第五章 安装 QT 与测试

本章需要安装 QT 开发工具，以及包括 QT 的动态编译环境。

一 安装 qt 官方的动态库与 IDE

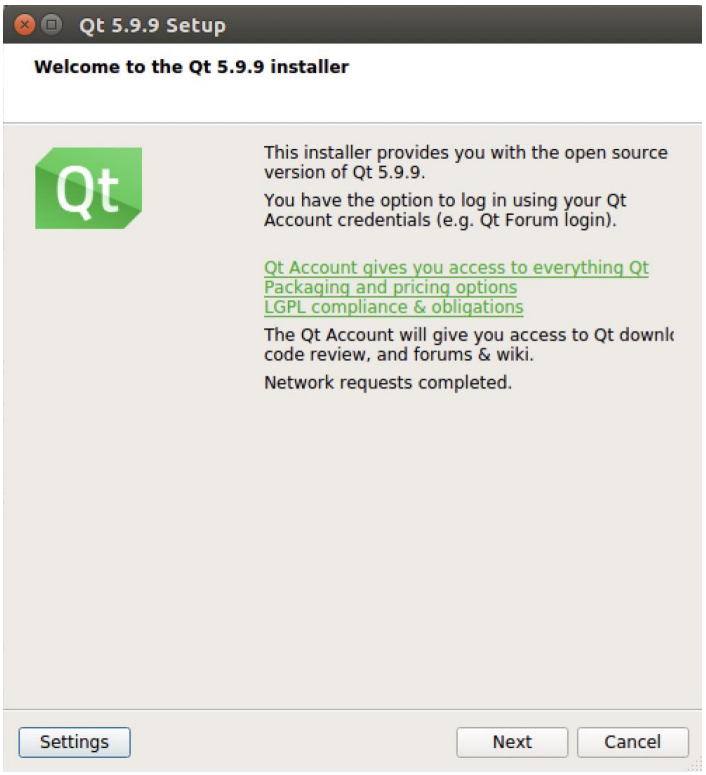
qt-5.15.1 官方没有提供动态库，我们可以临时使用较低的版本 5.9.9，里面包含了 qt-creator。如果是 Mac 系统，则需要下载官方的 **dmg** 格式，直接双击 **dmg** 文件进行安装。

下面对 Linux 系统进行安装介绍。下载 qt-opensource-linux-x64-5.9.9.run，然后对该文件增加执行权限。

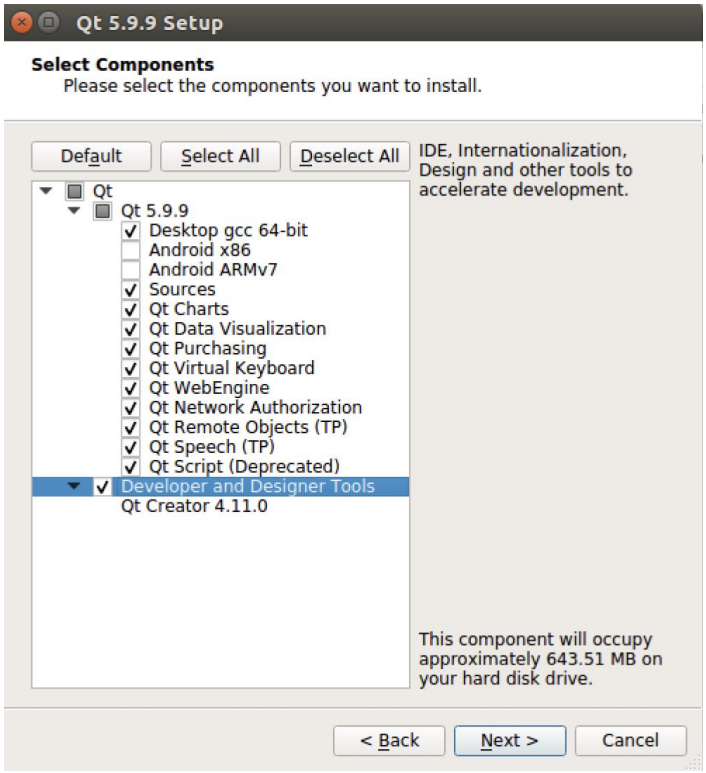
chmod +x qt-opensource-linux-x64-5.9.9.run

