

Purple Pi R1 应用开发

LVGL的开发

简介

下载LVGL源码

下载LVGL_8.2源码

编译LVGL源码

使用GUI Guider开发LVGL应用

安装JDK

安装GUI Guider

在GUI Guider上开发应用

交叉编译lvgl应用

在开发板上运行lvgl应用

ffmpeg的移植测试

软路由配置

场景一 WiFi工作为AP模式，其他设备通过连接WiFi热点能够上网

场景二 WiFi工作为STA模式，连接其他热点，2个网口通过WiFi上网

场景三 网口0作为WAN口，其他设备连接网口1后可以上网

Qt Creator开发流程

配置Qt开发环境

Qt Creator添加Qt4.8.7开发套件

Qt Creator添加Qt4.8.7交叉编译

编写Qt 程序在开发板中运行



Purple Pi R1

应用开发

深圳触觉智能科技有限公司

www.industio.cn

LVGL的开发

简介

LVGL是一个轻量级的，开源的图形库。本文详细介绍了如何在Purple Pi开发板上运行lvgl应用，同时介绍了如何使用著名的LVGL IDE工具GUI Guider来开发应用程序。

下载LVGL源码

下载LVGL_8.2源码

基于lvgl v8.2，我们已经适配了Purple Pi开发板，并且放在github上面，你可以通过以下方法下载：

```
1  industio@industio$:git clone https://github.com/industio/ido-lvgl8.2.git
```

编译LVGL源码

在编译LVGL源码之前，请确保已经安装了交叉编译链arm-linux-gnueabi-gcc：

Plain Text

复制代码

```
1  industio@industio$: which arm-linux-gnueabihf-gcc
2  /home/ronnie/work3/ssd201/gcc-arm-8.2-2018.08-x86_64-arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gcc
```

Plain Text

复制代码

```
1  industio@industio$:cd ido-lvgl8.2
2  industio@industio$:mkdir objs
3  industio@industio$:mkdir output
4  industio@industio$:make clean
5  industio@industio$:make -j16
```

如果生成output/app，说明LVGL源码编译成功。

使用GUI Guider开发LVGL应用

安装JDK

GUI Guider依赖JDK工具，因此需要先安装JDK工具。

到<https://www.oracle.com/java/technologies/downloads/>下载并安装最新的JDK。

安装GUI Guider

登录<https://www.nxp.com/design/software/development-software/gui-guider:GUI-GUIDER>，往下翻页找到GUI Guider v1.3.0并下载：

Downloads

☒ NXP (14)

FILTER BY

Development Software

☐ UI Developer

1-5 of 14 downloads

Sort by Relevance ▾

UI DEVELOPER

GUI Guider v1.3.1 installer for MacOS 11 **FEATURED**

PKG Rev 1.3.1 Mar 31, 2022 270636 KB Gui-Guider-Setup-1.3.1-GA-Mac
Sign in required

DOWNLOAD ▾

UI DEVELOPER

GUI Guider v1.3.1 installer for Ubuntu 20.04 **FEATURED**

DEB Rev 1.3.1 Mar 31, 2022 129512 KB Gui-Guider-Setup-1.3.1-GA-Lin
Sign in required

DOWNLOAD ▾

UI DEVELOPER

GUI Guider v1.3.1 installer for Windows 10 **FEATURED**

EXE Rev 1.3.1 Mar 31, 2022 359519 KB Gui-Guider-Setup-1.3.1-GA-Win
Sign in required

DOWNLOAD ▾

UI DEVELOPER

GUI Guider v1.3.0 installer for Windows 10

EXE Rev 1.3.0 Jan 24, 2022 337539 KB Gui-Guider-Setup-1.3.0-GA-Win
Sign in required

DOWNLOAD ▾

UI DEVELOPER

GUI Guider v1.0.0 Installer (Windows 10)

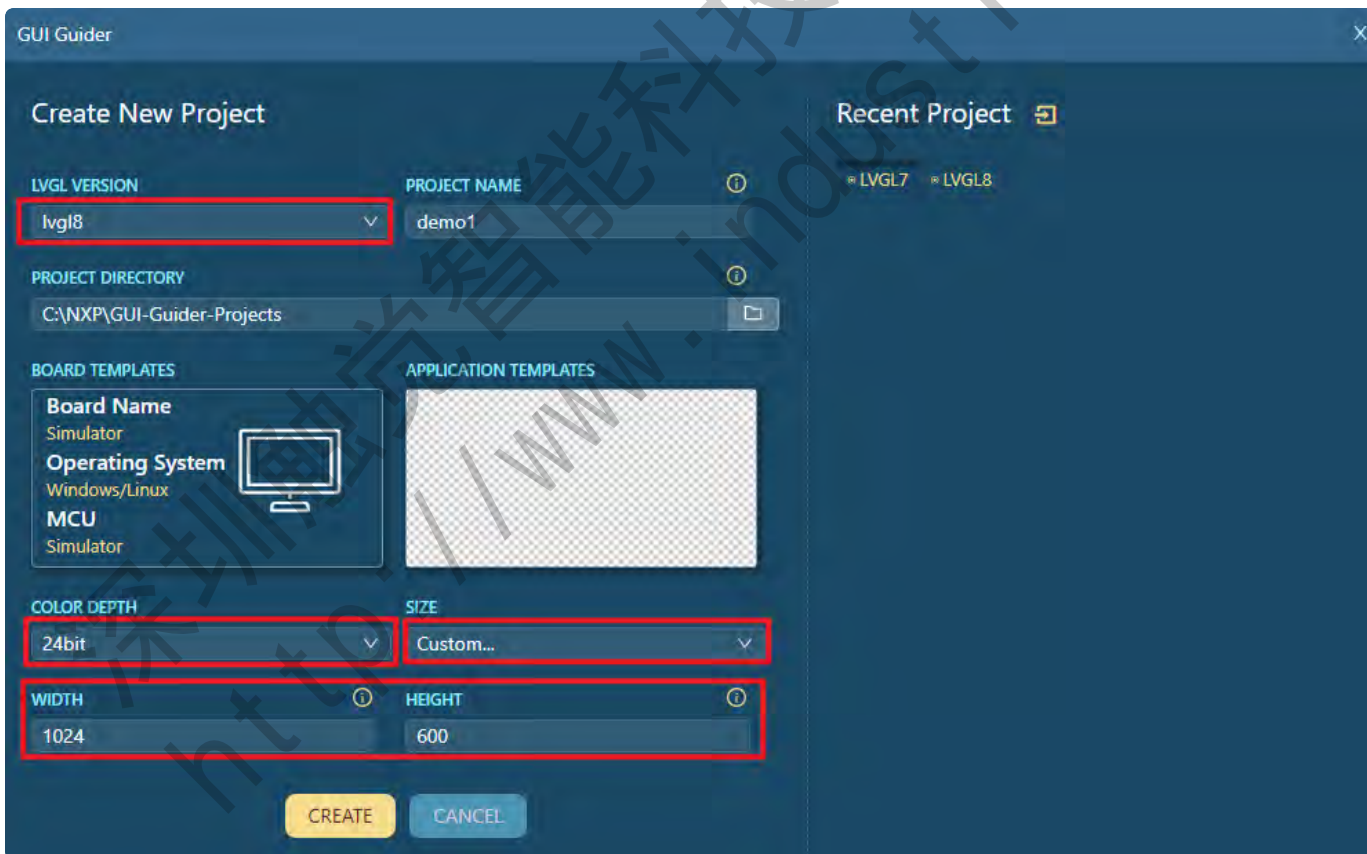
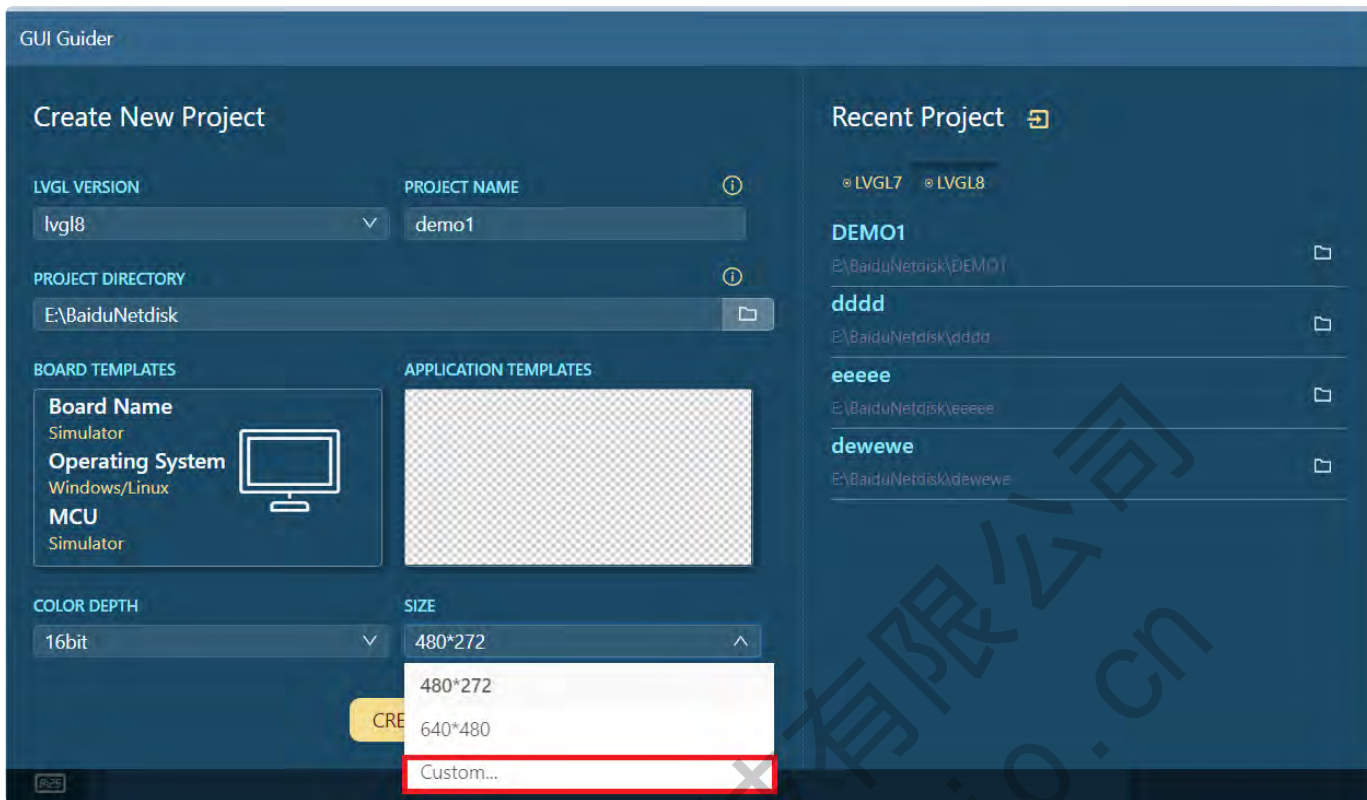
EXE Rev 0 Jan 15, 2021 288781 KB Gui-Guider-Setup-1.0.0-GA-Win
Sign in required

DOWNLOAD ▾

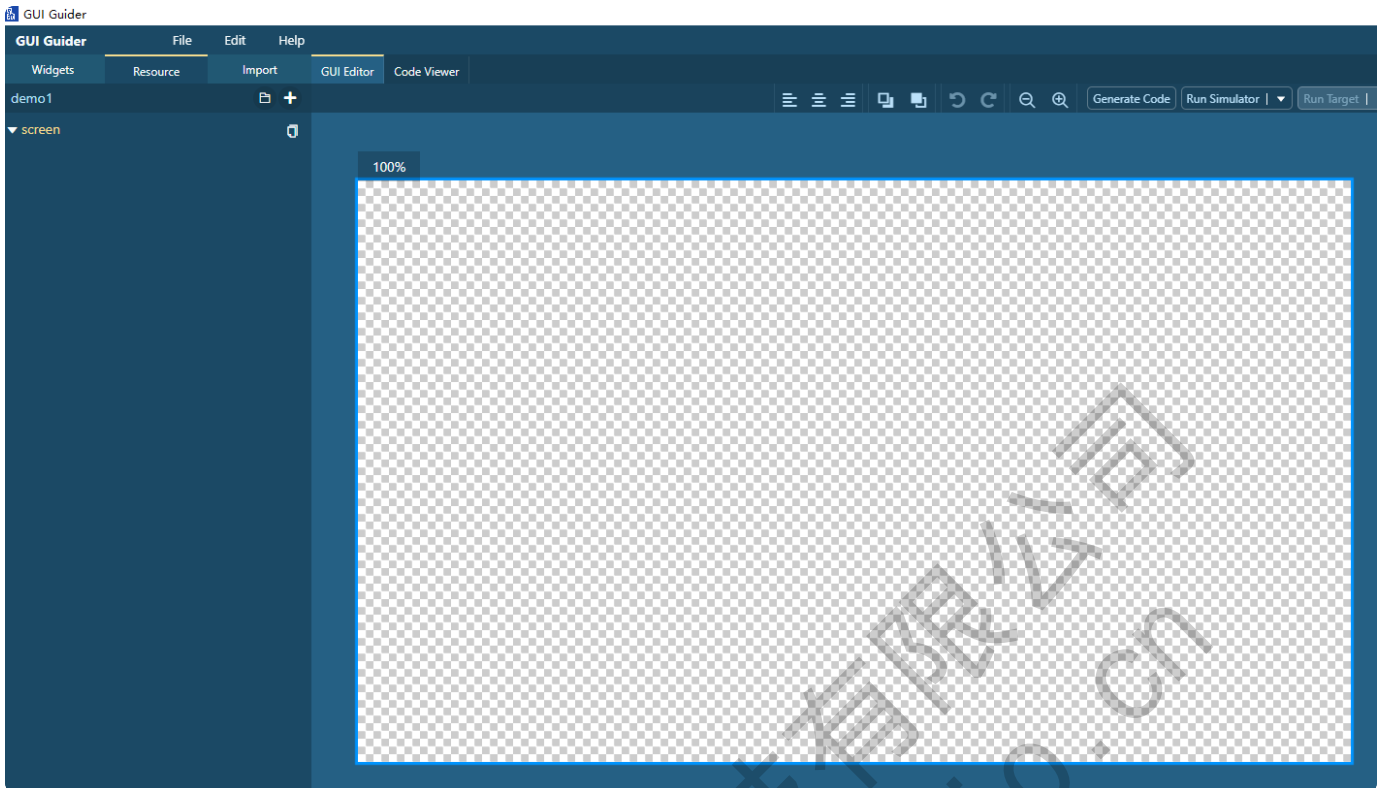
安装包下载完成后，双击安装即可。

在GUI Guider上开发应用

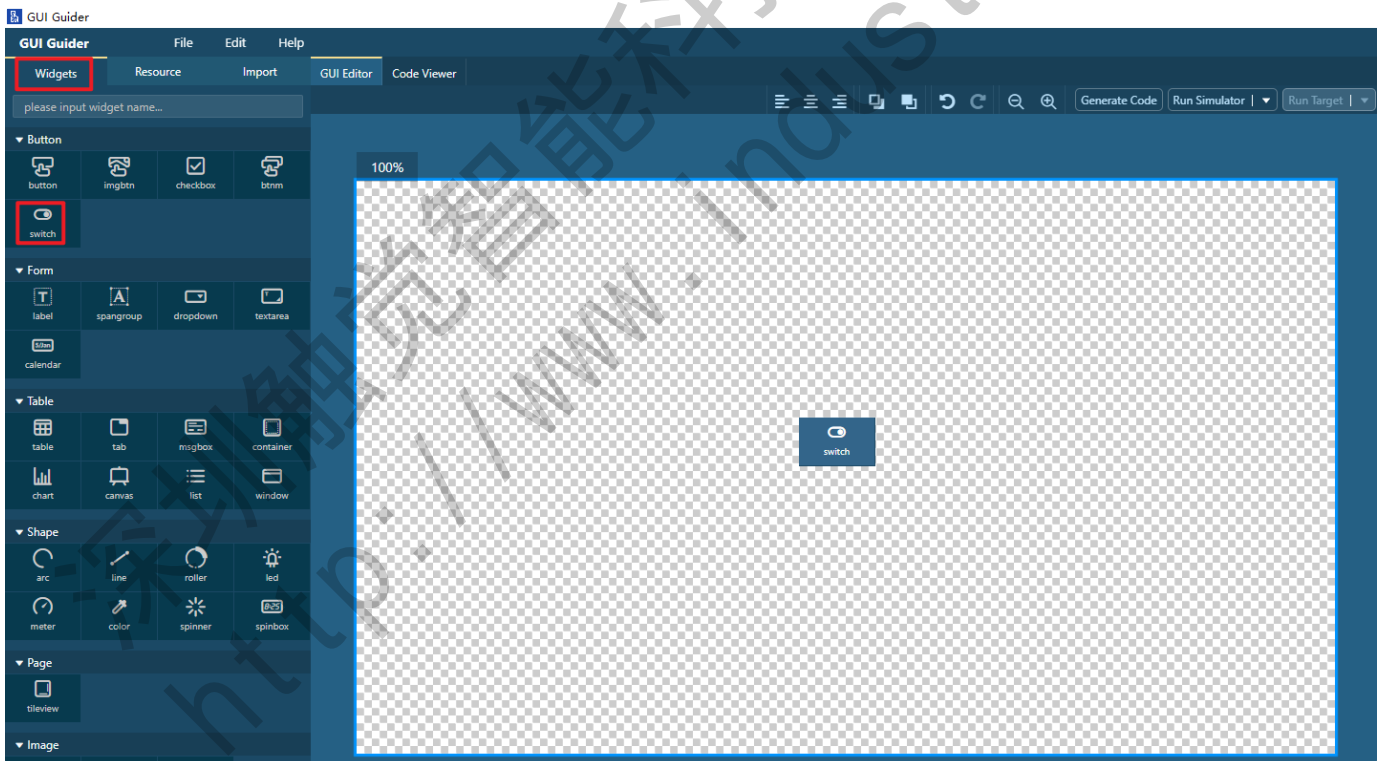
打开GUI Guider，新建一个demo1项目，参数设置请保持和下面图片一致：



