，现在我们要编译 X86 平台的 Qt4.8.4。

```
make distclean
```

在执行 configure 前，需要安装以下库。因为配置时会出现缺少 Xlib 相关错误。建议分行复制，下面为一行指令。

```
sudo apt-get install libfontconfig1-dev libfreetype6-dev libx11-dev libxcursor-dev libxext-dev libxfixes-dev libxft-dev libxi-dev libxrandr-dev libxrender-dev
```

直接执行 configure，configure 会自动根据宿主机平台设置。之后输入“o”，再输入“yes”。配置需要一段时间。

```
./configure
```

```
alientek@ubuntu:~/qt-everywhere-opensource-src-4.8.4$ ./configure
Which edition of Qt do you want to use ?

Type 'c' if you want to use the Commercial Edition.
Type 'o' if you want to use the Open Source Edition.

o


This is the  Open Source Edition.

You are licensed to use this software under the terms of
the Lesser GNU General Public License (LGPL) versions 2.1.
You are also licensed to use this software under the terms of
the GNU General Public License (GPL) versions 3.

Type '3' to view the GNU General Public License version 3.
Type 'L' to view the Lesser GNU General Public License version 2.1.
Type 'yes' to accept this license offer.
Type 'no' to decline this license offer.

Do you accept the terms of either license? yes

Creating qmake. Please wait...
make: Nothing to be done for `first'.
```

图 5.1 1 执行 configure

```
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/arthurplugin/arthurplugin.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/books/books.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/composition/composition.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/macmainwindow/macmainwindow.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/deform/deform.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/declarative/photoviewer/photoviewer.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/declarative/twitter/twitter.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/declarative/webbrowser/webbrowser.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/declarative/minehunt/minehunt.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/declarative/rssnews/rssnews.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/declarative/calculator/calculator.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/declarative/samegame/samegame.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/declarative/snake/snake.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/declarative/flickr/flickr.pro
  for /home/alientek/qt-everywhere-opensource-src-4.8.4/demos/sub-attaq/sub-attaq.pro

Qt is now configured for building. Just run 'make'.
Once everything is built, you must run 'make install'.
Qt will be installed into /usr/local/Trolltech/Qt-4.8.4

To reconfigure, run 'make confclean' and 'configure'.

alientek@ubuntu:~/qt-everywhere-opensource-src-4.8.4$
```

图 5.1 2 执行 configure 成功

看到上图的结果，说明配置成功。可以看到安装目录是/usr/local/Trolltech/Qt-4.8.4。可以执行 make 编译了。

直接执行 make 开始编译。

```
make -j 16   // -j 16 代表最多允许 16 条编译指令同时运行，参数 16，一般为个人分配给虚拟机的核心数的 2 倍
```

编译完成如下图，编译时长由个人计算机速度及分配给虚拟机核心数决定（编者编译了 12 分钟）。如按上面操作，遇到编译 Qt 源码出现错误，请重新配置！有可能是个人计算机分配给虚拟机内存太低所导致的，既然 Qt 源码能发布，说明 Qt 源码是几乎不可能有编译错误的！

编译完成如下图

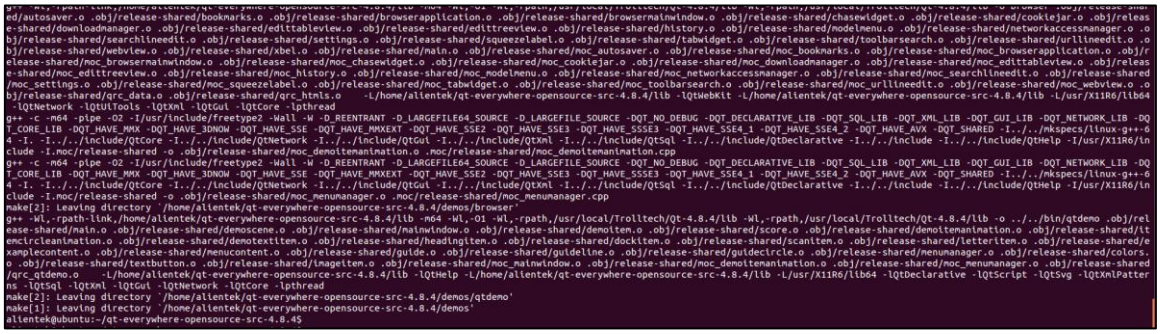


图 5.1 3 编译成功如上

安装，执行安装指令

```
sudo make install
```

```
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/win32-msvc2008 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/win32-msvc2010 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/win32-msvc2012 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince50standard-armv4i-msvc2005 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince50standard-armv4i-msvc2008 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince50standard-mipsii-msvc2005 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince50standard-mipsii-msvc2008 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince50standard-mipsiv-msvc2005 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince50standard-mipsiv-msvc2008 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince50standard-sh4-msvc2005 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince50standard-sh4-msvc2008 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince50standard-x86-msvc2005 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince50standard-x86-msvc2008 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince60standard-armv4i-msvc2005 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince60standard-x86-msvc2005 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince70embedded-armv4i-msvc2008 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wince70embedded-x86-msvc2008 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wincewm50pocket-msvc2005 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wincewm50pocket-msvc2008 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wincewm50smart-msvc2005 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wincewm50smart-msvc2008 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wincewm60professional-msvc2005 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wincewm60professional-msvc2008 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wincewm60standard-msvc2005 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wincewm60standard-msvc2008 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wincewm65professional-msvc2005 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
cp -f -r /home/alientek/qt-everywhere-opensource-src-4.8.4/mkspecs/wincewm65professional-msvc2008 /usr/local/Trolltech/Qt-4.8.4/mkspecs/
alientek@ubuntu:~/qt-everywhere-opensource-src-4.8.4$
```

图 5.1 4 安装成功

第六章 搭建 X86 Linux 平台 Qt Creator 环境

本章将介绍如何搭建 X86 Linux 平台上的 Qt Creator 环境。

6.1 Linux 安装 Qt Creator

此小节必须和第五章一样使用 Ubuntu12.04，否则配置不了 Qt Creator Kits。Qt Creator 是 Qt 的原生编程 IDE，并集成了很多开源的软件功能。我们使用 Qt 编程时所用工具就是 Qt Creator，Qt4 在 Linux 下需要另外安装 Qt Creator。根据 Windows 的 Qt 4.8.4 版本所配套的 Qt Creator 版本为 Qt Creator 2.6.1。所以我们在官网下载 Qt Creator 2.6.1。

打开 https://download.qt.io/archive/qtcreator/2.6/2.6.1/网址，编者的是 64 位的 Ubuntu 所以下载 qt-creator-linux-x86_64-opensource-2.6.1.bin，如下图。请根据个人的 Ubuntu 版本选择相应位数的 Qt Creator 位数版本。

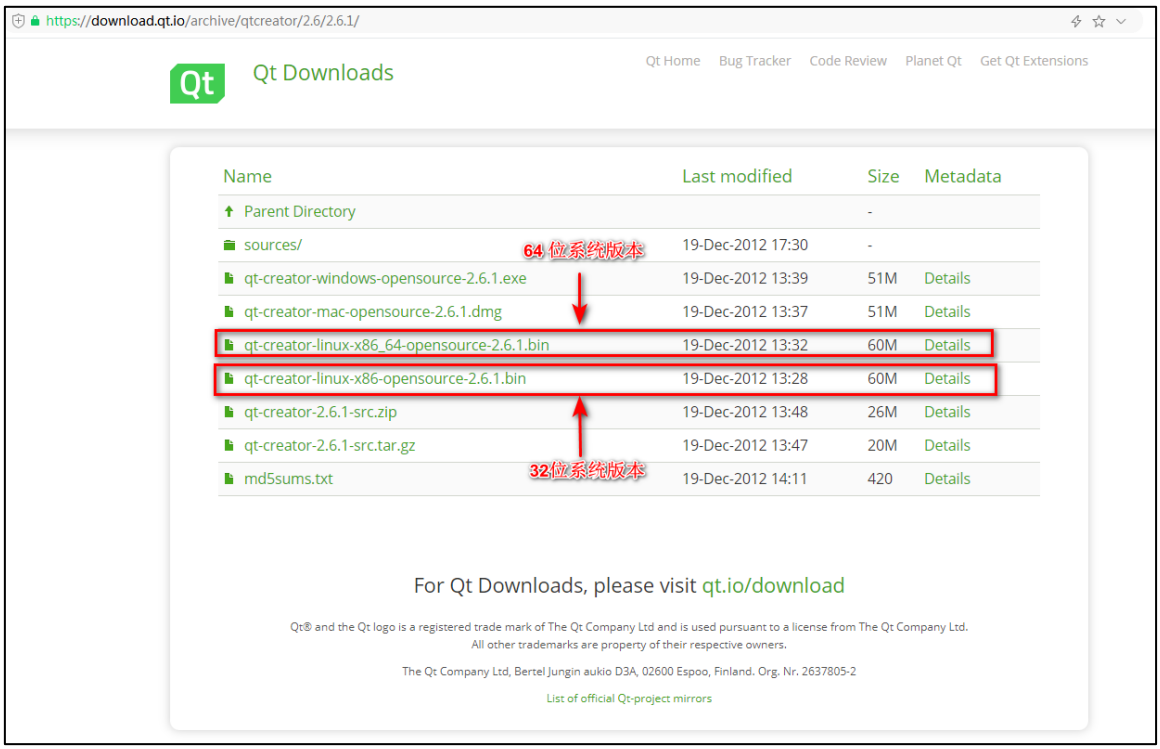


图 6.1 1 下载 Qt Creator

或者右键该项，选择复制链接下载地址，直接在 Ubuntu 上使用 wget 指令下载。

```
wget https://download.qt.io/archive/qtcreator/2.6/2.6.1/qt-creator-linux-x86_64-opensource-2.6.1.bin  // >=ubuntu14.04
wget http://download.qt.io/archive/qtcreator/2.6/2.6.1/qt-creator-linux-x86_64-opensource-2.6.1.bin  // ubuntu12.04
```