运行 qt-opensource-linux-x64-5.9.9.run 文件。

./qt-opensource-linux-x64-5.9.9.run



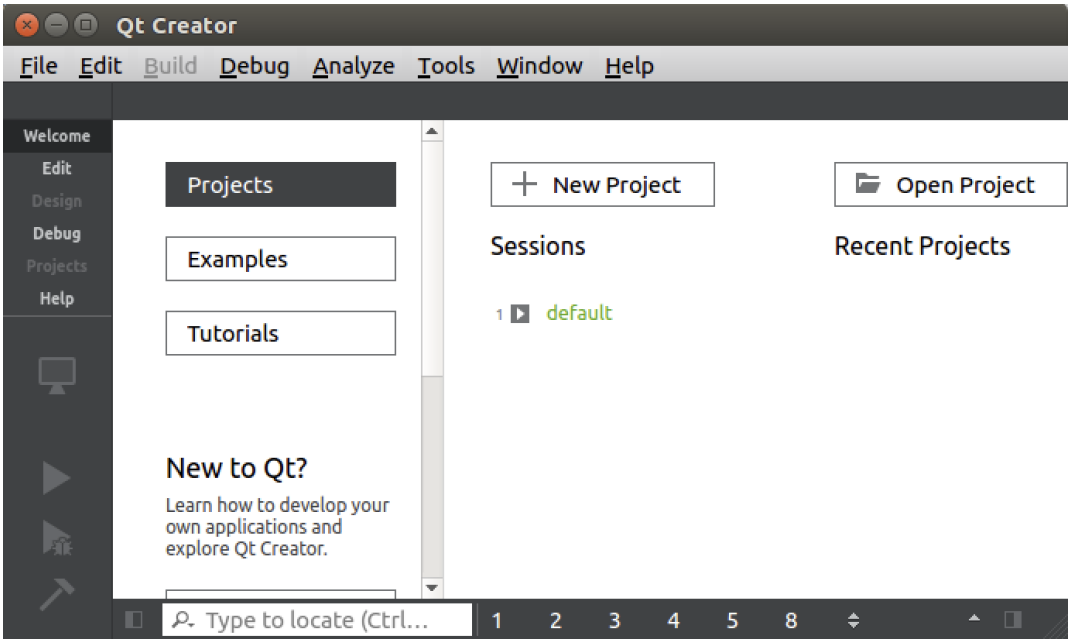
需要登陆 QT 的账号，如果没有，可以注册一个。将安装路径调整为/opt/Qt5.9.9。