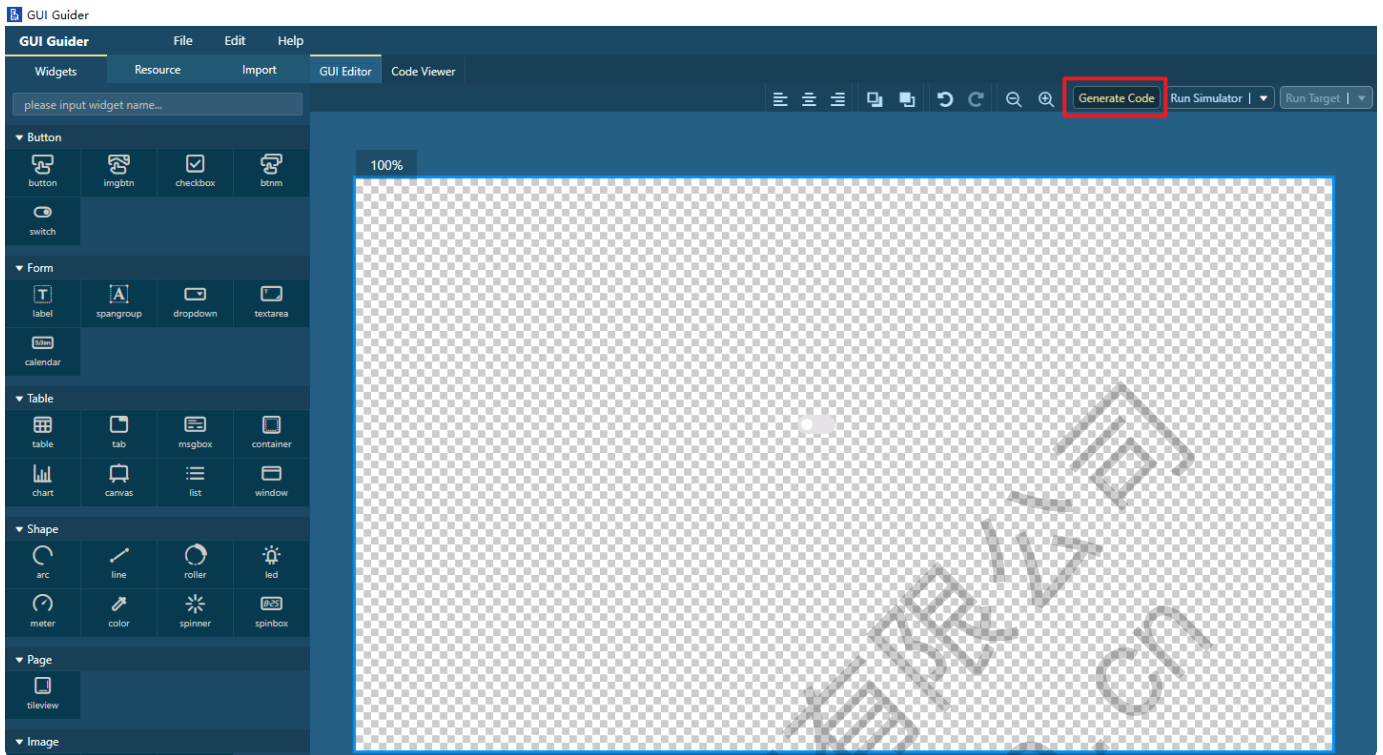
点击下方CREATE按钮，项目创建完成：



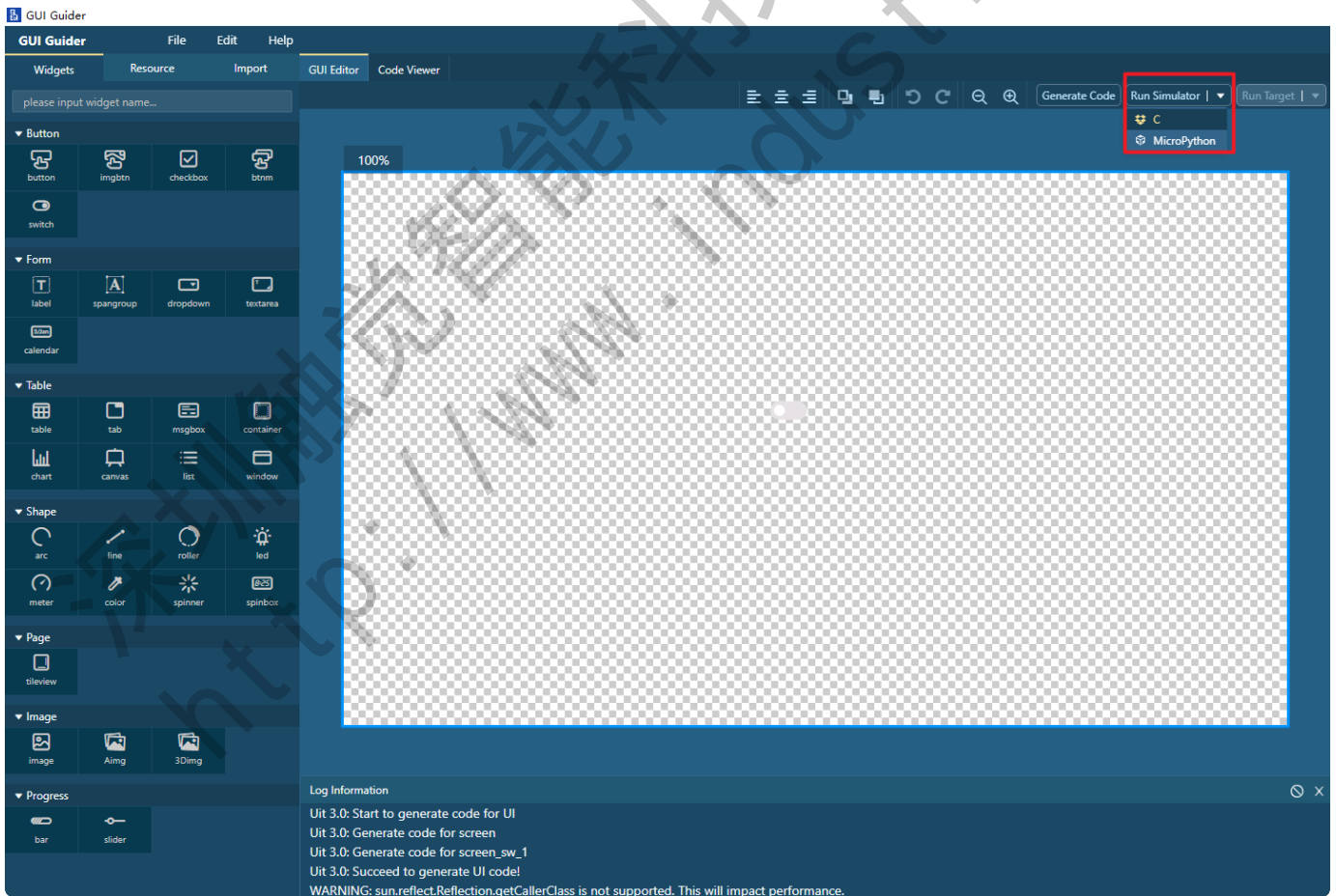
选择Widgets页，拖动放置一个switch器件：



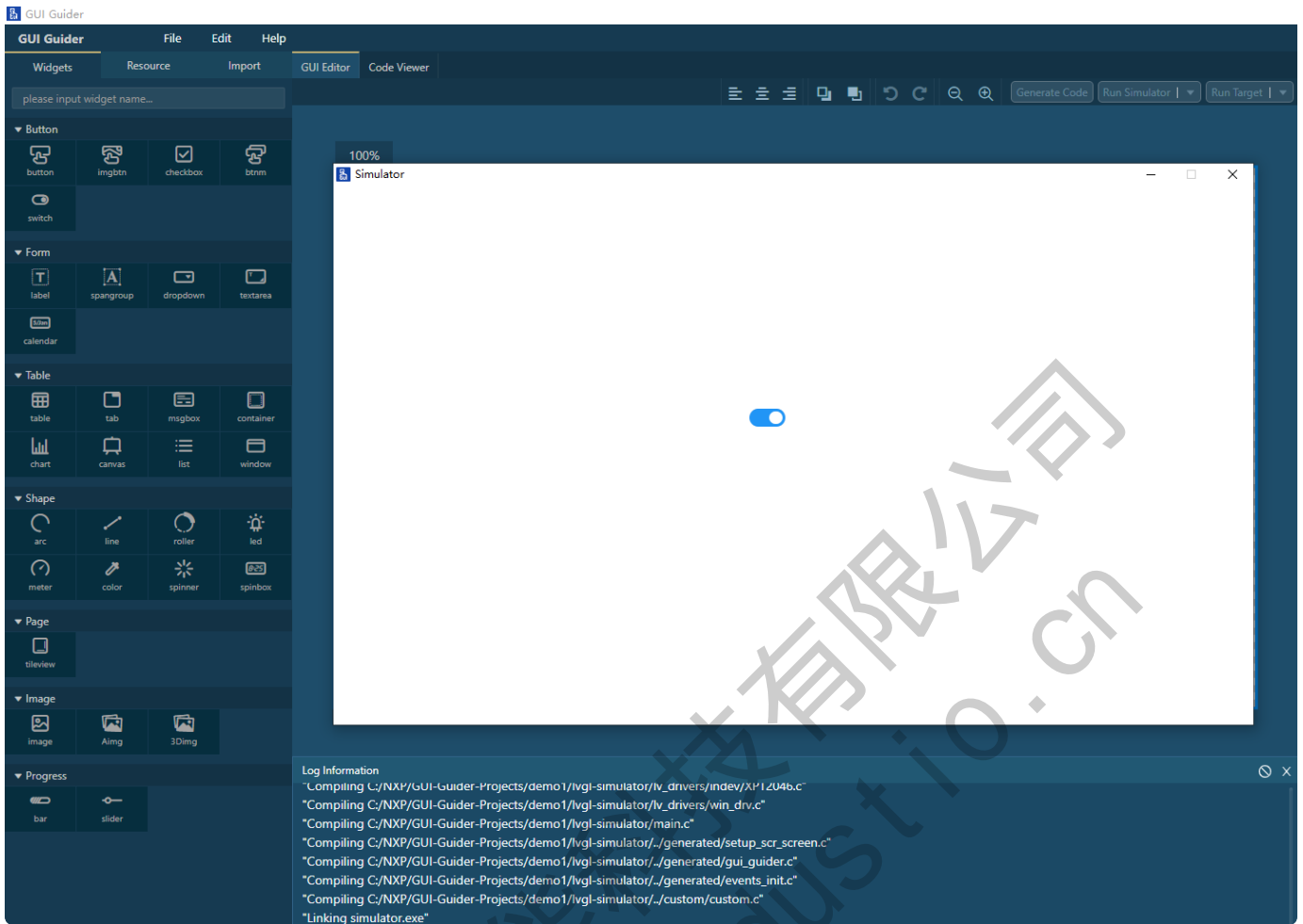
点击右上角Generate Code按钮：



Generate Code完成后，继续点击Run Simulator->C，在电脑上模拟运行：

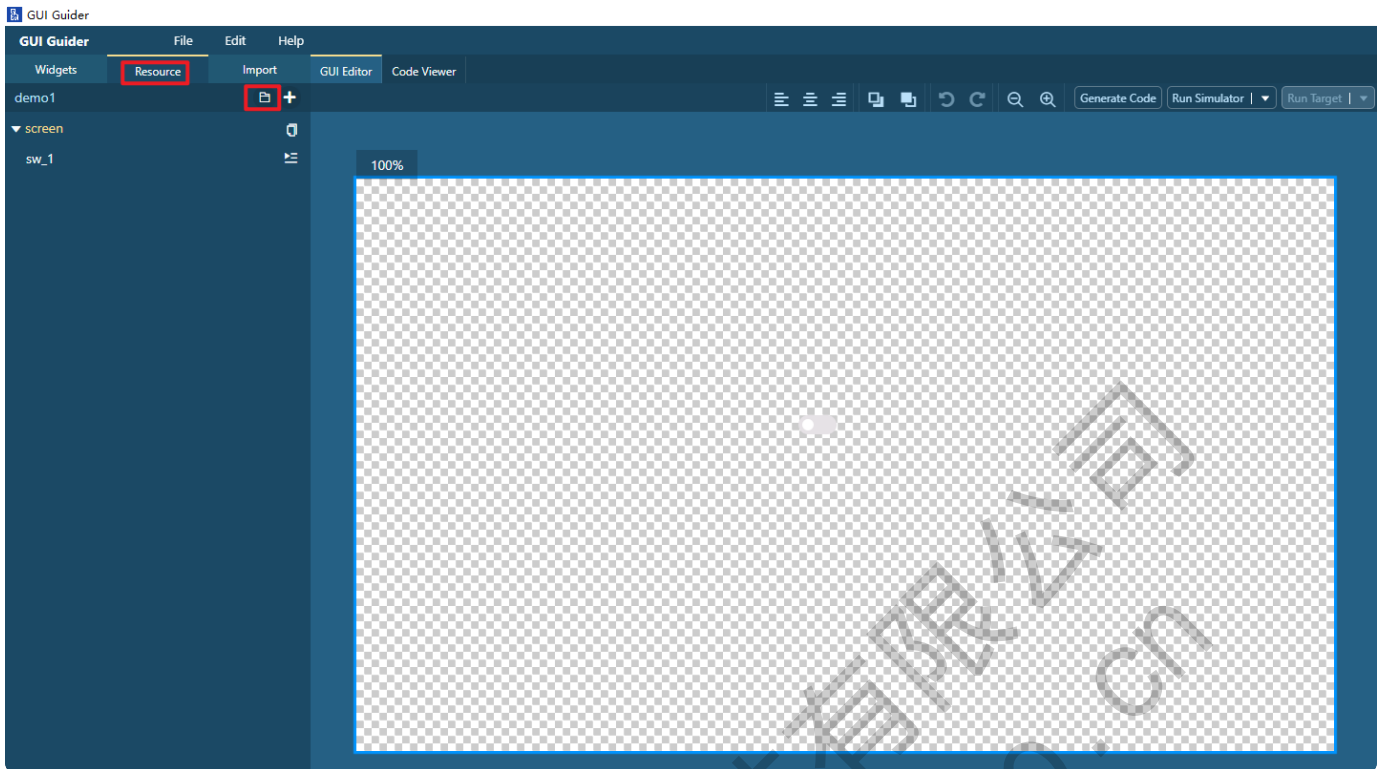


可以看到，刚刚放置的按钮显示在窗口上：

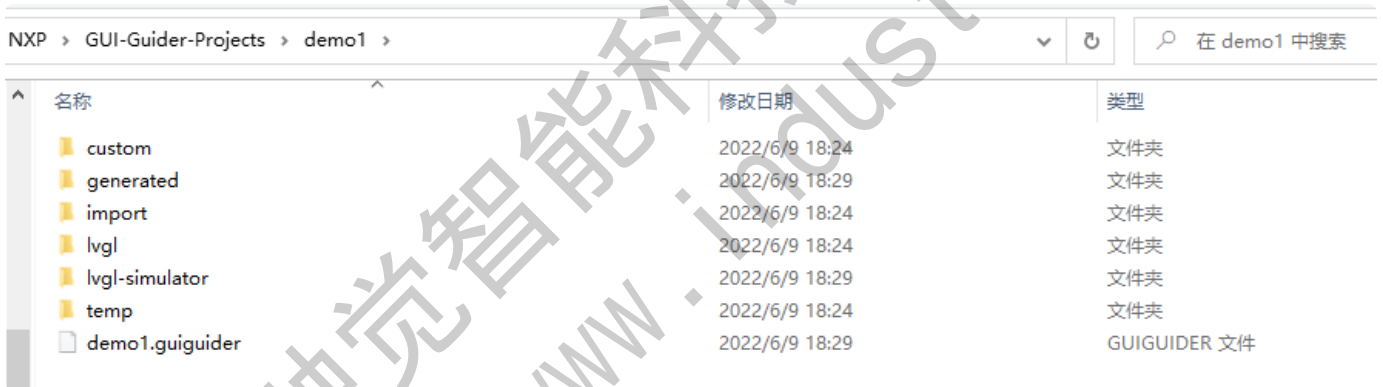


交叉编译lvgl应用

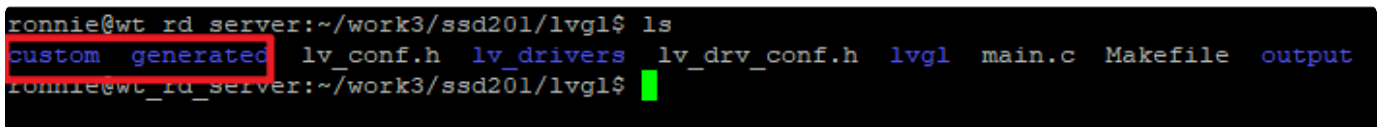
在仿真完成后，停止仿真，点击Resource按钮，然后点击文件夹图标：



在新打开的窗口中，展示了刚才新建的项目源代码所在的位置：



复制generated和custom文件夹，替换lvgl源码的generated和custom文件夹：



在编译前需要配置交叉编译链arm-linux-gnueabi-hf-gcc, 下载gcc-arm-8.2-2018.08-x86_64-arm-linux-gnueabi-hf.tar.gz解压

Plain Text 复制代码

```
1  industio@industio$:tar -xvf gcc-arm-8.2-2018.08-x86_64-arm-linux-gnueabi-hf.tar.gz
2  industio@industio$:export PATH=/home/xxx/gcc-arm-8.2-2018.08-x86_64-arm-linux-gnueabi-hf/bin:$PATH
```

替换完成后, 即可开始编译app:

Plain Text 复制代码

```
1  industio@industio$:make clean
2  industio@industio$:make -j16
```

编译成功, 将生成output/app。

在开发板上运行lvgl应用

将output/app拷贝到开发上, 并赋予可执行权限:

Plain Text 复制代码

```
1  #tftp -g -l app 192.168.1.147
2  #chmod a+x app
```

如果没有初始化屏幕, 先执行disp_init初始化屏幕:

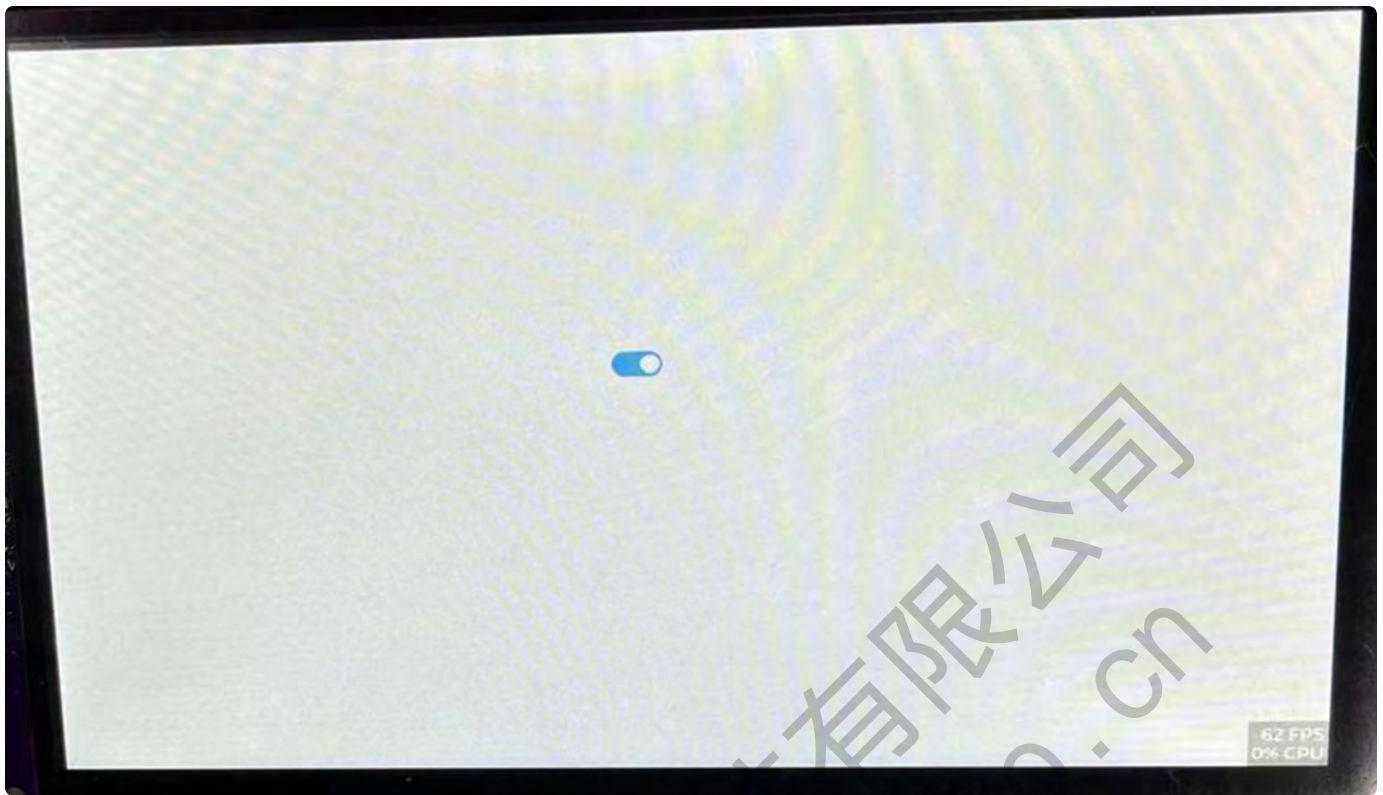
Plain Text 复制代码

```
1  # /customer/disp_init &
```

运行app:

Plain Text 复制代码

```
1  # ./app
```



如果你不想在右下角显示FPS & CPU信息，请修改lv_conf.h，将LV_USE_PERF_MONITOR设置为0：

```
1 /*1: Show CPU usage and FPS count*/
2 #define LV_USE_PERF_MONITOR 0
3 #if LV_USE_PERF_MONITOR
4     #define LV_USE_PERF_MONITOR_POS LV_ALIGN_BOTTOM_RIGHT
5 #endif
```

ffmpeg的移植测试

1、下载源码

git clone <https://github.com/aaron201912/ffmpeg.git>

将ffmpeg-master.zip 放到Ubuntu下解压

```
1 industio@industio$: unzip -x ffmpeg-master.zip ./
```

注意：解压出来的ffmpeg目录需要放在project同一目录下。

2、配置屏参

将7寸MIPI屏参头文件拷贝到ffplayer/app/下。并修改ffplayer/app/sd20xpanel.c和ui_app/player/playerWnd.c，将屏参头文件包含进去。

```
cc0702i50r_1024x600.h  libssplayer.a  main.c  Makefile  sd20xpanel.c  sd20xpanel.o  ss22x_panel.h  ss268_panel.c  ss268_panel.o
EQT700BKJ004P_1024x600_MIPI.h  libssplayer.so  main.o  SAT070CP50_1024x600.h  sd20xpanel.h  ss22x_panel.c  ss22x_panel.o  ss268_panel.h  ssplayer
```

```
1  industio@industio$: vi ffplayer/app/sd20xpanel.c
```

```
#if DISPLAY_1024_600
// #include "SAT070CP50_1024x600.h"
// #include "CC0702I50R_1024x600.h"
#include "EQT700BKJ004P_1024x600_MIPI.h"
#endif
```

```
1  industio@industio$: vi ui_app/player/playerWnd.c
```

```
// #include "SAT070CP50_1024x600.h"
// #include "CC0702I50R_1024x600.h"
#include "EQT700BKJ004P_1024x600_MIPI.h"
#include "usbdetect.h"
#include "frame.h"
```

