Purple Pi R1 应用开发

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：

Plain Text 复制代码

```
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-gneabihf-gcc：

Plain Text 复制代码

```
industio@industio$: which arm-linux-gnueabihf-gcc
/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

```
industio@industio$:cd ido-lvgl8.2
industio@industio$:mkdir objs
industio@industio$:mkdir output
industio@industio$:make clean
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

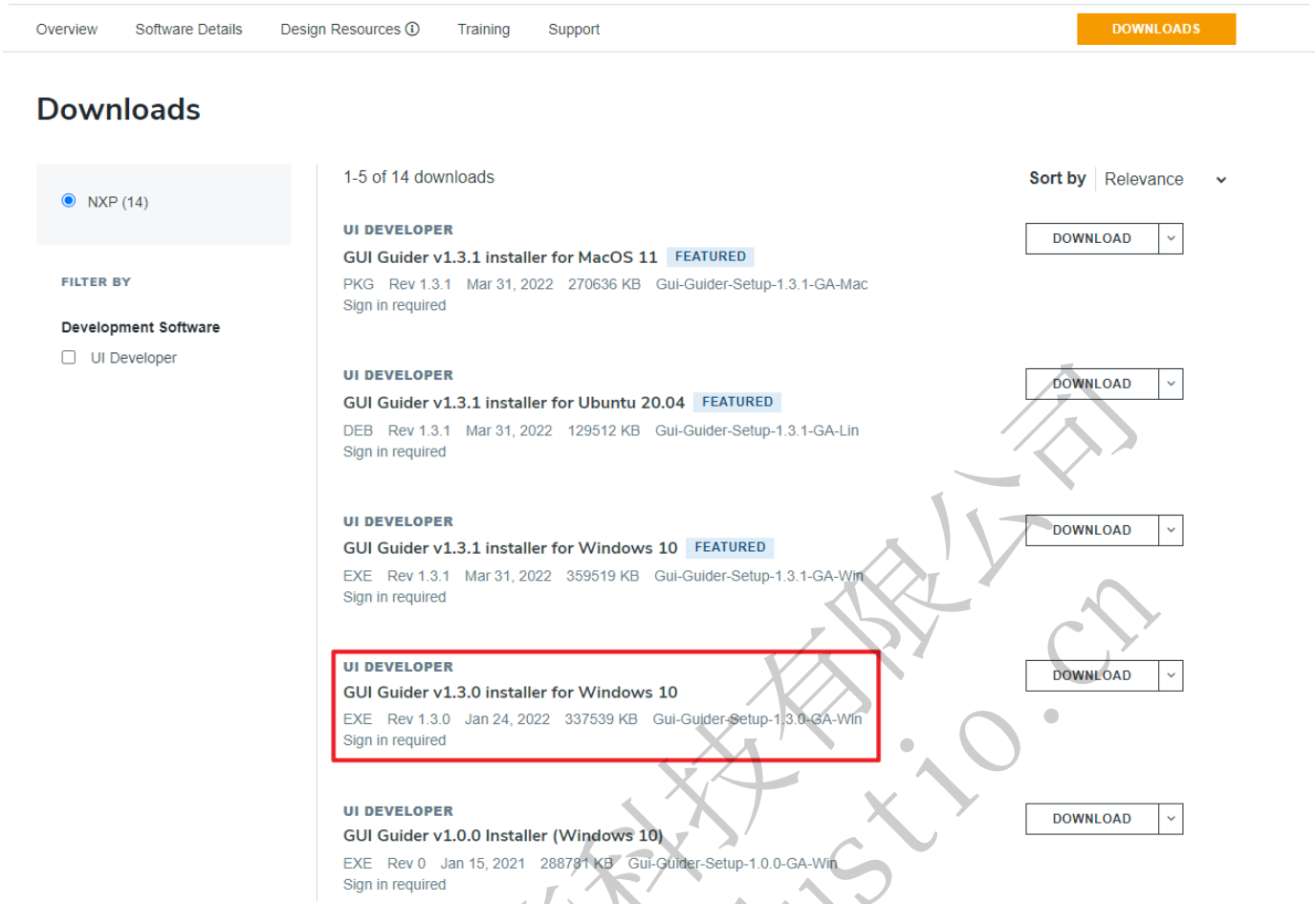
安装JDK

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

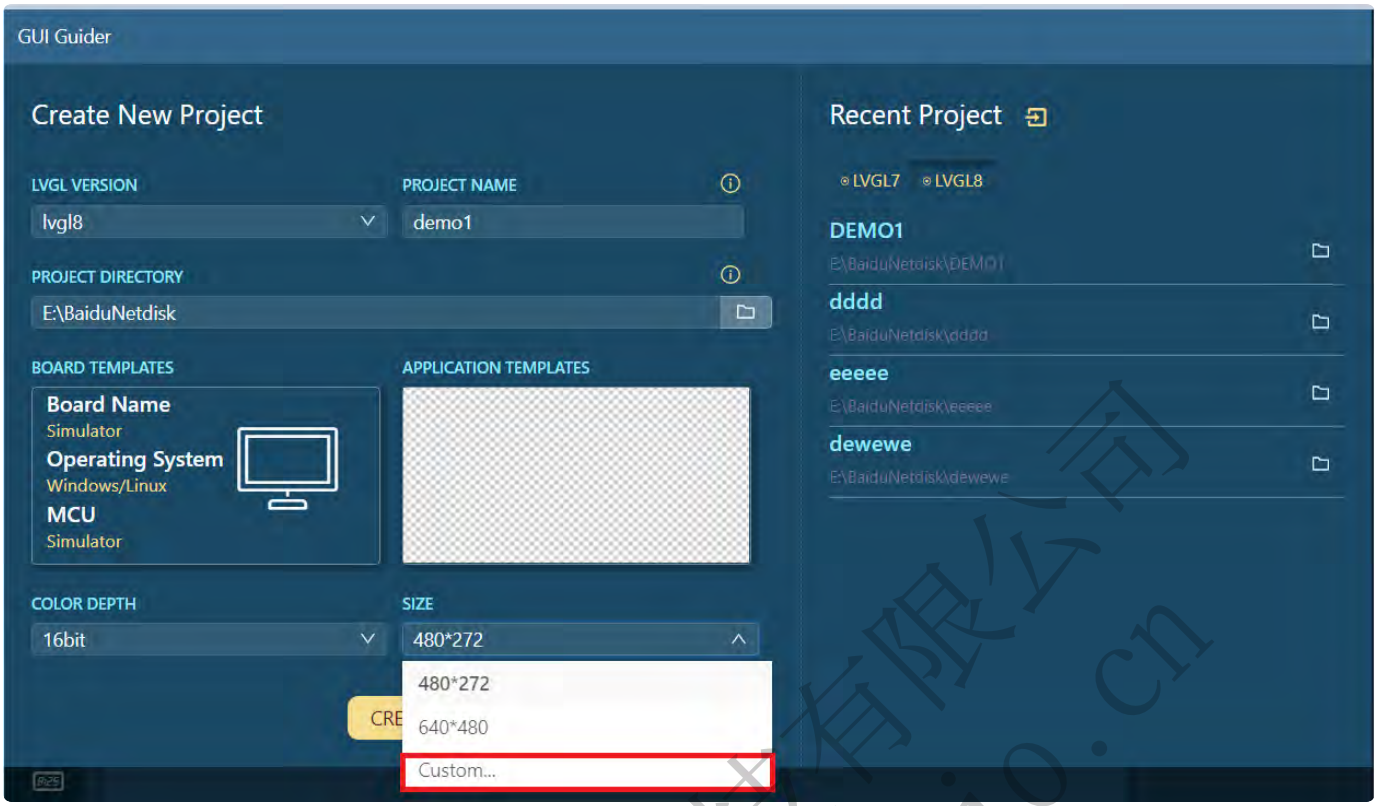
登录https://www.nxp.com/design/software/development-software/gui-guider:GUI-GUIDER，往下翻页找到GUI Guider v1.3.0并下载：

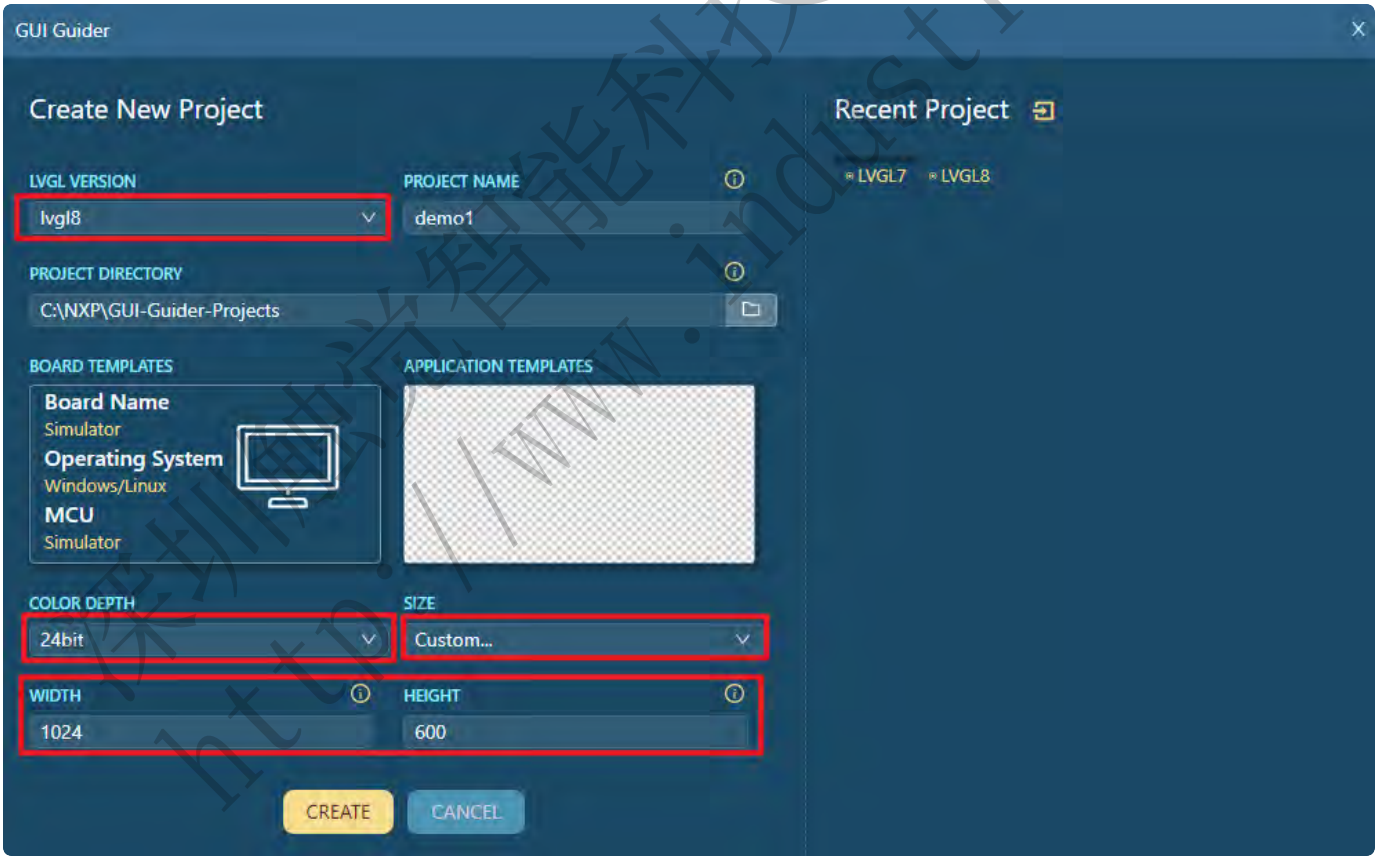


安装包下载完成后，双击安装即可。

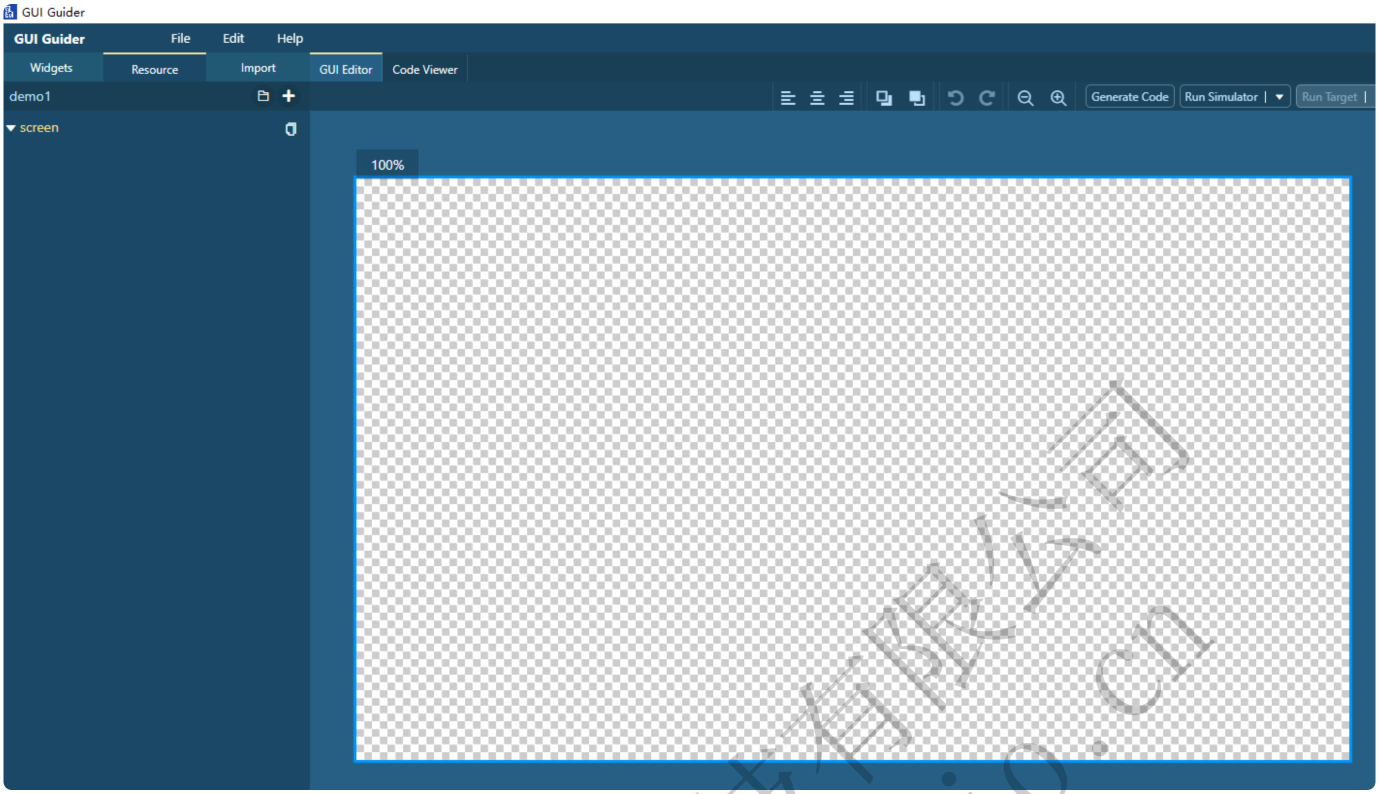
在GUI Guider上开发应用

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：

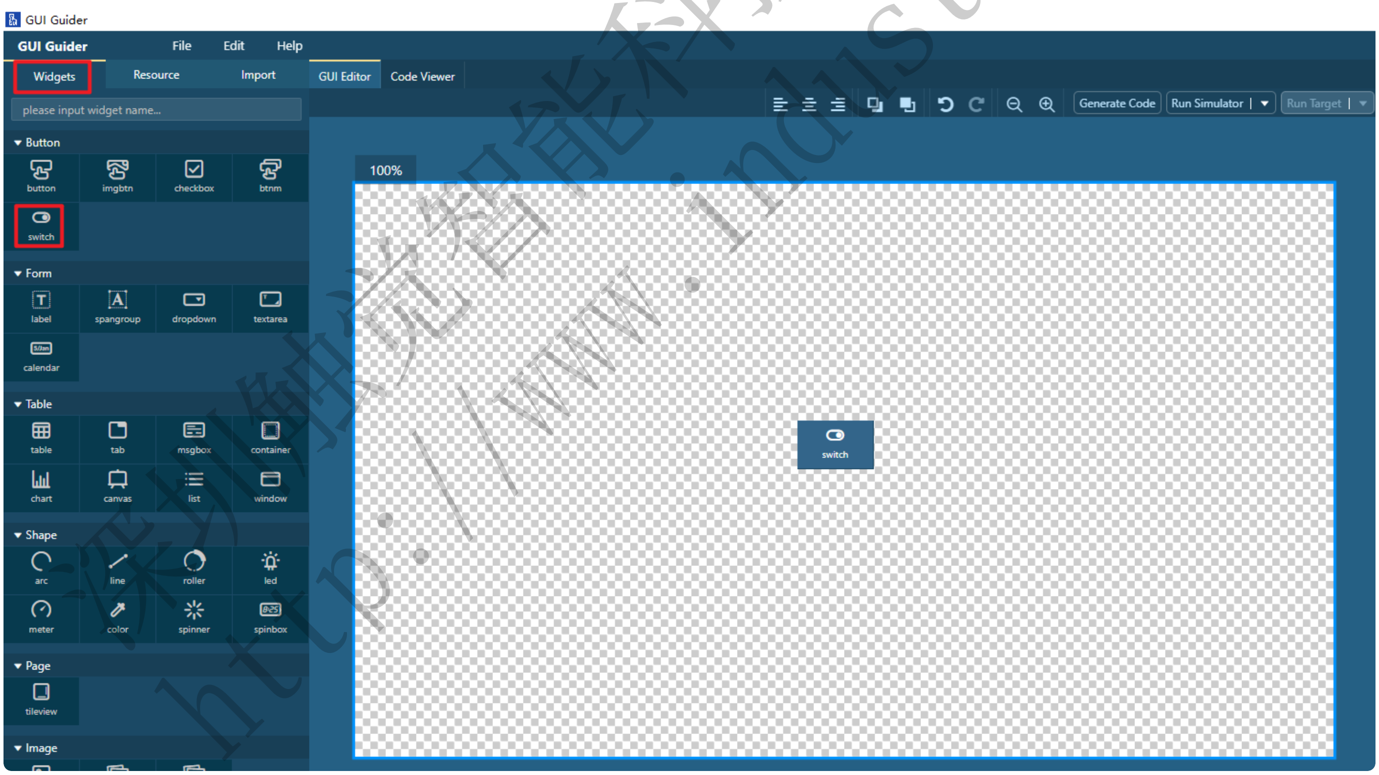




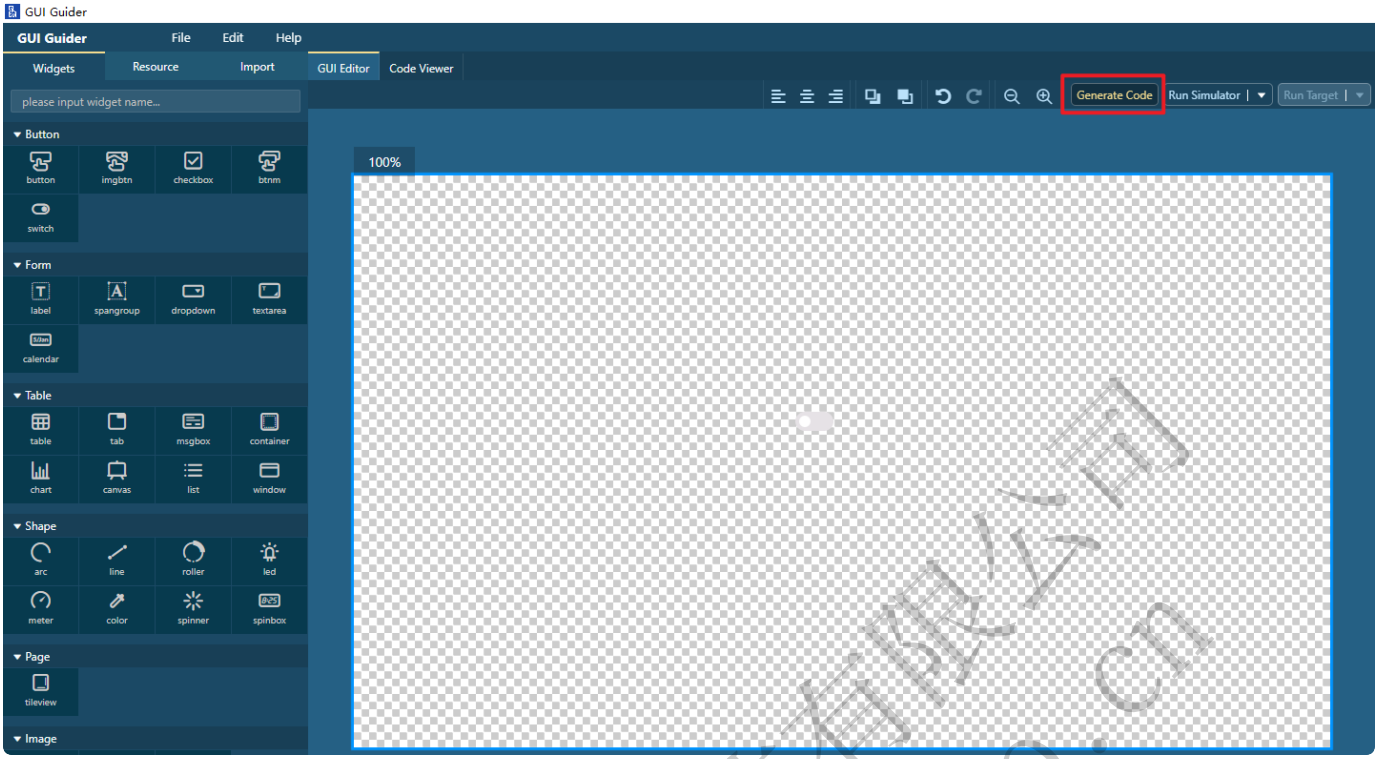
点击下方CREATE按钮，项目创建完成：



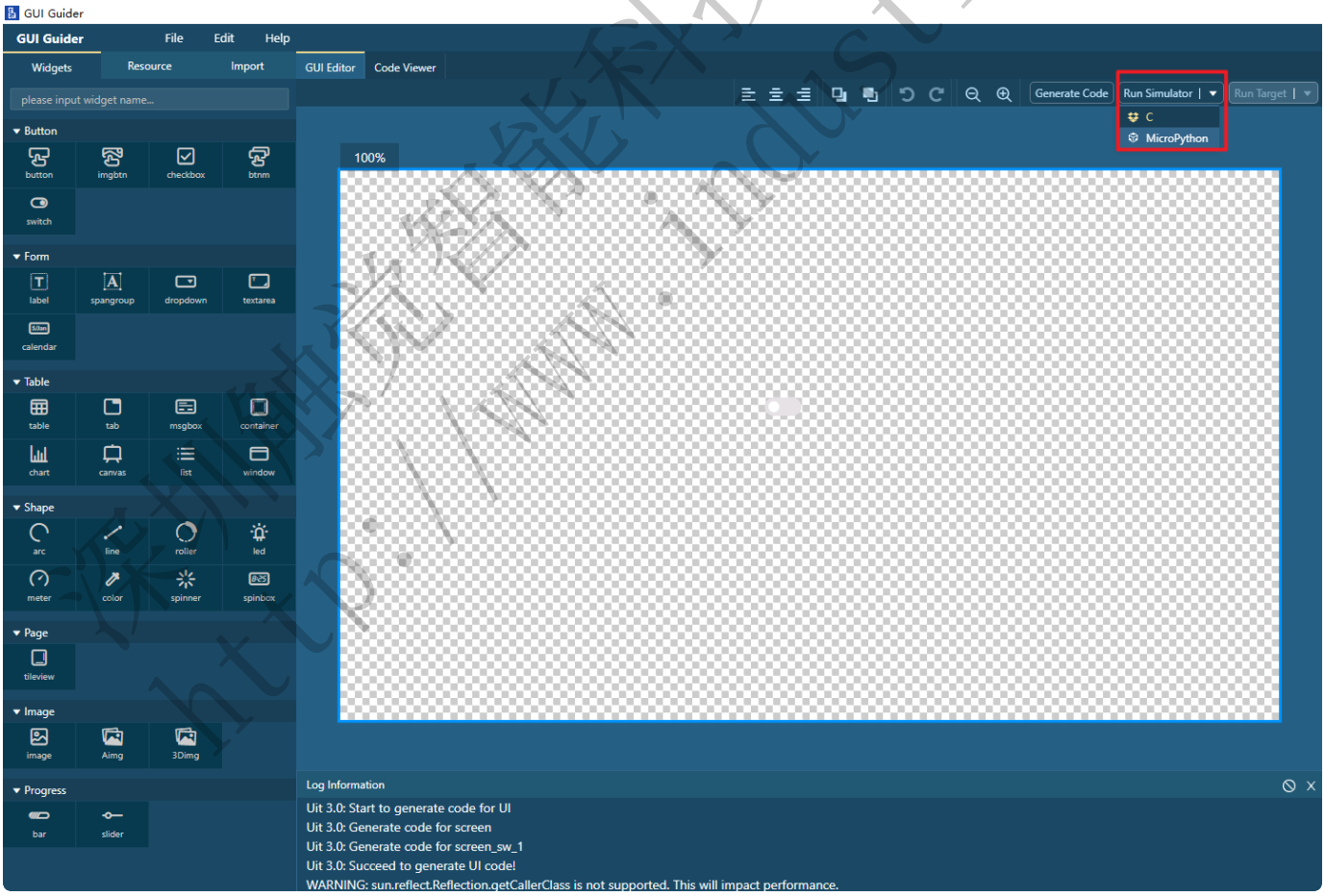
选择Widgets页，拖动放置一个switch器件：



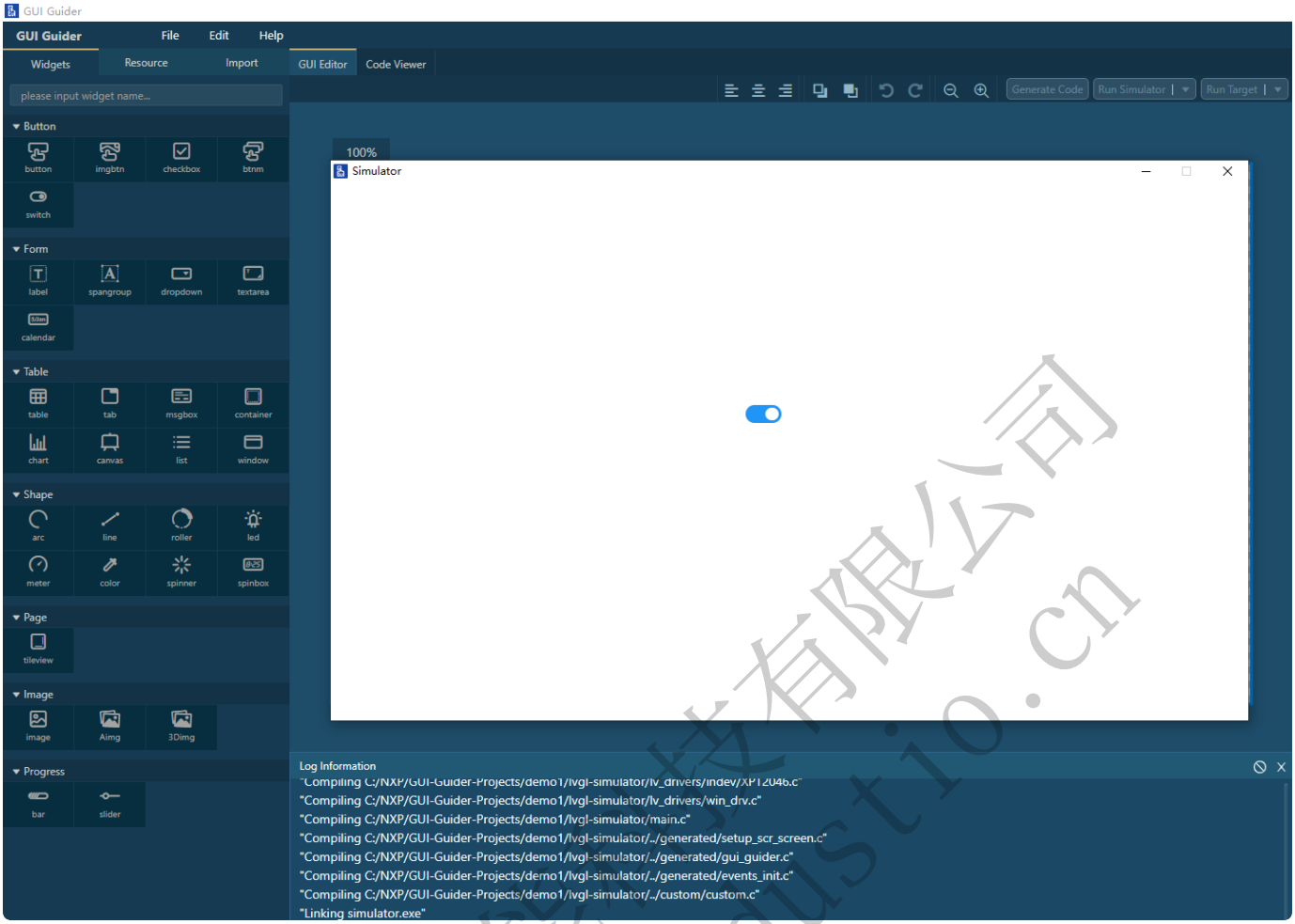
点击右上角Generate Code按钮：



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

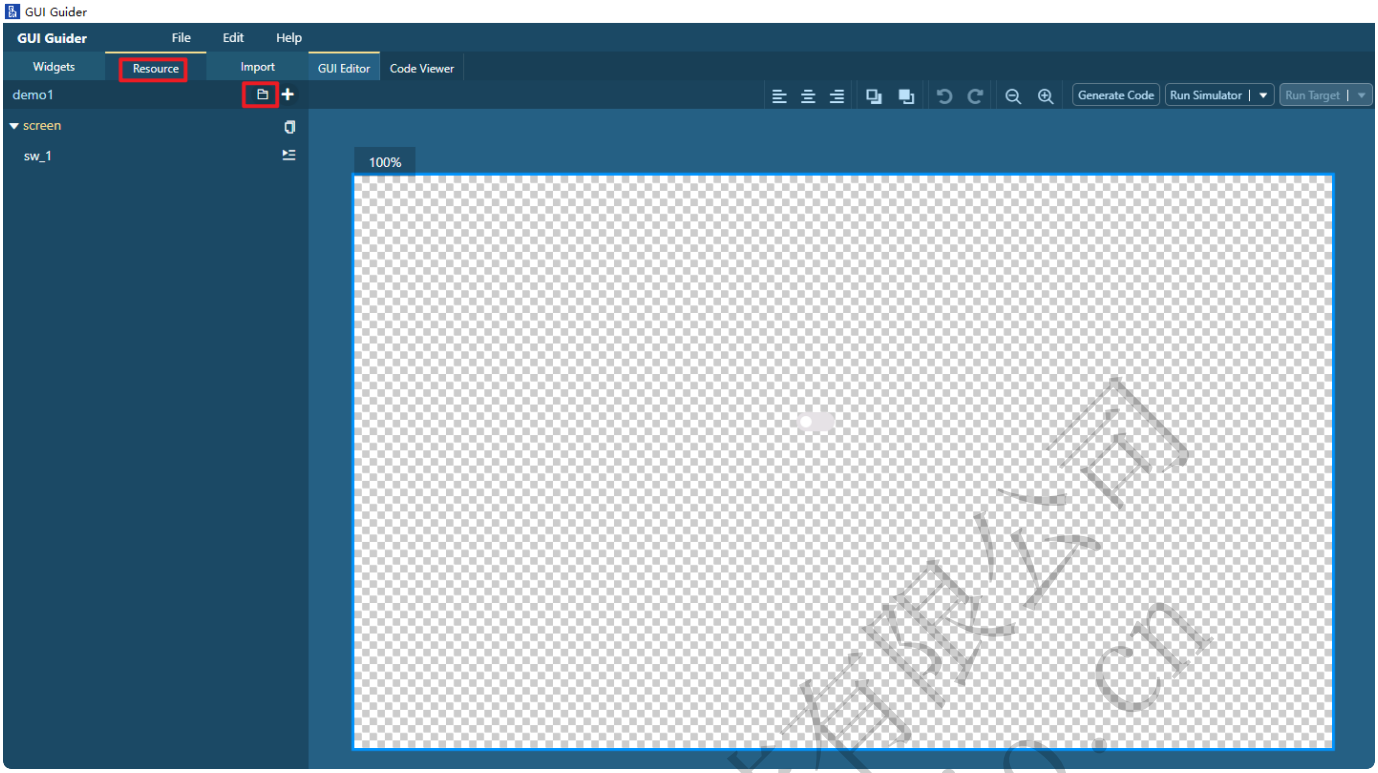


可以看到，刚刚放置的按钮显示在窗口上：



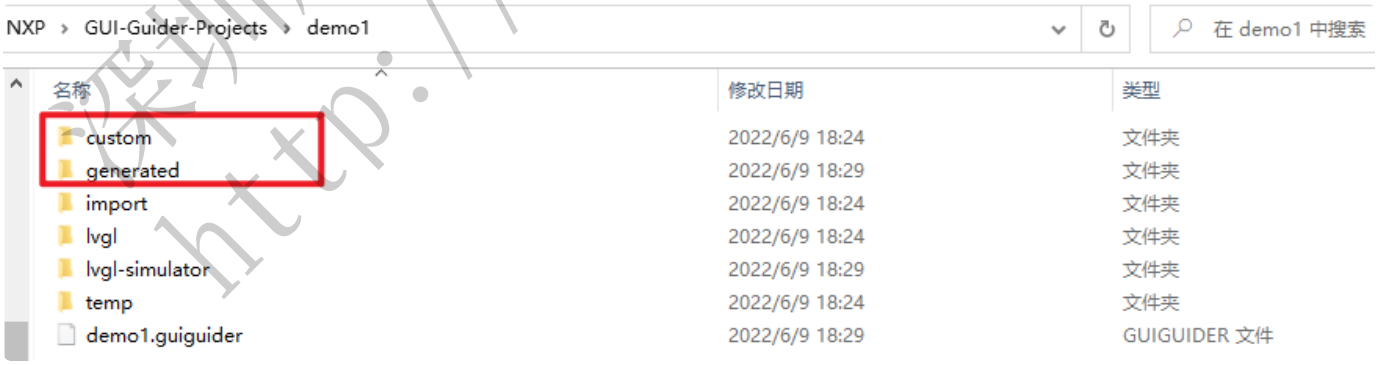
交叉编译lvgl应用

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：