因为不使用 android 平台的 sdk，其他的保留。如果硬盘空间不足，并且我们选择的源代码使用 5.15.1 自定义编译则可以忽略 Sources 选项。

二 启动 qt 的 IDE

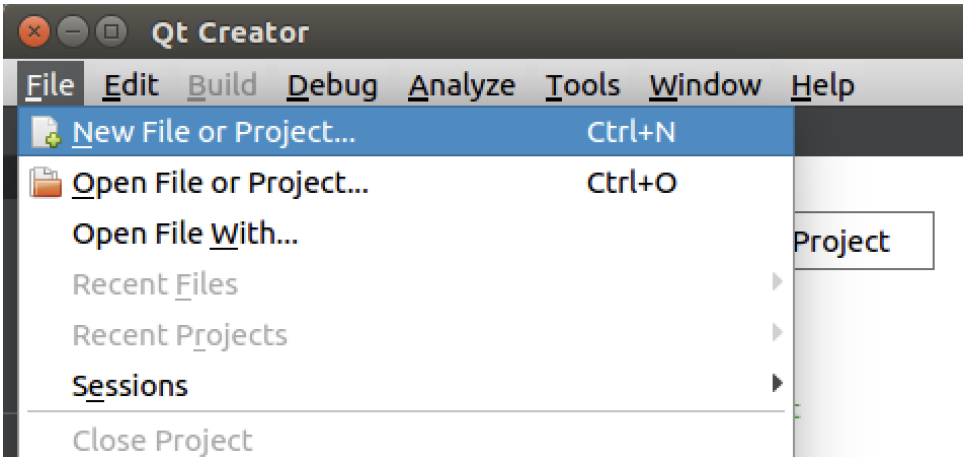
安装完成后，启动 qt-creator 如下图。



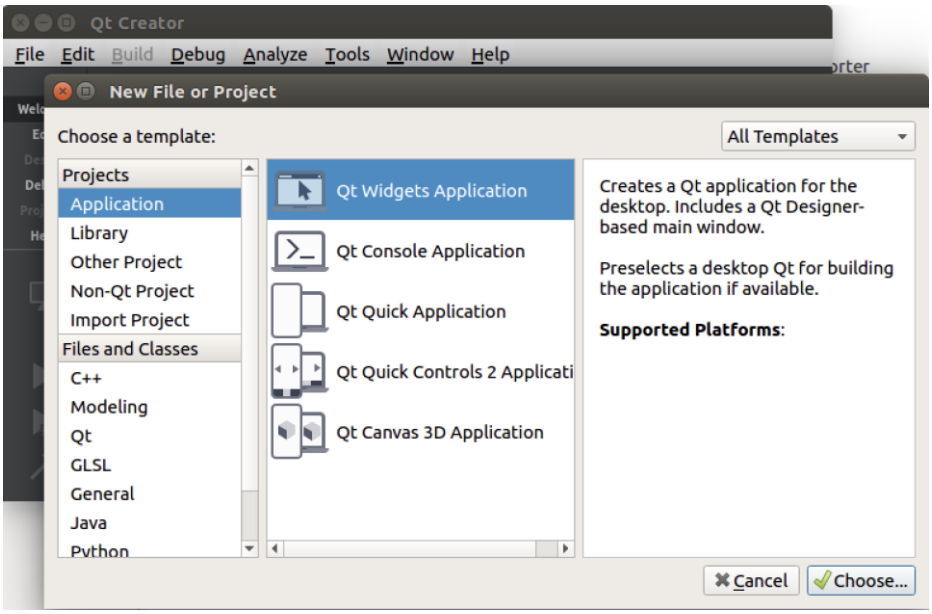
目前我们安装了 qt-creator，同时也安装了 qt 相关的库。这时候是可以开发与编译**动态库**模式的应用。

三 创建空白的应用

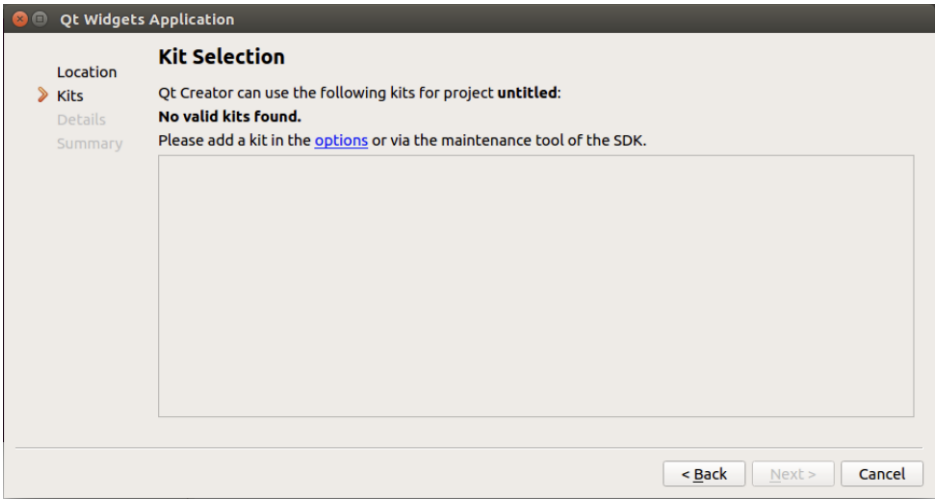
选择菜单“File”菜单中的“New File or Project”创建新的空白项目。