3、编译

```
1  industio@industio$: industio@industio$: cd ffmpeg-4.1.3/
2  industio@industio$: sh config_for_ssd20x.sh
3  industio@industio$: make clean 【可选】
4  industio@industio$: make -j4
5  industio@industio$: make install
```

生成文件位于host目录下：

```
dynamic include share static
```

4、验证

编译测试app

```
1 industio@industio$: cd ../ffplayer/app
2 industio@industio$: vi Makefile
```

注意:

CHIP ?= SSD20X (选择SSD20X)

```
CURRENT_PATH = $(shell pwd)
CROSS_COMPILE ?=arm-linux-gnueabi-
CC = $(CROSS_COMPILE)gcc
CPP = $(CROSS_COMPILE)g++
AR = $(CROSS_COMPILE)ar

ALKAID_PATH ?= ../../../../
CHIP ?= ssd20x
LIBRARY ?= dynamic
DISPLAY ?= panel
GIT_COMMIT_INFO:="ssplayer library version: git_commit.$(shell cd
t="%h") build_time.$(shell date +%Y%m%d)"

ifeq ($(CHIP),ssd20x)
    #$(shell echo "choose chip ssd20x" > $(shell tty))
else ifeq ($(CHIP),ss268)
    #$(shell echo "choose chip ss268" > $(shell tty))
else ifeq ($(CHIP),ss22x)
    #$(shell echo "choose chip ss22x" > $(shell tty))
else
    $(error invalid chip type)
endif
```

```
1 industio@industio$:make clean
2 industio@industio$:make -j4
```

编译生成可执行文件ssplayer

```
xxquan@rt_rdx_server: /work/ssd201/SSD20X-2D06/ido-sbc2d06-sdk/ffmpeg/ffplayer/app$ ls
EQ700BKJ004P_1024x600_MIP1.h  libssplayer.so  main.o  SAT070CP50_1024x600.h  sd20xpanel.h  ss22x_panel.c  ss22x_panel.o  ss268_panel.h  ssplayer
libssplayer.a                main.c          Makefile  sd20xpanel.c          sd20xpanel.o  ss22x_panel.h  ss268_panel.c  ss268_panel.o
xxquan@rt_rdx_server: /work/ssd201/SSD20X-2D06/ido-sbc2d06-sdk/ffmpeg/ffplayer/app$
```

将所需的库和执行文件拷贝至开发板。

ssplayer: ffplayer/app

clock.avi & cuc.flv: /resources/

libssplayer.so: ffplayer/app/

ffmpeg-4.1.3/host/dynamic/下的所有动态库，拷贝到开发板/usr/lib目录下

注意：拷贝之前先查看MMA大小，一般MMA大小设置为16M左右及以上，看需求设置
MMA大小设置请参考：http://doc.industio.com/docs/ssd20x-system/page_8

MMA设置结束重新编译更新系统

```
CHIP = i4m
BOARD = 011A
BOARD_NAME = SSC011A-S01A
PRODUCT = nvr
TOOLCHAIN = glibc
TOOLCHAIN_VERSION = 8.2.1
KERNEL_VERSION = 4.9.84
LIBC = libc-2.28
BUSYBOX = busybox-1.20.2-arm-linux-gnueabi-hf-glibc-8.2.1-dynamic
KERNEL_CONFIG = glibc
IMAGE_CONFIG = spinand.ubifs.p2.partition.config
CUSTOMER_OPTIONS = 011a.201_options.mk
CUSTOMER_TAILOR = nvr_i2m_display_glibc_tailor.mk
MMAP = MMAP_I2M_64M.h
MHAL = i2m
MERGE_BOOT = TRUE
BOOTLOGO_FILE = sigmastar1024_600.jpg
BOOTLOGO_ADDR = E_LX_LOGO_RESERVED_FB
DISP_OUT_NAME = CC0702I50R
EXBOOTARGS =
KERNEL_BOOT_ENV = LX_MEM=$(KERNEL_MEMLN) mma_heap=mma_heap_name0,miu=0,sz=0x1000000 mma_memblock_re
move=1 highres=off
TOOLCHAIN_REL = arm-linux-gnueabi-hf-
```

5、播放测试

```
1 # ./ssplayer ./cuc.flv
```

软路由配置

场景一 WiFi工作为AP模式，其他设备通过连接WiFi热点能够上网

执行以下脚本：

```
1 #/etc/route/WifiAP_Eth0Wan.sh
```

默认WiFi热点名称：industio_2d06，密码：12345678，上网数据通过eth0网口。

WiFi热点配置文件为/etc/hostapd.conf:

ssid: 热点名称

wpa_passphrase: 热点密码

场景二 WiFi工作为STA模式，连接其他热点，2个网口通过WiFi上网

执行以下脚本:

```
1 #/etc/route/WifiWan_2EthLan.sh
```

默认连接热点配置文件为/etc/wpa_supplicant.conf:

```
1 network={
2     ssid="TP-LINK_2G_RONNIE"
3     psk="12345678"
4     key_mgmt=WPA-PSK
5     priority=2
6 }
7
```

ssid: 热点名称

psk: 热点密码

默认2个网口的IP分配通过/etc/dhcp/dhcpd.conf配置:

```
1 subnet 192.168.3.0 netmask 255.255.255.0 {
2     range 192.168.3.100 192.168.3.200;
3     option domain-name-servers 114.114.114.114;
4     option domain-name "dhcp";
5     option routers 192.168.3.1;
6 }
```

两个网口分配的IP地址范围为192.168.3.100至192.168.3.200.