```
ronnie@wt_rd_server:~/work3/ssd201/lvgl$ ls
custom  generated  lv_conf.h  lv_drivers  lv_drv_conf.h  lvgl  main.c  Makefile  output
ronnie@wt_rd_server:~/work3/ssd201/lvgl$
```

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

Plain Text 复制代码

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

替换完成后，即可开始编译app：

Plain Text 复制代码

```
industio@industio$:make clean
industio@industio$: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

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

如果没有初始化屏幕，先执行disp_init初始化屏幕：

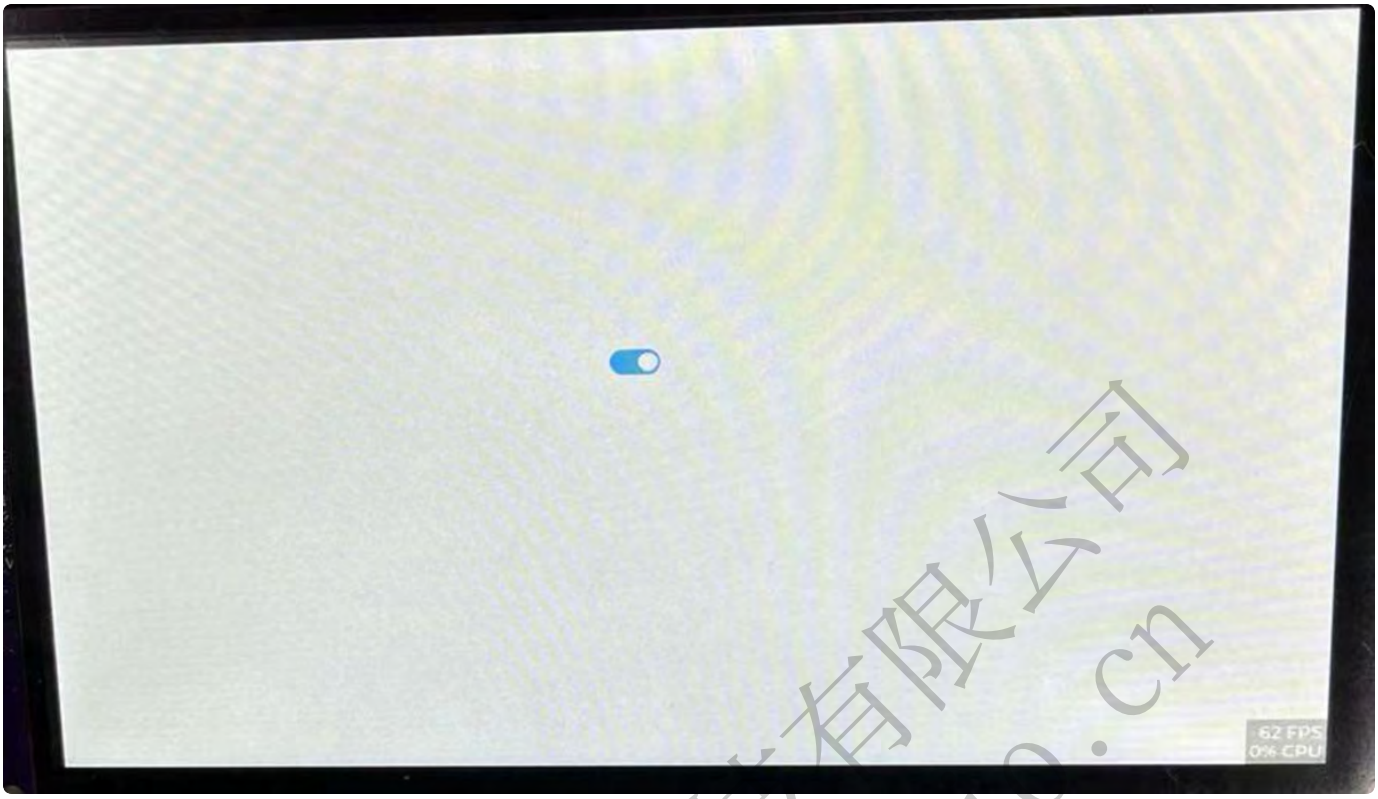
Plain Text 复制代码

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

运行app：

Plain Text 复制代码

```
#./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：

Plain Text 复制代码