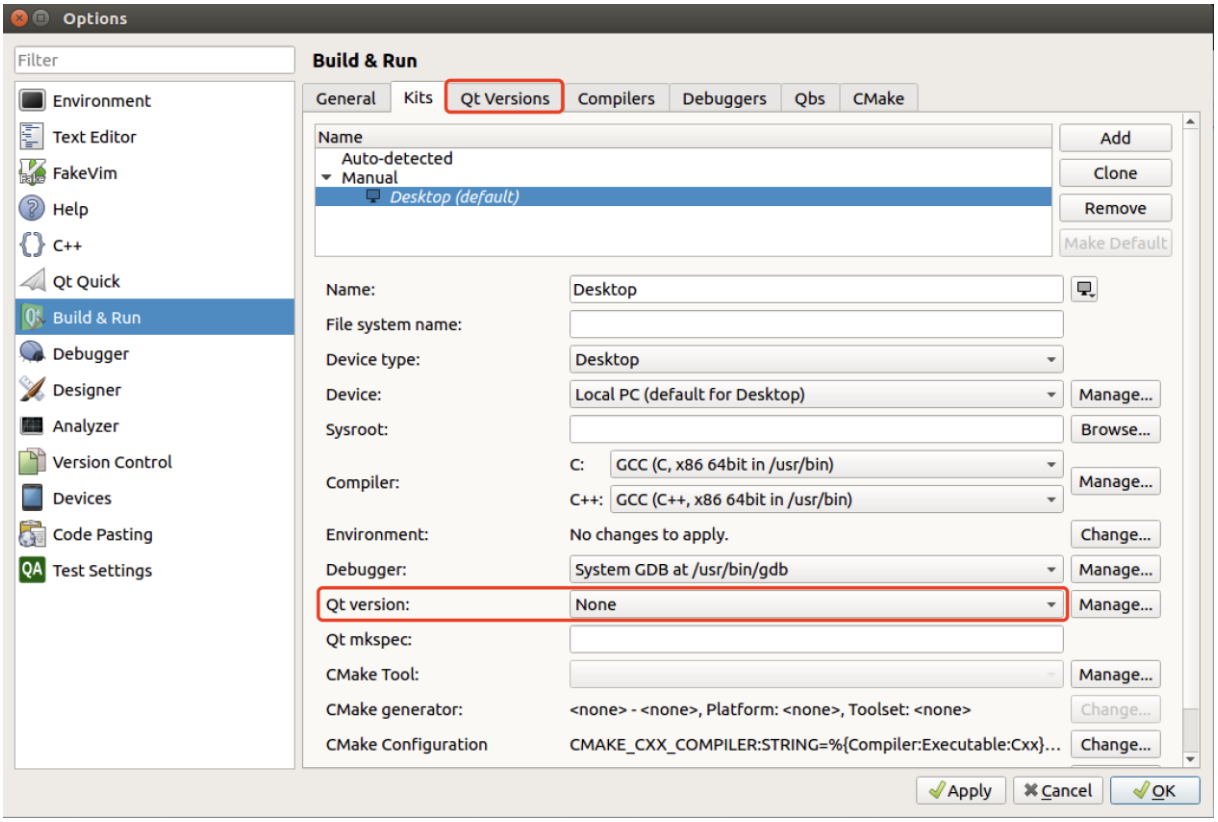
选择创建“Qt Widgets Application”，是用于桌面版本的应用，设置保存到/opt/qt-demo。