如下，编者已经下载好 qt-creator-linux-x86_64-opensource-2.6.1.bin 并上传至 Ubuntu。

图 6.1 2 拷贝下载好的 Qt Creator 到 Ubuntu

赋予可执行权限，然后使用 sudo 权限进行安装，用户最好也使用 sudo 权限进行安装，默认会安装到系统指定目录。

```
alientek@ubuntu:~$ chmod +x qt-creator-linux-x86_64-opensource-2.6.1.bin
alientek@ubuntu:~$ sudo ./qt-creator-linux-x86_64-opensource-2.6.1.bin
```

图 6.1 3 执行安装 Qt Creator

若无意外，将弹出安装界面。

图 6.1 4 开始安装 Qt Creator

选择安装目录，默认即可，点击 Next。

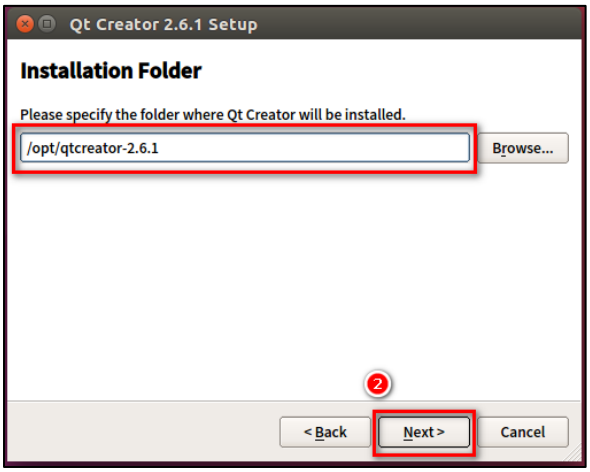


图 6.1 5 默认安装目录

选择同意 License，点击 Next

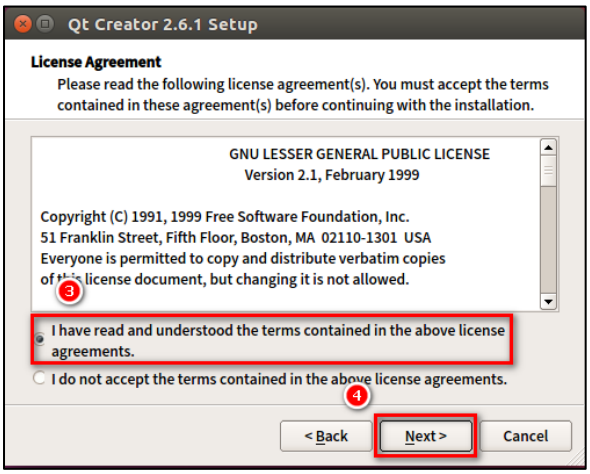


图 6.1 6 同意 License，点击 Next

点击 install，开始安装。

图 6.1 7 开始安装

安装过程较快，点击 Finish。完成安装。

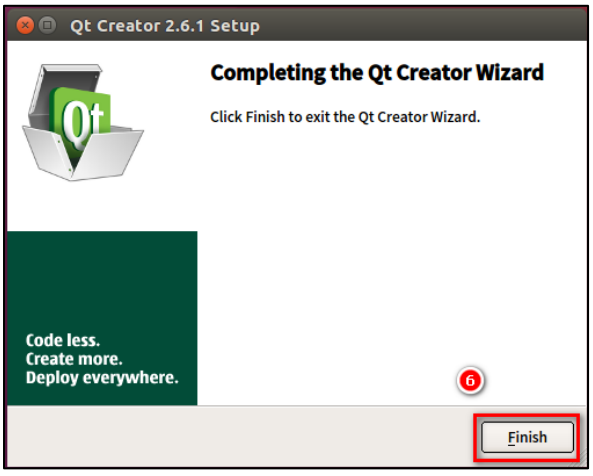


图 6.1 8 点击 Finish，完成安装

安装完成后，可以在左上角软件中心搜索栏，输入“qt”，搜索打开 Qt Creator。单击打开 Qt Creator。

图 6.1 9 搜索软件栏，打开 Qt Creator

打开 Qt Creator 界面如下。

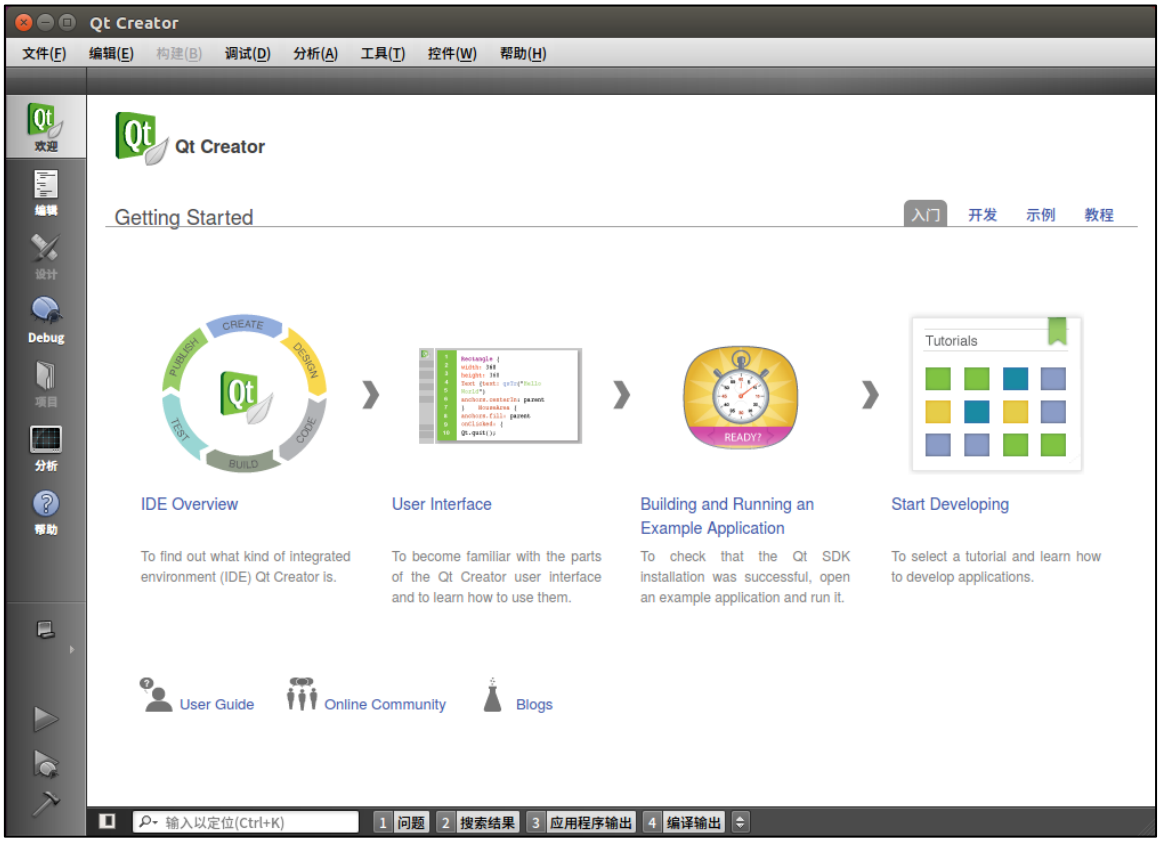


图 6.1 10 打开 Qt Creator 后的界面

6.2 配置 X86 平台的 Qt Creator Kits

Kits 这里指 Qt Creator 里的编译套件。我们要为 Qt Creator 配置 X86 平台的编译套件，这样我们在使用 Qt Creator 编写程序时可以方便在 Ubuntu（X86）平台上调试及运行。

打开 Qt Creator 界面，找到 Tools》Options。

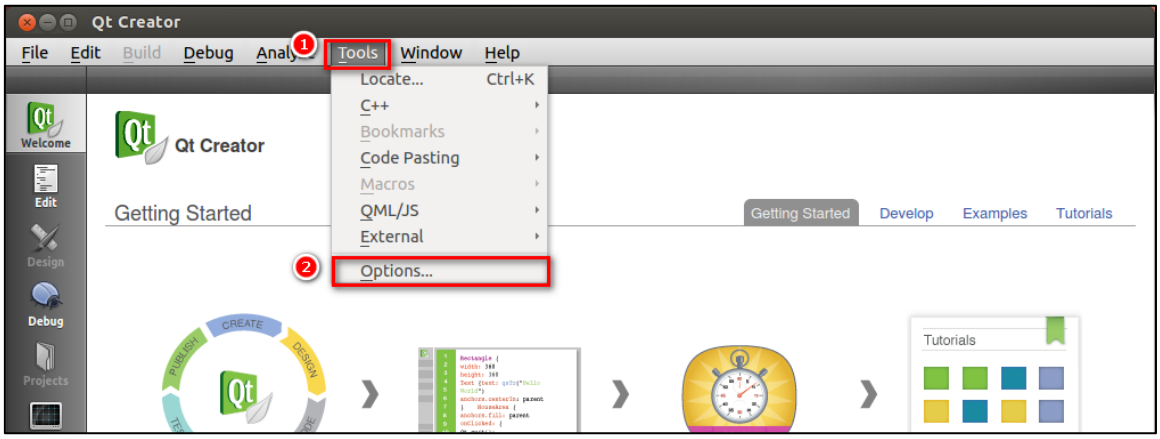


图 6.2 1 打开选项

添加 qmake 的路径，按如下步骤，选择 qmake 的路径，将第 5.1 小节编译好的 X86 平台的 qmake 添加进来。（qmkae 的作用是生成 Makefile，用于编译）