场景三 网口0作为WAN口，其他设备连接网口1后可以上网

执行以下脚本：

```
1  #/etc/route/Eth0Wan_Eth1Lan.sh
```

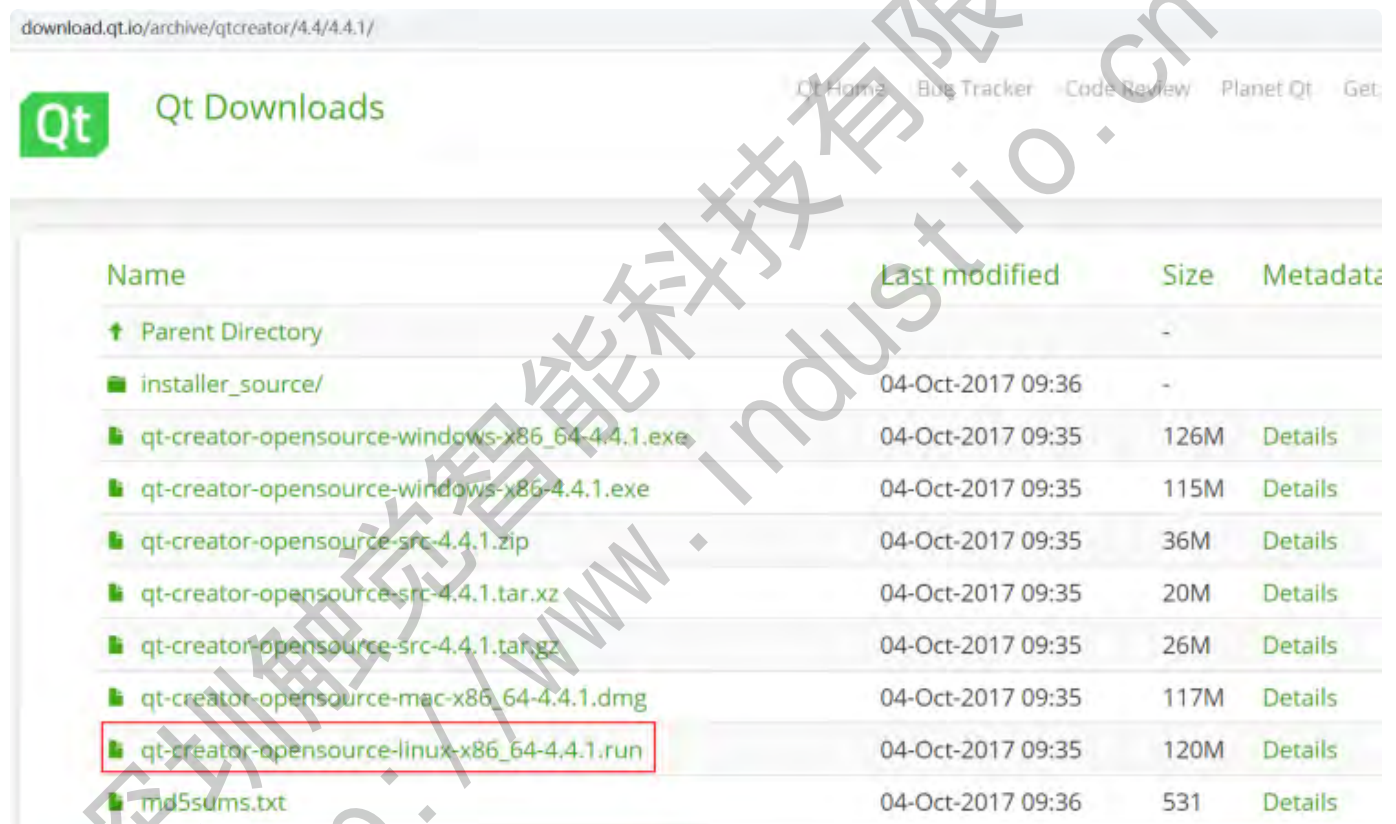
Plain Text

复制代码

Qt Creator开发流程

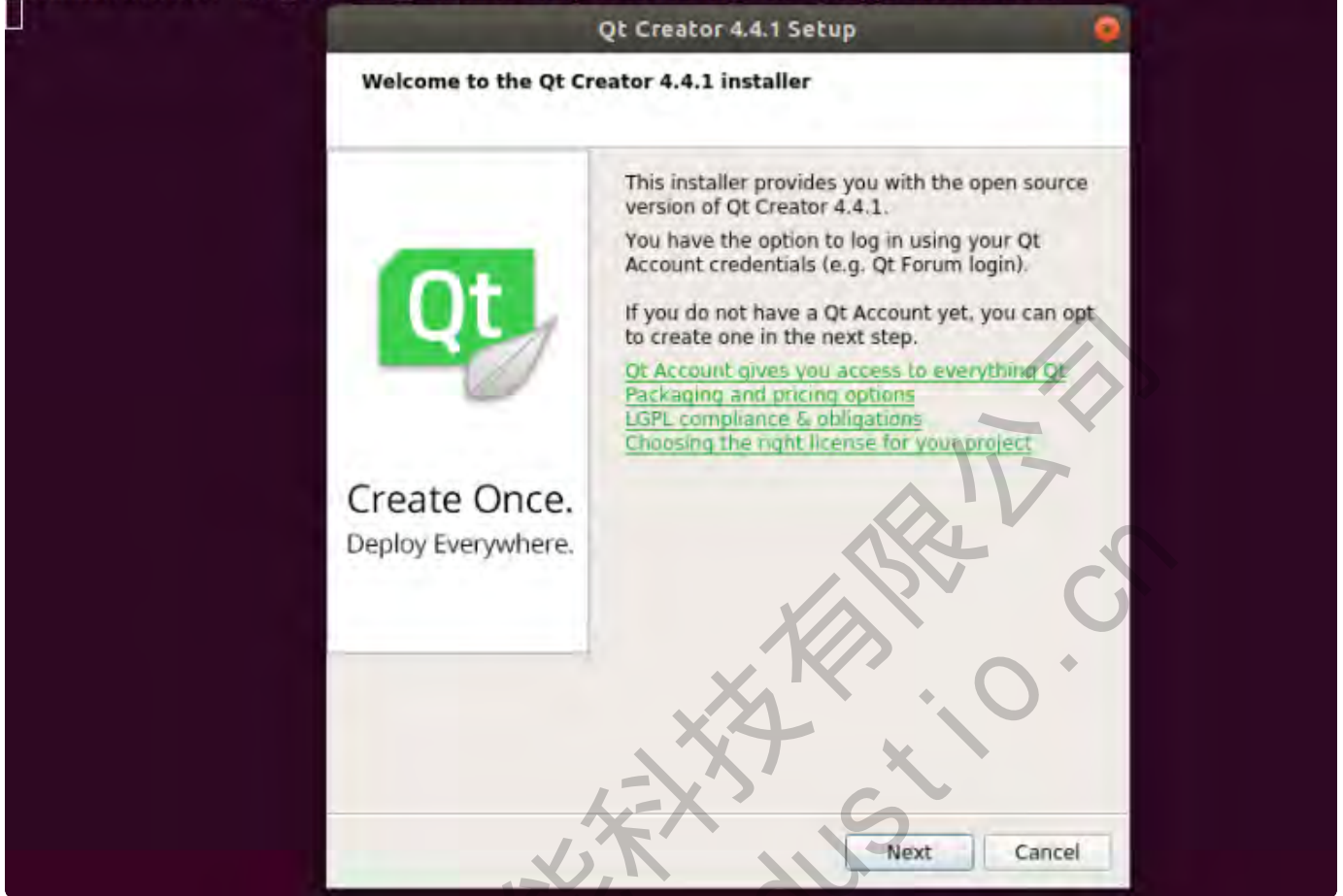
配置Qt开发环境

1、下载Qt Creator4.4.1下载地址：<https://download.qt.io/archive/qtcreator/4.4/4.4.1/>

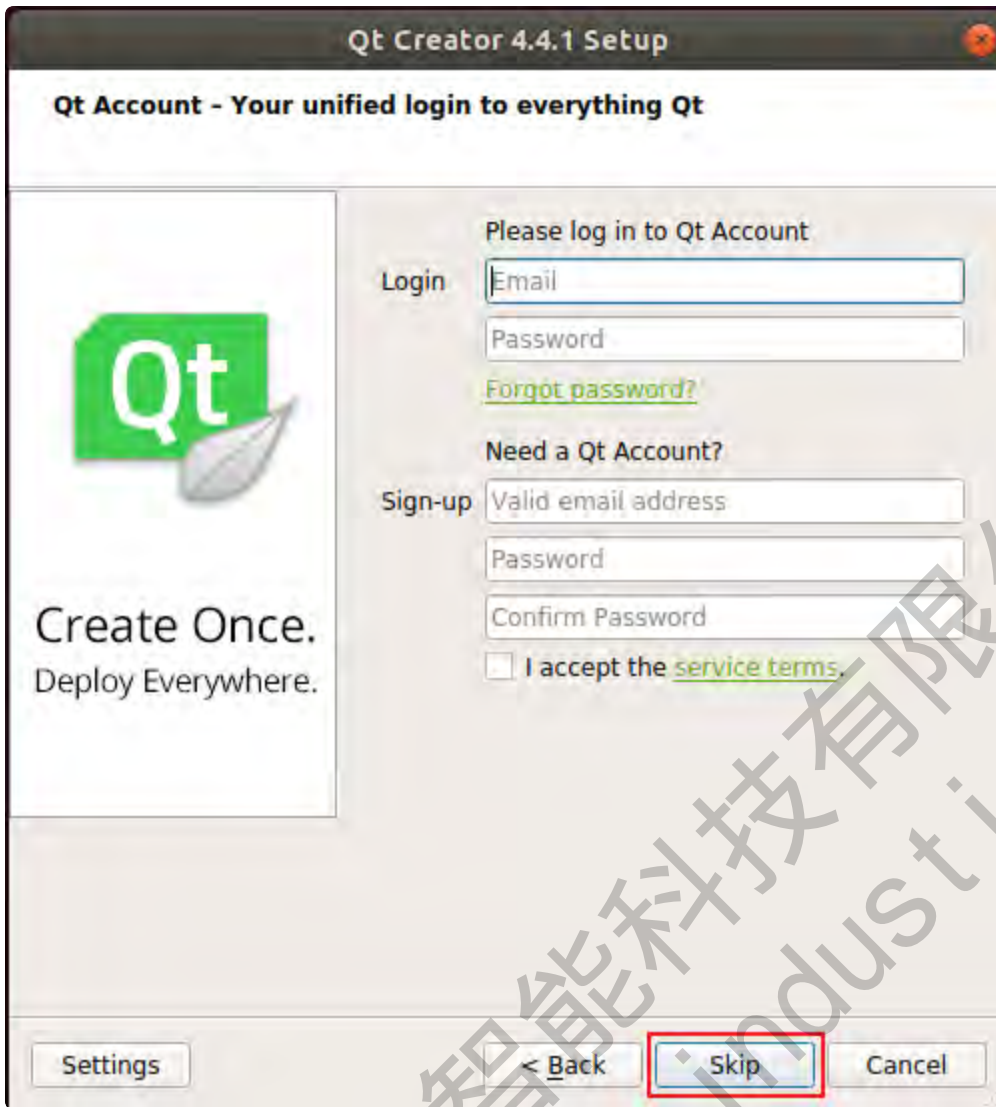


选择下载`qt-creator-opensource-linux-x86_64-4.4.1.run`，并将下载的文件拷贝至Ubuntu虚拟机中。在Ubuntu终端执行`sudo ./qt-creator-opensource-linux-x86_64-4.4.1.run`命令打开安装向导界面,点击“Next”按钮进入下一步。如下图所示

```
fu@fu-VirtualBox:/home/industio_work/Qt$ sudo ./qt-creator-opensource-linux-x86_64-4.4.1.run
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```



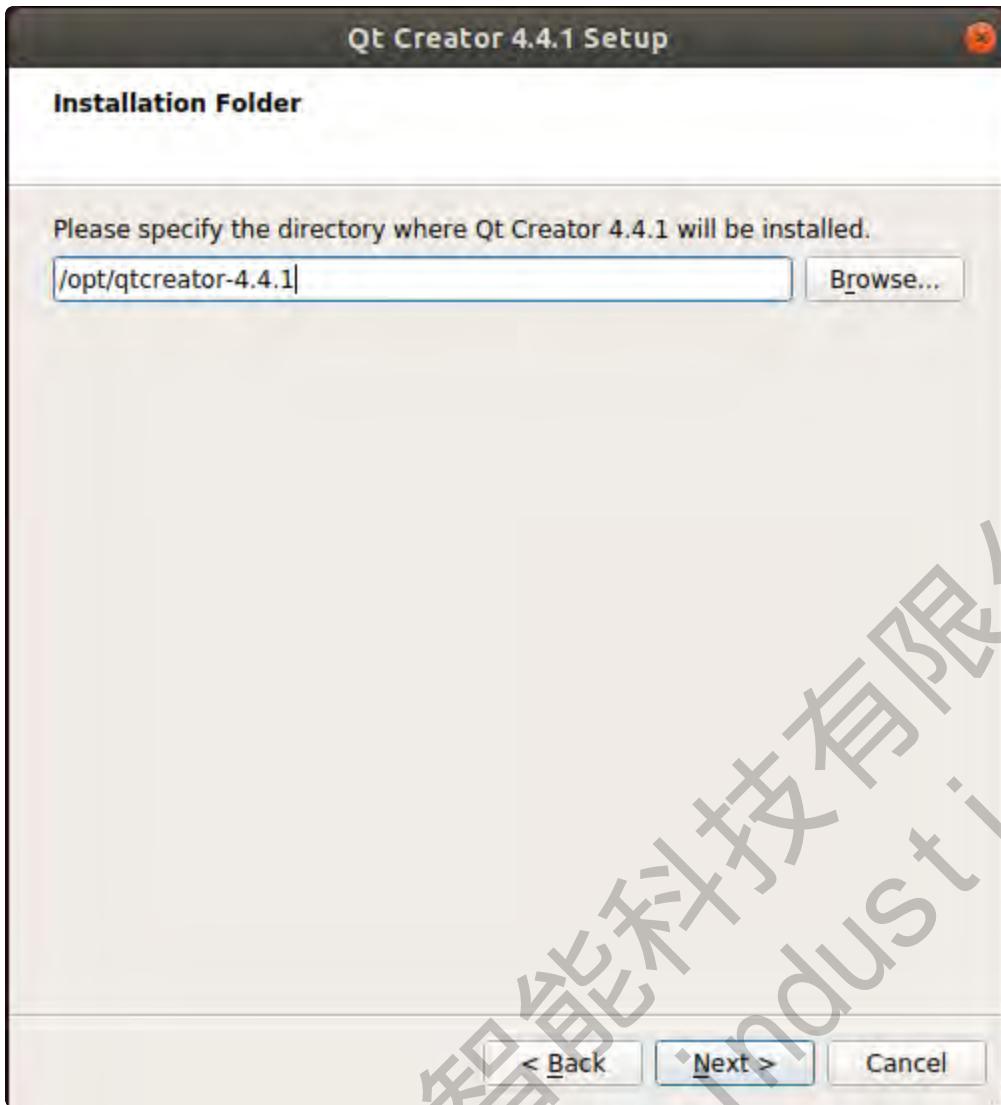
此处如果需要登陆，点击“skip”跳过即可；



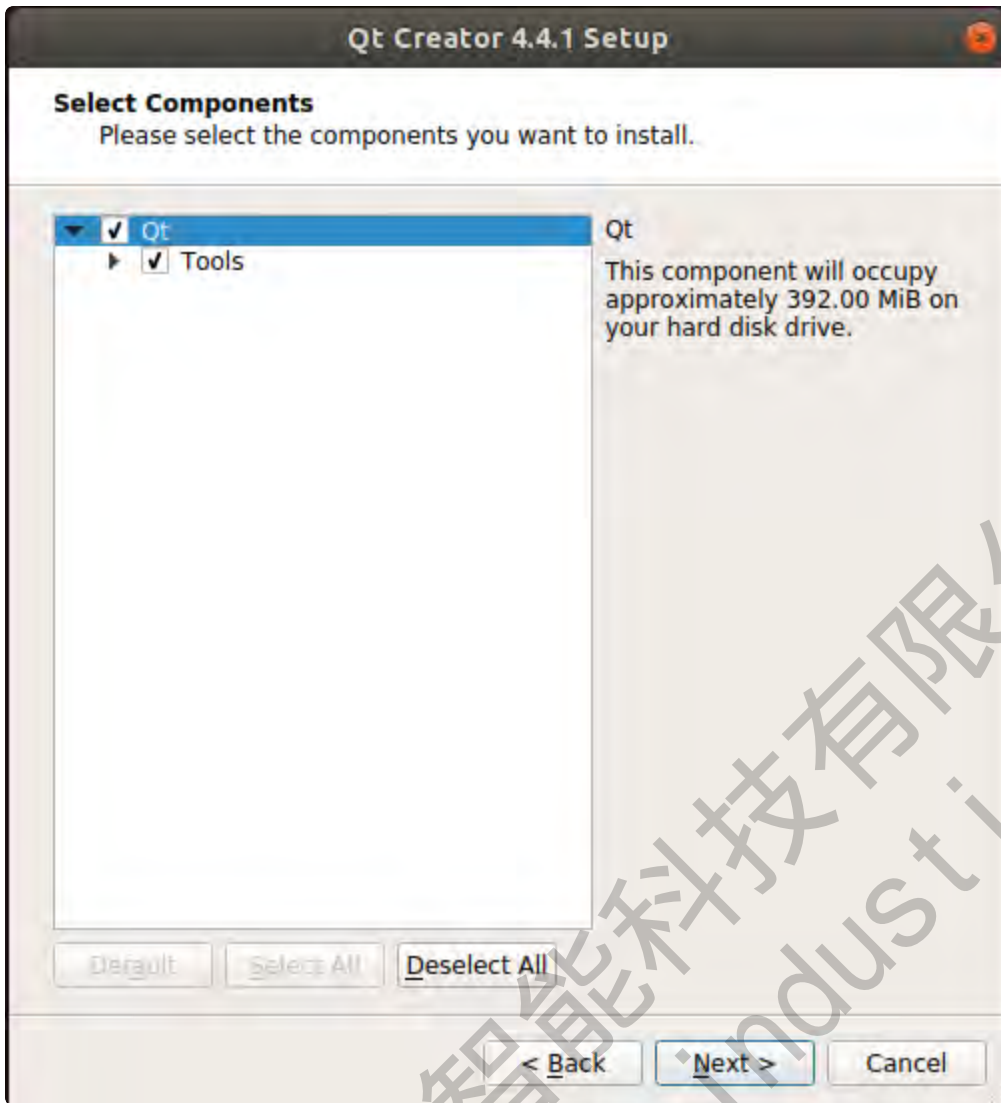
点击“Next”进入下一步；



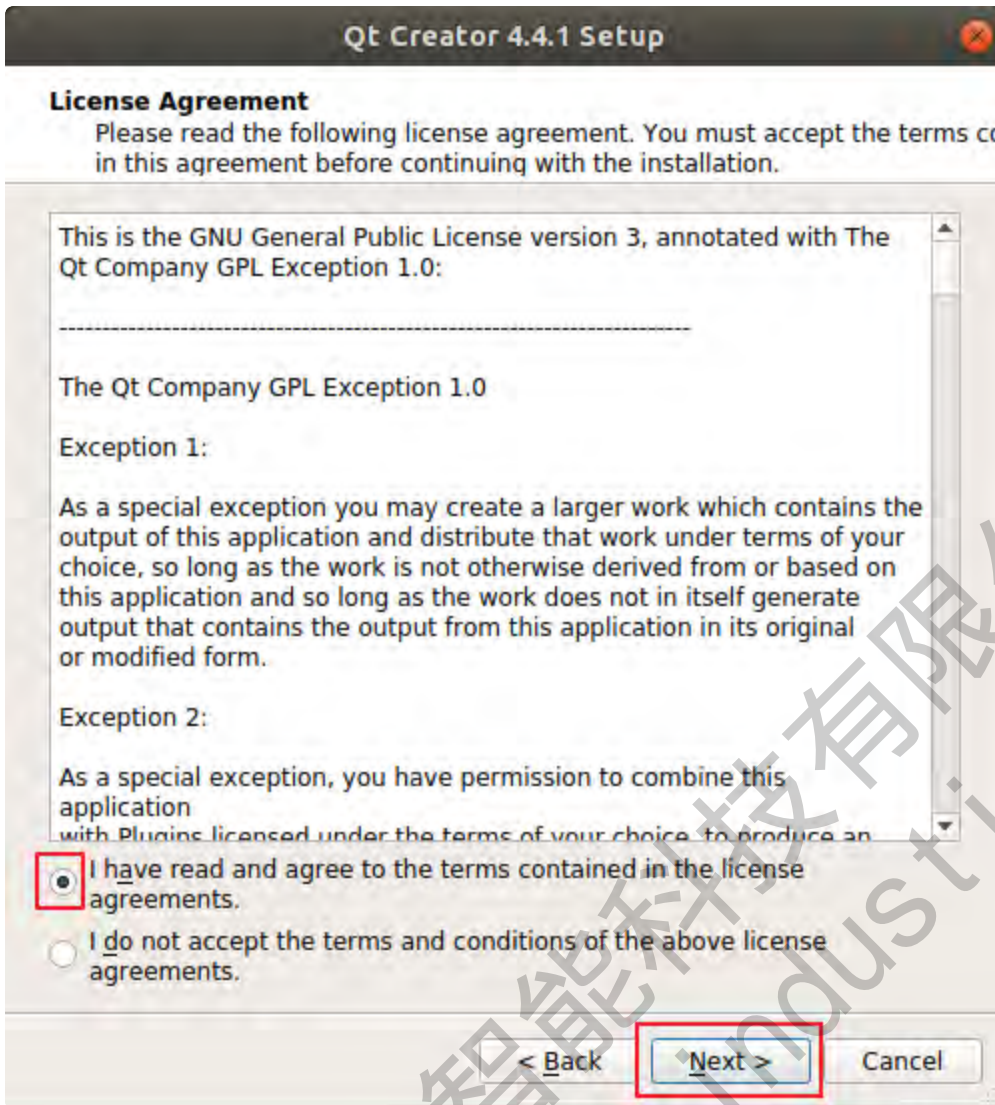
默认安装目录为“/opt/qtcreator-4.4.1”，点击“Next”进入下一步；



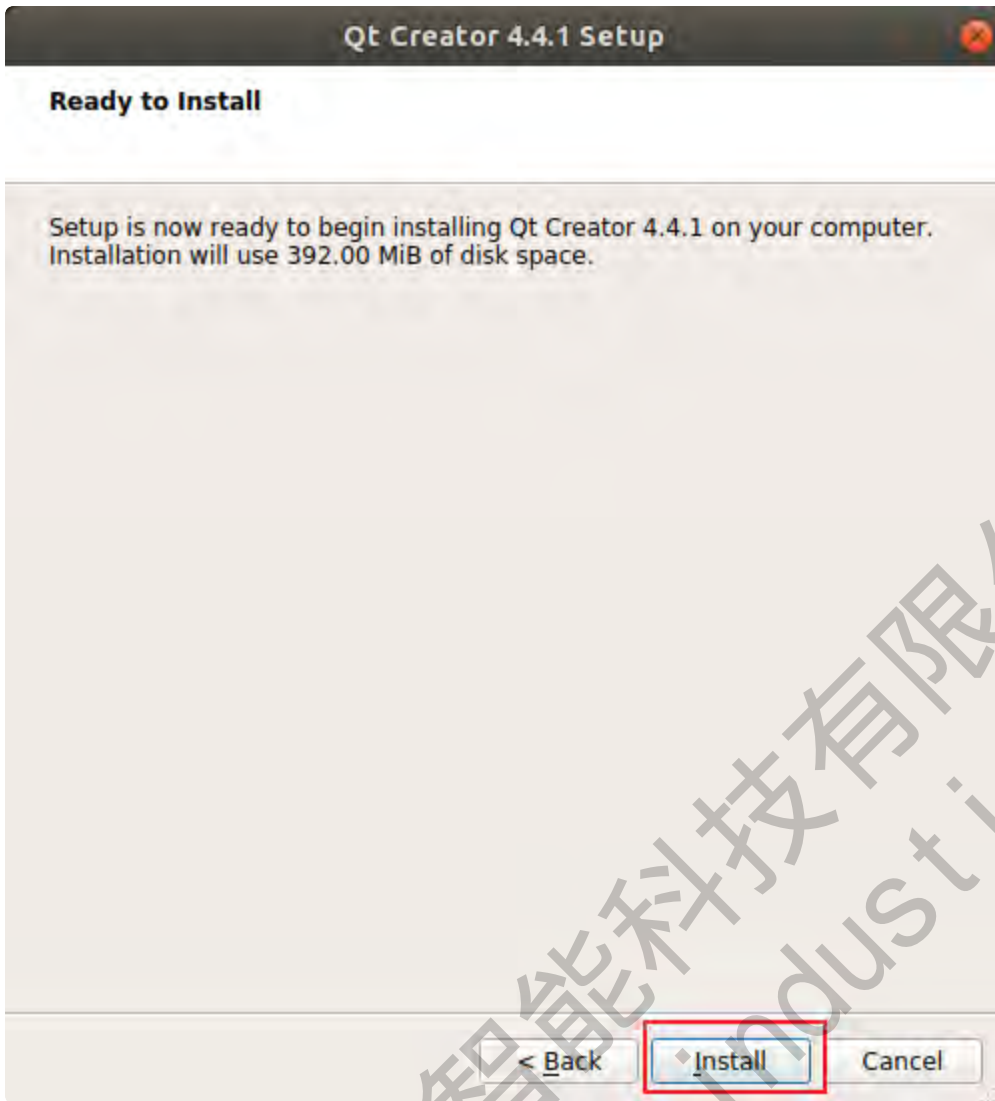
保留默认的勾选安装组件内容，点击“Next”进入下一步；



阅读协议条款内容后，选择同意，点击”Next“进入下一步；



点击”Install“开始安装；



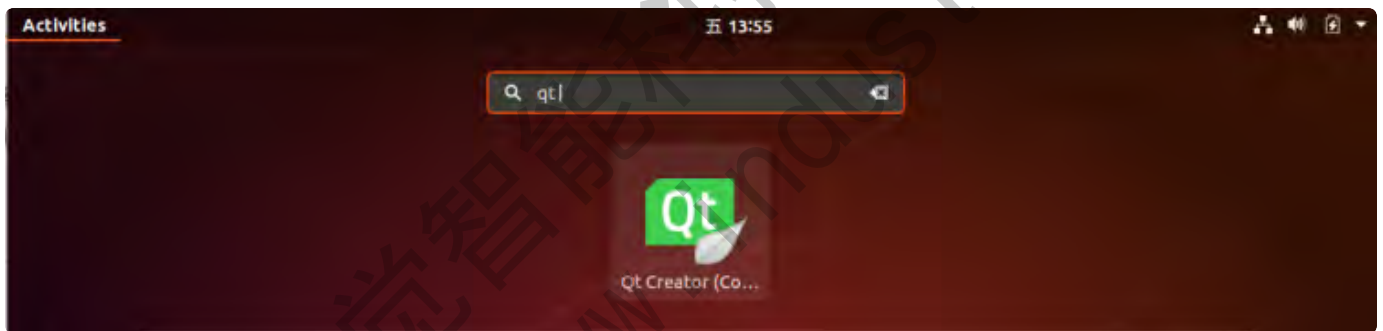
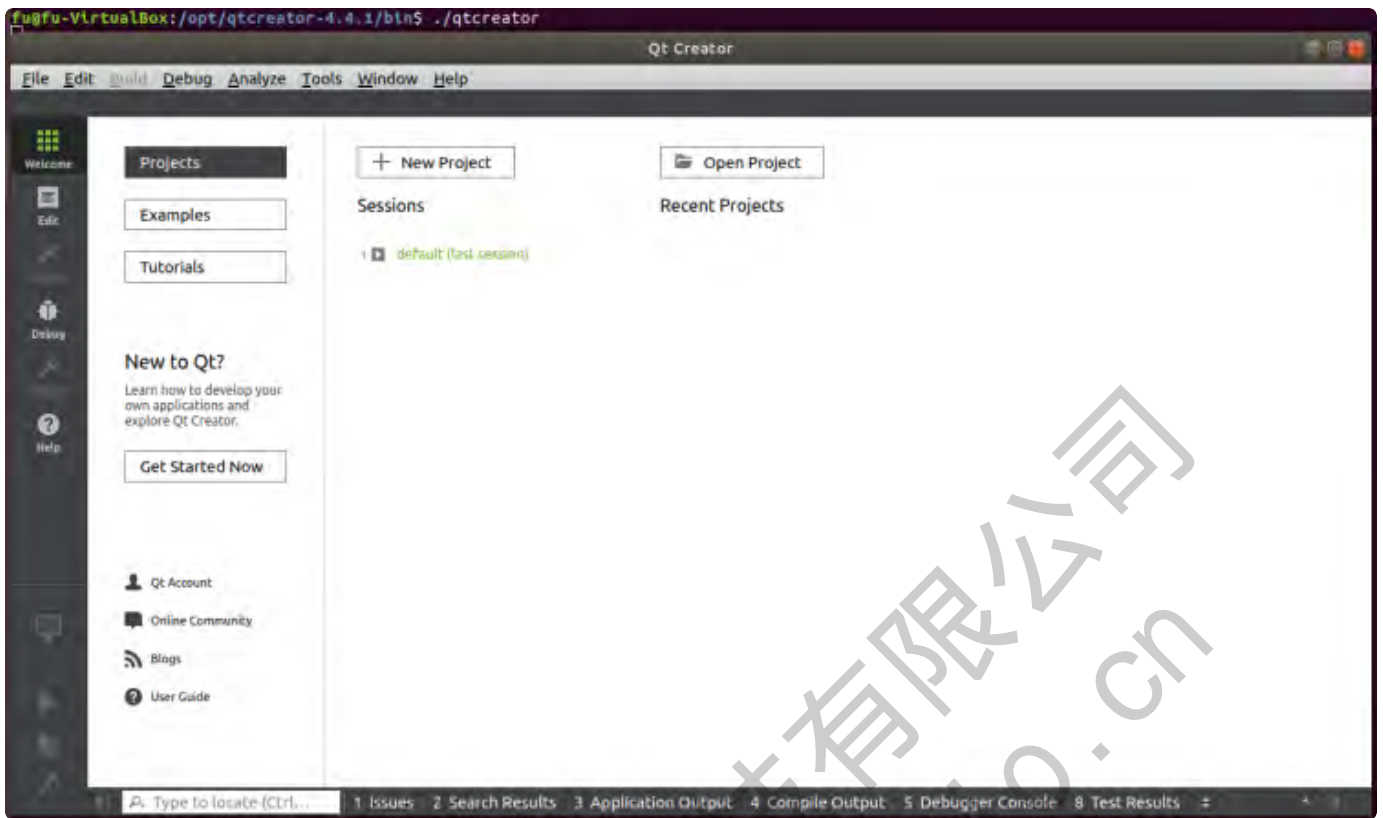
安装进度界面如下图；



安装完成后，即可在安装目录“/opt/qtcreator-4.4.1”的bin目录下找到可执行程序`qtcreator`

```
fu@fu-VirtualBox:/opt/qtcreator-4.4.1/bin$ ls
qbs          qbs-config-ui      qbs-qmltypes      qbs-setup-qt      qt.conf          qtcreator.sh
qbs-config   qbs-create-project qbs-setup-android  qbs-setup-toolchains qtcreator
```

可以在命令行的“/opt/qtcreator-4.4.1/bin”目录下，执行`./qtcreator`开启Qt Creator4.4.1集成开发环境；或者点击Ubuntu18.04界面的左上角“Activities”，在弹出的搜索框中输入“Qt”，在查找结果中找到Qt Creator的图标，双击图标开启Qt Creator程序。



安装Qt4.8.7

在Linux平台安装Qt4.8.7需要从官网下载源代码，通过编译源码获得可用的Qt4.8.7 sdk。

官方源码下载地址：<https://download.qt.io/archive/qt/4.8/4.8.7/>



Name	Last modified	Size	Metadata
↑ Parent Directory	-	-	-
qt-opensource-windows-x86-vs2010-4.8.7.exe	04-Jun-2018 17:26	236M	Details
qt-opensource-windows-x86-vs2008-4.8.7.exe	04-Jun-2018 17:27	235M	Details
qt-opensource-windows-x86-mingw482-4.8.7.exe	04-Jun-2018 17:27	329M	Details
qt-opensource-mac-4.8.7.dmg	04-Jun-2018 17:27	185M	Details
qt-opensource-mac-4.8.7-debug-libs.dmg	04-Jun-2018 17:27	476M	Details
qt-everywhere-opensource-src-4.8.7.zip	04-Jun-2018 17:26	268M	Details
qt-everywhere-opensource-src-4.8.7.tar.gz	04-Jun-2018 17:27	230M	Details
md5sums-4.8.7	04-Jun-2018 17:26	517	Details
changes-4.8.7	04-Jun-2018 17:27	12K	Details

选择下载“qt-everywhere-opensource-src-4.8.7.tar.gz”，将下载的文件拷贝至Ubuntu虚拟机中。

```
fu@fu-VirtualBox:/home/industio_work/Qt$ ls
qt-creator-opensource-linux-x86_64-4.4.1.run qt-everywhere-opensource-src-4.8.7.tar.gz
```

在终端执行下方命令，解压源码，并切换至解压后的源码目录。

```
1  industio@industio$:tar zxvf qt-everywhere-opensource-src-4.8.7.tar.gz
2  industio@industio$:cd qt-everywhere-opensource-src-4.8.7
```

```
fu@fu-VirtualBox:/home/industio_work/Qt/qt-everywhere-opensource-src-4.8.7$ ls
bin                configure.exe      include           LICENSE.GPL3      projects.pro      tools
changes-4.8.7      demos            INSTALL          LICENSE.LGPL       qmake            translations
config.profiles    doc              LGPL_EXCEPTION.txt LICENSE.LGPLv21    README           util
config.tests       examples         lib              LICENSE.LGPLv3     src
configure          imports          LICENSE.FDL      mkspecs           templates
```

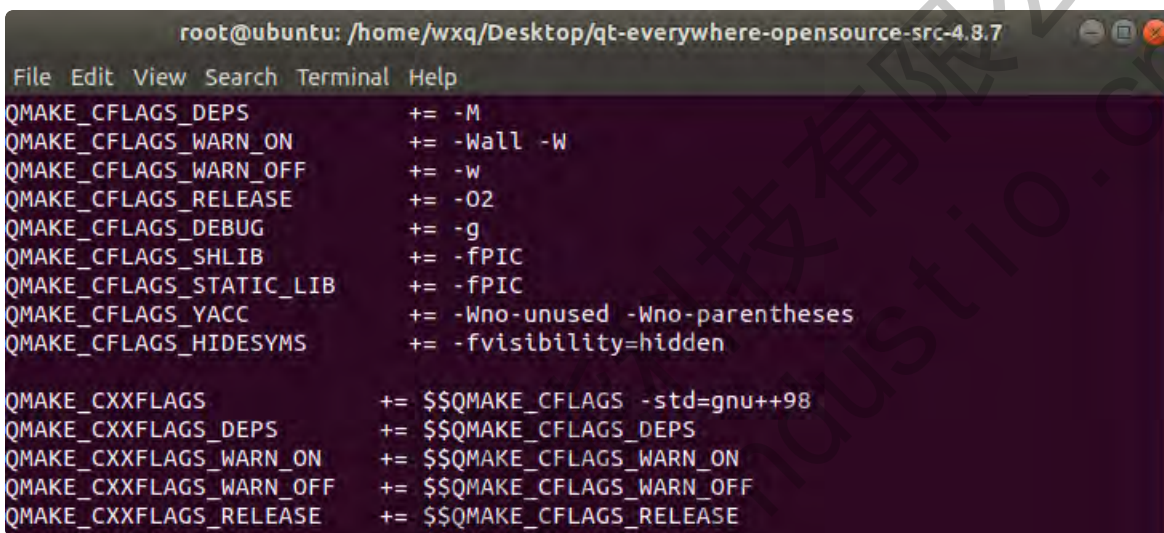
安装编译Qt需要的库