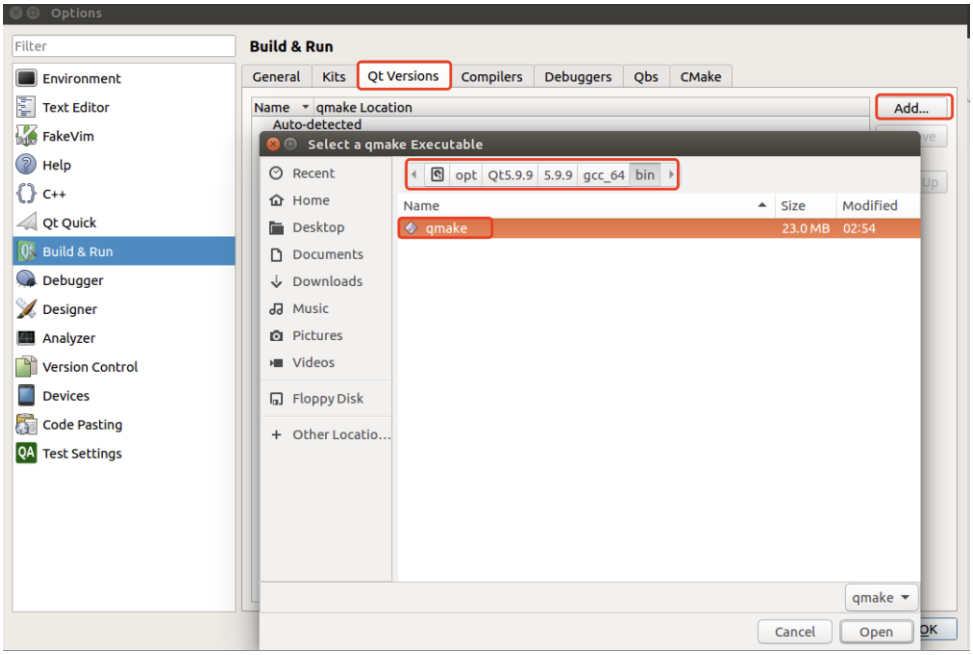
设置 qt 项目所依赖的开发套件。



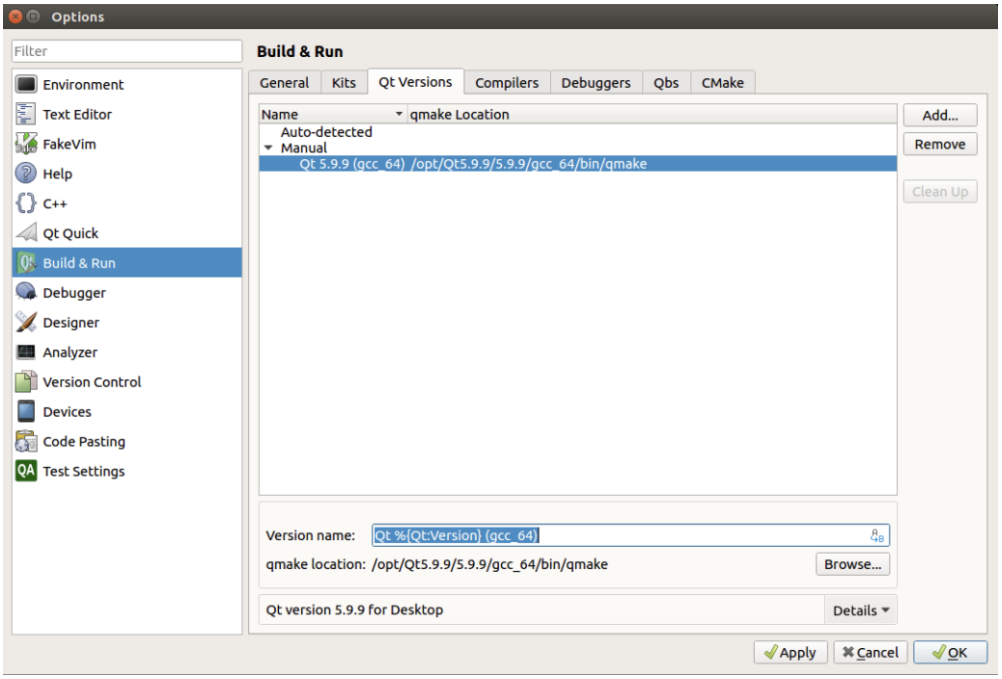
点击窗体里面的选项“options”进行设置。