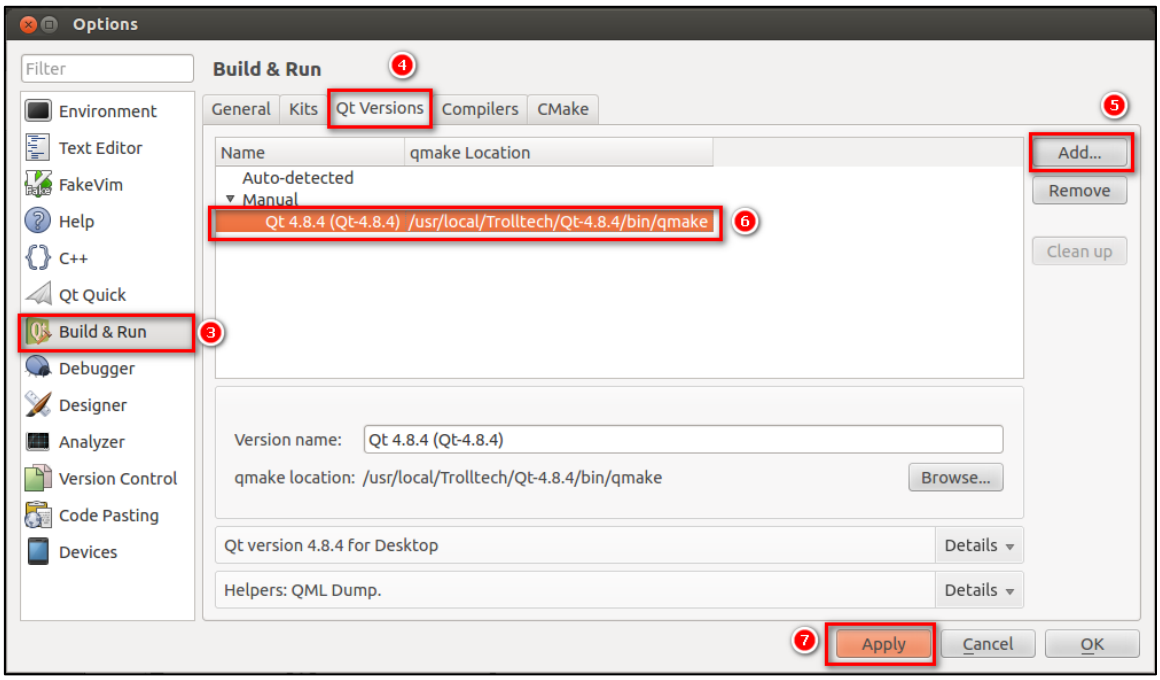


图 6.2 2 配置 qmake

配置 Kits（套件），选择上一步配置好的 Qt Versions。如下图步骤操作。

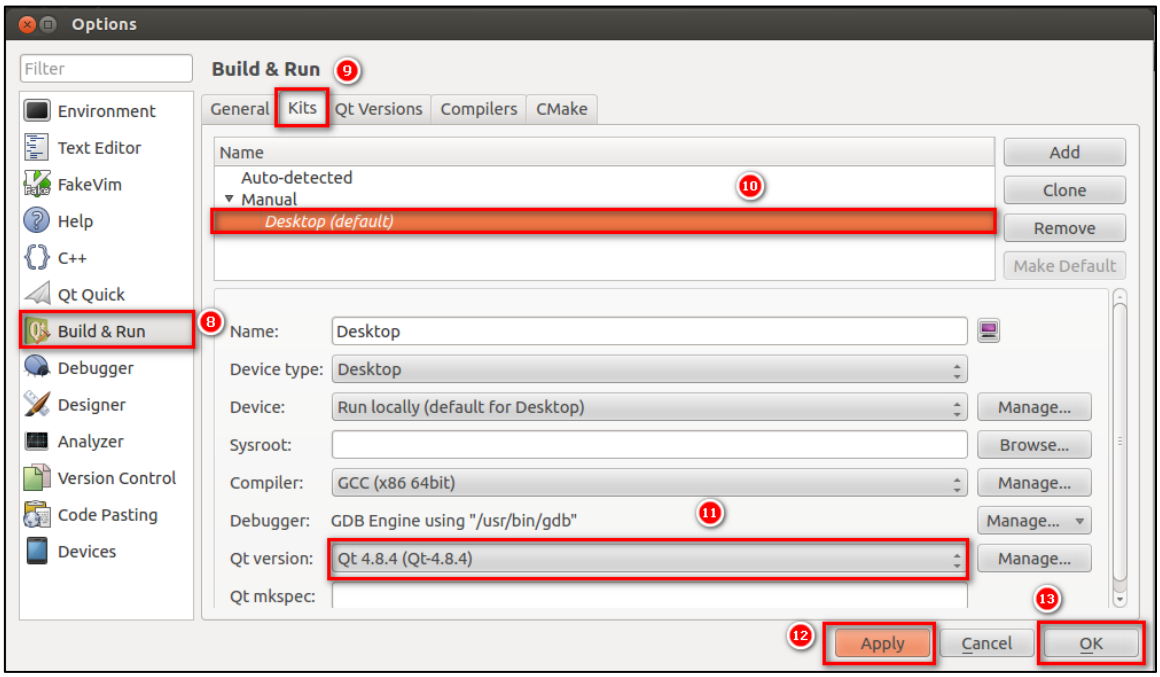


图 6.2 3 配置 Kits

6.3 验证 X86 平台的 Qt 运行

我们新建一个工程，验证 Qt 在 Ubuntu 平台的运行情况。按如下步骤新建工程，这些步骤和其他 IDE 新建工程的步骤基本都一样。

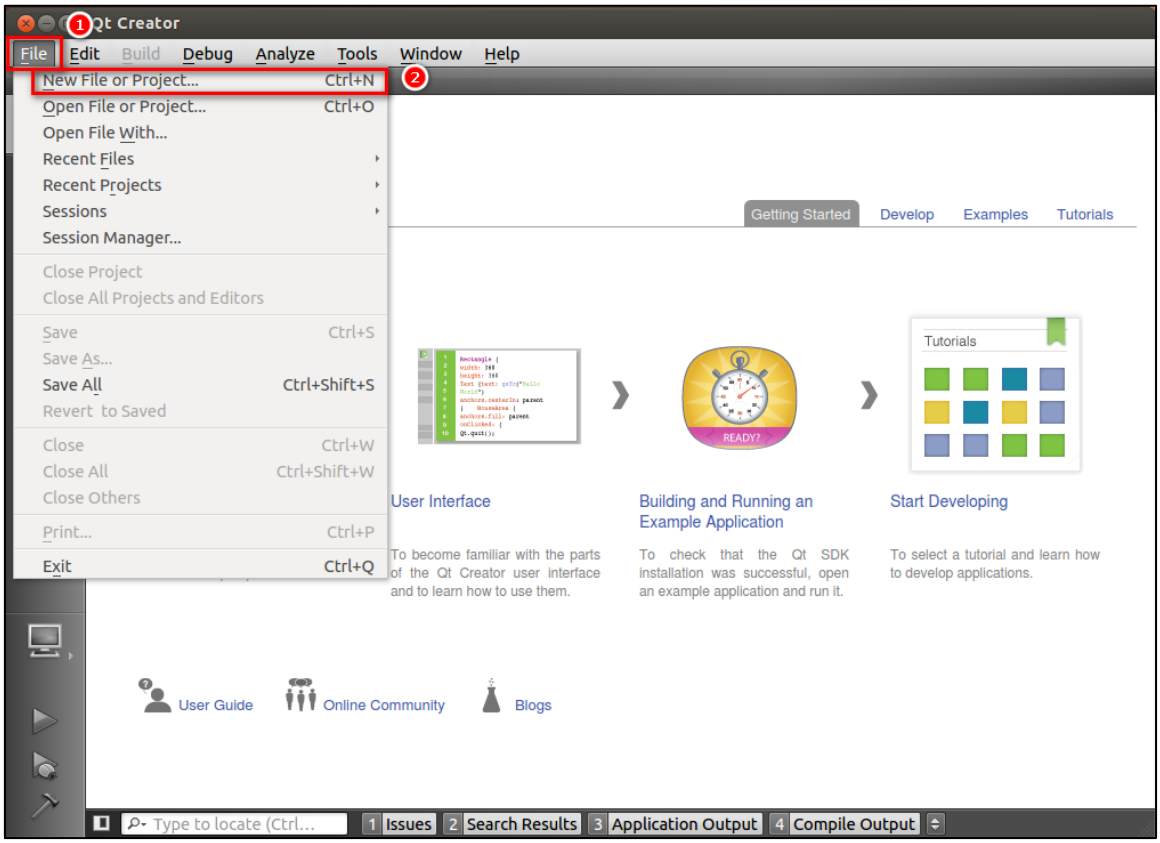


图 6.3 1 新建工程

选择 Qt Gui 工程

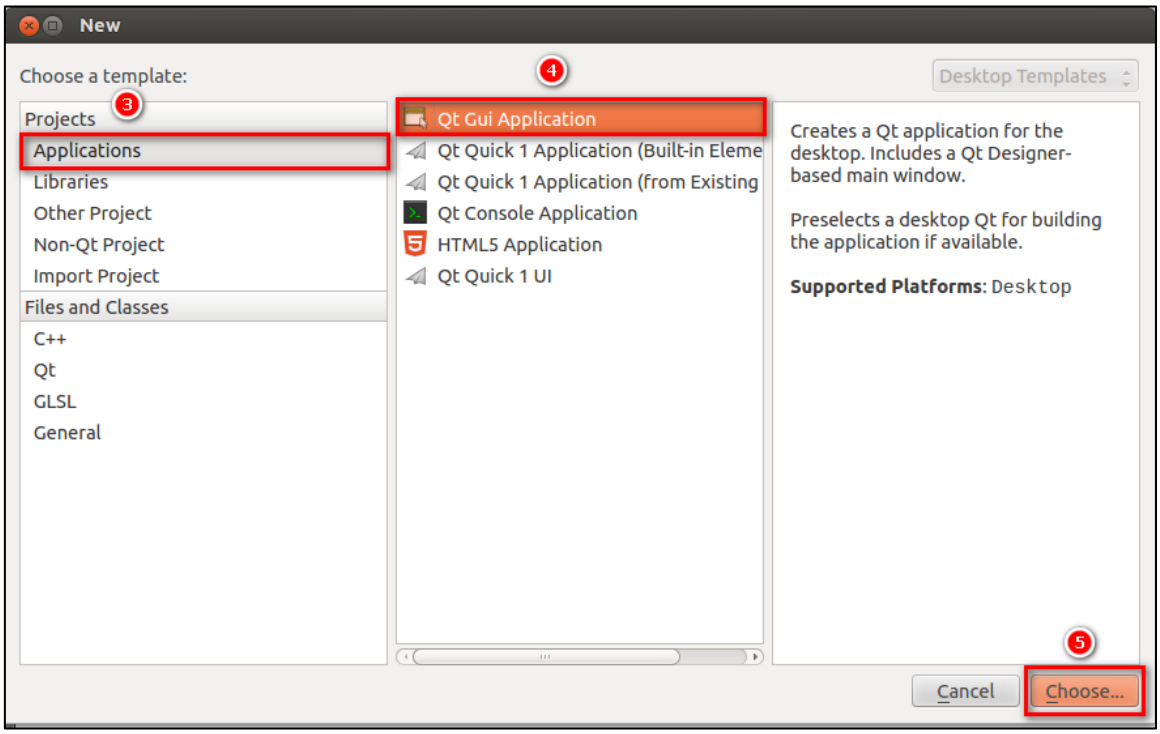


图 6.3 2 选择 Qt Cui 工程模板

工程命名为 test，选择工程的位置，这里位置不要随便选择，建议放在家目录下（/home/用户名）。

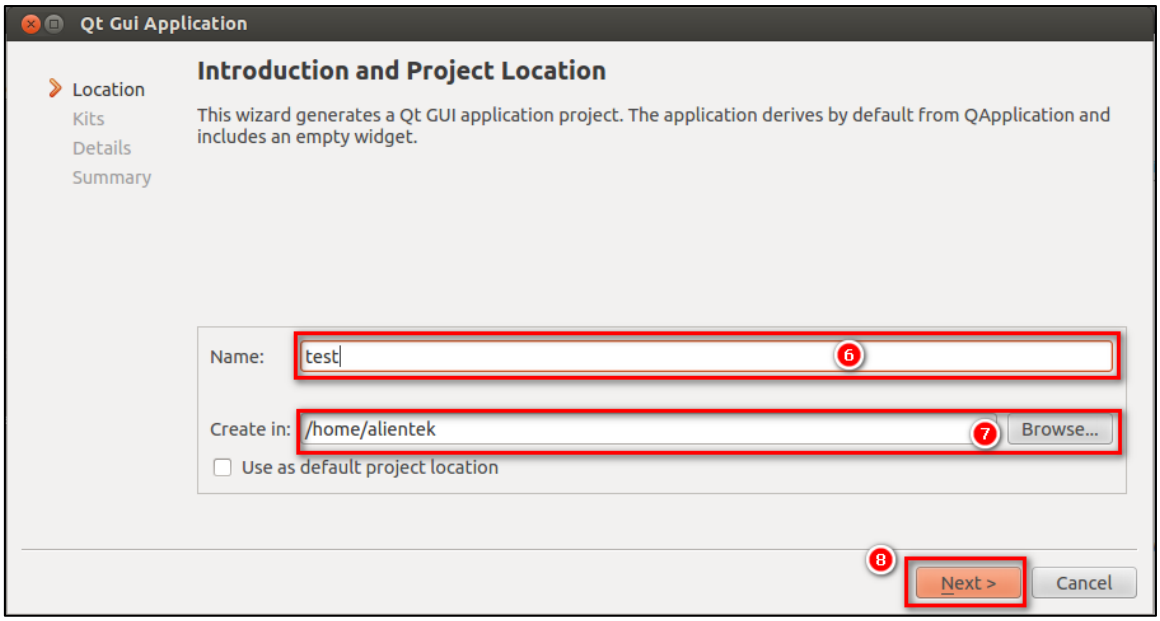


图 6.3 3 命名工程为 test

可以看到我们在第 6.2 小节配置的名为 Desktop 的 Kits 已经生效，点击 Next。