```

1  industio@industio$:sudo apt-get install gcc g++ g++-multilib make automake
2  industio@industio$:sudo apt-get install zlib1g-dev lib32ncurses5 lib32z1 li
    bpng-dev autoconf libtool
3  industio@industio$:sudo apt-get install libxext-dev libx11-dev libxext-dev
    libxtst-dev
4  industio@industio$:sudo apt install libgstreamer1.0-dev libgstreamer1-perl
    libgstreamer-plugins-base0.10-dev libgstreamer1.0-dev
5  industio@industio$:sudo apt install freeglut3-dev mesa-utils

```

修改源码gcc/g++编译配置，指定标准为“gnu++98”，修改源码“修改源码“mkspecs/common/gcc-
base.conf”文件中的“QMAKE_CXXFLAGS”参数，在参数后面添加-std=gnu++98



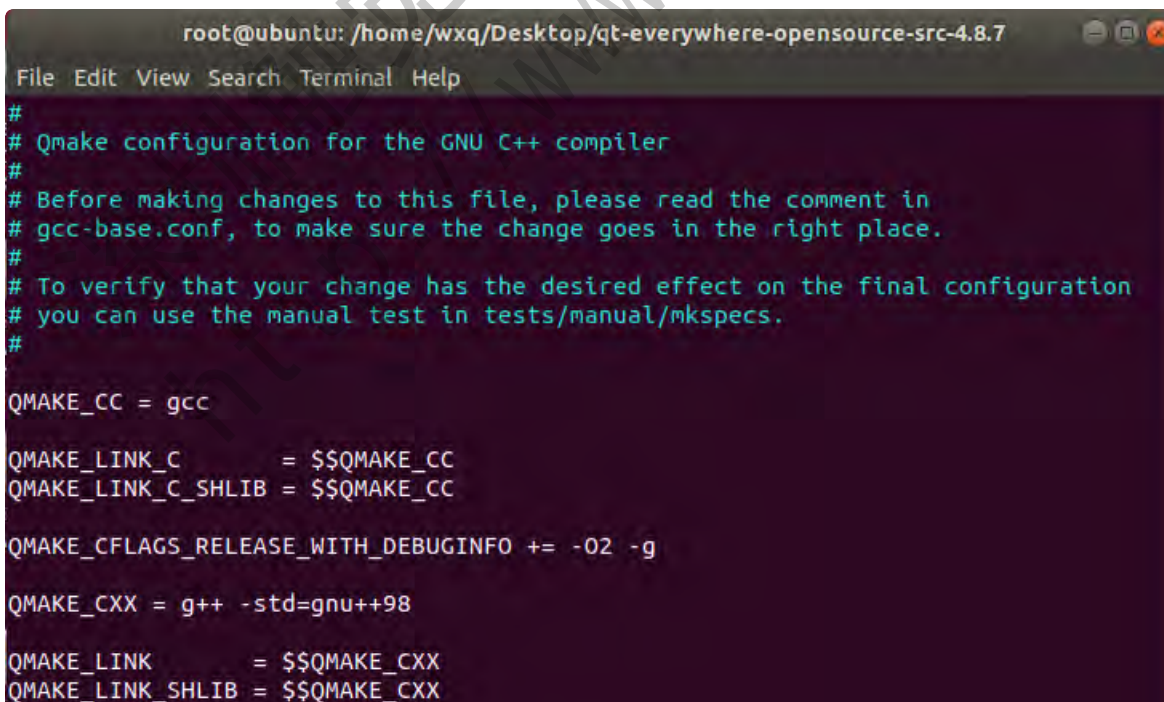
```

root@ubuntu: /home/wxq/Desktop/qt-everywhere-opensource-src-4.8.7
File Edit View Search Terminal Help
QMAKE_CFLAGS_DEPS      += -M
QMAKE_CFLAGS_WARN_ON   += -Wall -W
QMAKE_CFLAGS_WARN_OFF  += -w
QMAKE_CFLAGS_RELEASE   += -O2
QMAKE_CFLAGS_DEBUG     += -g
QMAKE_CFLAGS_SHLIB     += -fPIC
QMAKE_CFLAGS_STATIC_LIB += -fPIC
QMAKE_CFLAGS_YACC       += -Wno-unused -Wno-parentheses
QMAKE_CFLAGS_HIDESYMS  += -fvisibility=hidden

QMAKE_CXXFLAGS         += $$QMAKE_CFLAGS -std=gnu++98
QMAKE_CXXFLAGS_DEPS    += $$QMAKE_CFLAGS_DEPS
QMAKE_CXXFLAGS_WARN_ON += $$QMAKE_CFLAGS_WARN_ON
QMAKE_CXXFLAGS_WARN_OFF += $$QMAKE_CFLAGS_WARN_OFF
QMAKE_CXXFLAGS_RELEASE += $$QMAKE_CFLAGS_RELEASE

```

修改源码“mkspecs/common/g++-base.conf”文件中的“QMAKE_CXX”参数，在参数后面添加-std=gnu++98



```

root@ubuntu: /home/wxq/Desktop/qt-everywhere-opensource-src-4.8.7
File Edit View Search Terminal Help
#
# Qmake configuration for the GNU C++ compiler
#
# Before making changes to this file, please read the comment in
# gcc-base.conf, to make sure the change goes in the right place.
#
# To verify that your change has the desired effect on the final configuration
# you can use the manual test in tests/manual/mkspecs.
#
QMAKE_CC = gcc

QMAKE_LINK_C      = $$QMAKE_CC
QMAKE_LINK_C_SHLIB = $$QMAKE_CC

QMAKE_CFLAGS_RELEASE_WITH_DEBUGINFO += -O2 -g

QMAKE_CXX = g++ -std=gnu++98

QMAKE_LINK      = $$QMAKE_CXX
QMAKE_LINK_SHLIB = $$QMAKE_CXX

```

```
1 industio@industio$ ./configure --static --debug-and-release -nomake demos -nomake examples -no-openssl -no-exceptions
```

```
root@ubuntu:/home/wxq/Desktop/qt-everywhere-opensource-src-4.8.7# ./configure --static --debug-and-release -nomake demos -nomake examples -no-openssl -no-exceptions
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?
```

如果没有报错，配置结束后提示内容如下：

执行“make”编译源码，执行“make install”在编译完成后，将Qt4.8.7安装到默认路径“/usr/local/Trolltech/Qt-4.8.7”目录。

```
for /home/industio_work/Qt/qt-everywhere-opensource-src-4.8.7/src/

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-4.8.7

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

编译安装

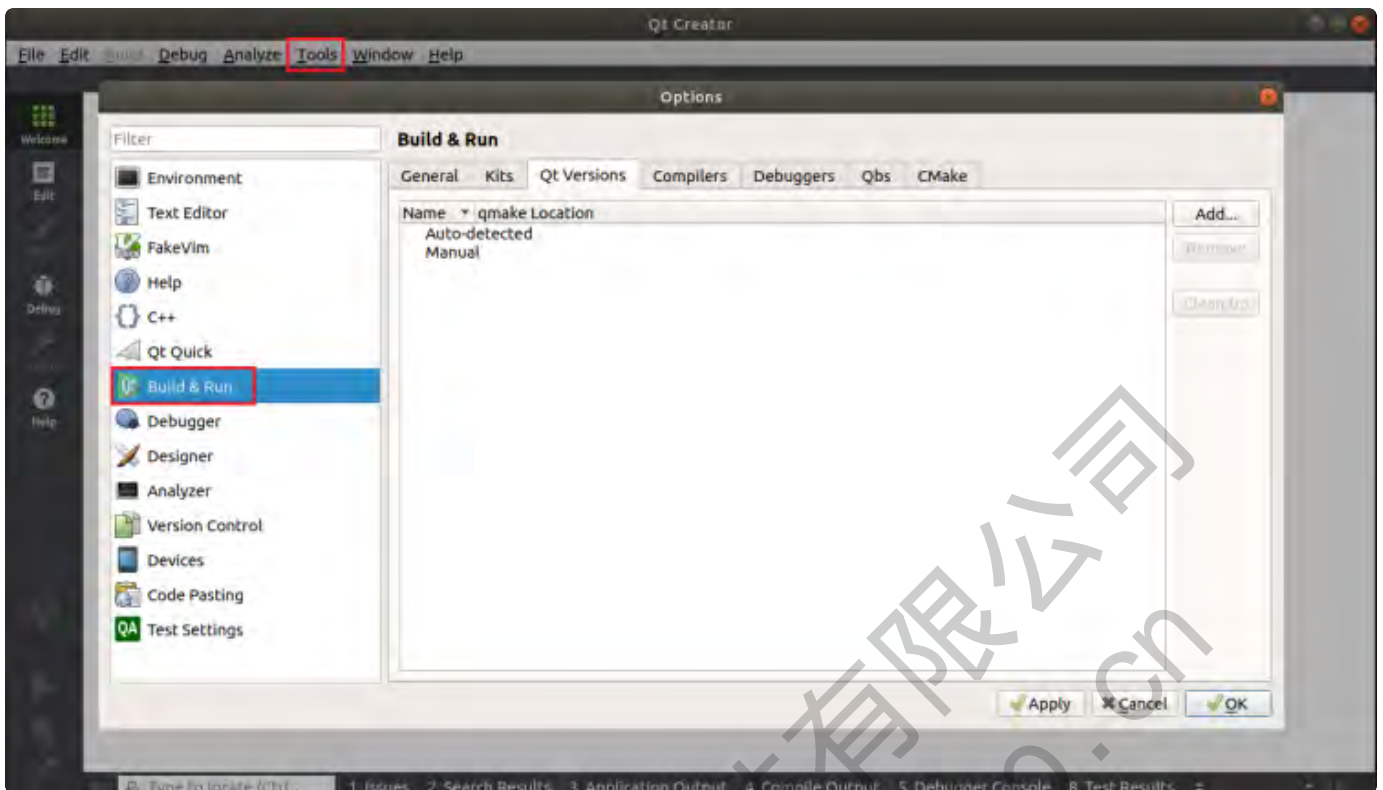
```
1 industio@industio$ make && make install
```

安装完成后，安装的文件位于“/usr/local/Trolltech/Qt-4.8.7”目录，进入sdk的bin目录，执行`./qmake -v`即可查看安装的qmake和Qt的版本。

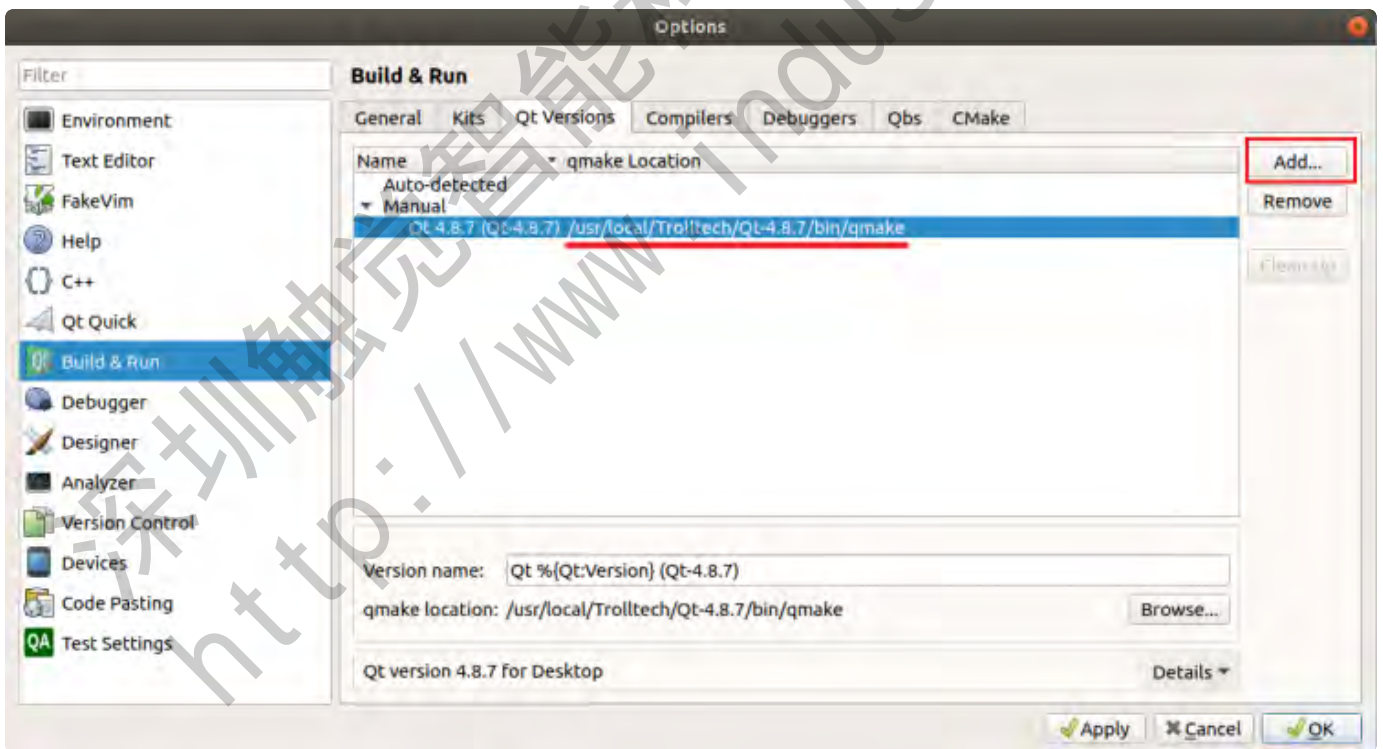
```
root@ubuntu:/usr/local/Trolltech/Qt-4.8.7/bin# ./qmake -v
QMake version 2.01a
Using Qt version 4.8.7 in /usr/local/Trolltech/Qt-4.8.7/lib
root@ubuntu:/usr/local/Trolltech/Qt-4.8.7/bin#
```

Qt Creator添加Qt4.8.7开发套件

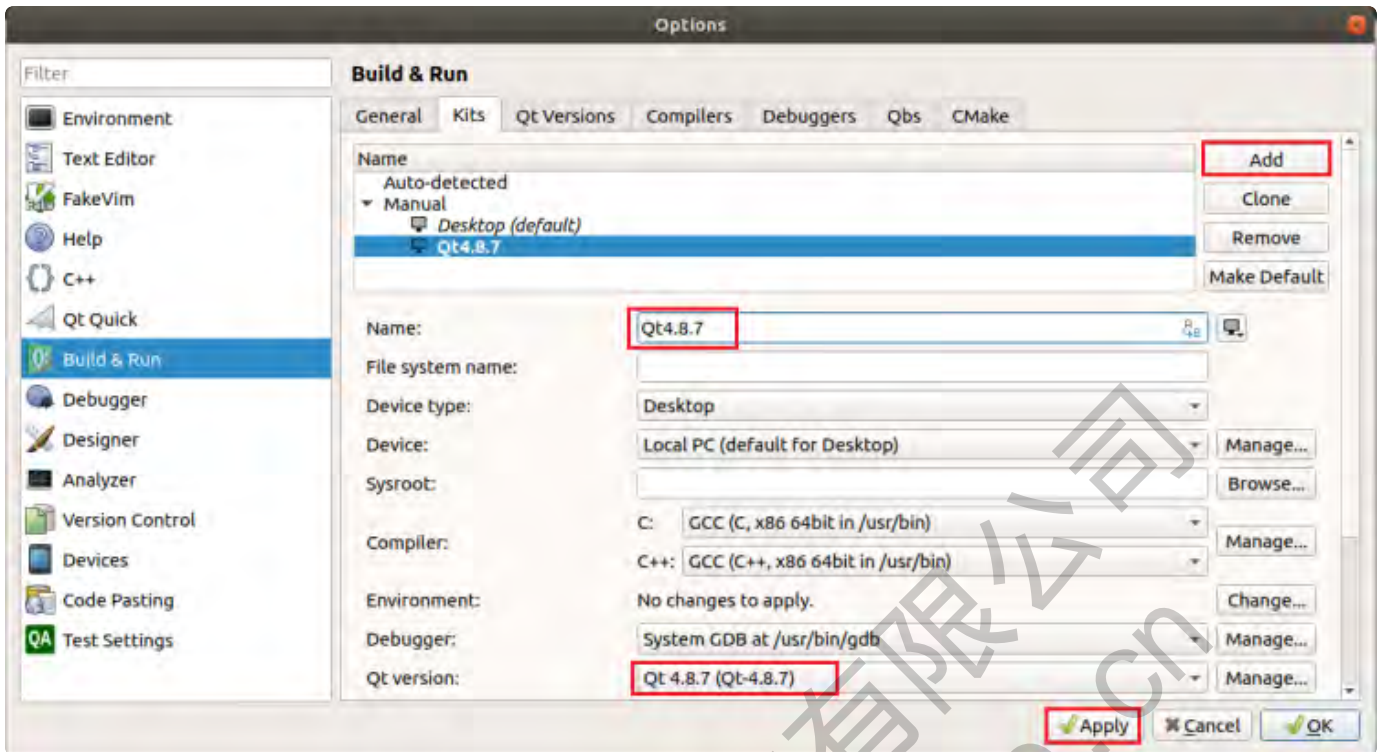
Qt Creator 4.4.1需要添加Qt4.8.7开发套件才可能在Ubuntu18.04 PC端编译、运行和调试Qt代码。打开Qt Creator 4.4.1软件，选择“Tool->Options”，在弹出的界面中选择“Build & Run”。



在”Qt Versions”栏点击右边的”Add..”按钮，进入文件资源浏览界面，选择”/usr/local/Trolltech/Qt-4.8.7/bin/qmake“，点击右下方的”Apply“按钮生效添加的Qt Version。



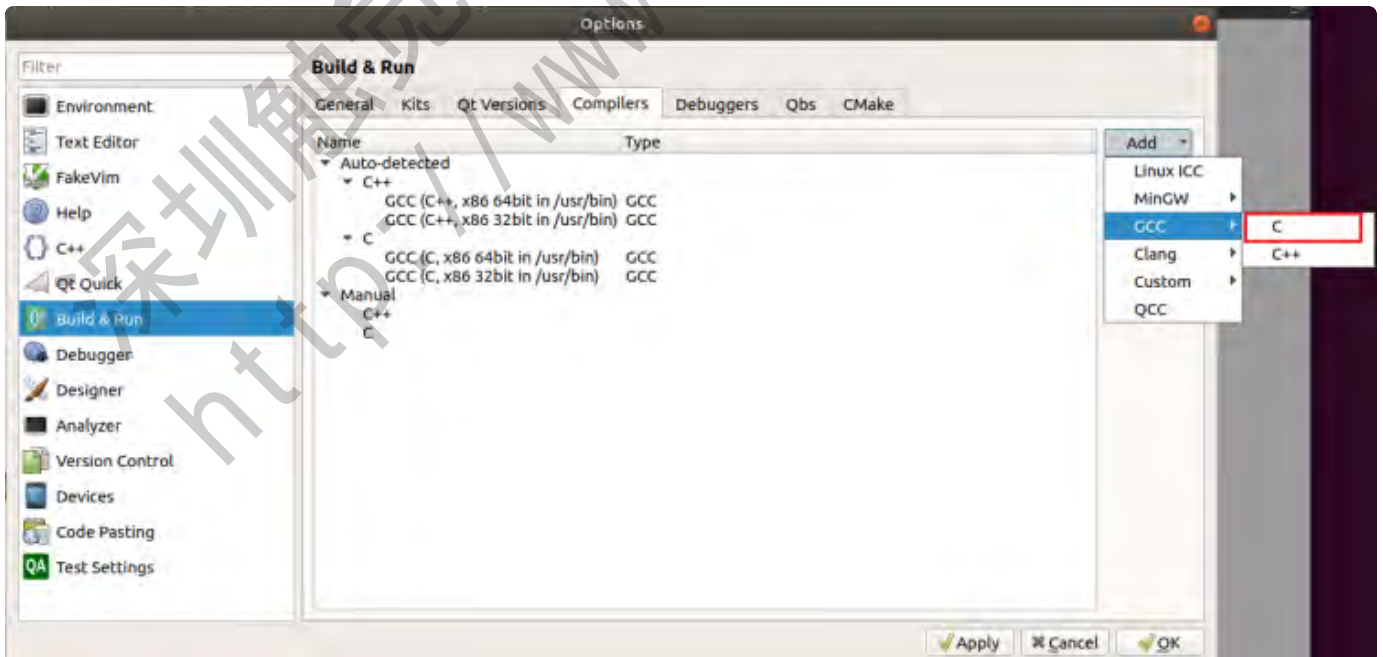
点击进入Kits栏，点击”add“按钮新增一项kit，修改名字为”Qt4.8.7“，Qt Version选择步骤2中添加的Qt-4.8.7，点击”Apply“按钮生效，点击”OK“按钮关闭窗口。



