

移植 QT5.6 到韦东山 JZ2240 嵌入式开发板

首先感谢 <http://blog.csdn.net/lizuobin2/article/details/52673494> 的博主，应该是韦东山团队的老师，说实话，拿到 JZ2240 的板子很长时间了，每次都有冲动去折腾它，其实初学者不是很喜欢从 0 开始学 linux，我们迫切想知道 linux 是哪样，以及自己拿这块板子可以玩点什么，那么初学者第一件想干且有热情干的事情就是不用懂原理，按照扎实的步骤编译 uboot、linux 内核、构建文件系统、移植 qt 程序到我们的开发板上！可惜啊，每次我的热情都被配套的使用手册 pdf、论坛的指导贴给迅速磨灭，初学者很看重入门这个阶段，不过说实话，韦东山的视频确实不错，耐心下来确实可以学到不少东西！机缘巧合这几天我的 zynq 板子（米联客的技术支持是目前线上最 nb 的，没有之一，强烈推荐）被我误擦错电源烧坏，送回去维修了，再加上正好韦东山出了新一期的教程加强版，最关键的是配套开发环境 Ubuntu 终于换成了 16.04（虽然下载有问题），废话不说了，开始老老实实重新优化上面博文的文章，再次感谢博文文章的指导。

1、开发环境：ubuntu 16.04 (<https://eyun.baidu.com/s/3miEaDza#sharelink/path=%2F>)

在全部文件>百度网盘分享的所有文件>005_ARM 裸机 1 期加强版里面，下载解压有问题，这个 ubuntu-16.04.2-x64-100ask-s002.vmdk 破坏了，需要和团队老师索要，我不知道为什么反正之前可以进入图形界面，现在下载的开机也不能登录图形界面，需要自己网上自己想办法搞定，吓得我赶紧把虚拟机备份了下，希望韦东山团队赶紧上传个可以用的，环境很重要！

2、编译器：

友善的 4.4.3 版本的交叉编译工具：arm-linux-gcc-4.4.3.tar.gz

链接：<http://pan.baidu.com/s/1nvJF8ud> 密码：oi57

将现在原有的交叉编译器路径替换为我们新解压的交叉编译器：

a. 设置 root 密码：sudo passwd

b. sudo gedit /etc/bash.bashrc

在最后加入

export JAVA_HOME=/work/qt/FriendlyARM-gcc-4.4.3/bin（这个根据你自己将 arm-linux-gcc-4.4.3.tar.gz 解压到哪边，千万不要直接复制）

#export JAVA_HOME=/work/tools/gcc-3.4.5-glibc-2.3.6/bin（//这是之前的 3.4.5，可以#注释掉，留着以后方便切换）

export PATH=\$JAVA_HOME/:\$PATH

c. source /etc/bash.bashrc

建议在 book 用户和 root 用户下面都用 arm-linux-gcc -v 命令试下，确保是 4.4.3 交叉编译环境！如果不放心，重启下电脑！

3.重新编译 uboot 2012.04.01（测试也可以使用最新的 U-Boot 1.1.6 enable Ethernet alltime）

Uboot 2012.04.01 打好补丁的源码：（下载以后使用务必在 linux 环境解压，不然就会出错，因为 windows 环境下 xx.h 和 XX.h 是默认覆盖的）

下载链接：<https://github.com/lizuobin/uboot-2012.04.01-jz2440.git>

编译步骤：

make smdk2410_config

make

uboot 成功如下：

```

U-Boot 2012.04.01 (Dec 25 2017 - 22:57:07)

CPUID: 32440001
FCLK:      400 MHz
HCLK:      100 MHz
PCLK:      50 MHz
DRAM: 64 MiB
WARNING: Caches not enabled
Flash: 0 KB
NAND: 256 MiB
In: serial
Out: serial
Err: serial
Net: dm9000
Hit any key to stop autoboot: 0

```

将 u-boot 源码下 tools 目录里的 mkimage 工具复制到/usr/bin 目录下去，这样在编译内核时“make ulmage”才会成功。

4.重新编译内核（不要直接使用下载的 ulmage 4.3，触摸无用）

内核 3.4.2 源码：

<https://github.com/lizuobin/linux-3.4.2-jz2440.git>

编译步骤：

cp config_ok .config

make ulmage

可能遇到的问题：

a.Can't use 'defined(@array)' (Maybe you should just omit the defined()?) at kernel/timeconst.pl line 373.

```

} else {
    print "\n);\n";
    $hz += 0;                                # Force to number
    if ($hz < 1) {
        die "Usage: $0 HZ\n";
    }

    @val = @{$scanned_values{$hz}};
    if (!defined(@val)) {
        @val = compute_values($hz);
    }
    output($hz, @val);
}
exit 0;

```

去掉 defined

b. 查看 xxx/linux-3.4.2-jz2440-master/include/linux 下的 input.h 里面的 EV_VERSION
查看 /usr/include/linux 下的 input.h 里面的 EV_VERSION

确保两者的值是一样的 !不然会出现“selected device is not a touchscreen I understand”
错误，此时触摸屏触摸没有反应，这是由于内核和编译器的一个宏定义不一致导致的！

c. make menuconfig 时可能出现

Makefile:416: *** mixed implicit and normal rules: deprecated syntax

Makefile:1449: *** mixed implicit and normal rules: deprecated syntax

make: *** No rule to make target 'menuconfig'. Stop.