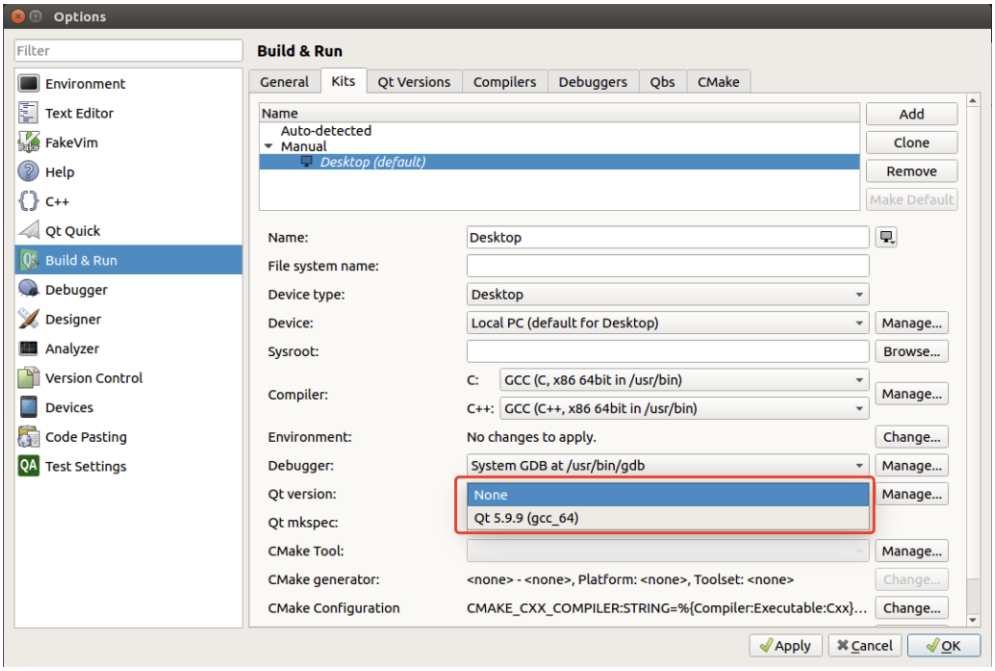
发现“Build & Run”功能的“Kits”页签中，“Qt Version”未找到，需要手动点击“Qt Versions”页签进行设置，找到 qt 编译的 qmake 可执行文件，根据第一步所安装的路径进行添加设置。如下图：