```
/*1: Show CPU usage and FPS count*/
#define LV_USE_PERF_MONITOR 0
#if LV_USE_PERF_MONITOR
    #define LV_USE_PERF_MONITOR_POS LV_ALIGN_BOTTOM_RIGHT
#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

Plain Text 复制代码

```
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
```

Plain Text 复制代码

```
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
```

Plain Text 复制代码

```
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

Plain Text 复制代码

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

生成文件位于host目录下：

```
dynamic  include  share  static
```

4、验证

编译测试app

Plain Text 复制代码

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

注意:

CHIP ?= SSD20X （选择SSD20X）

```
CURRENT_PATH = $(shell pwd)
CROSS_COMPILE ?=arm-linux-gnueabihf-
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
```

Plain Text 复制代码

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

编译生成可执行文件ssplayer

```
wxquan@wt_rd_server: /work/ssd201/SSD20X-2D06/ido-sbc2d06-sdk/ffmpeg/ffplayer/app$ ls
EQT700BKJ004P_1024x600_MIPI.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
wxquan@wt_rd_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 = i6m
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-gnueabihf-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_MEMLEN) mma_heap=mma_heap_name0,miu=0,sz=0x1000000 ma_memblock_re
move=1 highres=off
TOOLCHAIN_REL = arm-linux-gnueabihf-
```

5、播放测试

Plain Text 复制代码

```
# ./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：

Plain Text 复制代码

```
#/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：

Plain Text 复制代码

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

默认连接热点配置文件为/etc/wpa_supplicant.conf：

Plain Text 复制代码

```
network={
        ssid="TP-LINK_2G_RONNIE"
        psk="12345678"
        key_mgmt=WPA-PSK
        priority=2
}

```

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：

Plain Text 复制代码