解决方法:

将顶层 Makefile 中的:

config %config: scripts_basic outputmakefile FORCE

改为:

%config: scripts_basic outputmakefile FORCE

将顶层 Makefile 中的:

/ %/: prepare scripts FORCE

改为:

%/: prepare scripts FORCE

测试生成的内核时记得在 uboot 里 : set bootargs console=ttySAC0,115200 (后面是 nfs 启动还是 yaffs 启动用户自己配置), 不然内核串口显示出来是乱码。

以我自己的 nfs 启动为例我的设置如下 : (后面讲 NFS 再具体解释)

```
set      bootargs      noinitrd      root=/dev/nfs      console=ttySAC0,115200
nfsroot=192.168.1.2:/work/my_rootfs
ip=192.168.1.3:192.168.1.2:192.168.1.1:255.255.255.0::eth0:off init=/linuxrc
```

编译成功的内核如下 :

```
NAND read: device 0 offset 0x600000, size 0x400000
4194304 bytes read: OK
## Booting kernel from Legacy Image at 30007fc0 ...
Image Name: Linux-3.4.2
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 2305960 Bytes = 2.2 MiB
Load Address: 30008000
Entry Point: 30008000
Verifying Checksum ... OK
XIP Kernel Image ... OK
OK
Using machid 0x16a from environment

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
Booting Linux on physical CPU 0
Linux version 3.4.2 (root@book-virtual-machine) (gcc version 4.4.3 (ctng-1.6.1))
CPU: ARM920T [41129200] revision 0 (ARMv4T), cr=c0007177
CPU: VIVT data cache, VIVT instruction cache
Machine: SMDK2440
Memory policy: ECC disabled, Data cache writeback
CPU S3C2440A (id 0x32440001)
S3C24XX Clocks, Copyright 2004 Simtec Electronics
S3C244X: core 400.000 MHz, memory 100.000 MHz, peripheral 50.000 MHz
CLOCK: Slow mode (1.500 MHz), fast, MPLL on, UPLL on
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 16256
```

5. 制作文件系统

按照之前博主写的就行, 省事的话可以下载之前博主已经做好的。

链接 : <http://pan.baidu.com/s/1skLtj3J> 密码 : lbo2

该文件系统没有打包成镜像, 请使用 nfs 启动, 解压之后在 dev 目录创建 console null

设备节点，此外，/etc/profile 环境变量中的参数请修改成符合自己开发板的参数，默认为 3.5 寸（我的是 4.3 寸，该为 480 272，还有确定该文件系统里面的 my_rootfs/usr/local/Qt5.6/lib/fonts 里面是 有没有中文字体 DroidSansFallback.ttf 的，如果想测试配套的 qt 程序，需要加入博主提供的字体，不然会出错!!!!，我下载的是编译好的 Qt 链接：<http://pan.baidu.com/s/1pLCznr9> 密码：l6pz，里面没有那个中文字体，调了我快 1 小时。。。。。。。。）。

自己制作文件系统步骤：

Busybox 源码下载：

busybox-1.22.1.tar.bz2 .链接：<http://pan.baidu.com/s/1cee6Cl> .密码：lv81

Step1、解压

```
tar jxvf busybox-1.22.1.tar.bz2
```

Step 2、配置

```
make menuconfig
```

```
Busybox Settings ->
```

```
general configuration ->
```

```
[*] don't use /usr //选中它 否则会破坏虚拟机
```

```
build options->
```

```
cross compiler prefix = arm-linux- //选择交叉编译工具
```

```
installtion options->
```

```
busybox installation prefix = /work/my_rootfs //指定安装路径
```

step 3、编译&安装

```
mkdir -p /work/my_rootfs //安装路径
```

```
make
```

```
make install
```

```
cd /work/my_rootfs
```

```
ls //查看是否安装成功
```

编译安装完毕之后，我们的文件系统就生成在 /work/my_roofs 目录下了

Step4、创建 /etc/inittab

```
mkdir -p /work/my_rootfs/etc
```

```
gedit /work/my_rootfs/etc/inittab
```