点击“Open”后，如下图：

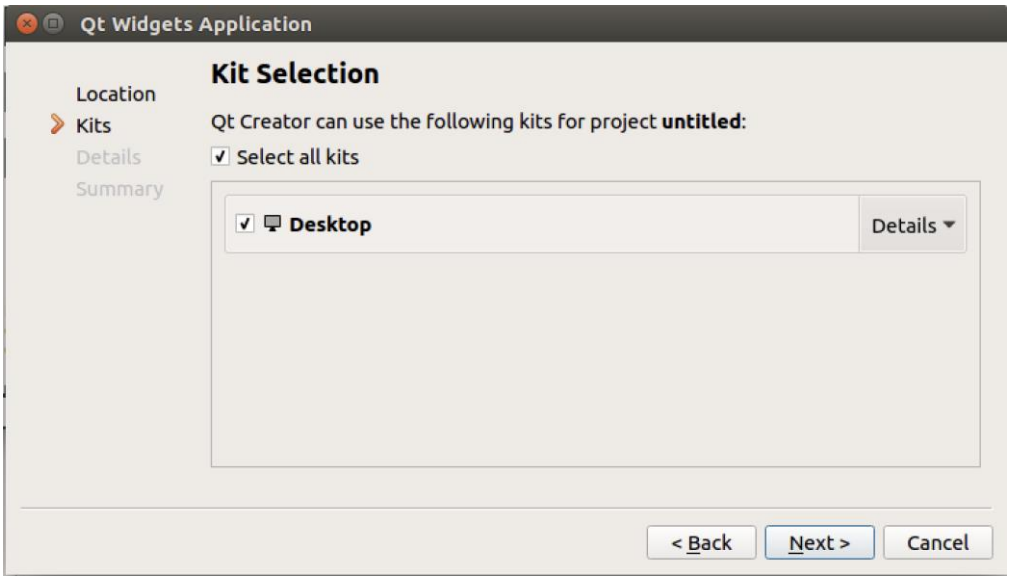


然后切换回“Kits”页签，调整“Qt Version”。



选择“Qt 5.9.9 (gcc_64)”，然后点击“Ok”保存 Kits 的设置。

这时候在 kits 列表中已经出现内容，说明 IDE 的设置正确，显示如下：



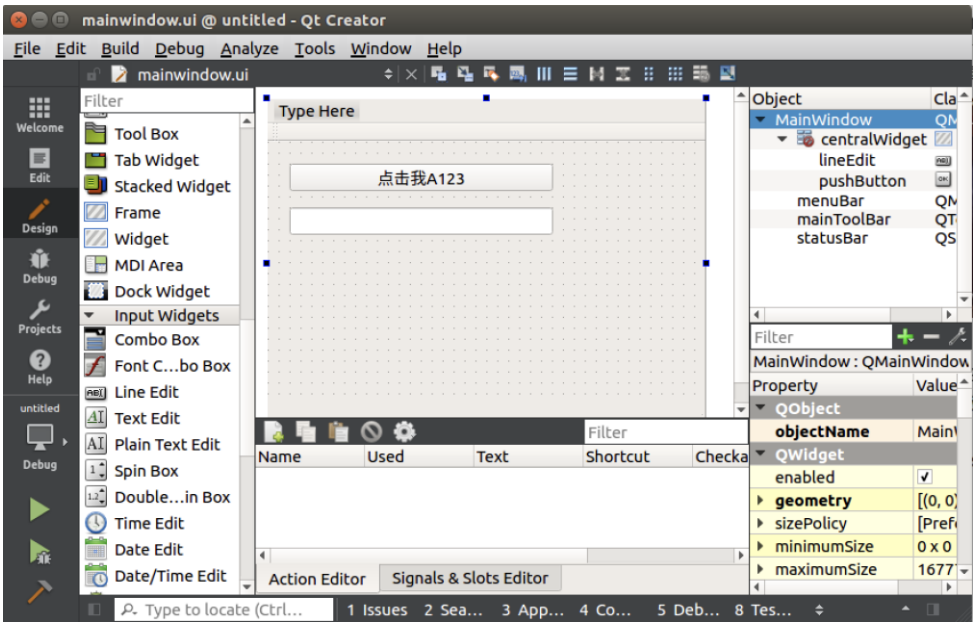
四 测试项目

为测试简单的 Qt 项目能正常编译，在第三步的在空白项目的 ui 中，增加一个 QPushButton，修改显示为“点击我 A123”，再增加一个文本输入框 QLineEdit 控件，主要的目的是在新电脑上测试以下几点：

1. 在新的 Linux 操作系统中是否能正常启动
2. 测试显示中文是否正常

3. 测试输入框是否能正常输入中文

应用的 UI 的控件如下图：



1. 设置 Qt 依赖的库

Qt 默认的动态库是依赖 libGL.so 的，所以需要设置好 GL 库软链接。

sudo ln -s /usr/lib/x86_64-linux-gnu/mesa/libGL.so.1 /usr/lib/x86_64-linux-gnu/libGL.so

将/usr/lib/x86_64-linux-gnu/mesa/libGL.so.1 链接到默认的 lib 所在目录/usr/lib/x86_64-linux-gnu 中。

2. 编译输出

然后编译项目，在 qt-creator 的编译输出中可以看到最终执行的命令是：

g++ -Wl,-rpath,/opt/Qt5.9.9/5.9.9/gcc_64/lib -o untitled main.o mainwindow.o moc_mainwindow.o -L/opt/Qt5.9.9/5.9.9/gcc_64/lib -lQt5Widgets -lQt5Gui -lQt5Core -lGL -lpthread