```
subnet 192.168.3.0 netmask 255.255.255.0 {
range 192.168.3.100 192.168.3.200;
option domain-name-servers 114.114.114.114;
option domain-name "dhcp";
option routers 192.168.3.1;
}
```

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

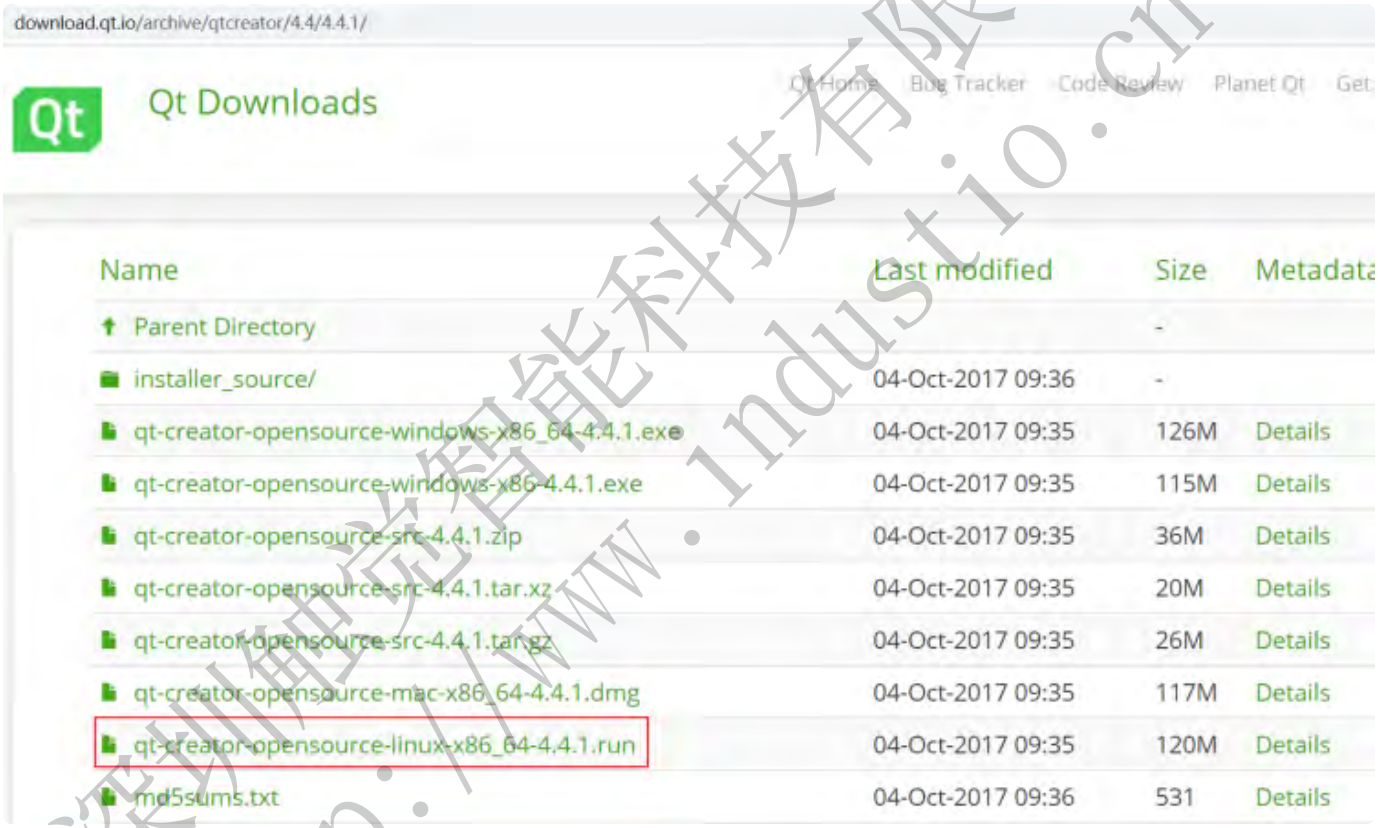
执行以下脚本：

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

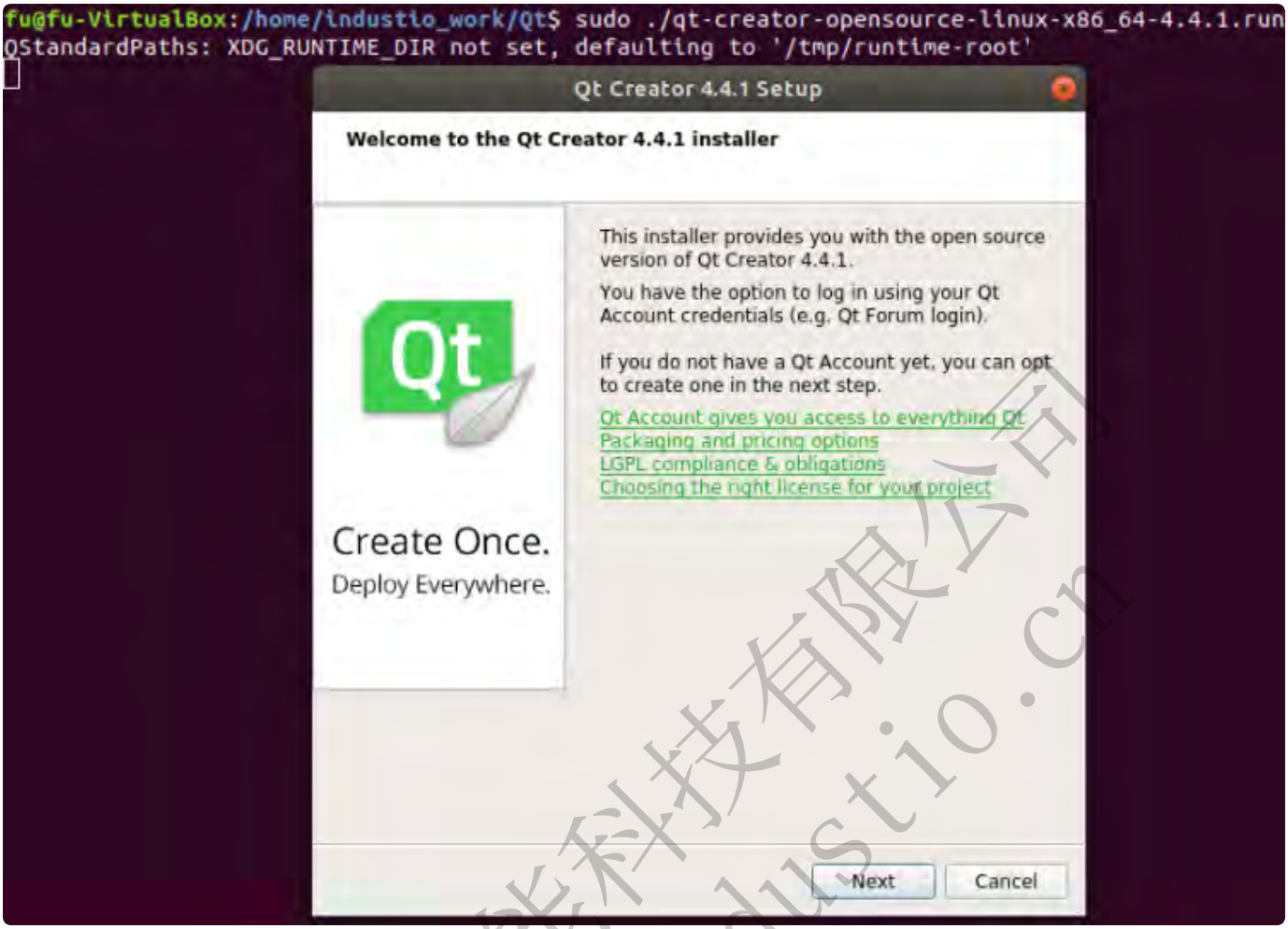
Qt Creator开发流程

配置Qt开发环境

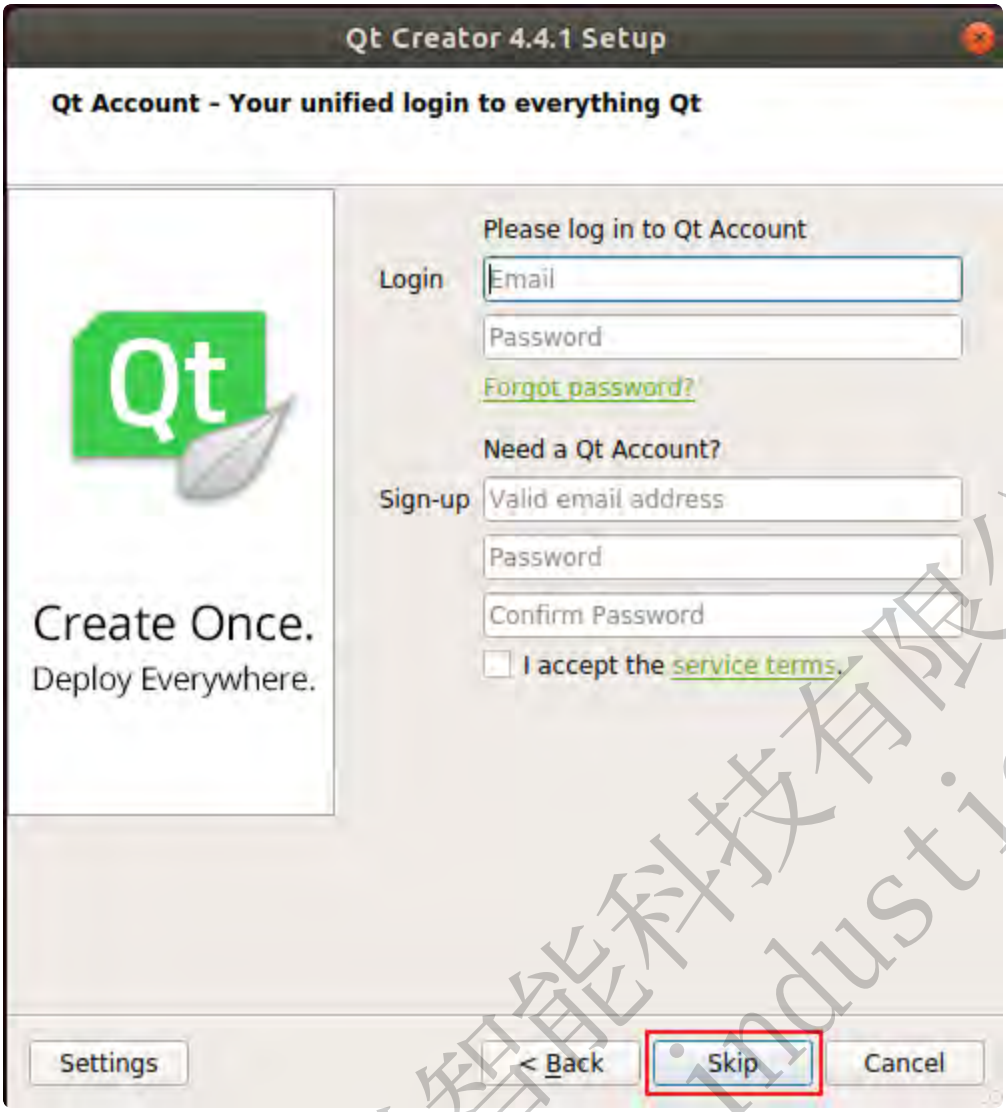
1、下载Qt Creator4.4.1下载地址：https://download.qt.io/archive/qtcreator/4.4/4.4.1/



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”按键进入下一步。如下图所示

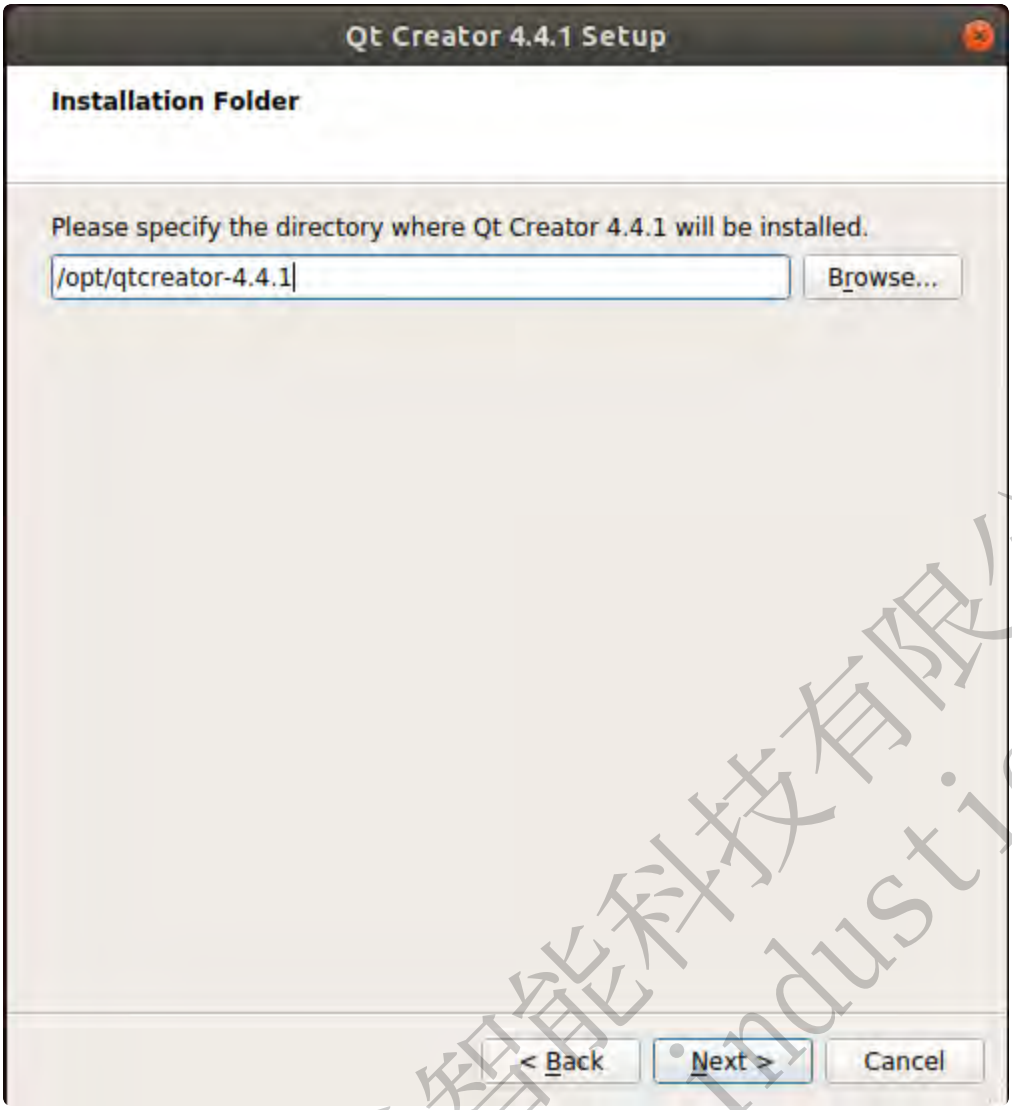


此处如果不需要登陆，点击“skip”跳过即可;

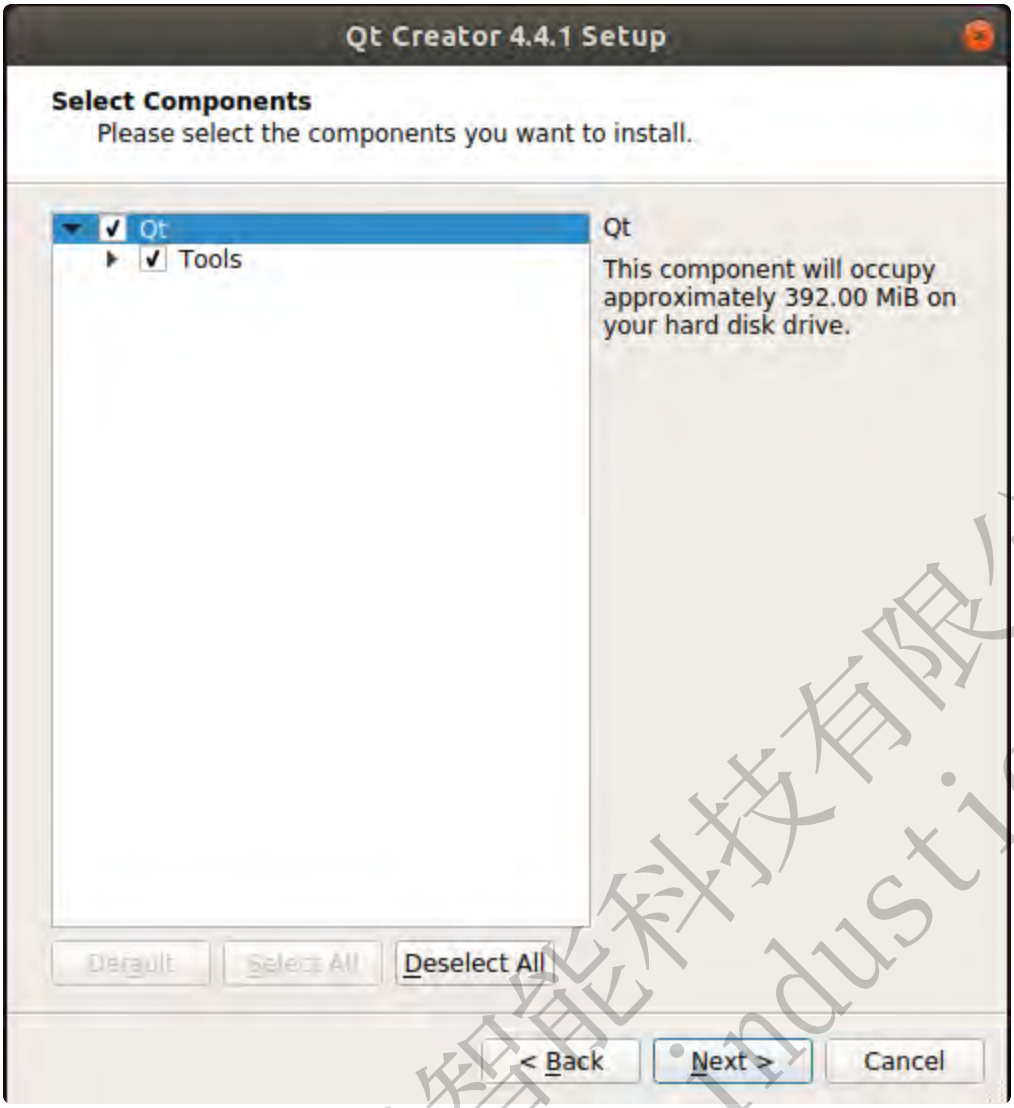


点击“Next”进入下一步；

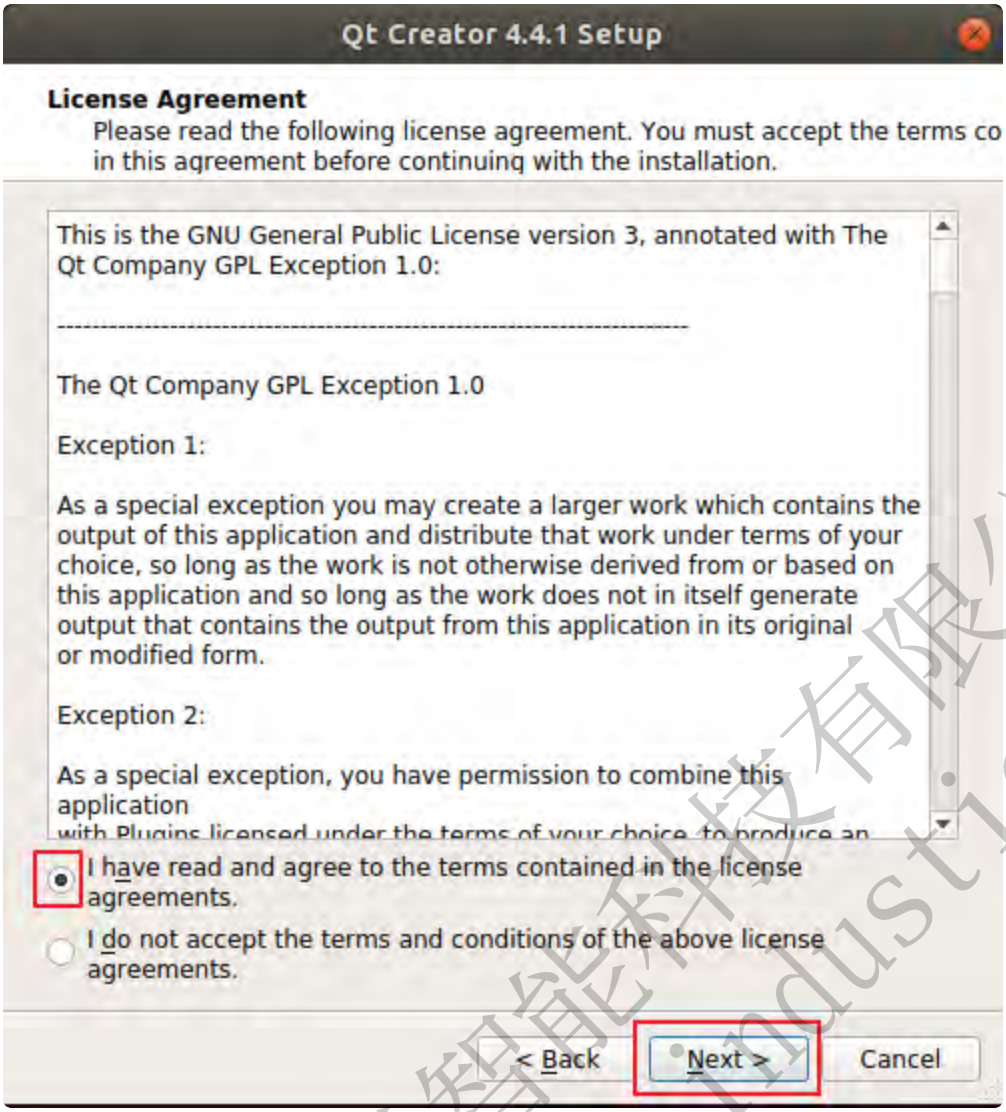
默认安装目录为“/opt/qtcreator-4.4.1”，点击“Next”进入下一步；



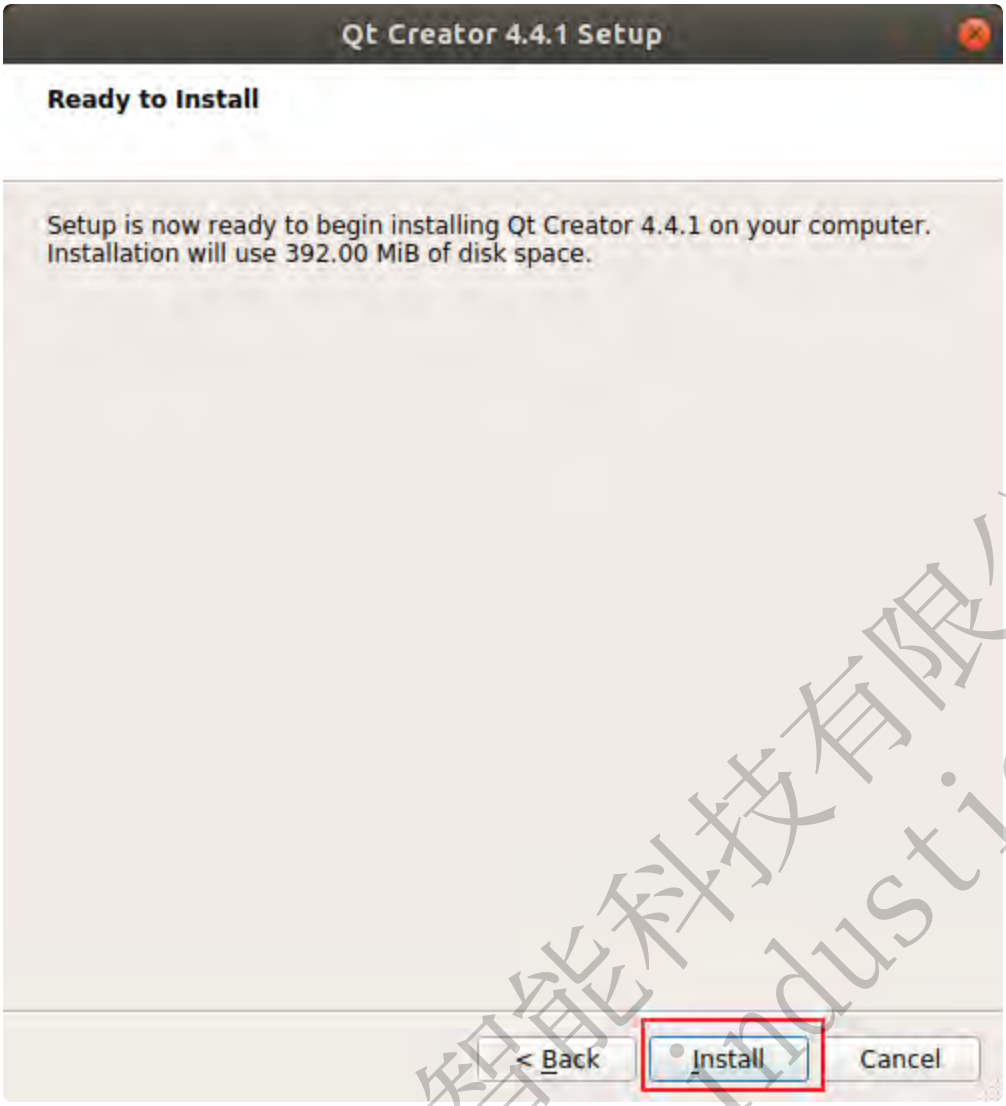
保留默认的勾选安装组件内容，点击“Next”进入下一步；



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；

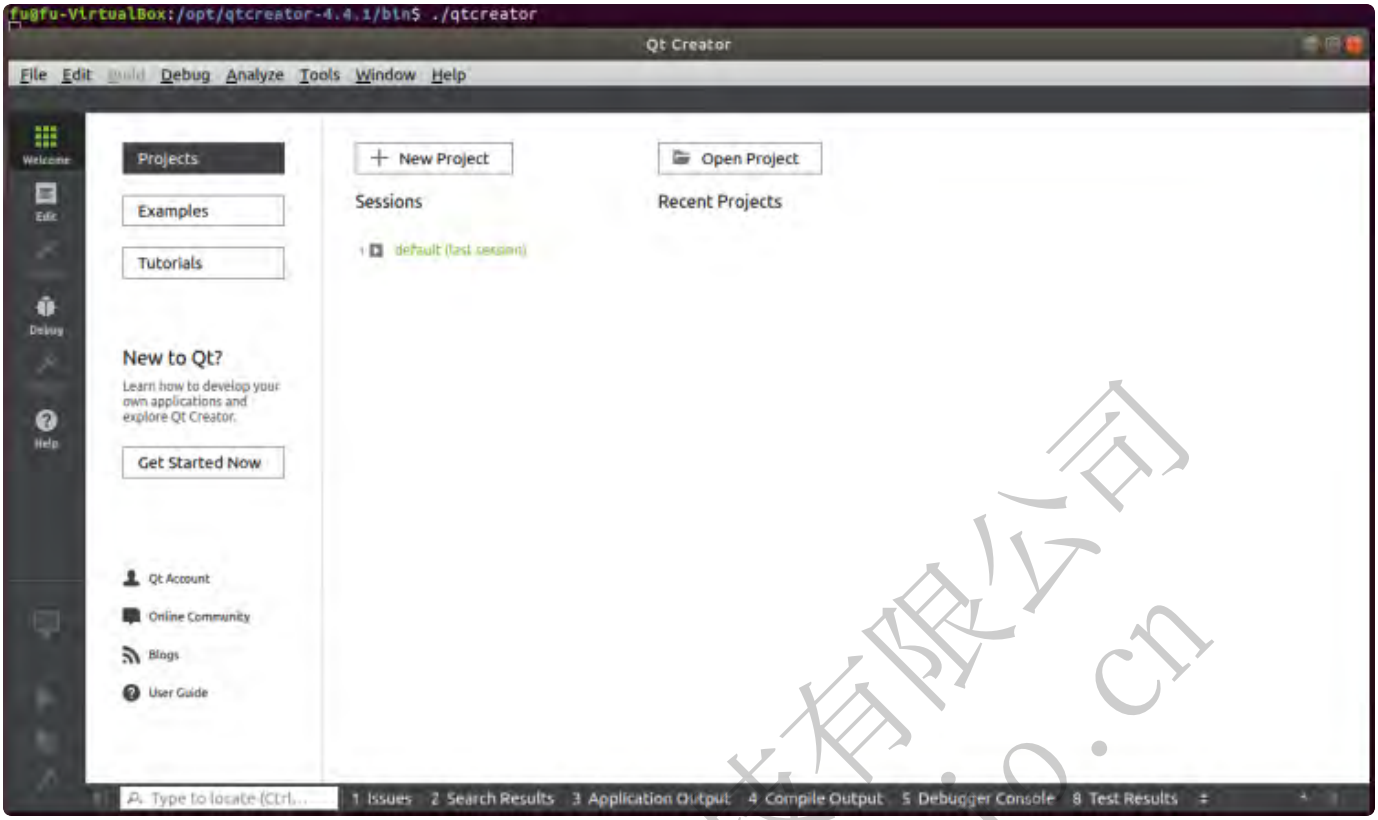


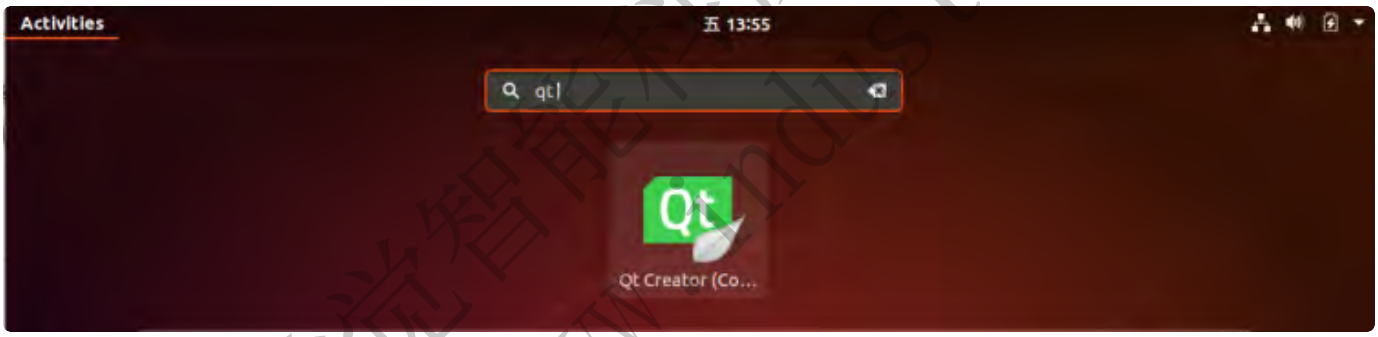
安装进度界面如下图；

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。

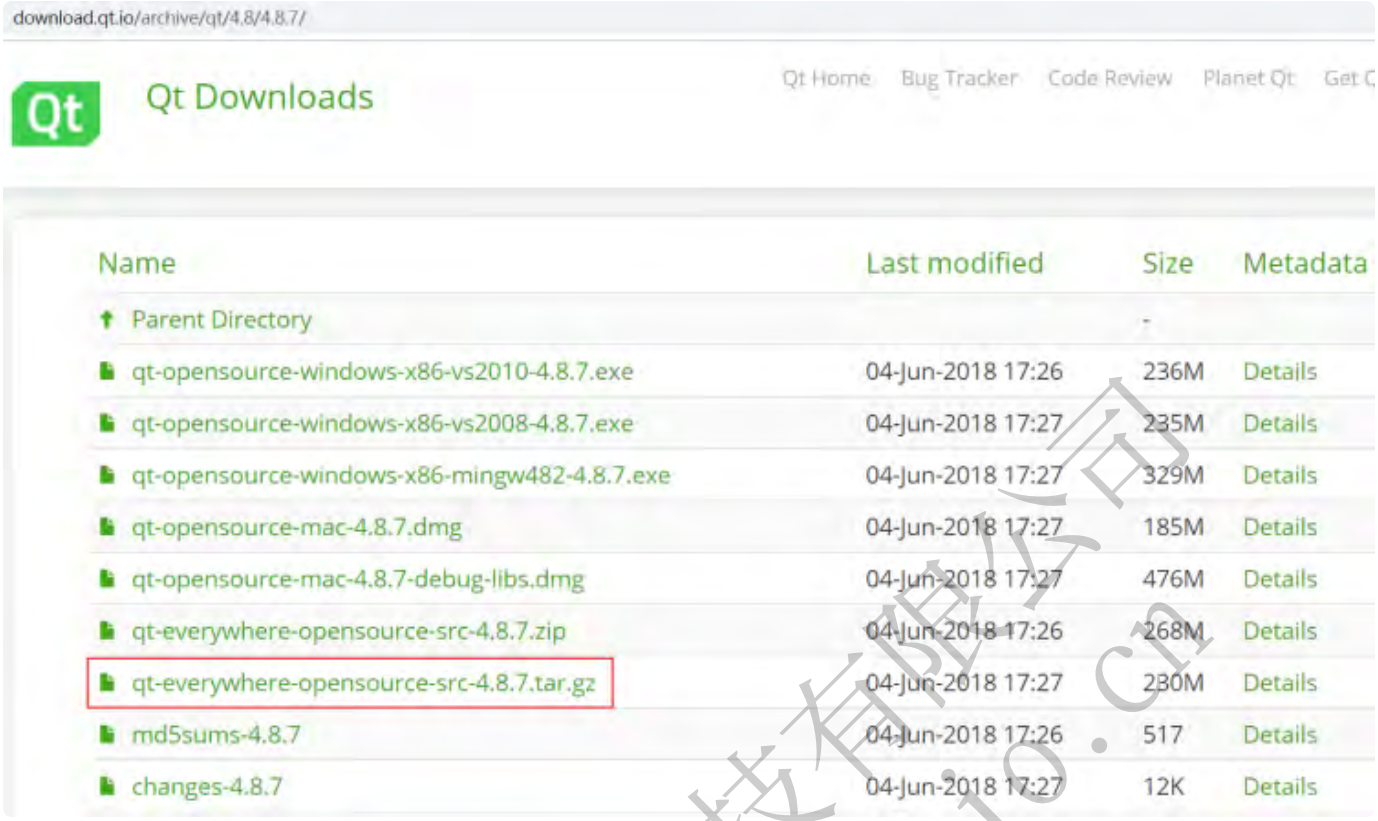




安装Qt4.8.7

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/



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。

Plain Text 复制代码

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

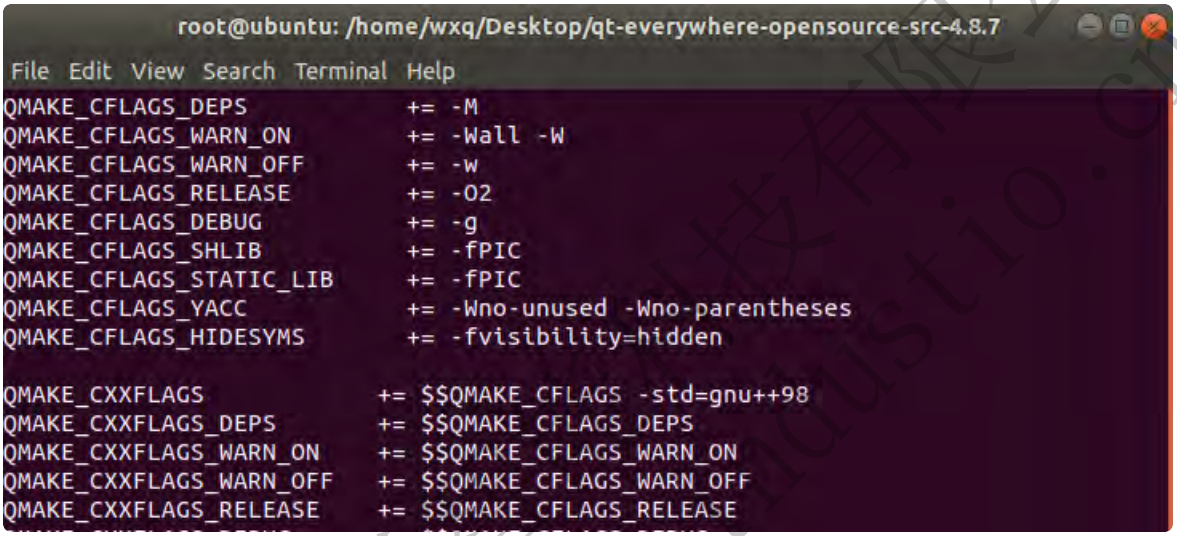
```
fu@fu-VirtualBox:/home/industio_work/Qt/qt-everywhere-opensource-src-4.8.7$ ls
bin              configure.exe  include           LICENSE.GPL3     projects.pro  tools
changes-4.8.7    demos          INSTALL           LICENSE.LGPL     qmake         translations
config.profiles  doc            LGPL_EXCEPTION.txt  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