输入以下内容：

```
# /etc/inittab
```

```
# 启动脚本/etc/init.d/rcS
```

```
::sysinit:/etc/init.d/rcS
```

```
# 启动 shell
```

```
::askfirst:/bin/sh
```

```
# 重启关机前 卸载文件系统
```

```
::ctrlaltdel:/sbin/reboot
```

```
::shutdown:/bin/umount -a -r
```

Step5、创建 /etc/init.d/rcS

```
gedit /etc/init.d/rcS
```

输入以下内容：

```
# 这是一个脚本文件，用/bin/sh 解析
```

```
#!/bin/sh
```

```
# 挂载文件系统
mount -a
# 使用内存文件系统
mkdir /dev/pts
mount -t devpts devpts /dev/pts
echo /sbin/mdev > /proc/sys/kernel/hotplug
mdev -s
# 设置 IP
#/sbin/ifconfig eth0 192.168.1.3    #nfs 不需要
# 挂载 /etc/fstab 中的文件系统
mount -a
exec /etc/rc.local
```

Step6、创建 rc.local

```
gedit /work/my_rootfs/etc/rc.local
```

输入以下内容：

```
#!/bin/sh
./etc/profile ////注意.后边有个空格！
```

Step7、创建/etc/fstab

```
gedit /work/my_rootfs/etc/fstab
```

输入以下内容：

# device	mount-point	type	options	dump	fsck	order
proc	/proc	proc	defaults	0	0	
sysfs	/sys	sysfs	defaults	0	0	
tmpfs	/tmp	tmpfs	defaults	0	0	
tmpfs	/dev	tmpfs	defaults	0	0	

Step8、构建 /dev 目录

```
mkdir -p /work/my_rootfs/dev
cd /work/my_rootfs/dev
sudo mknod console c 5 1
sudo mknod null    c 1 3
```

Step9、创建其它目录

```
mkdir proc mnt tmp sys root usr
```

Step10、拷贝 Lib

```
mkdir -p /work/my_rootfs/lib
cd /work/qt/FriendlyARM-gcc-4.4.3/arm-none-linux-gnueabi/lib (根据自己的交叉编译环境决定路径，不要直接复制)
cp *.so* /work/my_rootfs/lib -d
cd /work/qt/FriendlyARM-gcc-4.4.3/lib
cp *.so* /work/my_rootfs/lib -d
```

6. 移植 tslib

按照之前博主写的就行

tslib-1.4.tar.gz 源码下载

链接：<http://pan.baidu.com/s/1jINj3lQ> 密码：6kkc

步骤：(su 进入 root 用户)

Step1、解压&配置&编译

```
mkdir -p /usr/local/tslib
tar zxvf tslib-1.4.tar.gz (自己放在哪解压到哪自己决定)
cd tslib
./autogen.sh
./configure --host=arm-linux ac_cv_func_malloc_0_nonnull=yes CC=arm-none-linux-
gnueabi-gcc CXX=arm-none-linux-gnueabi-g++ -prefix=/usr/local/tslib
make
sudo make install
```

如果编译过程中遇到 undefined reference to 'rpl_malloc', 前面配置完成之后修改 config.h.in 文件, 注释掉文件最后的 #undef malloc, 重新 make 即可。

安装完成之后, tslib 就安装在虚拟机 /usr/local/tslib 目录下

Step2、更改 tslib 配置文件

```
cd /usr/local/tslib/etc/
```

```
sudo gedit ts.conf
```

去掉# module_raw input 前面的“#”和空格

Step 3、将制作好的 tslib 移动到我们制作的文件系统

```
cd /usr/local
```

```
sudo tar zcvf tslib.tar.gz tslib
```

```
mkdir -p /work/my_rootfs/usr/local
```

```
cp tslib.tar.gz /work/my_rootfs/usr/local
```

tar zxvf tslib.tar.gz (这个目录是你制作的文件系统/work/my_rootfs/usr/local 里面的而不是 ubuntu 系统的 usr/local)

```
rm tslib.tar.gz (与上面一致)
```

Step 4、添加 tslib 环境变量

```
gedit /work/my_rootfs/etc/profile
```

添加以下代码

```
#!/bin/sh
```

```
export T_ROOT=/usr/local/tslib
```

```
export LD_LIBRARY_PATH=/usr/local/tslib/lib:$LD_LIBRARY_PATH
```

```
export TSLIB_CONSOLEDEVICE=none
```

```
export TSLIB_FBDEVICE=/dev/fb0
```

```
export TSLIB_TSDEVICE=/dev/input/event0
```

```
export TSLIB_PLUGINDIR=$T_ROOT/lib/ts
```

```
export TSLIB_CONFFILE=$T_ROOT/etc/ts.conf
```

```
export POINTERCAL_FILE=/etc/pointercal
```

```
export TSLIB_CALIBFILE=/etc/pointercal
```