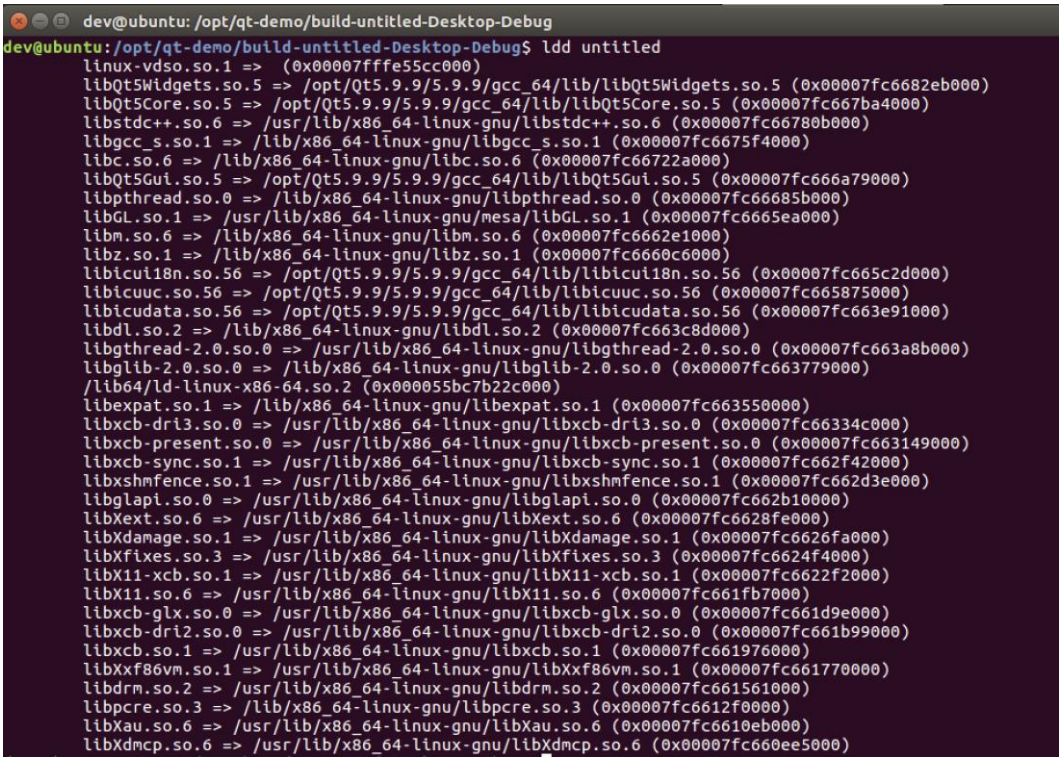
简要说明：

-L 参数：是设置动态库或静态库的路径，如果有多个则需要设置多个，默认的是包括系统默认的 lib 路径。

-l 参数：是用于指定依赖库的名称，比如-lQt5Gui，则说明依赖 libQt5Gui.so。里面因为有-lGL，所以要先设置好在系统默认的 lib 路径中能找到 libGL.so，否则会编译失败。

3. 分析依赖

对编译后的程序进行分析所依赖的动态库，如果使用该方法打包，则需要将依赖的大部分外部库都要打包。



发现依赖的动态库很多，将这些动态库打包同样也很复杂。而最终的目标是实现静态库编译，减少这么多动态库的依赖。

4. 测试验证

在一台全新的 Linux 系统中，将该可执行文件复制过去，可以通过 scp 命令进行复制。

用法 scp 本机文件名 远程用户名@远程主机 ip:远程主机路径

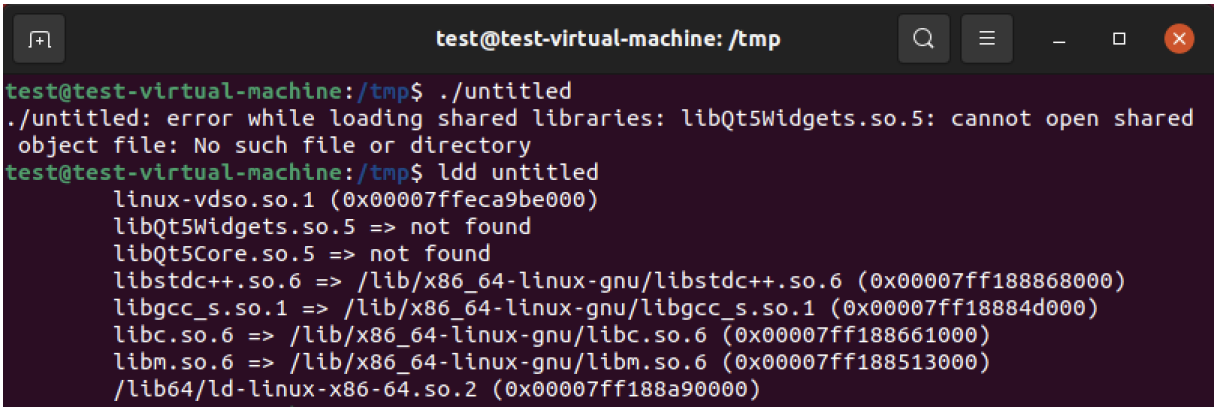
比如执行：

scp untitled test@192.168.131.7:/tmp

如果出现提示连接失败，请确保 2 台机器的网络是相通的以及在目标主机上安装好 openssh-server。

切换到新主机上验证/tmp 目录下的 untitled 文件。

1. 直接运行，提示缺失库。
2. 使用 ldd 查看依赖，发现存在 “not found” ，是因为缺失该动态库。



在新的主机上无法运行导致无法测试，为了解决该依赖的问题，后面的章节将会对 QT 进行静态编译。