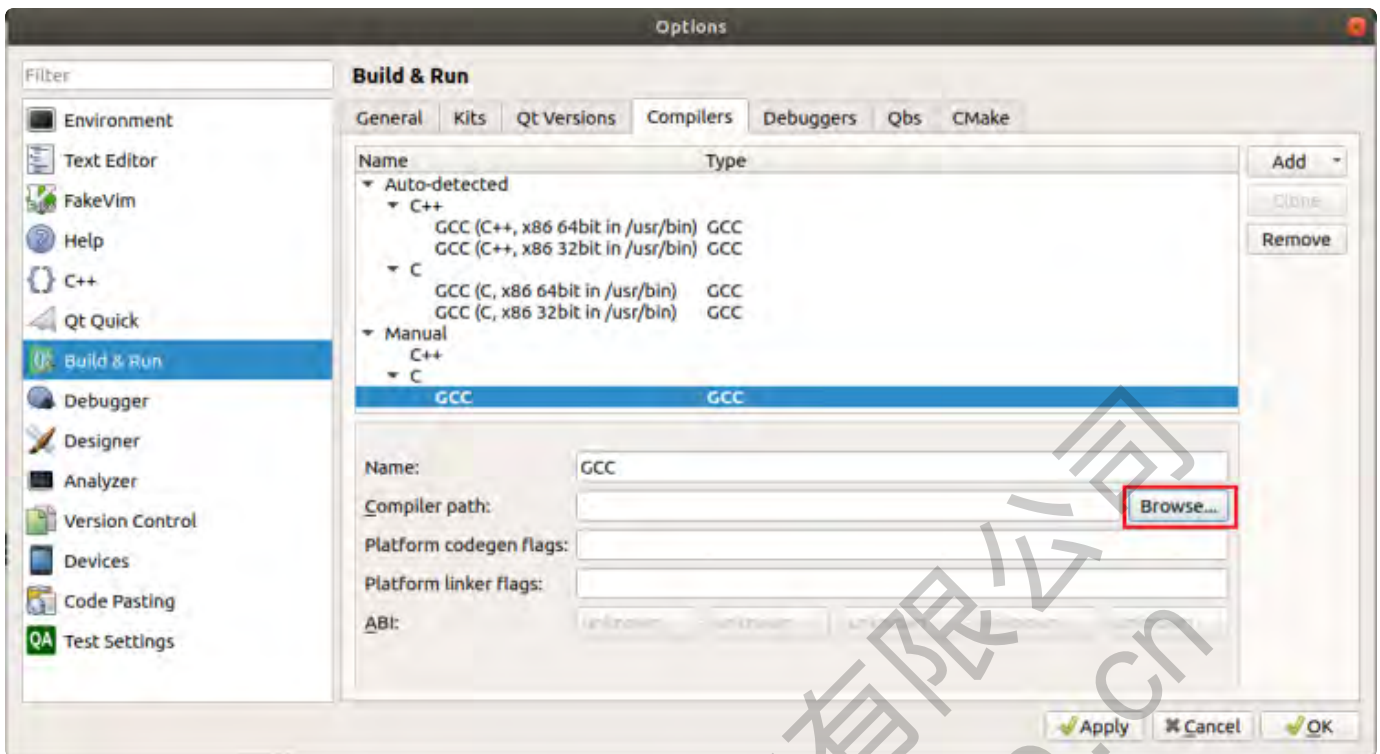
Qt Creator添加Qt4.8.7交叉编译

开发的Qt程序需要经过交叉编译工具链编译后，才可以在SSD20x开发板的Linux ARM平台上运行。QT Creator交叉编译工具链的设置界面位于的“Tools”->“Options”->“Build & Run”，需要在Compilers、Debuggers、QtVersions和Kits四配置中添加对应的文件。

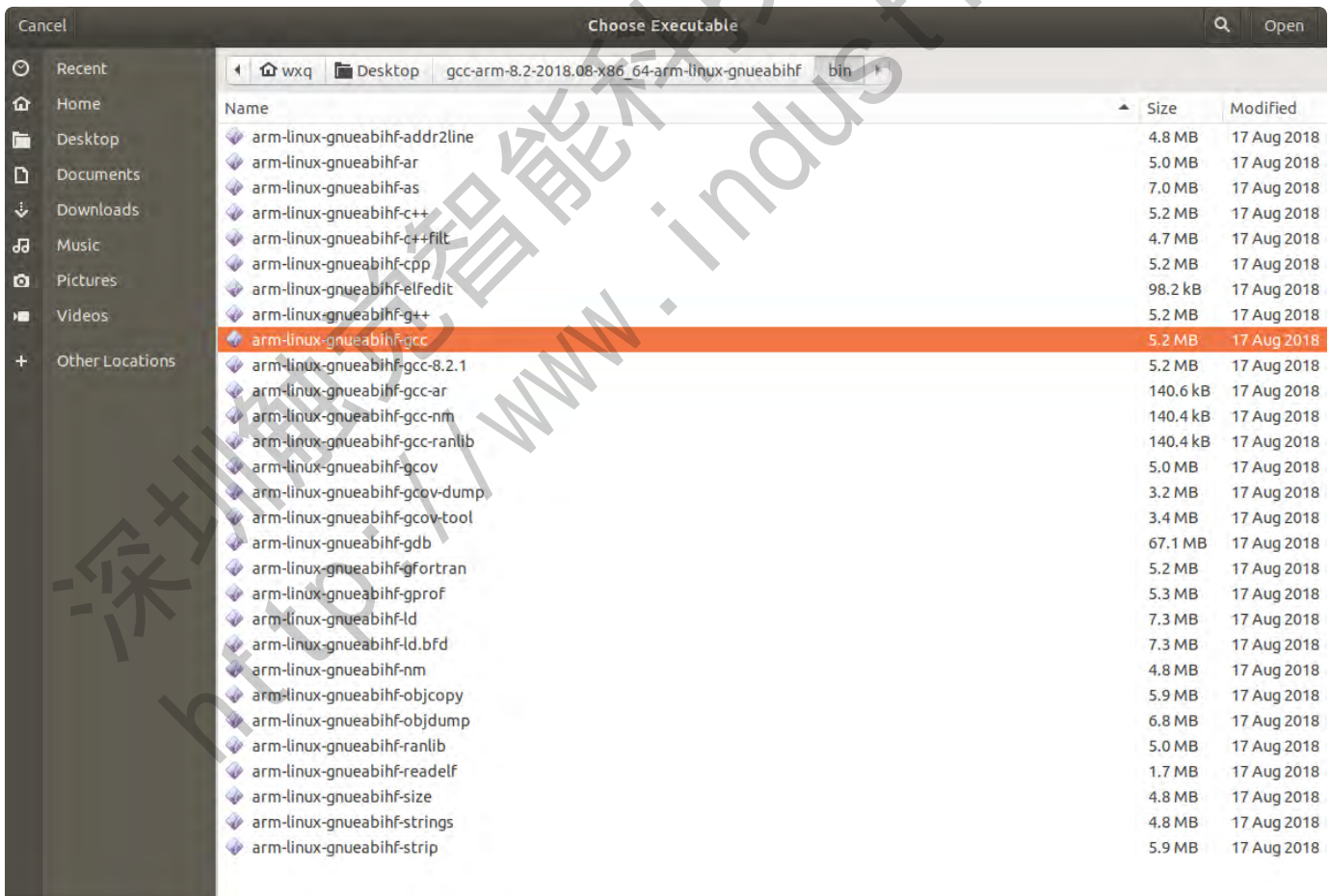
添加gcc在界面的右侧点击Add->GCC->C，添加一个GCC编译器。



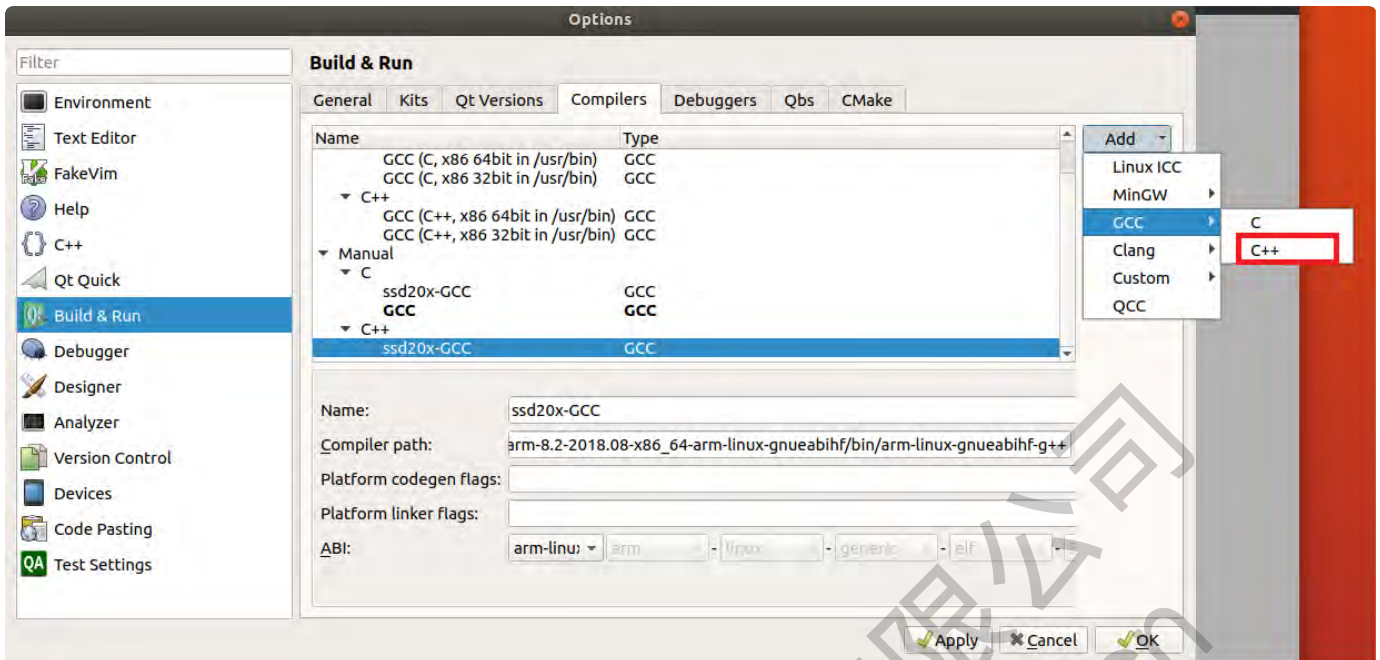
选中新添加的GCC编译器，然后点击下方的“Browse”按钮开启文件资源浏览器；



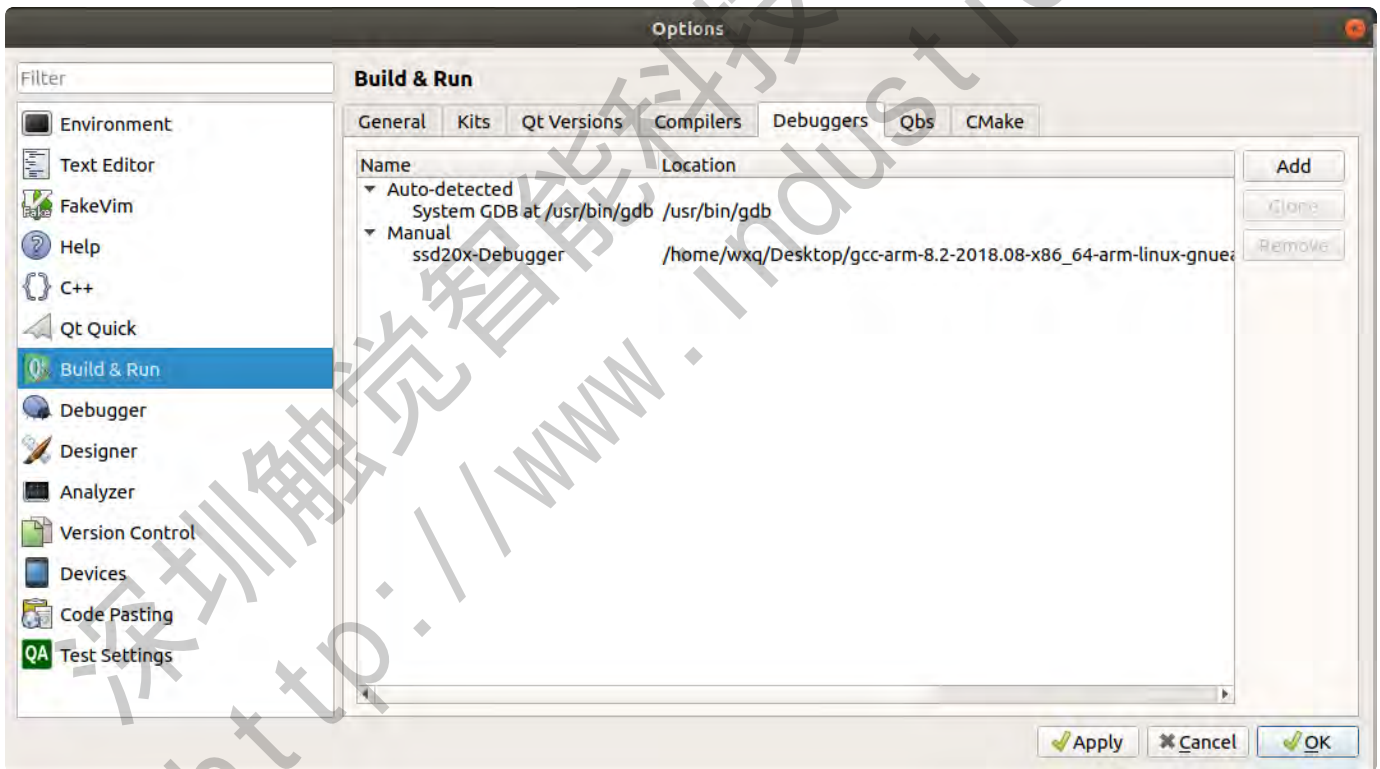
选中gcc编译工具“arm-linux-gnueabi-hf-gcc”，添加完成点击“Apply”生效设置



按照GCC的添加方法，添加g++



在Debuggers栏点击“Add”按钮新增一个Debugger，选择“arm-linux-gnueabi-gdb”并修改名字为“ssd20x-Debugger”，点击“Apply”按钮生效设置。

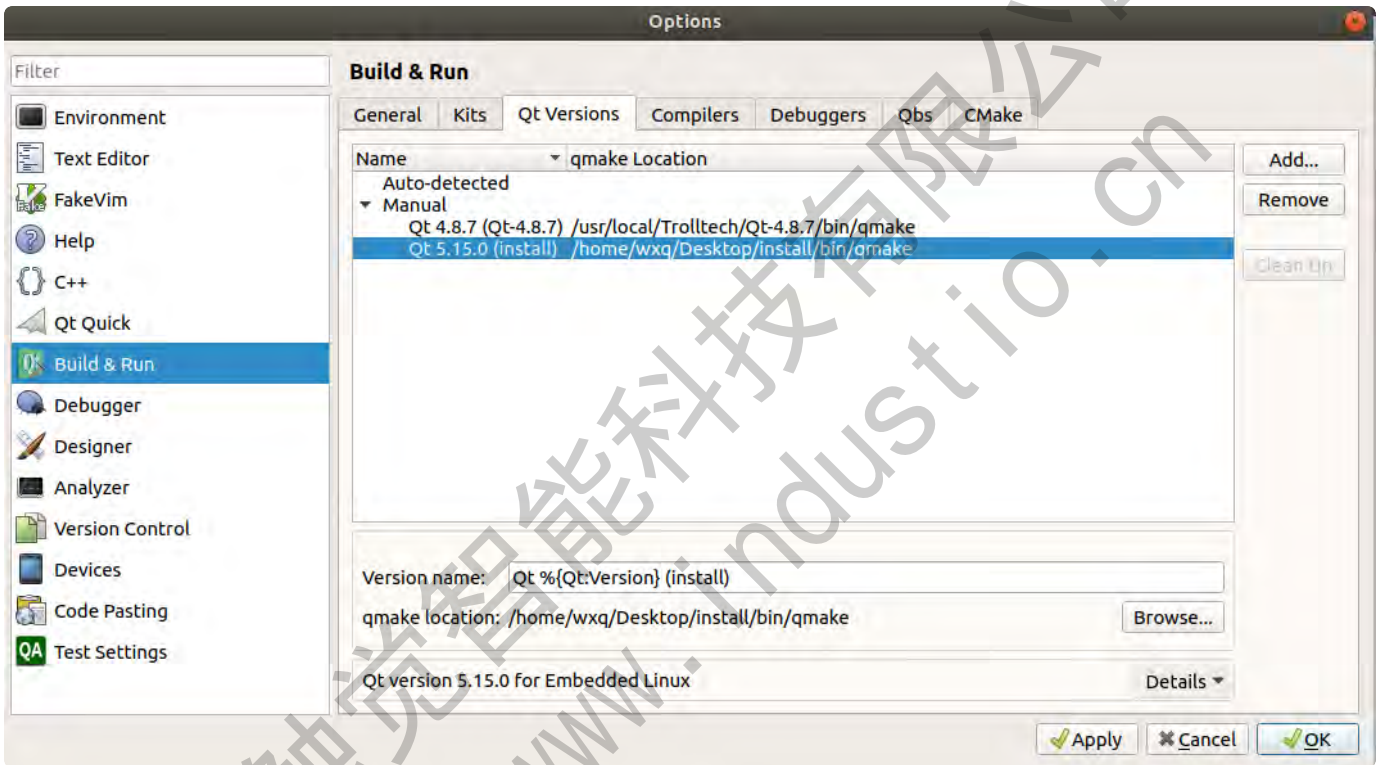


在配置Qt Versions时把网盘中的Qt5.15解压到Ubuntu中

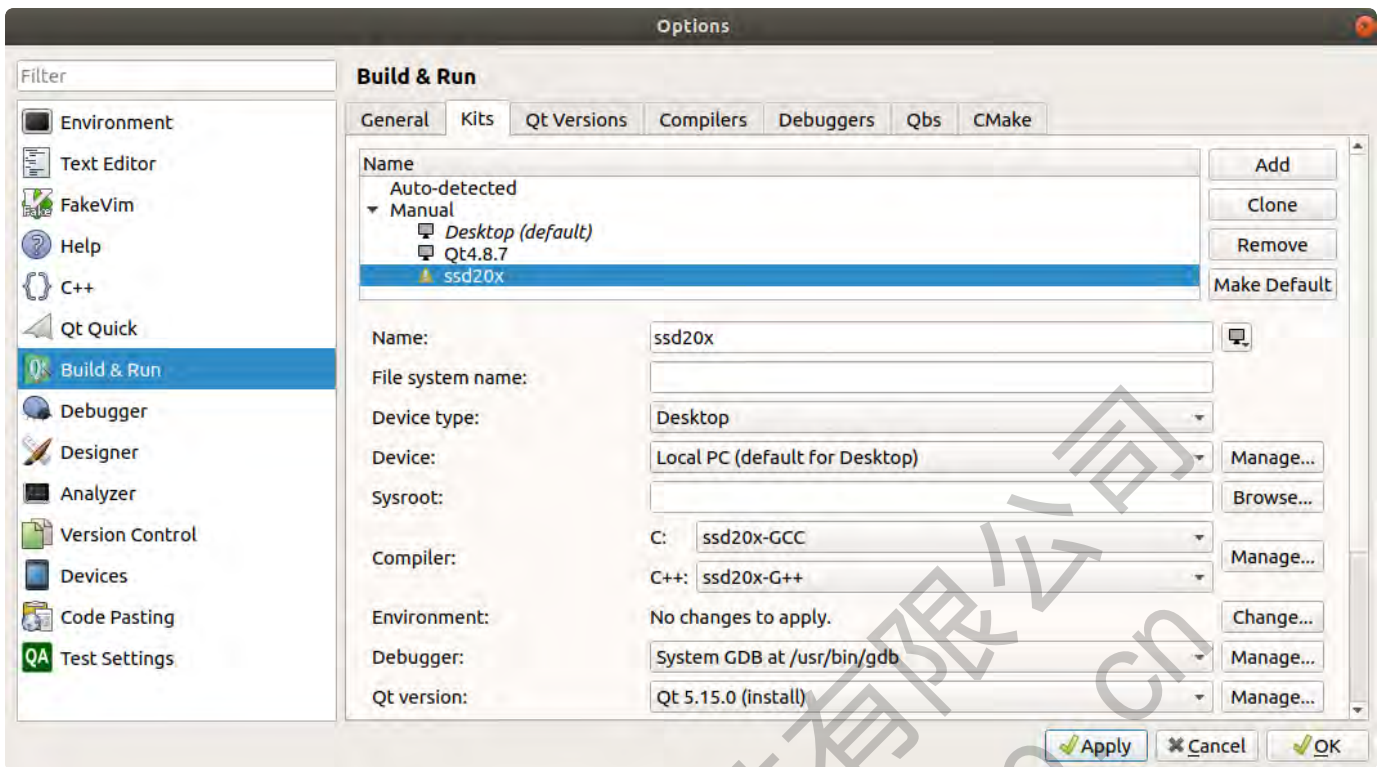
返回上一级 | ... > 开发板文档 > test > qt_5.15... > share