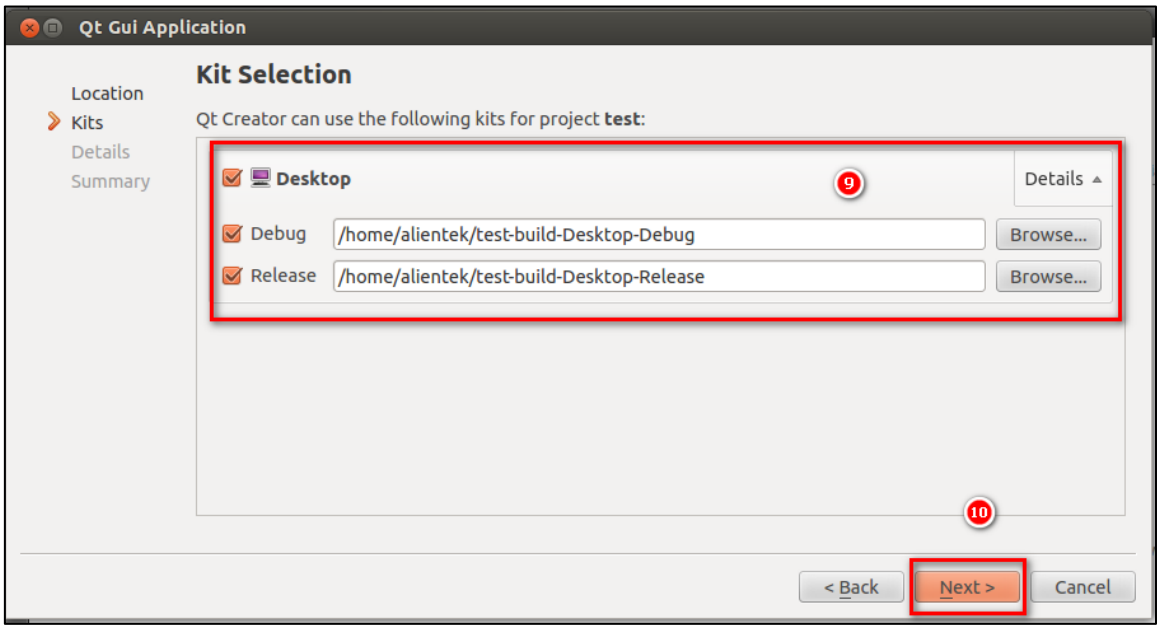


图 6.3 4 选择 Destop 套件

类的设置，默认让它继承 QMainWindow 类，点击 Next。

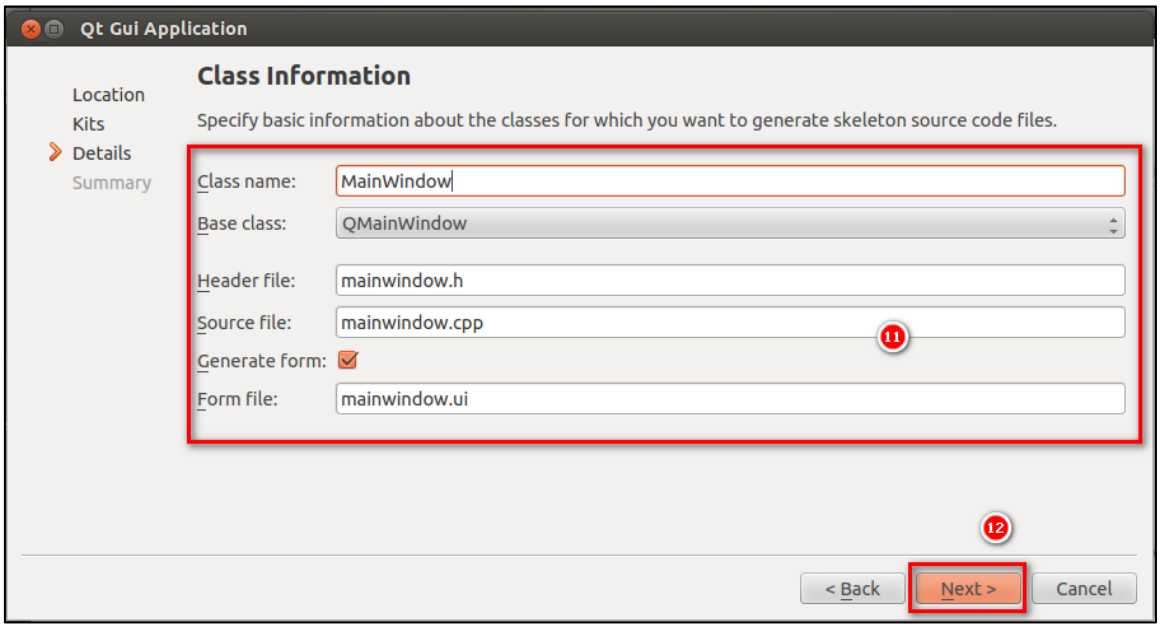


图 6.3 5 默认类的模板

询问是否添加版本控制，我们不用版本管理，直接点击下一步。

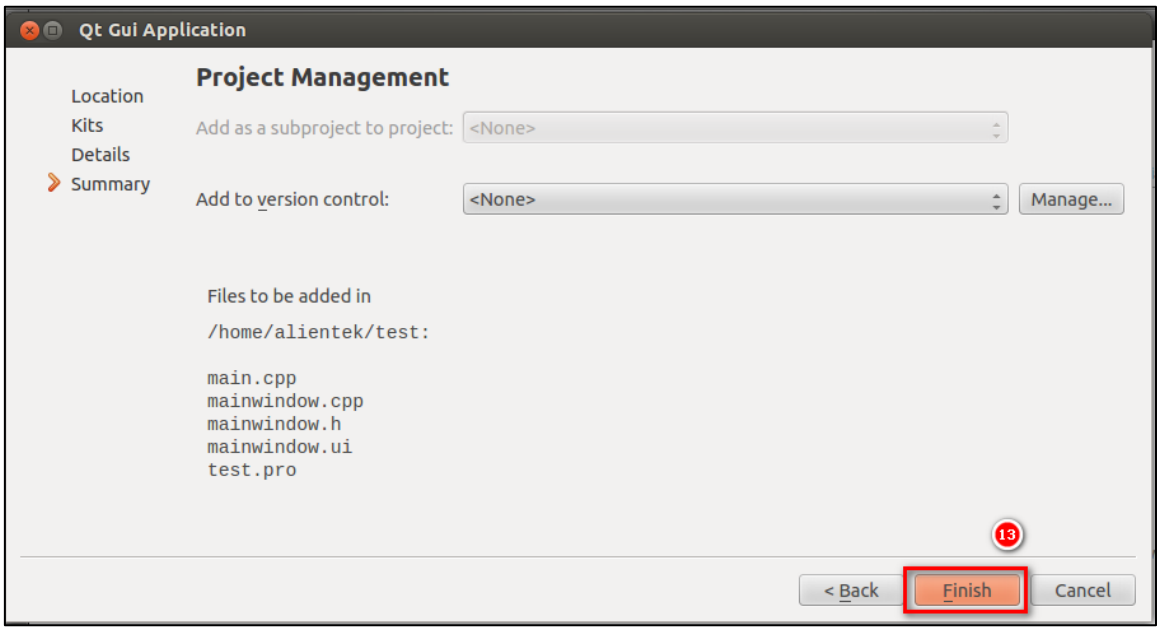


图 6.3 6 完成工程的创建

按如下图步骤编译工程，可以看到我们新建的工程能够运行在 Ubuntu 平台上了。编译程序的运行的结果是一个空白窗口。

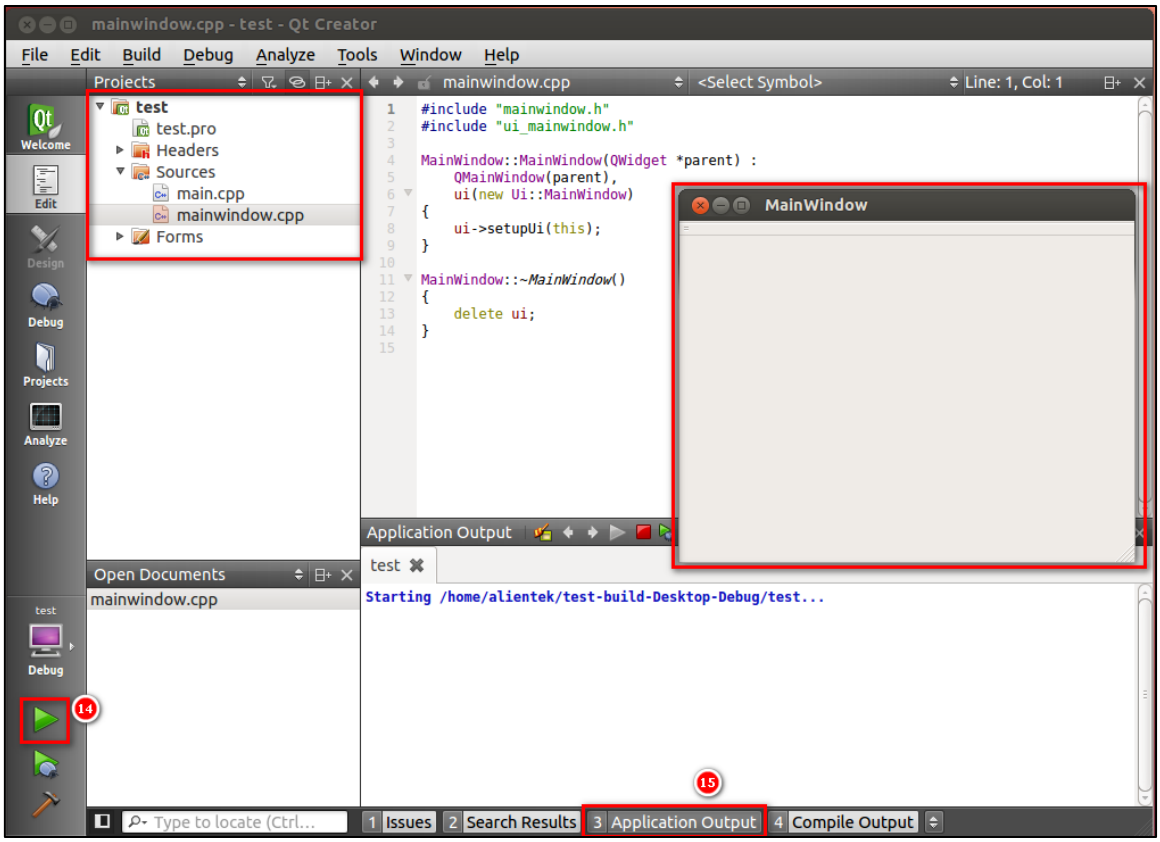


图 6.3 7 运行工程

第七章 搭建 ARM 平台的 Qt Creator 环境

本章将手把手搭建 ARM 平台的 Qt Creator 环境。

7.1 配置 ARM 平台的 Qt Creator Kits

和 6.2 小节一样，我们只需要选择 ARM 的 qmake 及 ARM 平台的编译器即可！
打开 Qt Creator 界面，找到 Tools》Options。