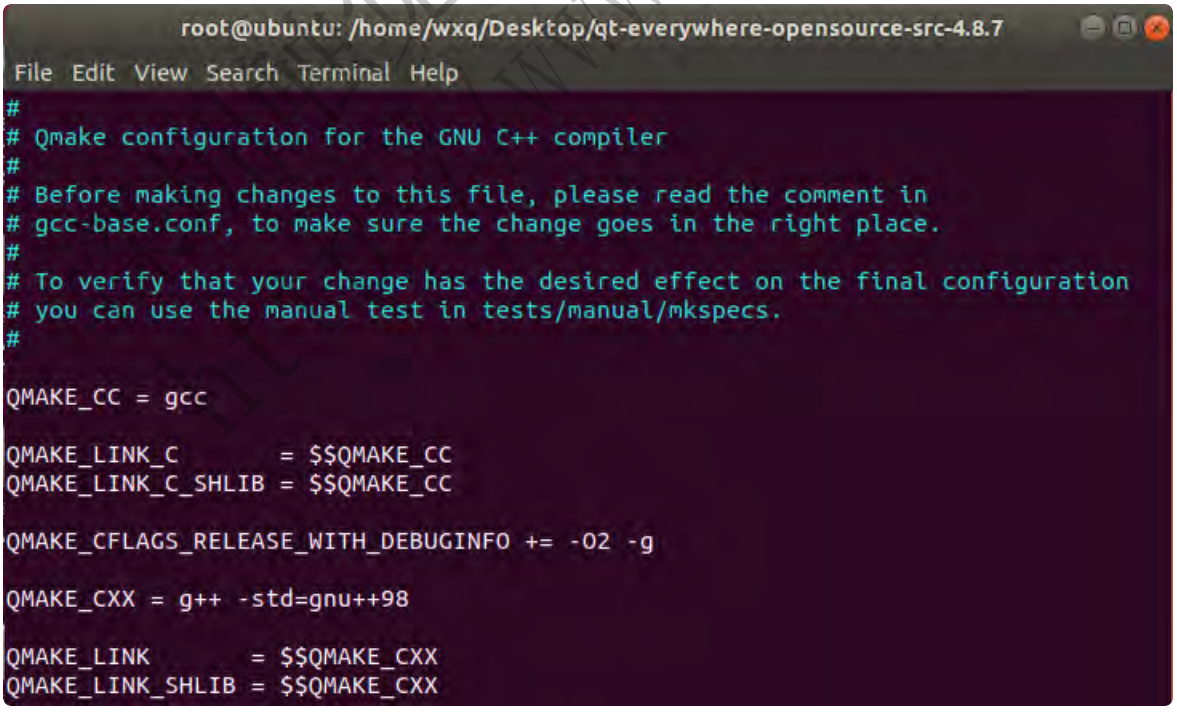
Plain Text 复制代码

```
industio@industio$:sudo apt-get install gcc g++ g++-multilib make automake
industio@industio$:sudo apt-get install zlib1g-dev lib32ncurses5 lib32z1 libpng-dev autoconf libtool
industio@industio$:sudo apt-get install libxext-dev libx11-dev libxext-dev libxtst-dev
industio@industio$:sudo apt install libgstreamer1.0-dev libgstreamer1-perl libgstreamer-plugins-base0.10-dev libgstreamerd-3-dev
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



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

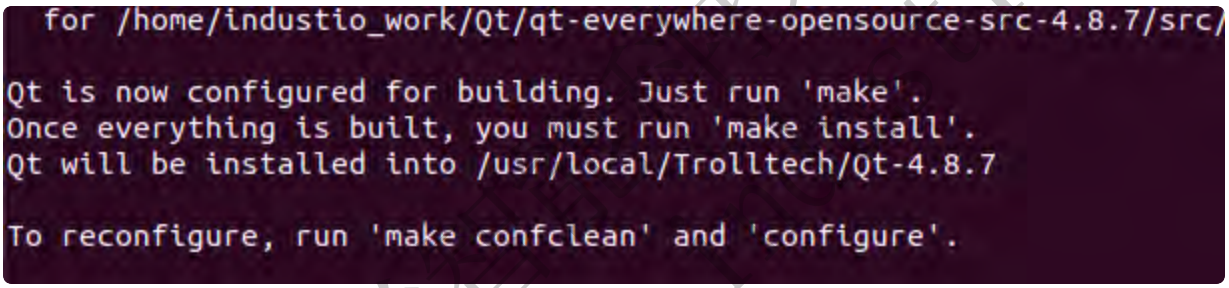


源码编译配置

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

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。



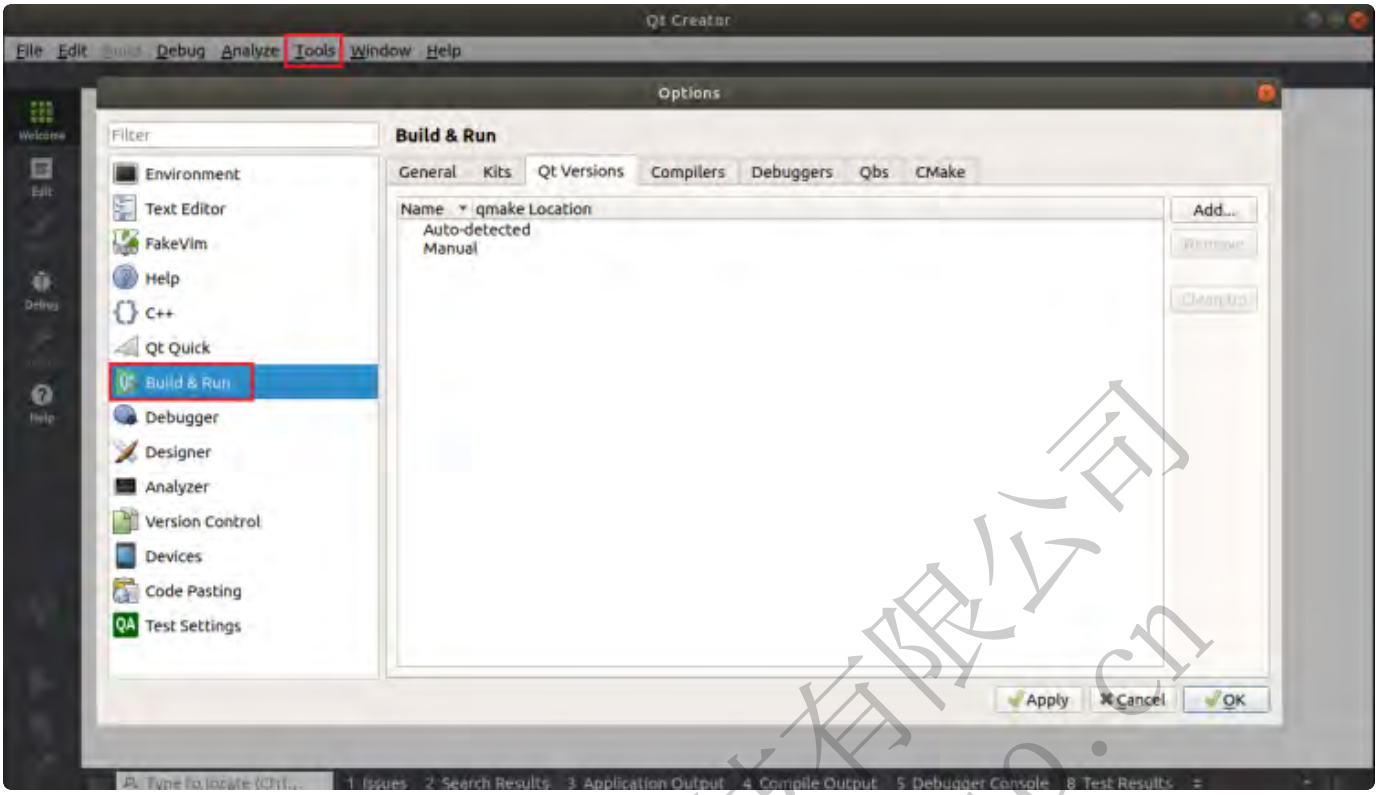
编译安装