文件名	大小	修改日期
readme.txt	571B	2021-04-26 11:51
qt_share_env.sh	290B	2021-04-26 11:51
qt_install_share_5.15.0.tar.bz2	104.4M	2021-04-26 11:51
qmake.conf	808B	2021-04-26 11:51
make_share.sh	576B	2021-04-26 11:51

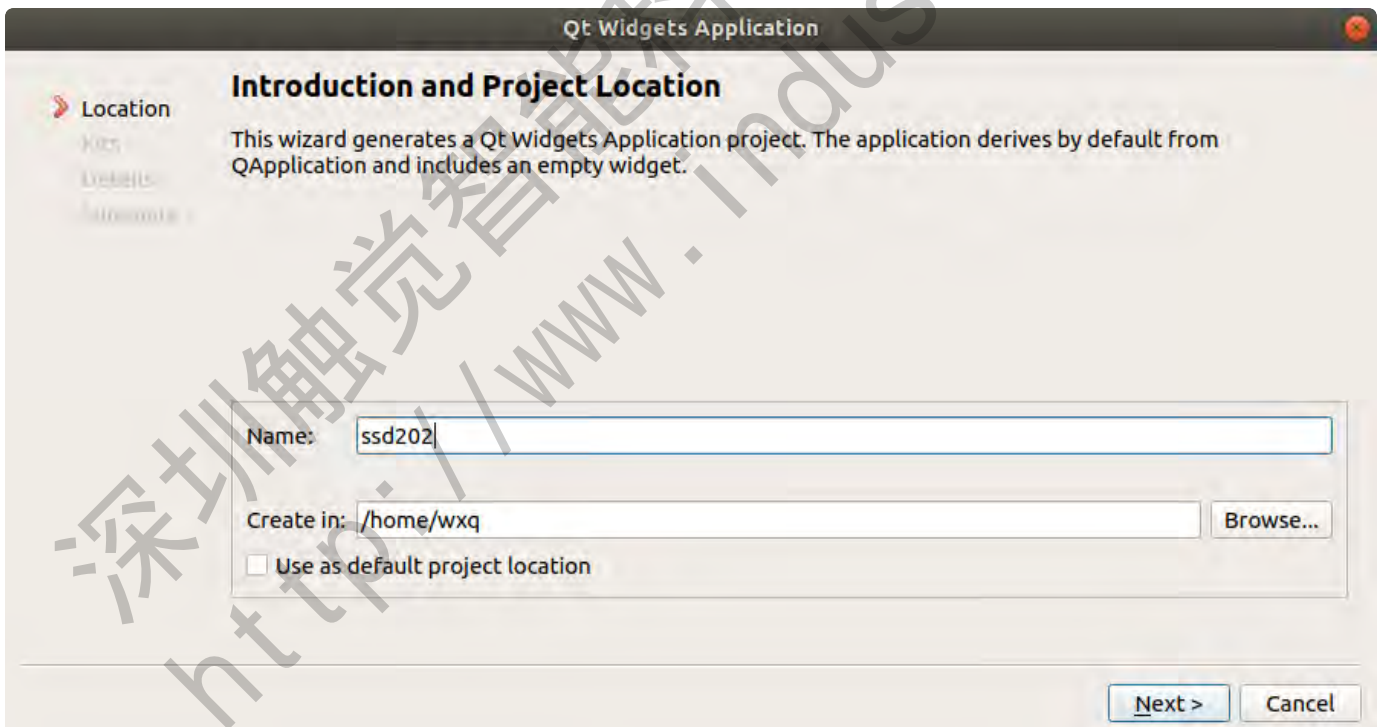
在Qt Versions栏点击“add”打开文件资源浏览器，选择“/install/bin/qmake”，点击“Apply”按钮生效设置。



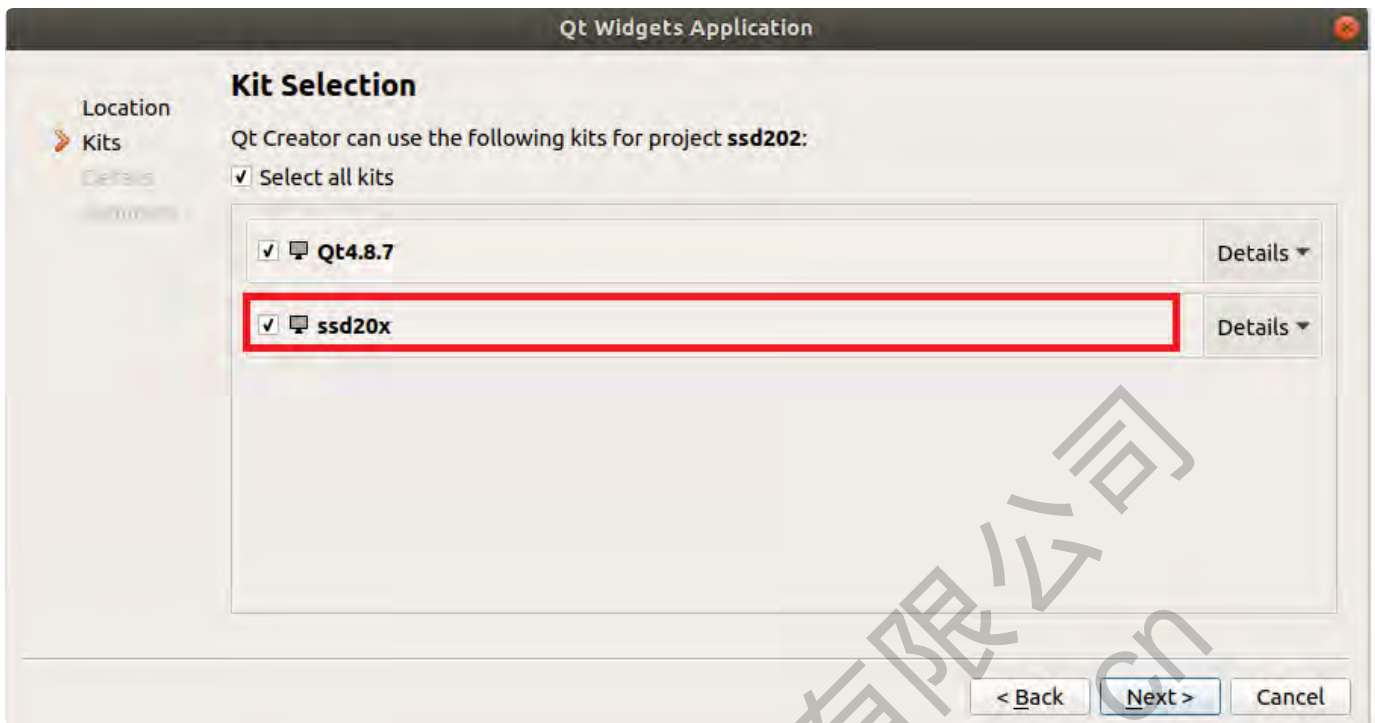
在Kits栏，点击“Add”按钮添加一个kit，并修改名字为“ssd20x”，选择Compiler C: ssd20x-GCC；Compiler C++: ssd20x-G++；Qt Version :“Qt5.15.0 (install) ”，最后点击“Apply”生效配置。



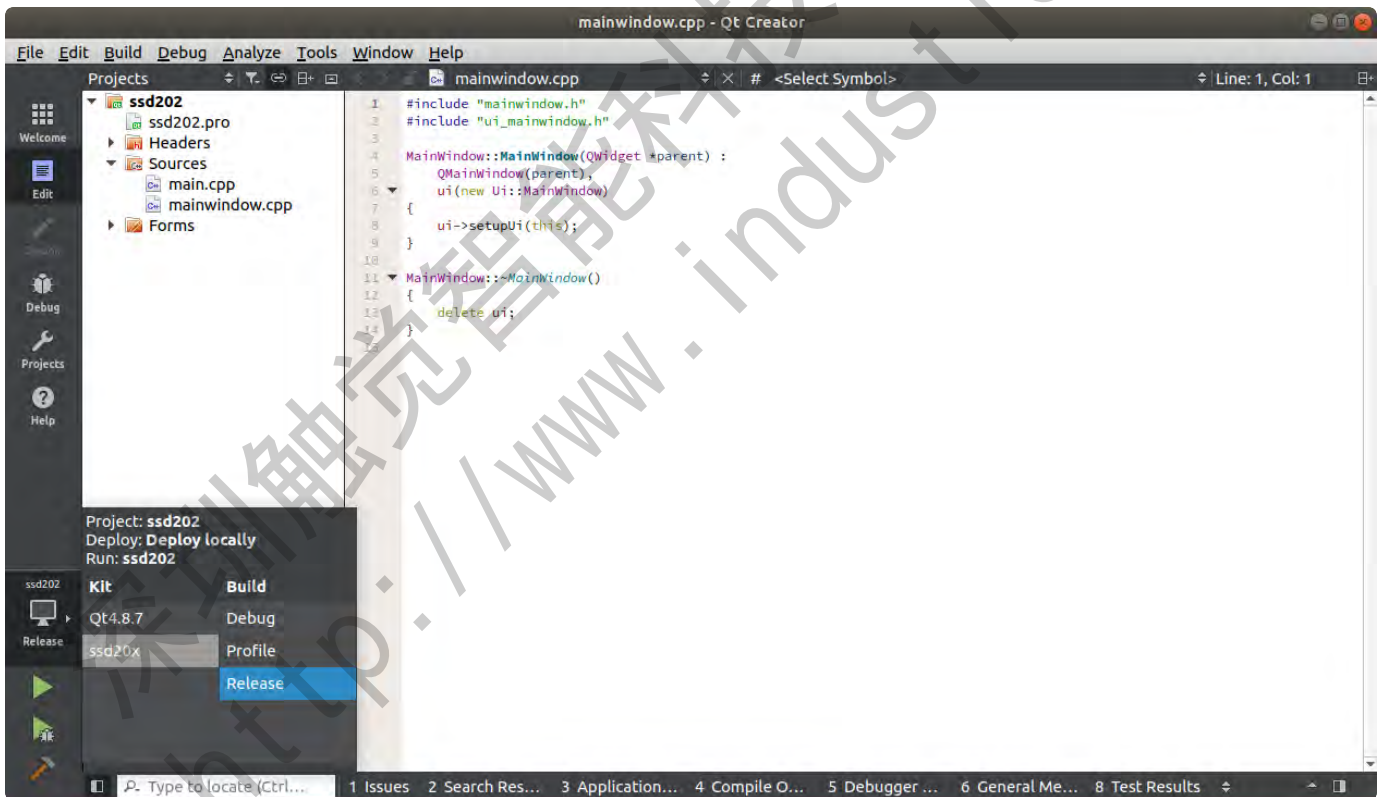
验证Qt Creator交叉编译配置，点击Qt Creator软件的“File→New File or Project”，新建一个“Qt widget Application”工程。



在“Kit Selection”界面，选择添加的“ssd202x”



在Qt Creator左侧选择“ssd20x->Release”

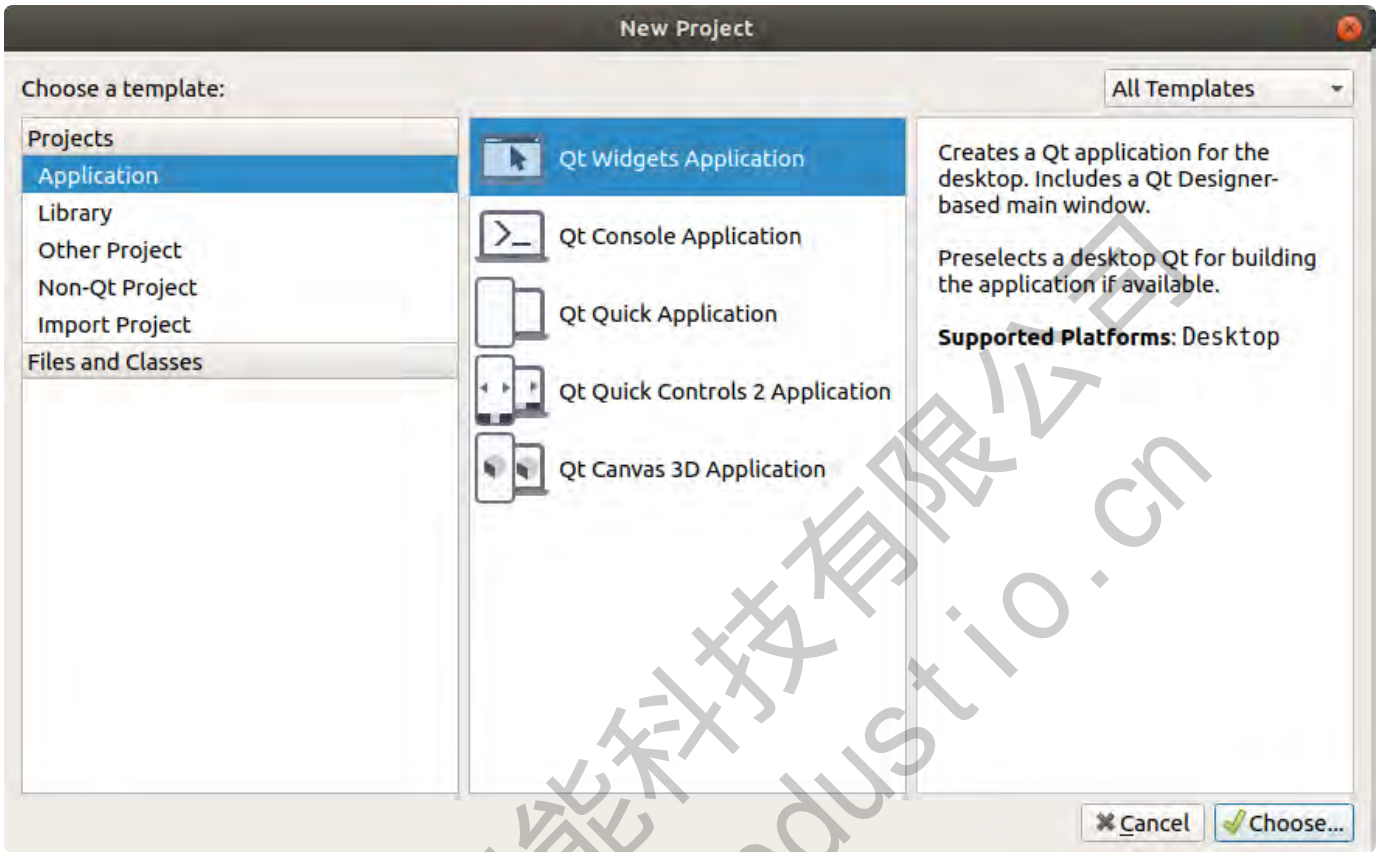


点击“Build”按钮编译；正常编译完，在新建工程的“/home/wxq”目录下，产生一个“build-ssd202-ssd20x-Release”文件夹，文件夹中包含编译的中间文件和开发板上运行的可执行文件“ssd202”。

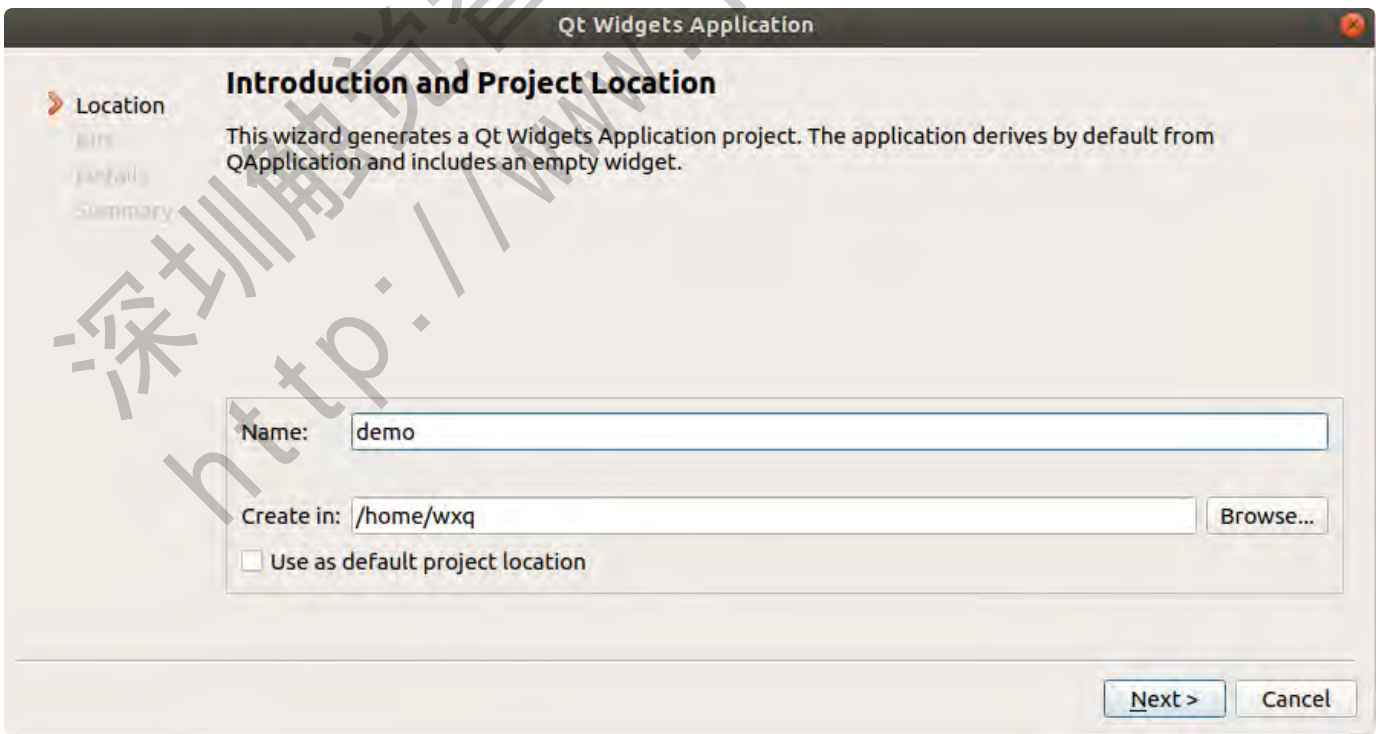
```
wxq@ubuntu:~/build-ssd202-ssd20x-Release$ ls
main.o      Makefile      moc_mainwindow.o  ssd202
mainwindow.o moc_mainwindow.cpp moc_predefs.h      ui_mainwindow.h
wxq@ubuntu:~/build-ssd202-ssd20x-Release$
```