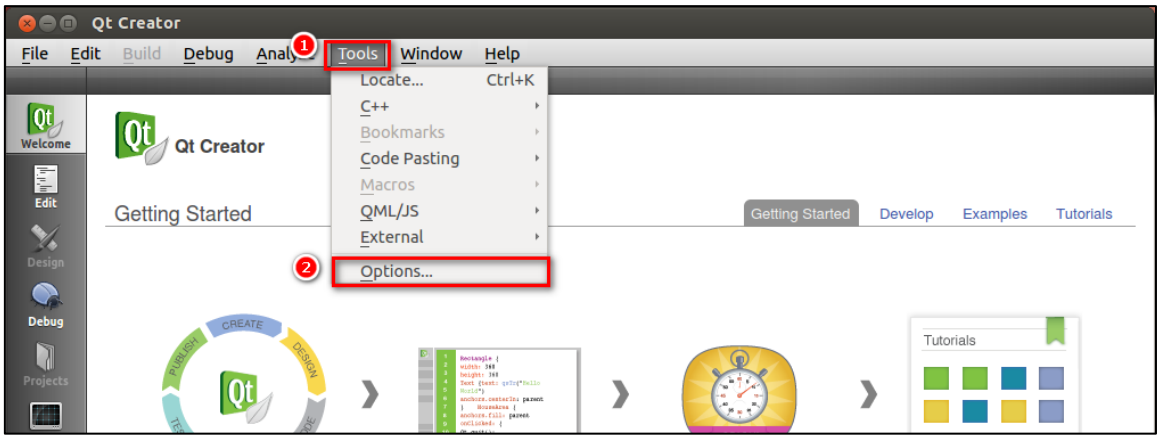


图 7.1 1 打开选项（Options）

➢ 配置 qmake

按下图步骤，找到我们第 3.4 小节编译出来的 ARM 平台的 Qt 库安装文件夹 arm-qt 下的 qmake。

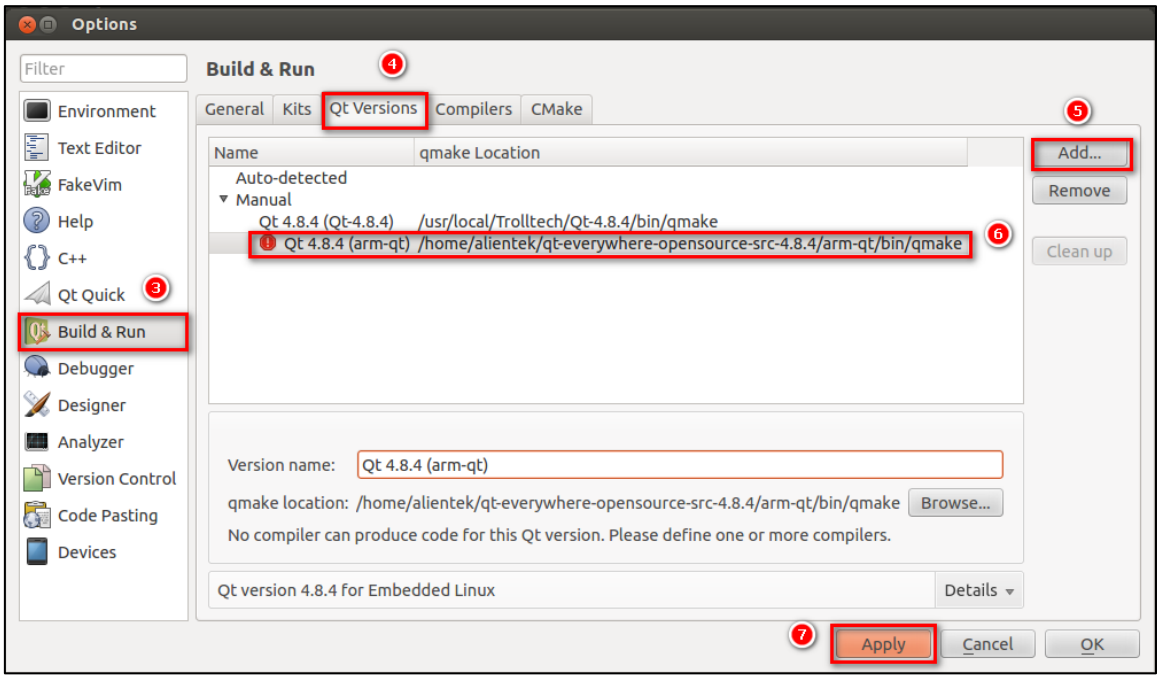


图 7.1 2 配置 Arm 平台的 qmake

➢ 配置 C++编译器

我们在 1.3 小节安装的 C++编译器为 arm-linux-gnueabihf-g++。按如下步骤配置，选择 Compiler path 的路径为我们 1.2 小节安装的交叉编译器路径/usr/local/arm/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-g++。并在第 13 步修改 Name 为 ARM-GCC。