```
industio@industio$:make && make install
```

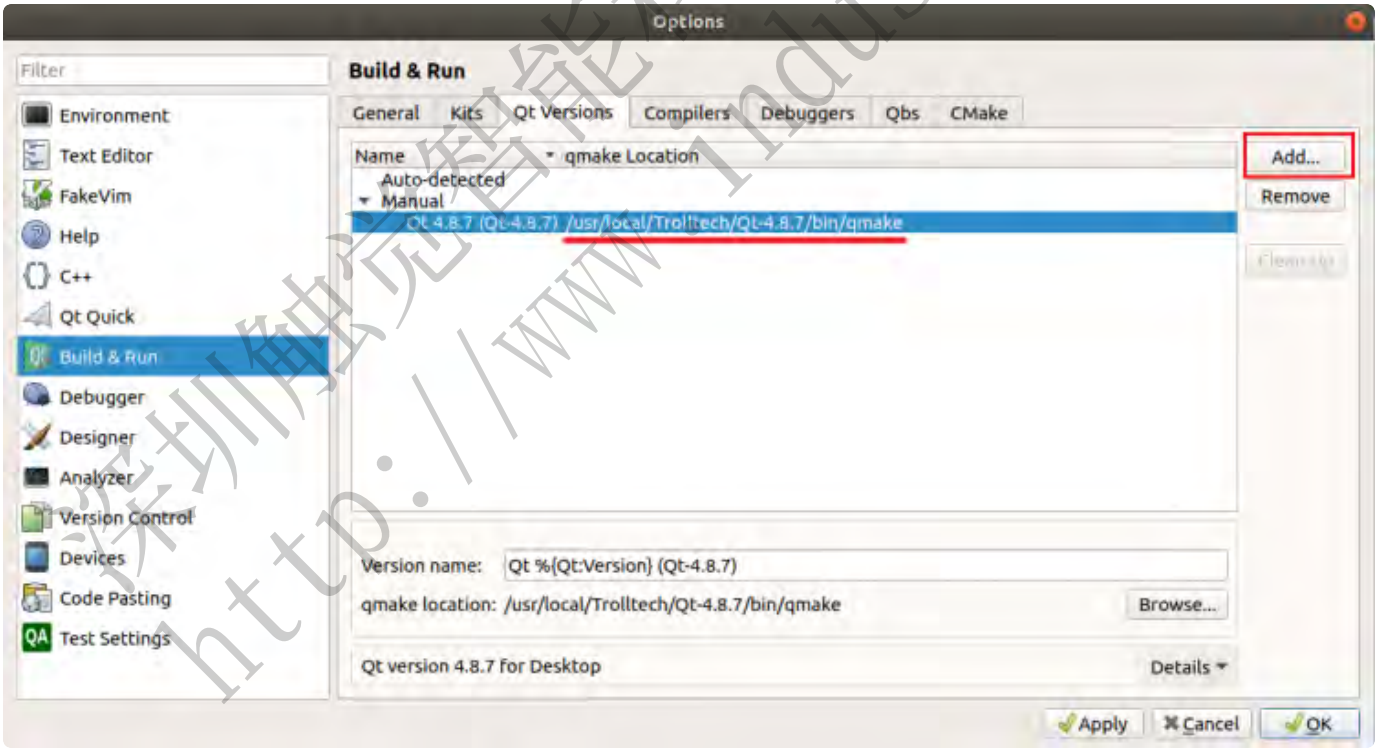
安装完成后，安装的文件位于”/usr/local/Trolltech/Qt-4.8.7“目录，进入sdk的bin目录，执行./qmake -v即可查看安装的qmake和Qt的版本。

Qt Creator添加Qt4.8.7开发套件

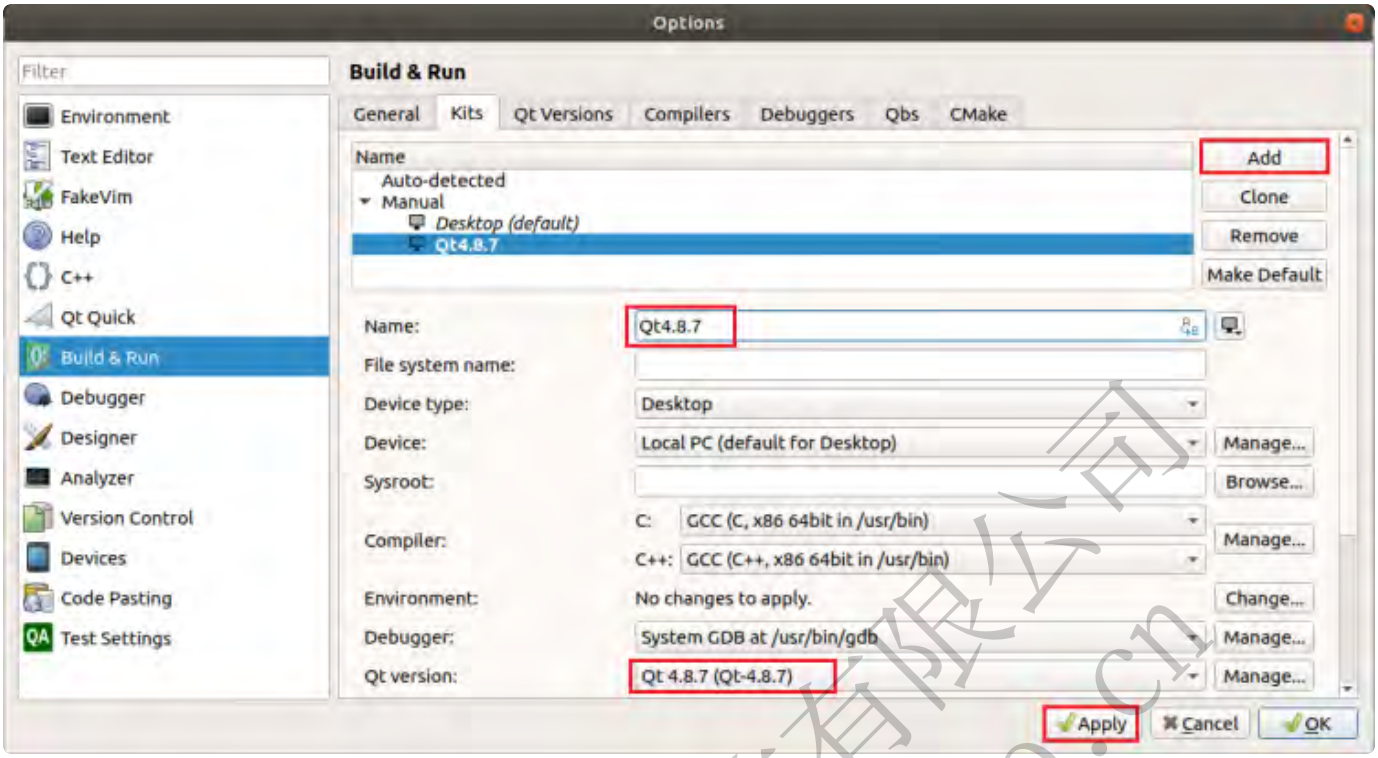
Qt Creator 4.4.1需要添加Qt4.8.7开发套件才可能在Ubuntu18.04 PC端编译、运行和调试Qt代码。打开Qt Creator 4.4.1软件，选择”Tool“->”Options”，在弹出的界面中选择”Build & Run”。



在”Qt Versions”栏点击右边的”Add..”按键，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“按键生效添加的Qt Version。



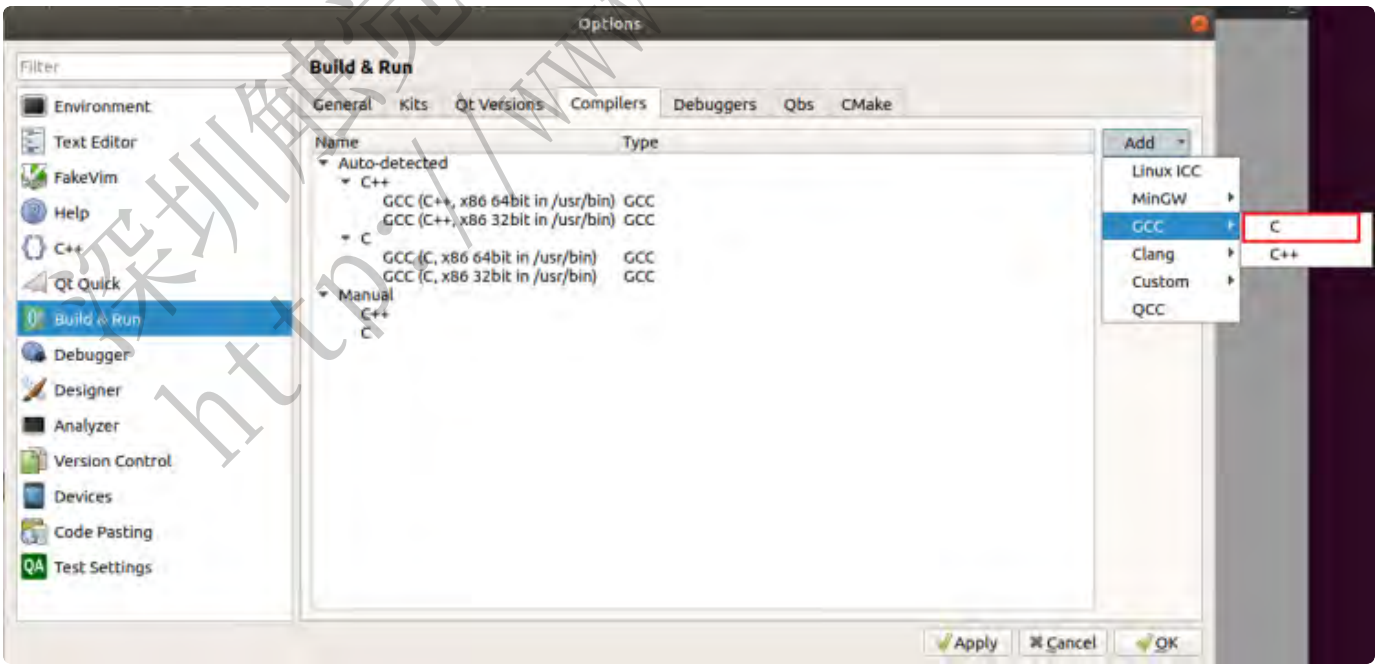
点击进入Kits栏，点击”add“按键新增一项kit，修改名字为”Qt4.8.7“，Qt Version选择步骤2中添加的Qt-4.8.7，点击”Apply“按键生效，点击”OK”按键关闭窗口。



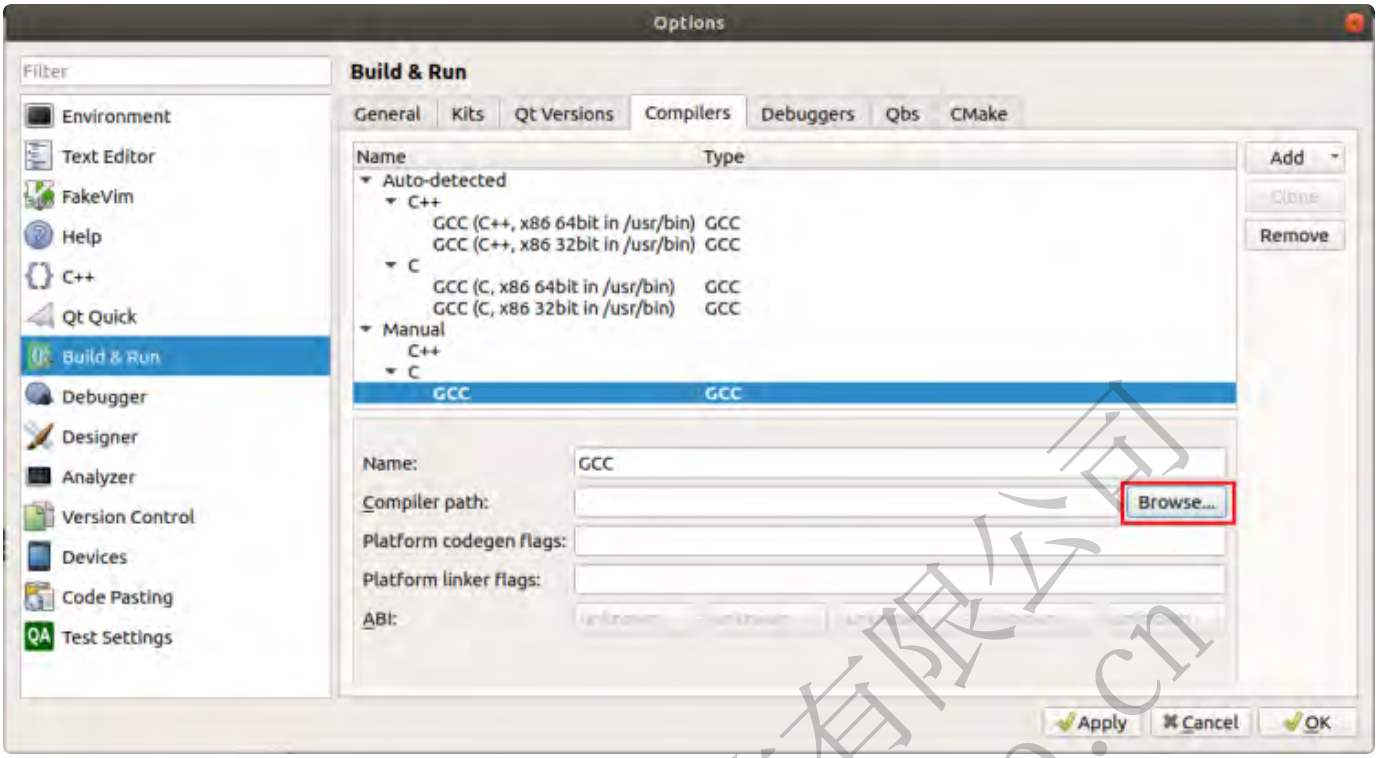
Qt Creator添加Qt4.8.7交叉编译

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。

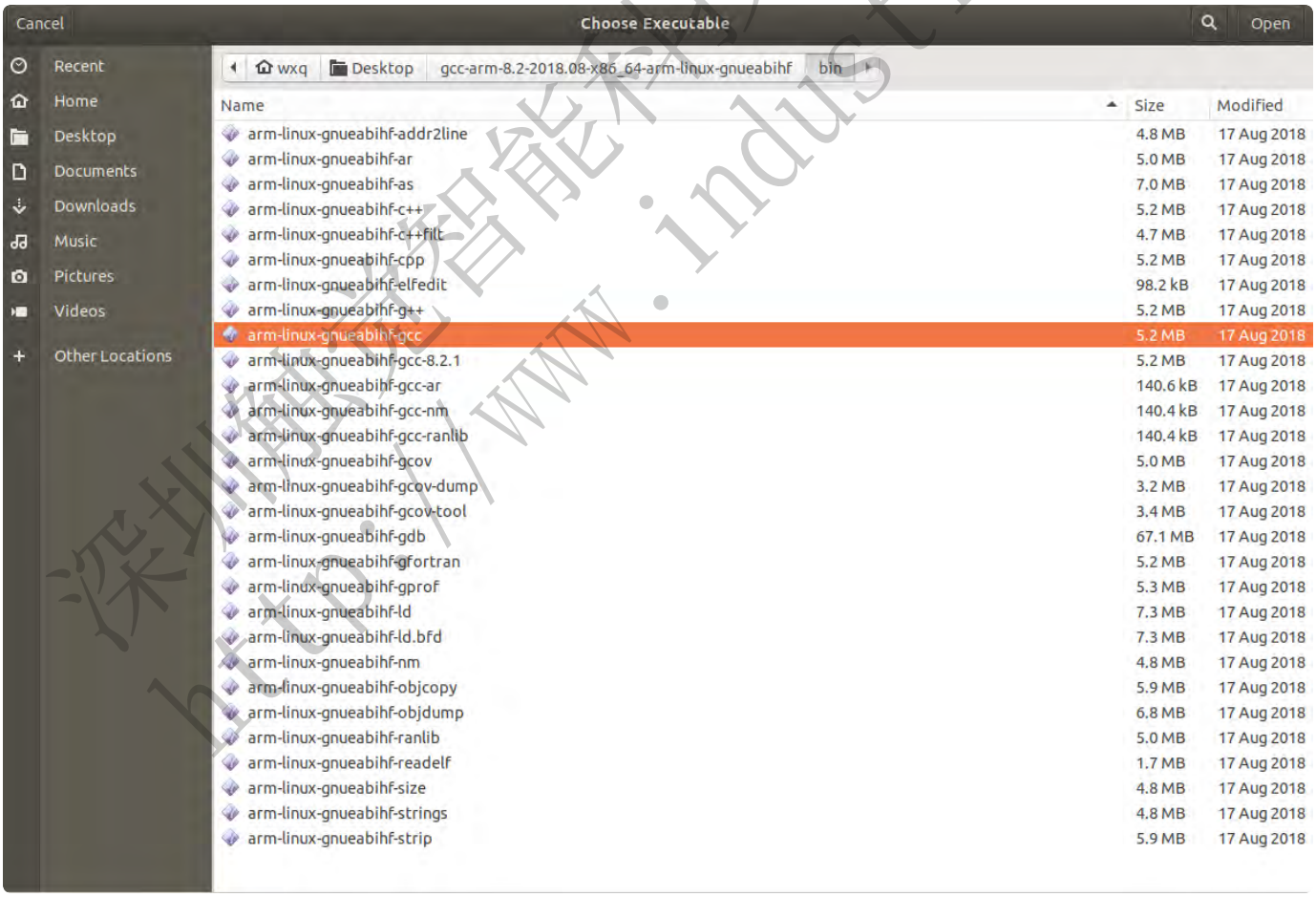
添加gcc在界面的右侧点击Add–>GCC–>C，添加一个GCC编译器。



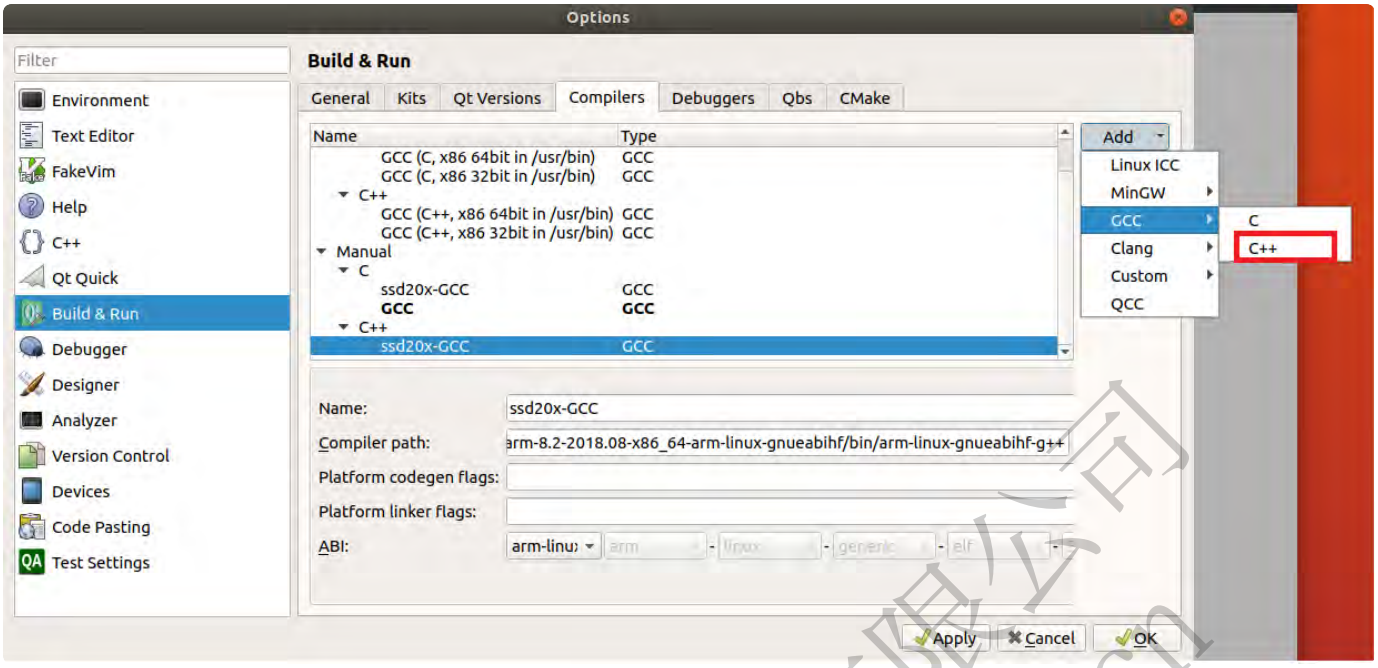
选中新添加的GCC编译器，然后点击下方的“Browse”按键开启文件资源浏览器；