编写Qt 程序在开发板中运行

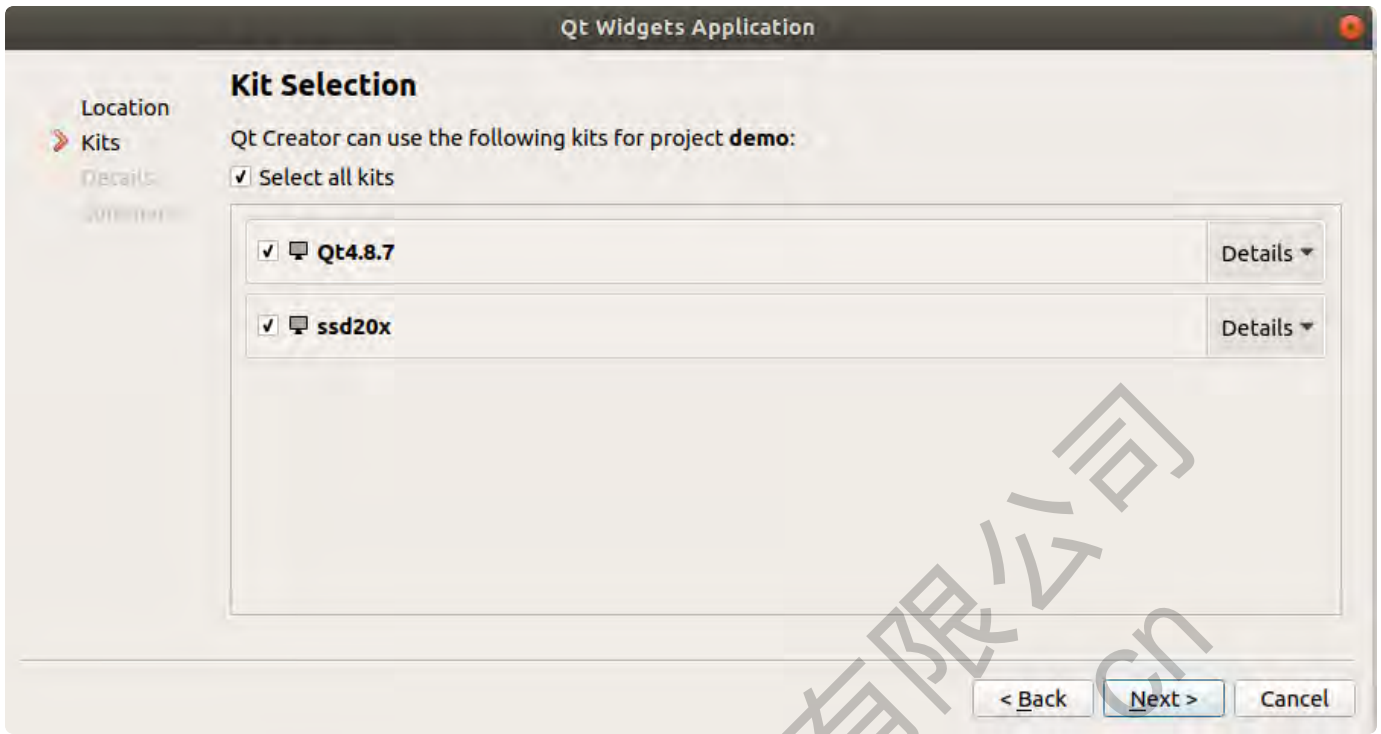
这里以编写hello world为例，点击新建工程，选择“Qt Widgets Application”,之后点击“choose”



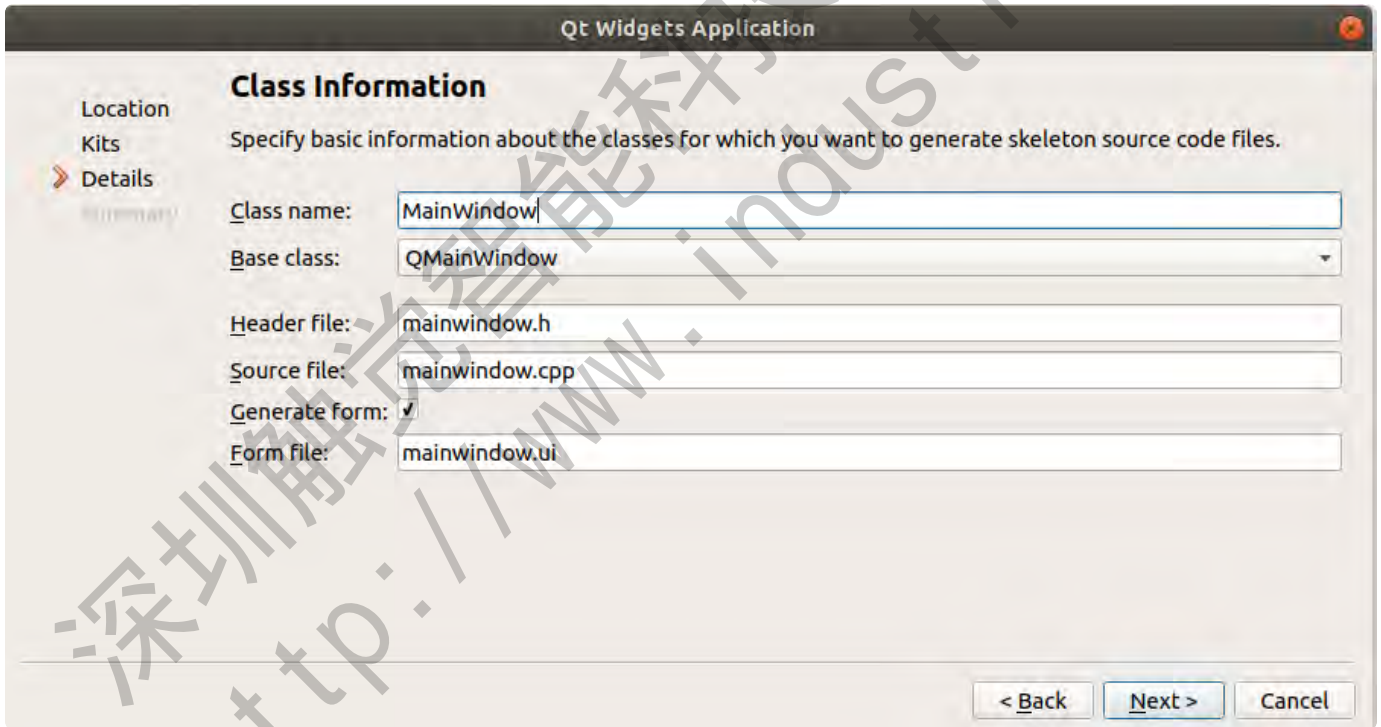
名字可以随便一个



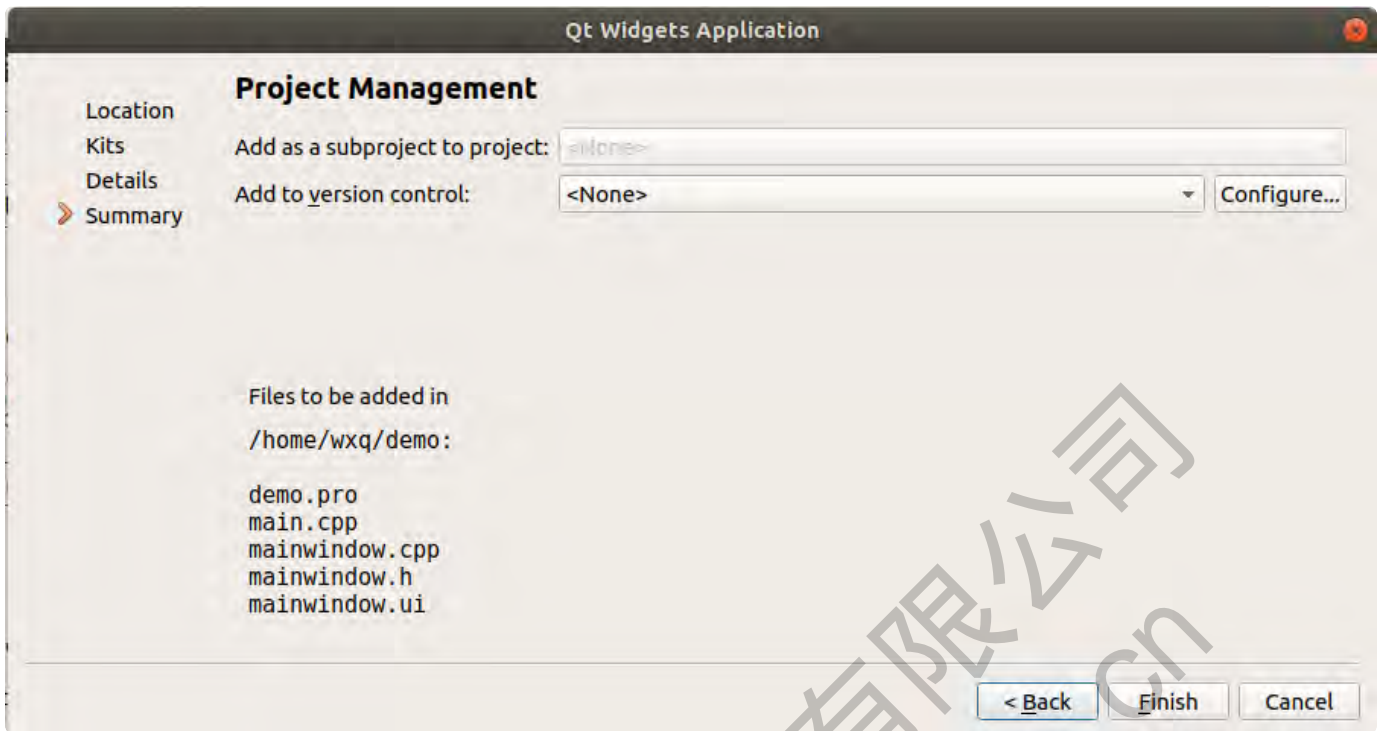
添加ssd20x



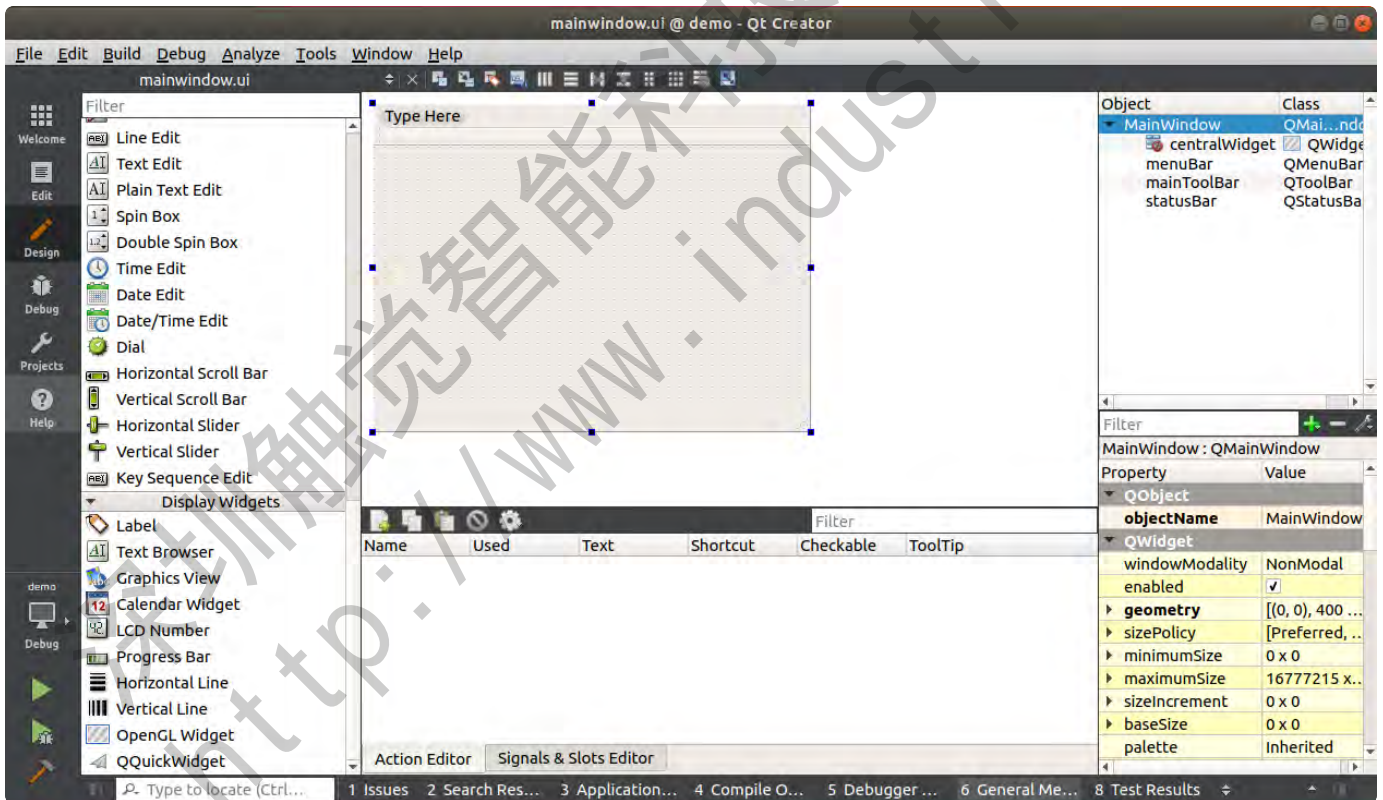
默认即可



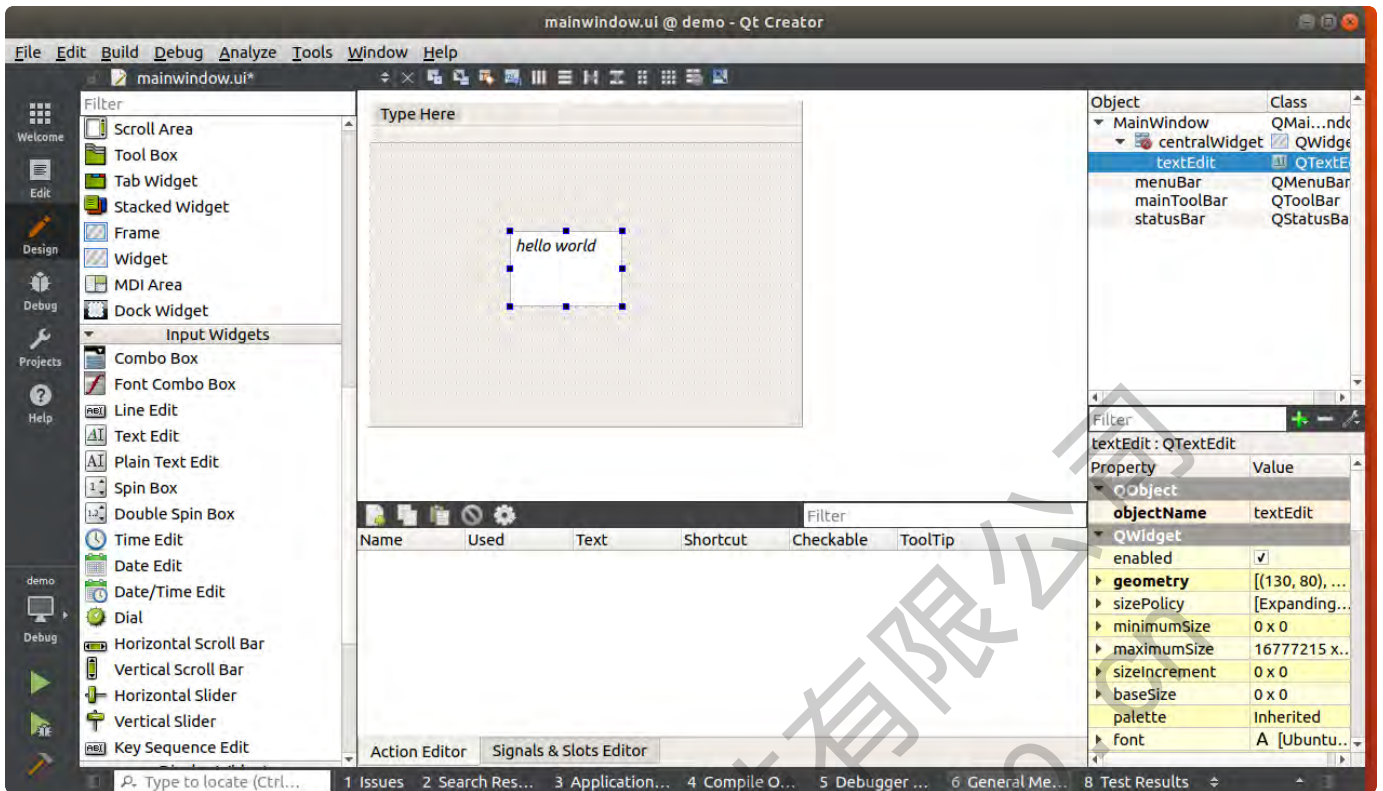
点击“Finish”



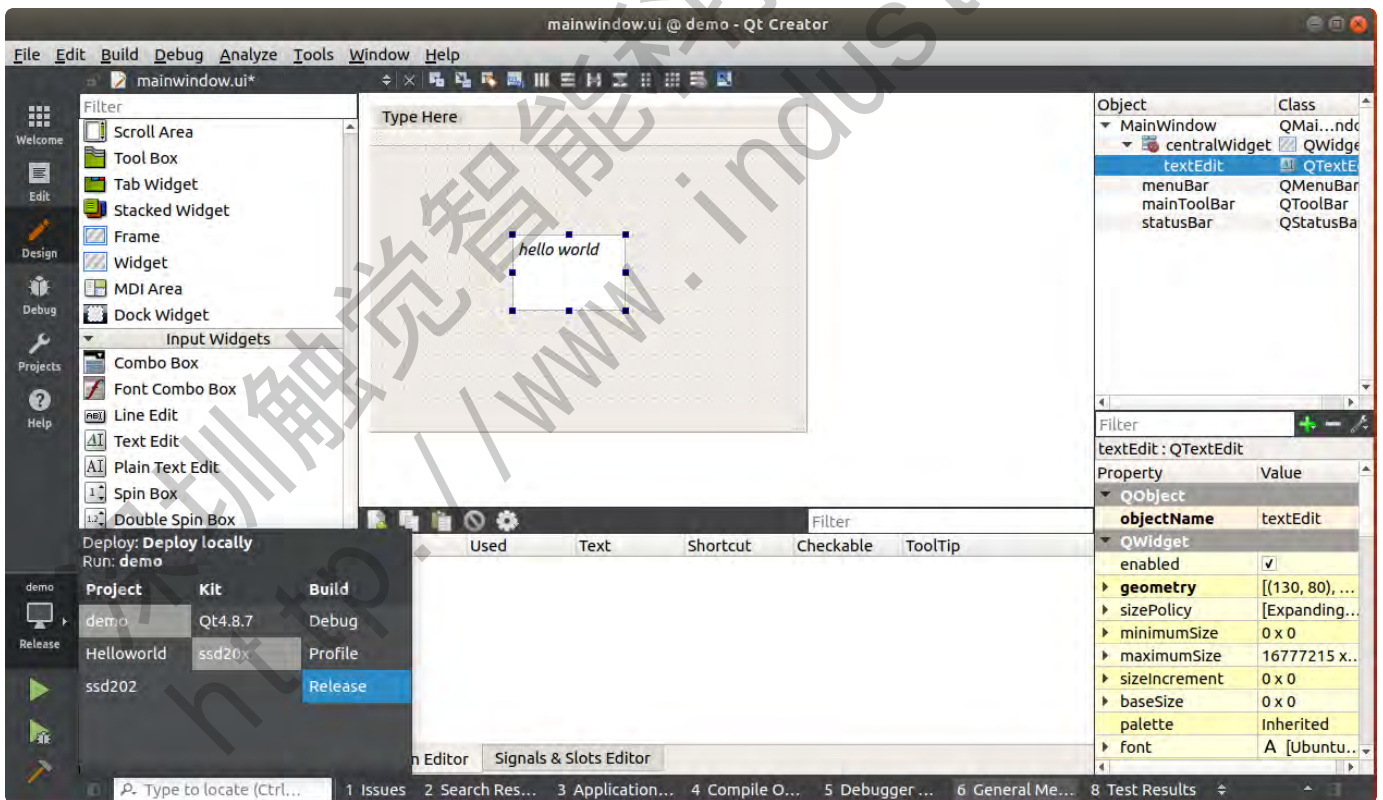
点击“Forms->mainwindow.ui”



选择“Text Edit”拉到“Type Here”中，写入“hello world”



debug选择“ssd20x->Release”



点击“Build”按钮编译；生产可执行文件，把文件拷贝到开发中运行