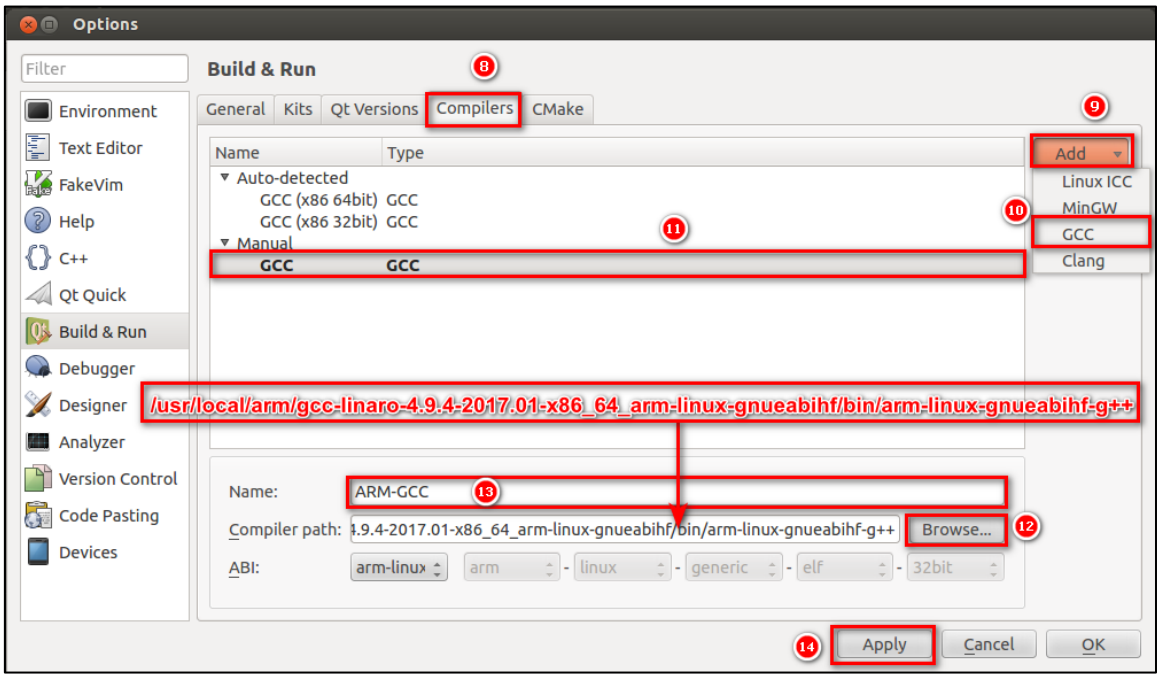


图 7.1 3 配置 C++编译器

➢ 配置 Kits

按如下图步骤操作，在 Name 处改名为 Arm-Qt4.8.4。选择我们配置好的 Compiler 和 Qt Version 即可。

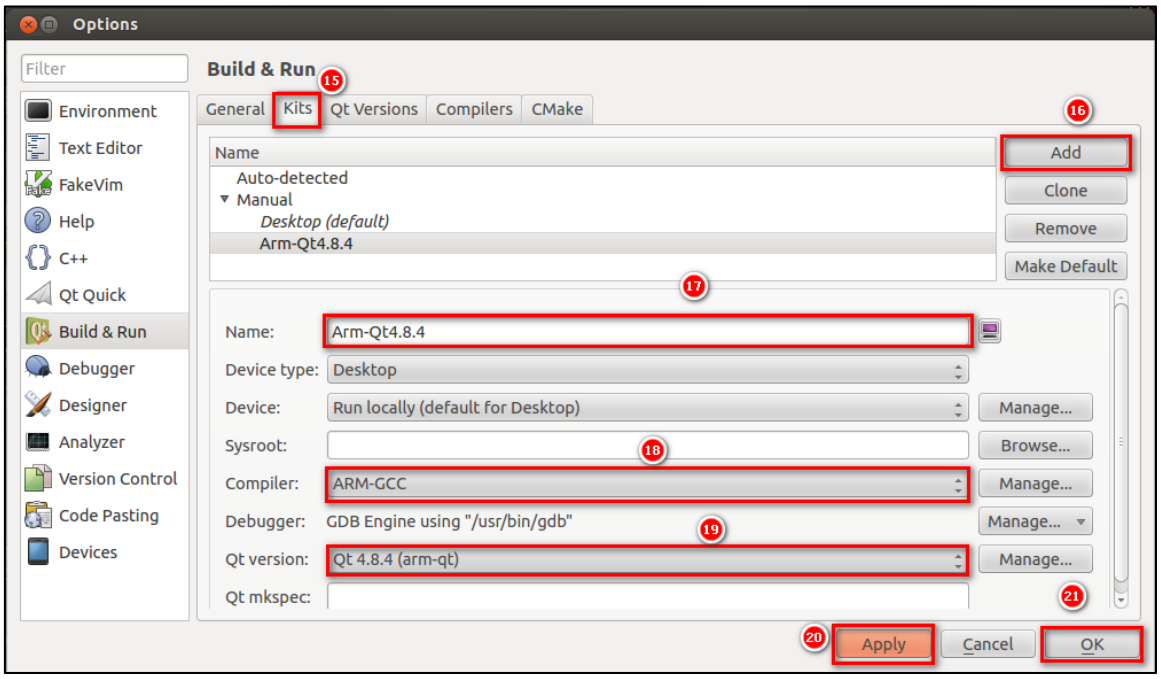


图 7.1 4 配置 Kits

7.2 验证 ARM 平台的 Qt 编译

注意，这里只能编译，不能点击运行，这里是编译出 ARM 平台的 Qt 应用程序，所以是不能在 Ubuntu 这样的 X86 平台上跑的。

和 6.3 小节一样，新建一个工程为 test2，如下图，按步骤操作。

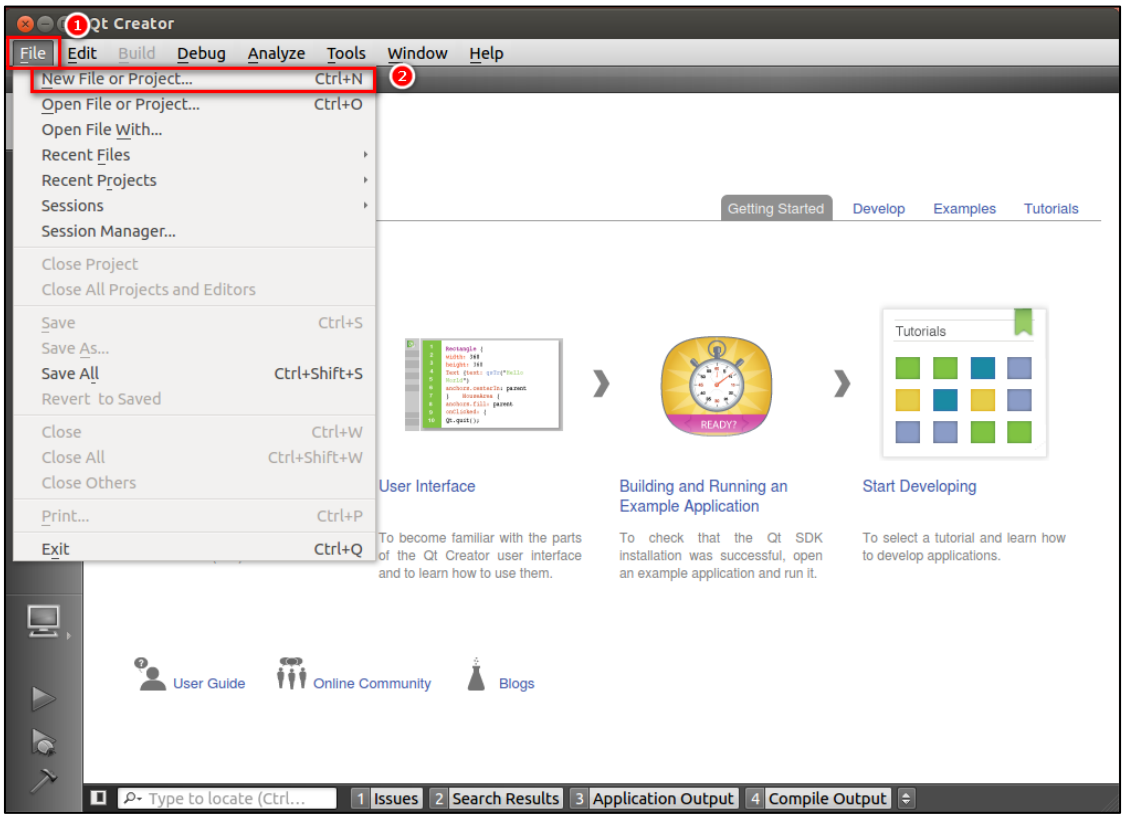


图 7.2 1 新建工程

选择 Qt Gui 工程。

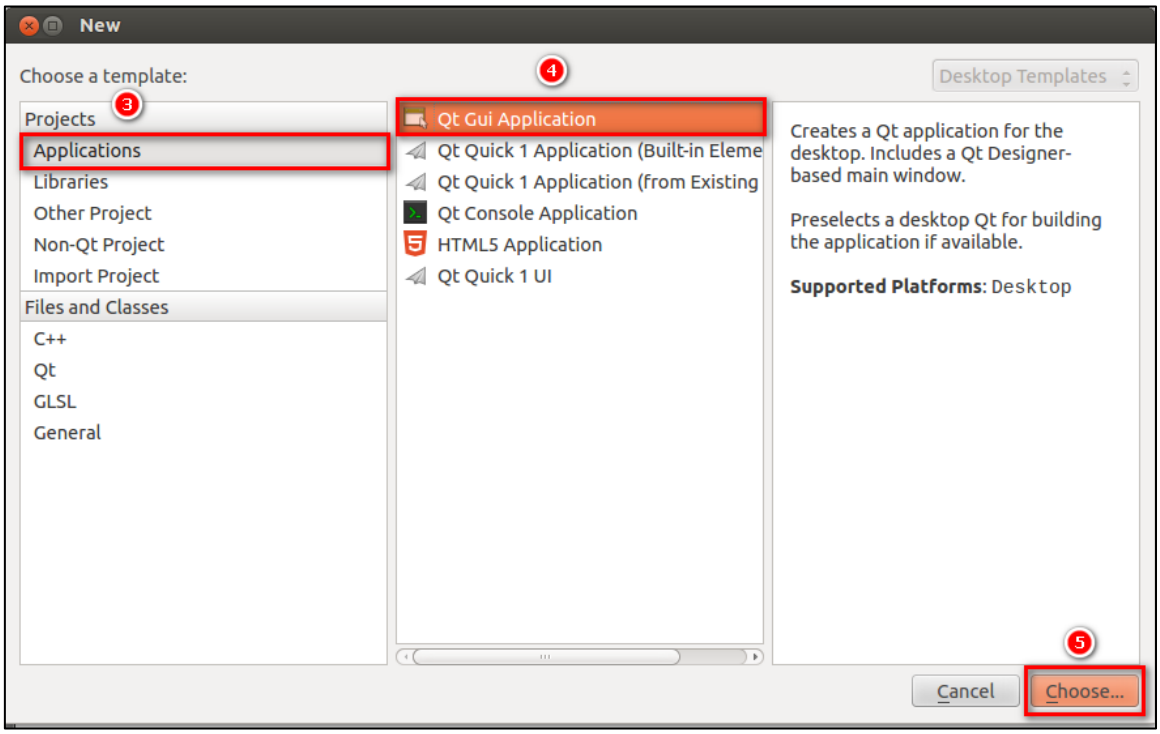


图 7.2 2 选择 Gui 模板