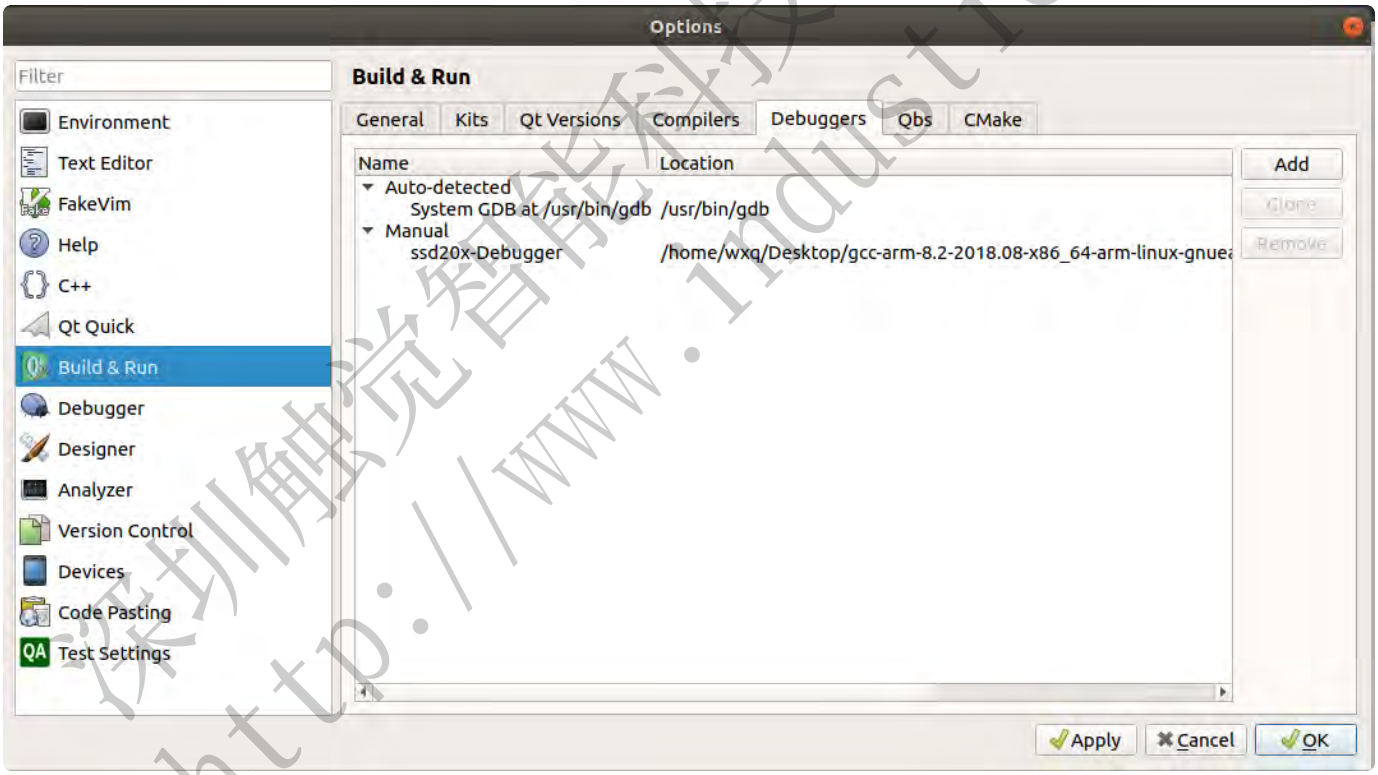
选中gcc编译工具“arm-linux-gnueabihf-gcc”，添加完成点击“Apply”生效设置



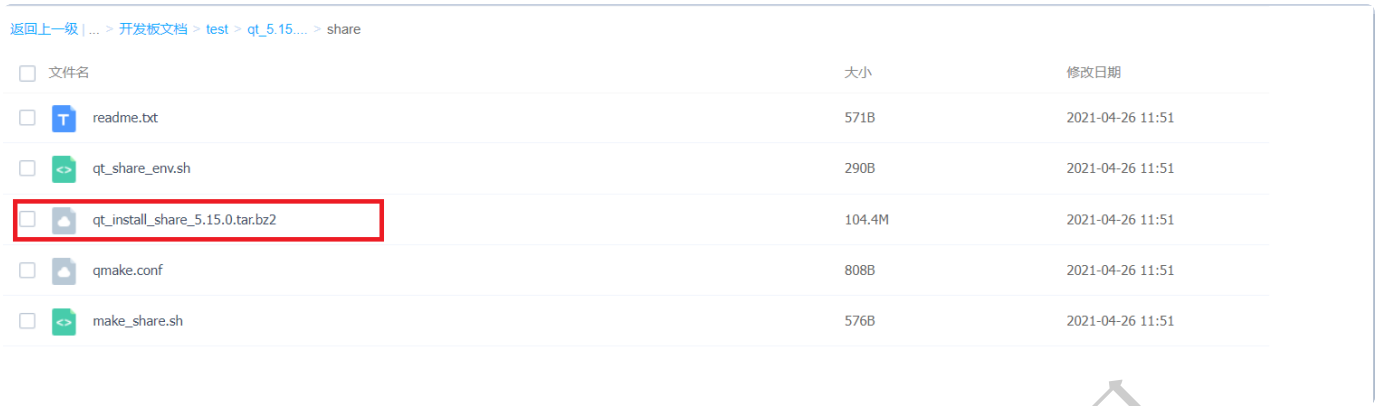
按照GCC的添加方法，添加g++



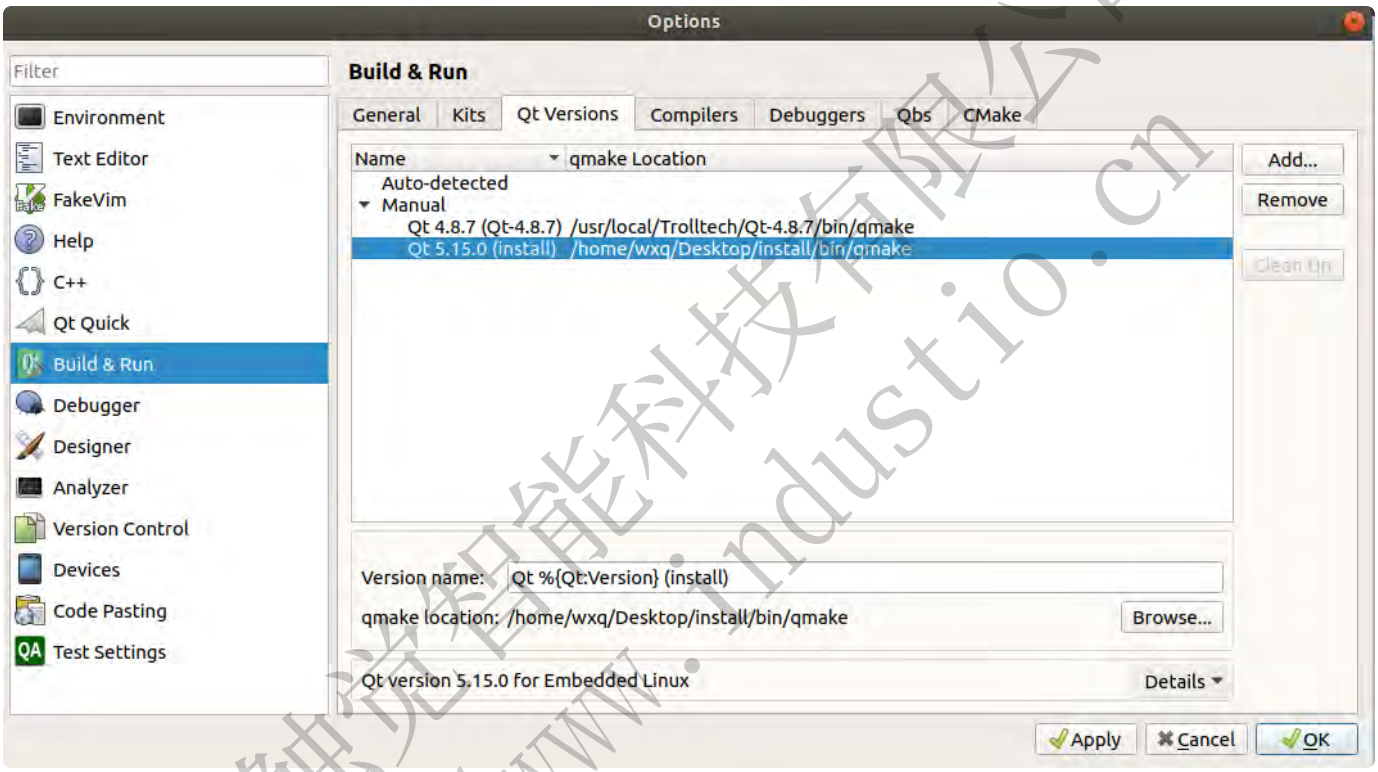
在Debuggers栏点击“Add”按键新增一个Debugger，选择“arm-linux-gnueabihf-gdb”并修改名字为“ssd20x-Debugger”，点击“Apply”按键生效设置。



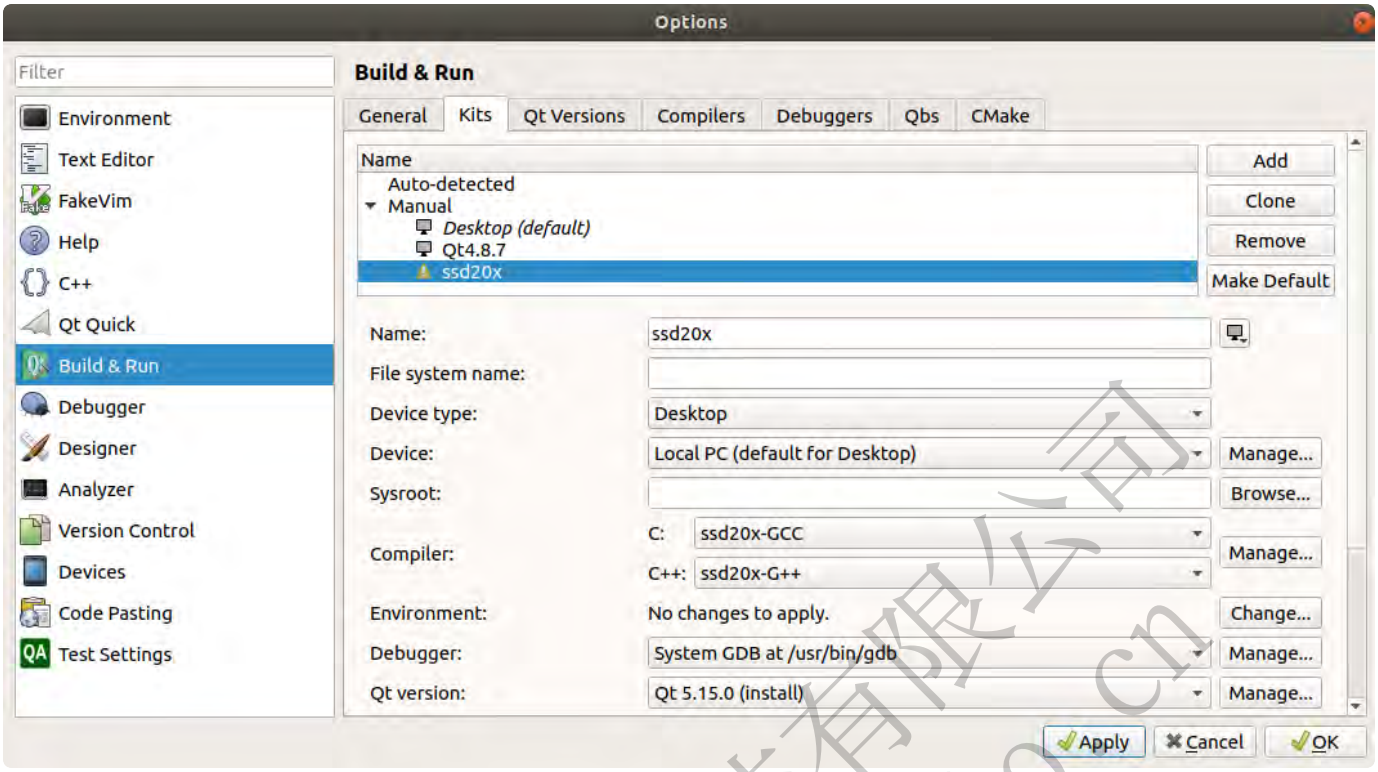
在配置Qt Versions时把网盘中的Qt5.15解压到Ubuntu中



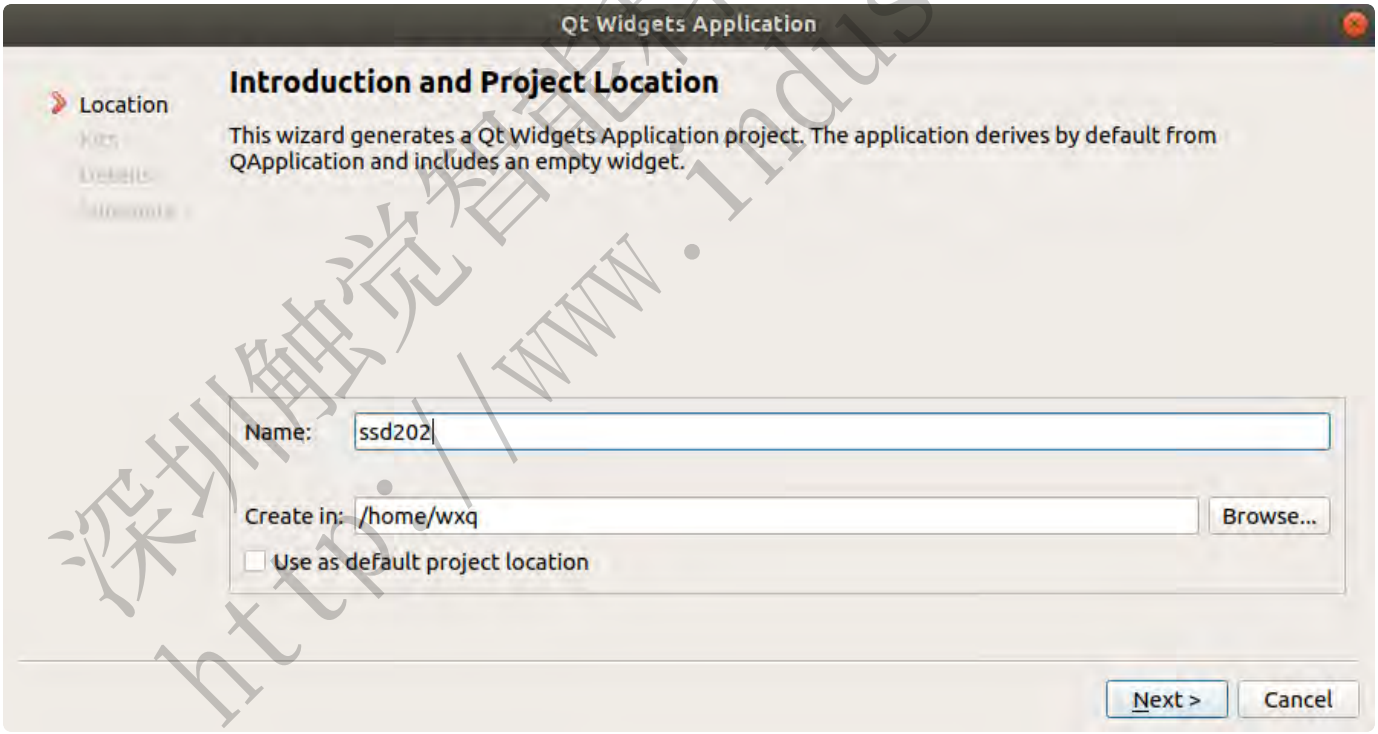
在Qt Versions栏点击“add”打开文件资源浏览器，选择“/install/bin/qmake”，点击“Apply”按键生效设置。



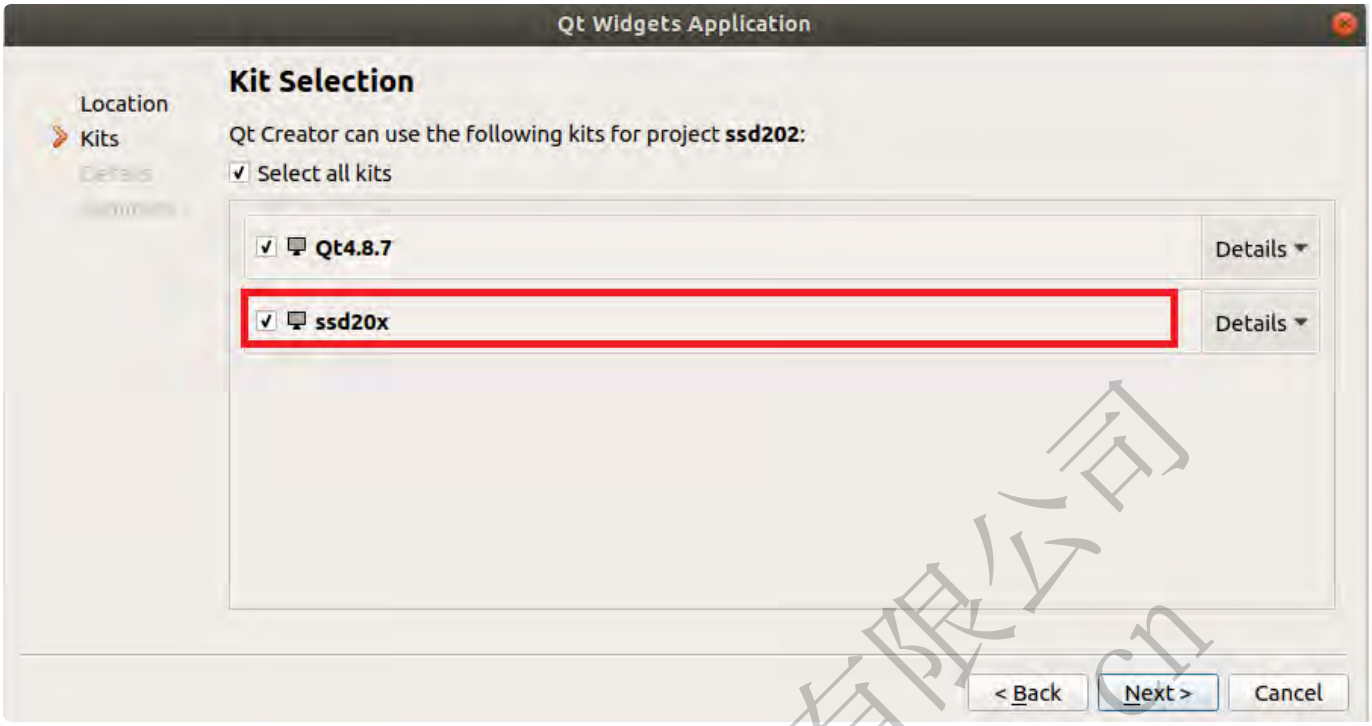
在Kits栏，点击“Add”按键添加一个kit，并修改名字为“ssd20x”，选择Compiler C: ssd20x-GCC；Compiler C++: ssd20x-G++；Qt Version :“Qt5.15.0（install）”，最后点击“Apply”生效配置。



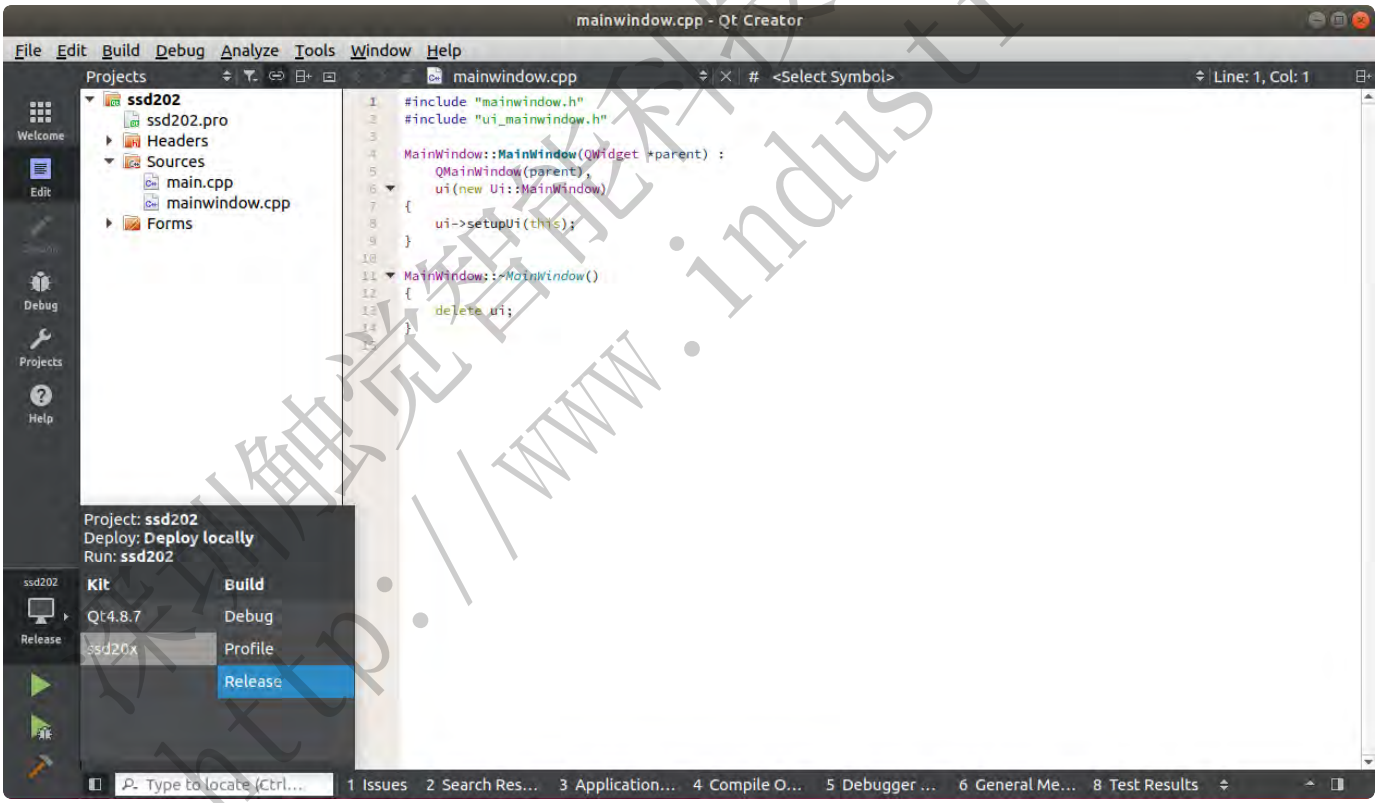
验证Qt Creator交叉编译配置，点击Qt Creator软件的“File–>New File or Project”，新建一个“Qt widget Application”工程。



在“Kit Selection”界面，选择添加的“ssd202x”



在Qt Creator左侧选择“ssd20x->Release”