工程命名为 test2，选择工程的位置，这里位置不要随便选择，建议放在家目录下（/home/用户名）。

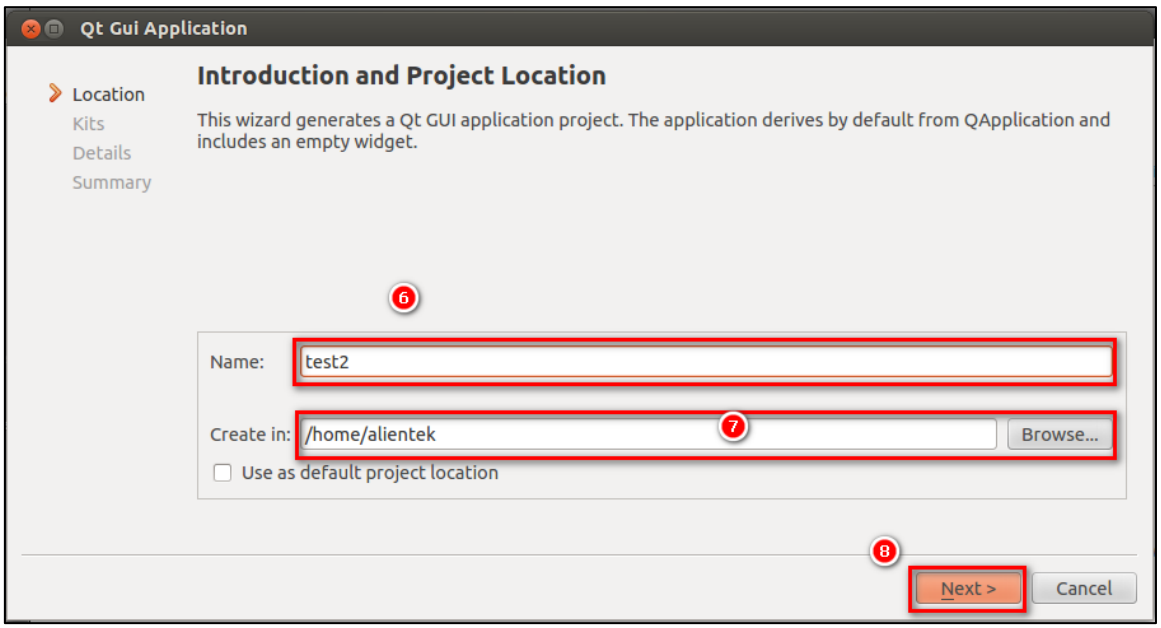


图 7.2 3 命名工程为 test2

选择套件，这里可以两个套件一起选，为了演示，我们只选择 ARM 平台的套件，如下。

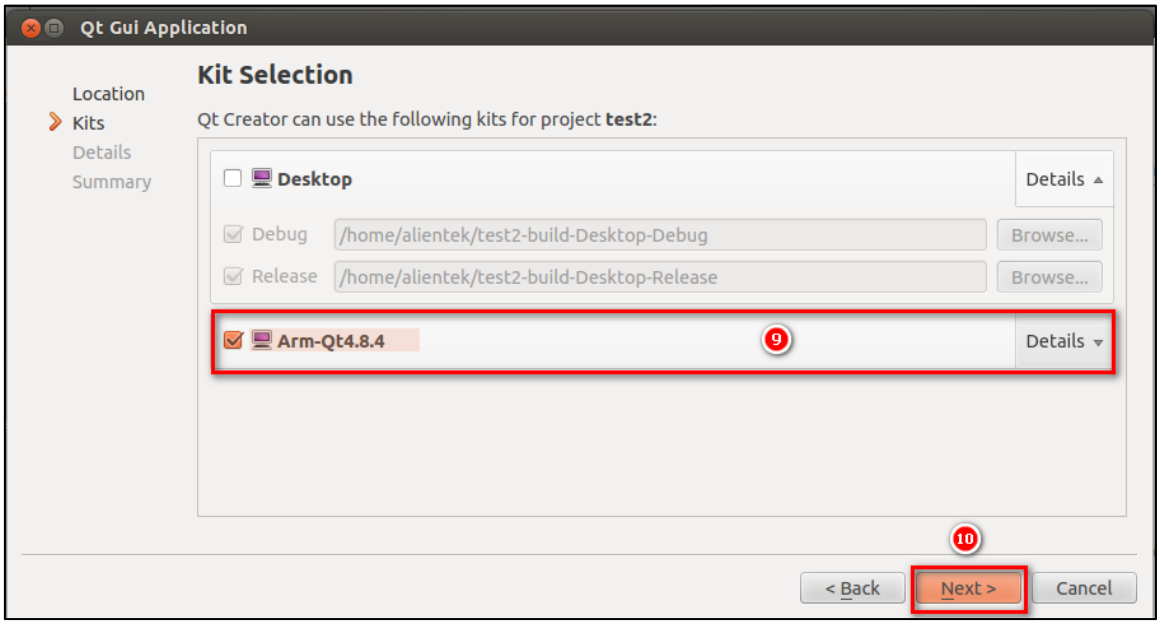


图 7.2 4 选择 Arm 平台的套件

默认，继续点击 Next

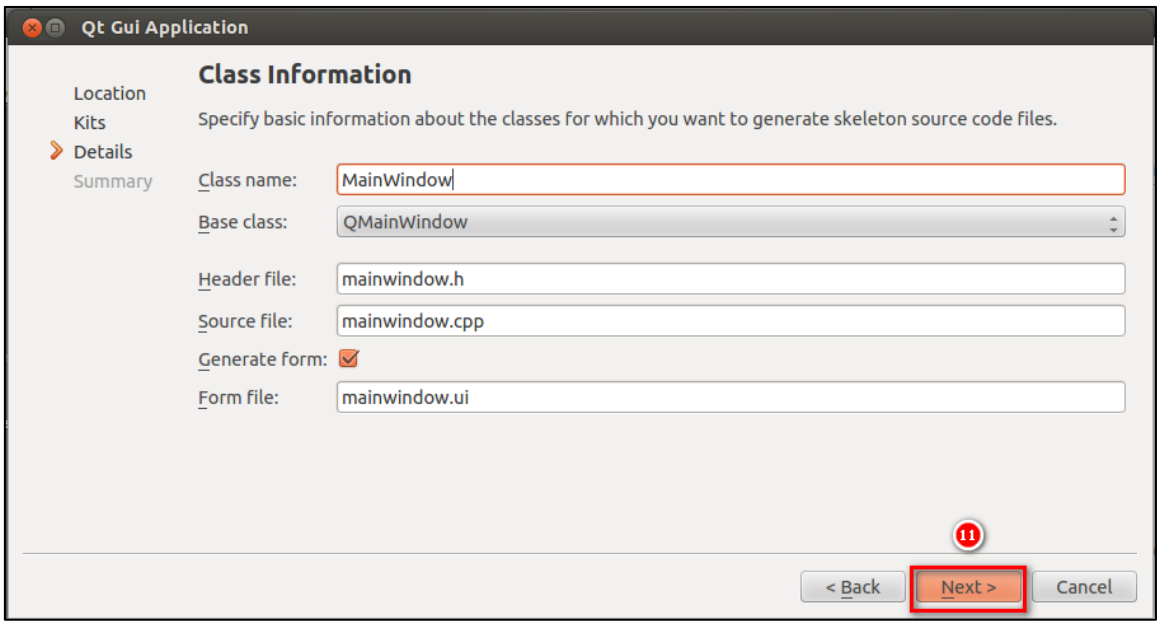


图 7.2 5 默认工程模板，点击 Next

默认，点击 Finish，完成。

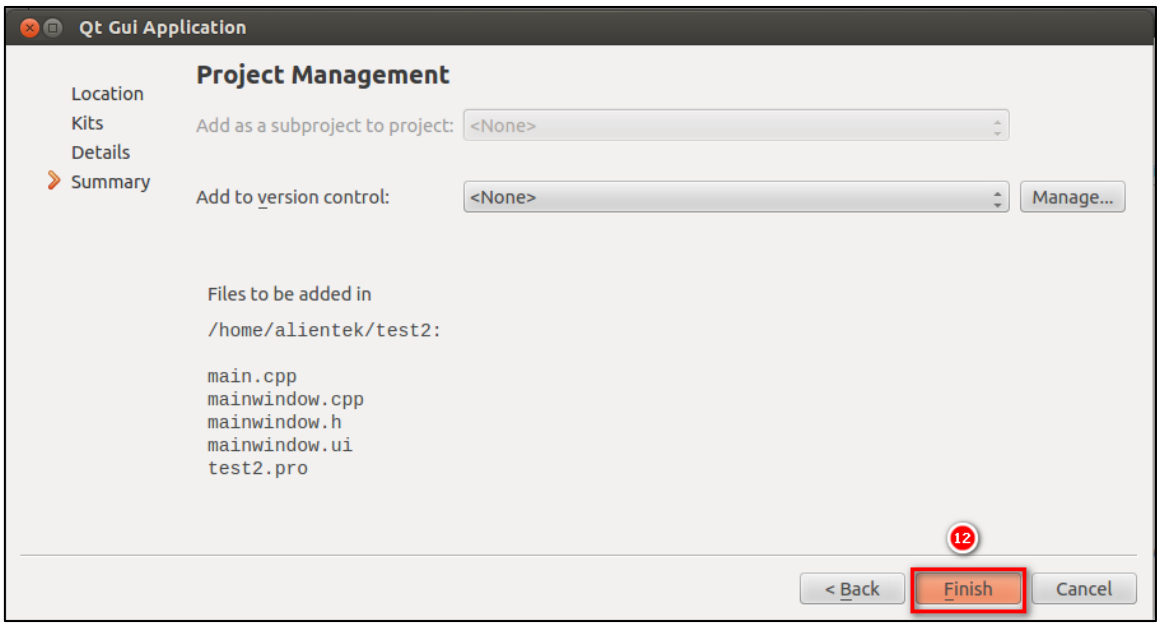


图 7.2 6 创建工程完成

按如下步骤点击 Build（构建），不要点击运行！也不要点击左下角的绿色三角符号。编译完成如下图，编译输出窗口。

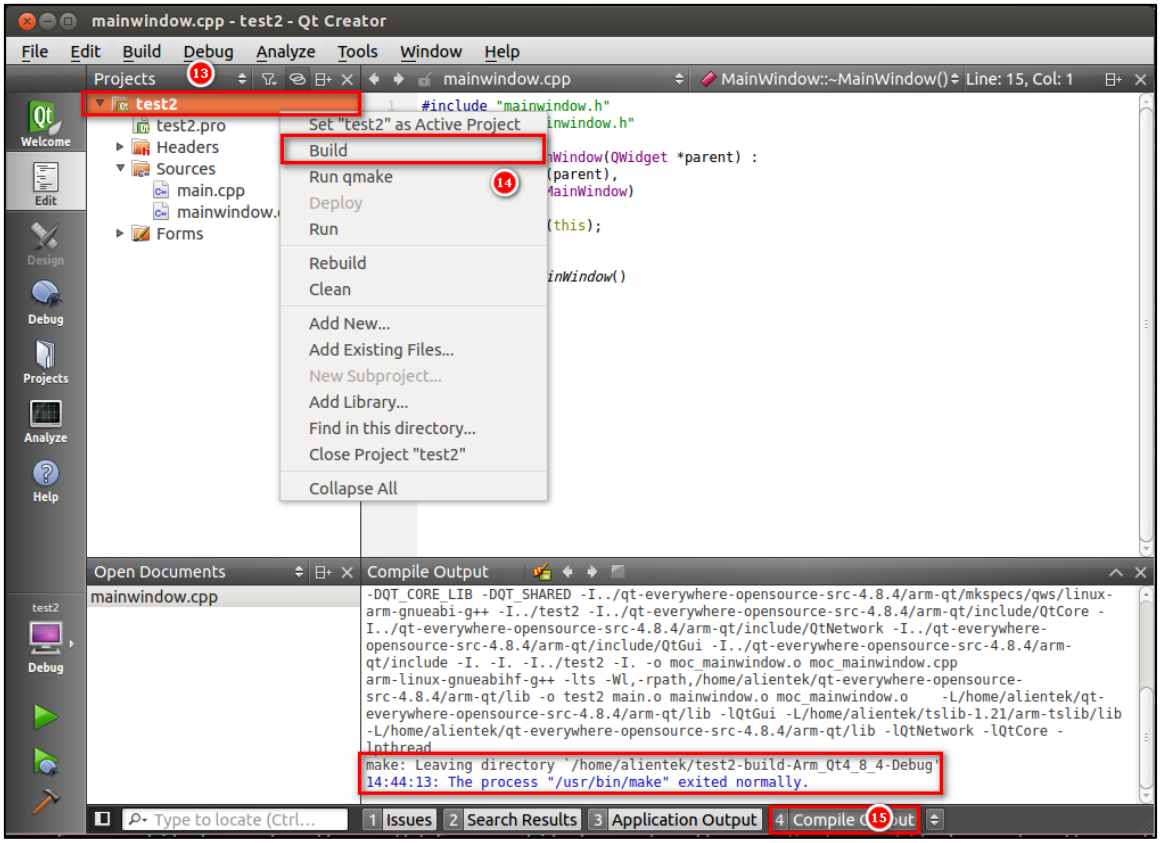


图 7.2 7 编译完成，注意不要点运行！

我们可以在工程目录所在的同一级目录下，找到 test2-build-Arm_Qt4_8_4-Debug 文件夹，找到编译可执行文件，拷贝 test2 文件到 4.1 小节开发板文件系统上执行即可！执行命令为./test -qws。

```
alientek@ubuntu:~$ cd test2-build-Arm_Qt4_8_4-Debug/
alientek@ubuntu:~/test2-build-Arm_Qt4_8_4-Debug$ ls
main.o  mainwindow.o  Makefile  moc_mainwindow.cpp  moc_mainwindow.o  test2  ui_mainwindow.h
alientek@ubuntu:~/test2-build-Arm_Qt4_8_4-Debug$
```

图 7.2 8 Arm 平台下的可执行文件 test2

至此，已经完成关于 Qt4.8.4 的移植及 Qt Creator 环境搭建。

附录-A