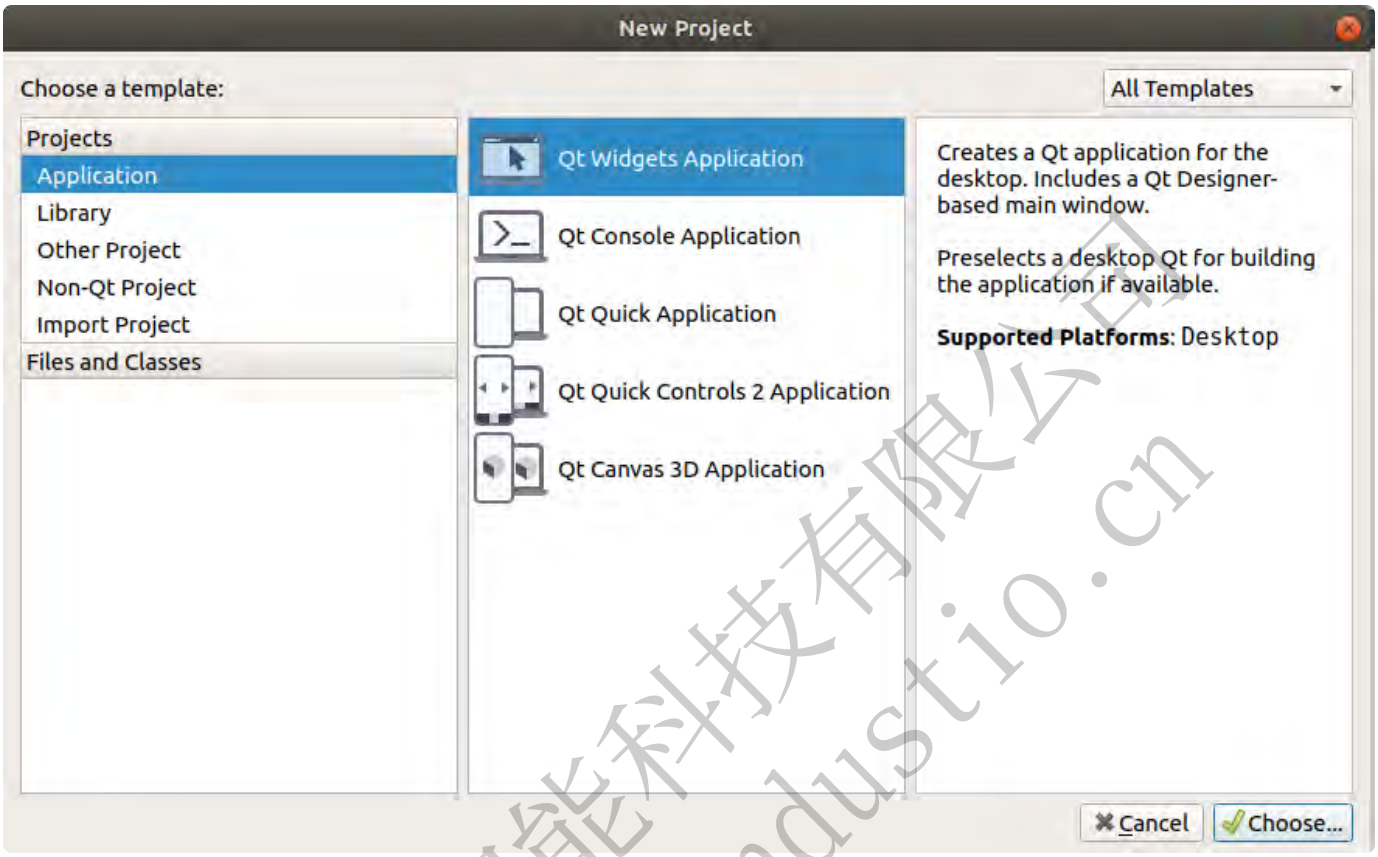


点击”Build”🔨按键编译；正常编译完，在新建工程的“/home/wxq”目录下，产生一个“build-ssd202-ssd20x-Release”文件夹，文件夹中包含编译的中间文件和开发板上运行的可执行文件“ssd202”。

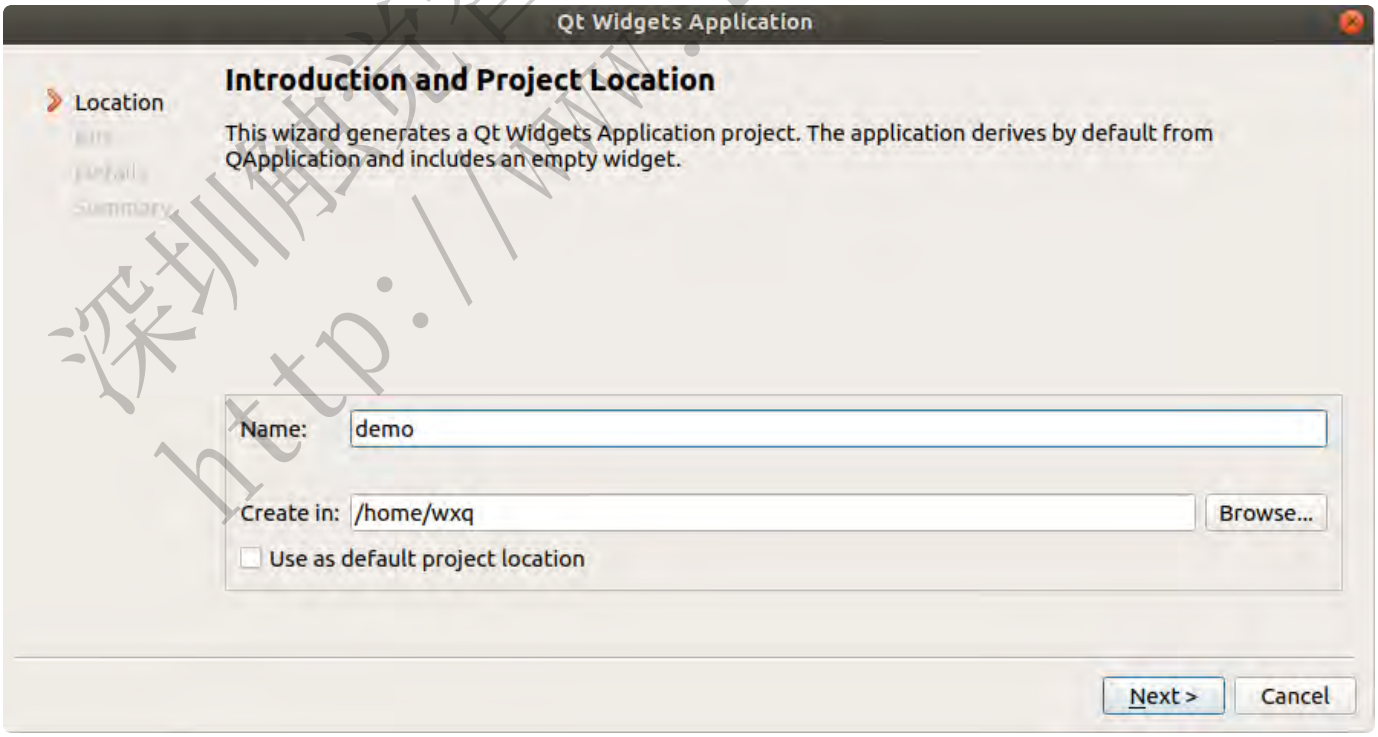
```
wxq@ubuntu:~/build-ssd202-ssd20x-Release$ ls
main.o          Makefile             moc_mainwindow.o  ssd202
mainwindow.o    moc_mainwindow.cpp   moc_predefs.h     ui_mainwindow.h
wxq@ubuntu:~/build-ssd202-ssd20x-Release$
```

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”

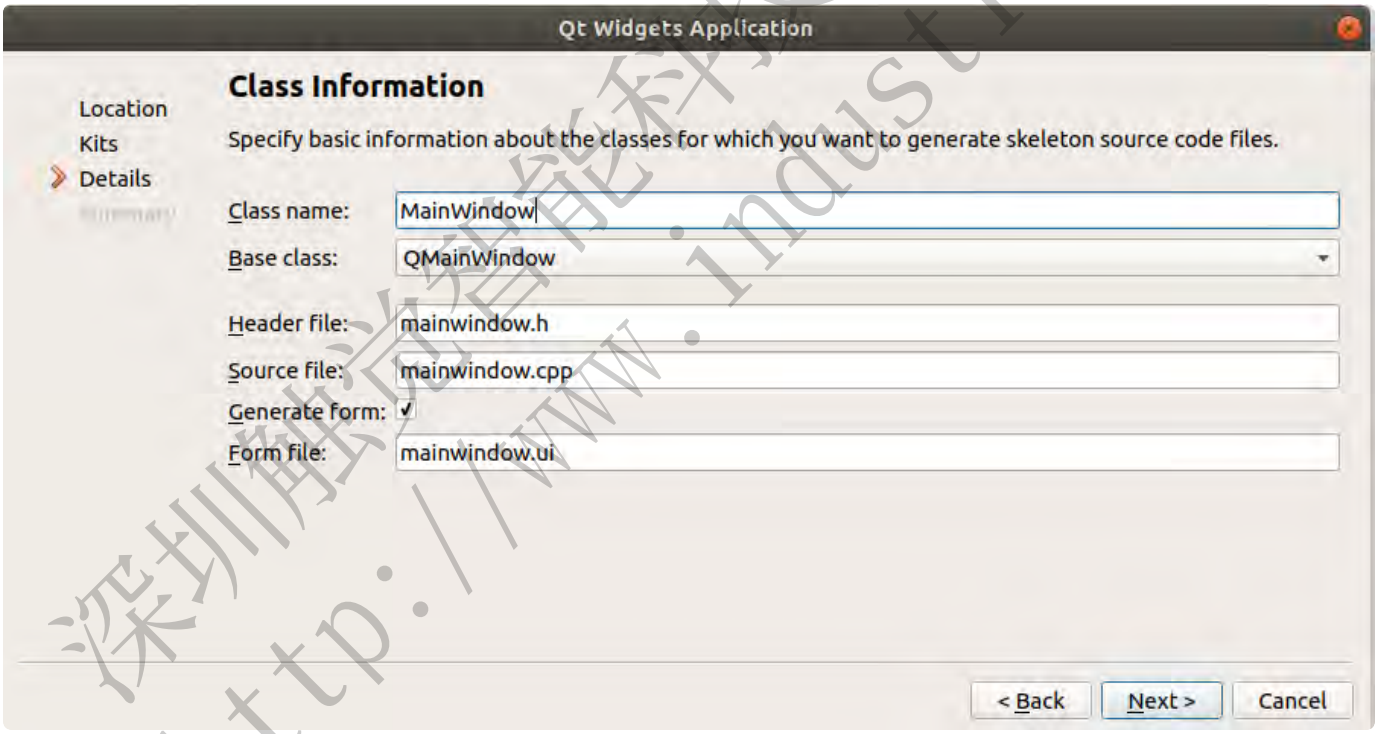


名字可以随便一个

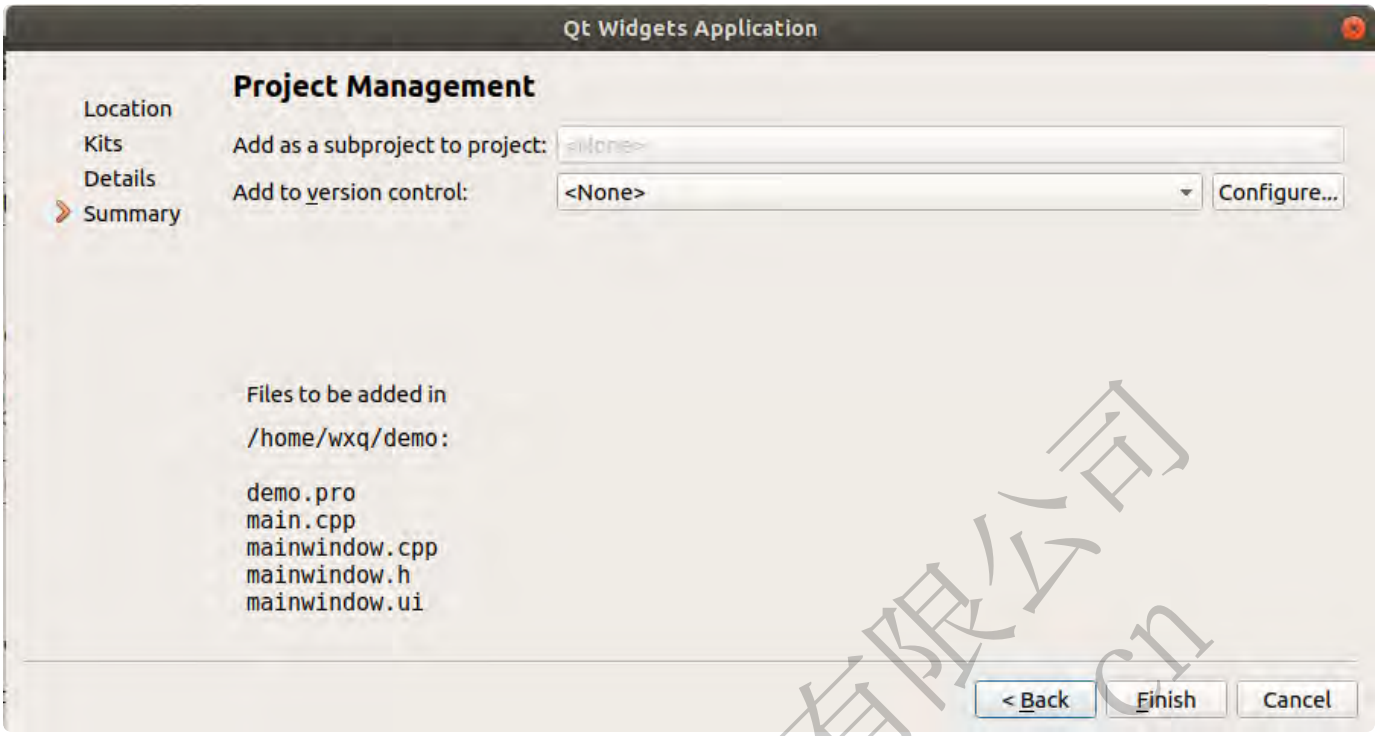


添加ssd20x

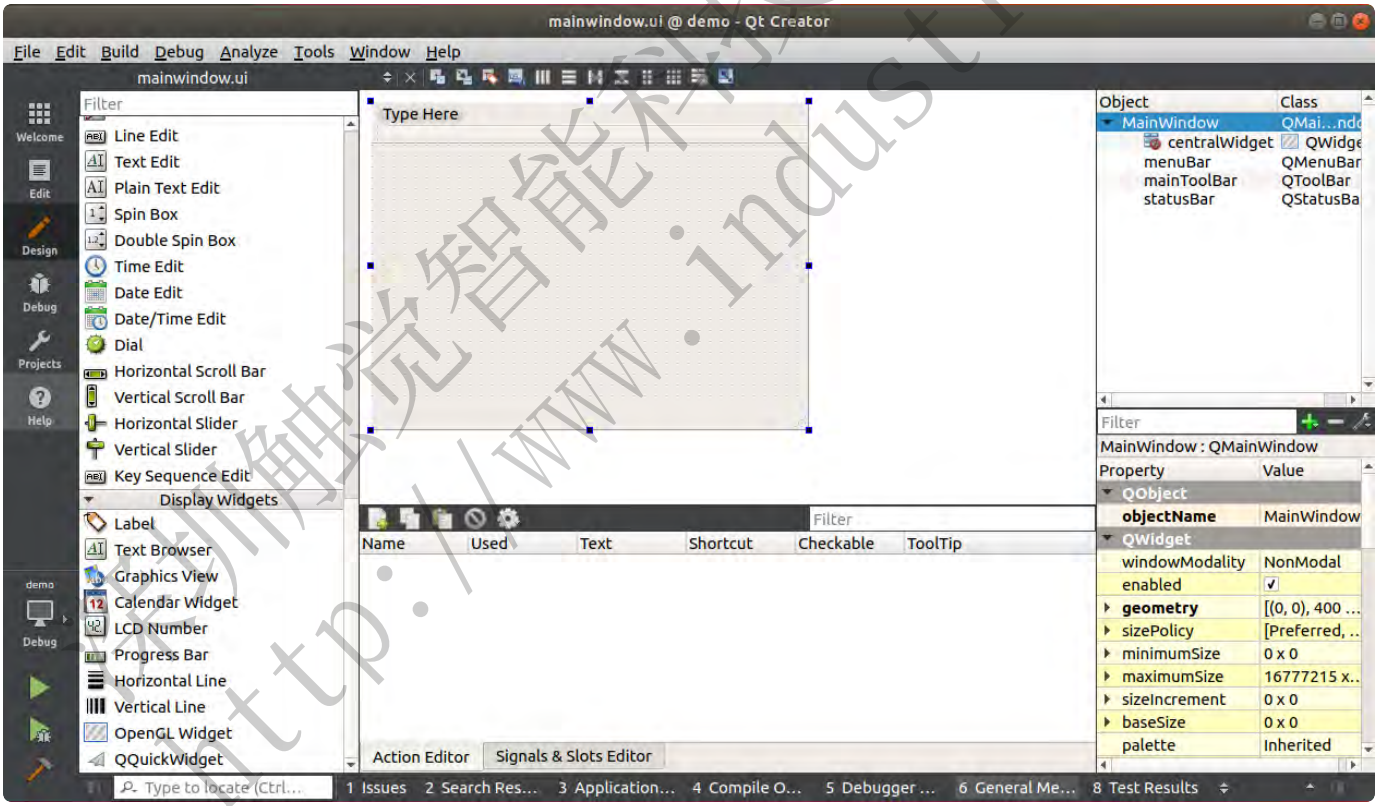
默认即可



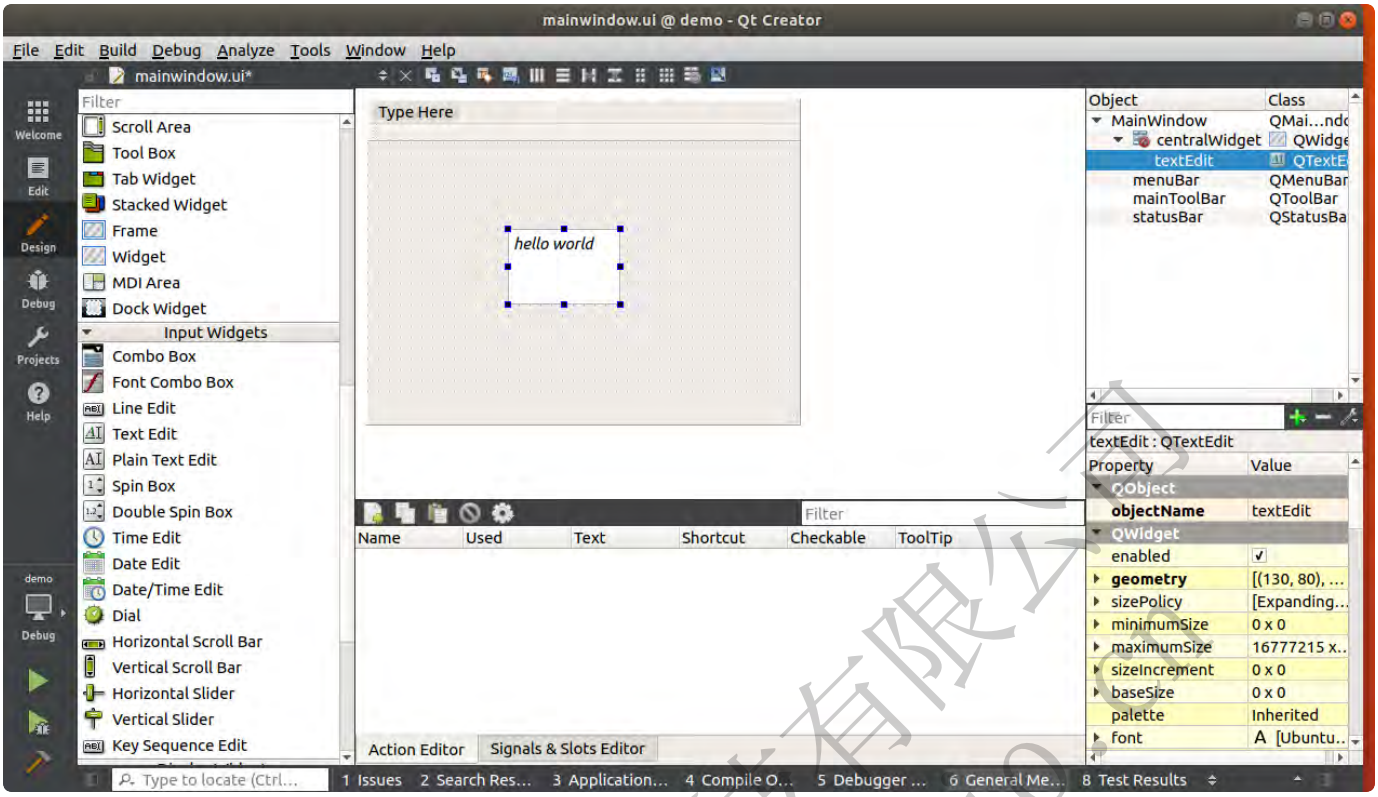
点击“Finish”



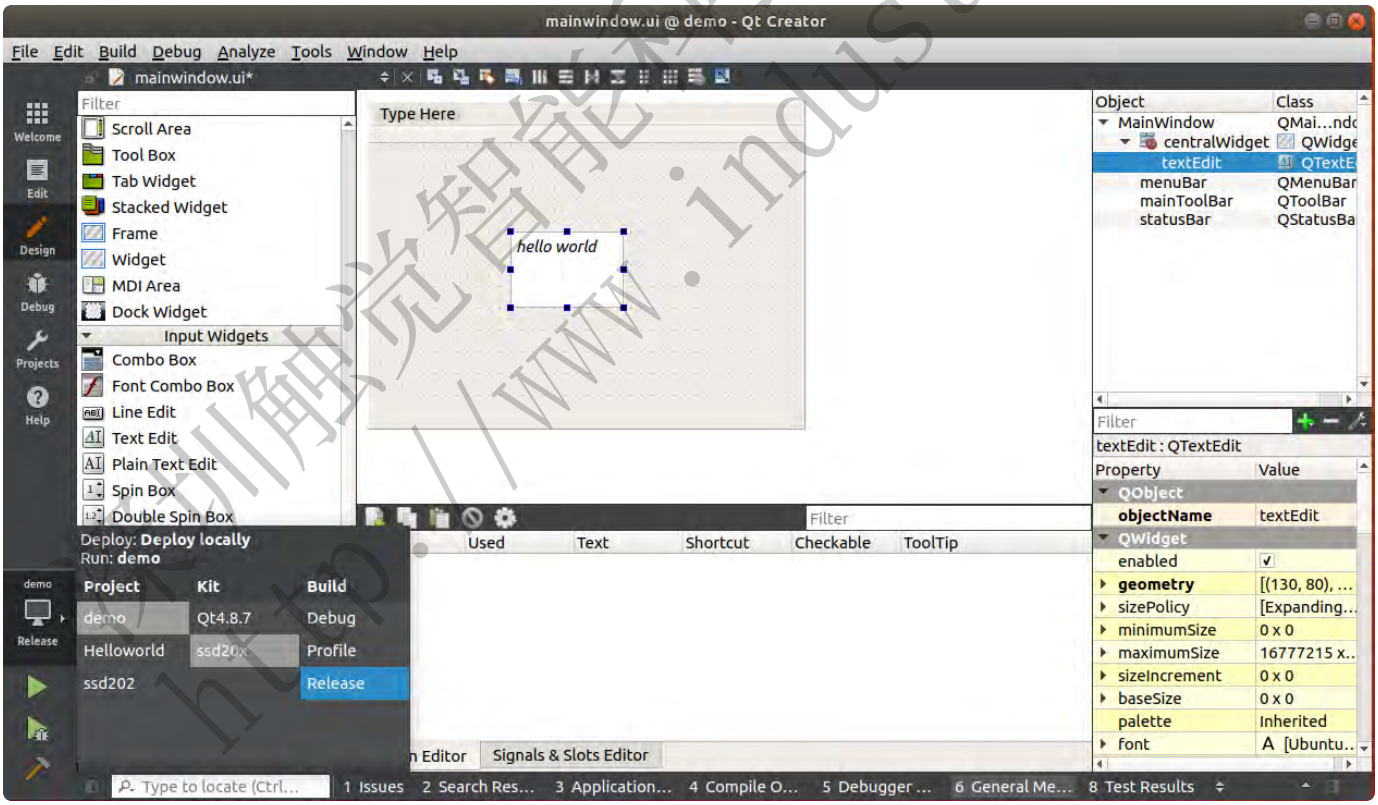
点击“Forms–>mainwindow.ui”



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



debug选择“ssd20x–>Release”



点击”Build”🔨按键编译；生产可执行文件，把文件拷贝到开发中运行