此时, tslib 就已经移植好了, 你可以挂载 nfs 文件系统启动, cd /usr/local/tslib/bin ./ts_test 来测试。

7. 移植 qt5.6

按照之前博主写的就行,

Step1、解压

```
tar zxvf qt-everywhere-opensource-src-5.6.0.tar
```

Step 2、修改编译配置

```
cd /work/qt-everywhere-opensource-src-5.6.0/qtbase/mkspecs/linux-arm-gnueabi-  
g++
```

```
gedit qmake.conf
```

针对于 2440 增加：

```
QT_QPA_DEFAULT_PLATFORM = linuxfb
```

```
QMAKE_CFLAGS += -msoft-float -D_GCC_FLOAT_NOT_NEEDED -march=armv4t -  
mtune=arm920t
```

```
QMAKE_CXXFLAGS += -msoft-float -D_GCC_FLOAT_NOT_NEEDED -march=armv4t  
-mtune=arm920t
```

march 指的 cpu 架构, 针对 2440 来说是 armv4t

mtune 指的 cpu 名字, 针对 2440 来说是 arm920t

将以下部分

```
# modifications to g++.conf
```

```
QMAKE_CC = arm-linux-gnueabi-gcc
```

```
QMAKE_CXX = arm-linux-gnueabi-g++
```

```
QMAKE_LINK = arm-linux-gnueabi-g++
```

```
QMAKE_LINK_SHLIB = arm-linux-gnueabi-g++
```

```
# modifications to linux.conf
```

```
QMAKE_AR = arm-linux-gnueabi-ar cqs
```

```
QMAKE_OBJCOPY = arm-linux-gnueabi-objcopy
```

```
QMAKE_NM = arm-linux-gnueabi-nm -P
```

```
QMAKE_STRIP = arm-linux-gnueabi-strip
```

修改为：-lts 是指在链接时链接 tslib 库

```
# modifications to g++.conf
```

```
QMAKE_CC = arm-none-linux-gnueabi-gcc -lts
```

```
QMAKE_CXX = arm-none-linux-gnueabi-g++ -lts
```

```
QMAKE_LINK = arm-none-linux-gnueabi-g++ -lts
```

```
QMAKE_LINK_SHLIB = arm-none-linux-gnueabi-g++ -lts
```

```
# modifications to linux.conf
```

```
QMAKE_AR = arm-none-linux-gnueabi-ar cqs
```

```
QMAKE_OBJCOPY = arm-none-linux-gnueabi-objcopy
```

```
QMAKE_NM = arm-none-linux-gnueabi-nm -P
```

```
QMAKE_STRIP = arm-none-linux-gnueabi-strip
```

Step 3、配置编译

```
sudo mkdir -p /usr/local/Qt5.6
```

```
cd ../../.. (这是要回到 qt-everywhere-opensource-src-5.6.0)
```

因为之前进入了 /work/qt-everywhere-opensource-src-5.6.0/qtbase/mkspecs/linux-arm-gnueabi-g++修改了 qmake.conf

```
./configure -prefix /usr/local/Qt5.6 \
```

```
-opensource \
```

```
-release \
```

```
-confirm-license \
```

```
-xplatform linux-arm-gnueabi-g++ \
-shared \
-qt-zlib \
-no-gif \
-qt-libjpeg \
-no-nis \
-no-opengl \
-no-cups \
-no-glib \
-no-dbus \
-no-rpath \
-no-sse2 -no-sse3 -no-ssse3 -no-sse4.1 -no-sse4.2 \
-no-avx \
-no-openssl \
-nomake tools \
-qreal float \
-qt-libpng \
-tslib \
-nomake examples \
-I /usr/local/tslib/include \
-L /usr/local/tslib/lib
make -j4
sudo make install
```

Step 4、将移植好的 qt 打包到开发板

```
cd /usr/local
sudo tar zcvf Qt5.6.tar.gz Qt5.6
cp Qt5.6.tar.gz /work/my_rootfs/usr/local/
tar zxvf Qt5.6.tar.gz (目录在你制作的文件系统 my_rootfs/usr/local)
rm Qt5.6.tar.gz (目录你制作的在文件系统 my_rootfs/usr/local)
rm -r doc include bin mkspecs qml translations (目录在你制作的文件系统
my_rootfs/usr/local/Qt5.6)
```

Step 5、设置 qt 相关的环境变量

此部分可以参考 qt 官方文档：<http://doc.qt.io/qt-5/embedded-linux.html>，这我这仅仅是设置支持了触摸屏，你可以参考官方设置支持键盘，鼠标等等。

在文件系统 /etc/profile 里添加

```
export QTDIR=/usr/local/Qt5.6
export LD_LIBRARY_PATH=/usr/local/Qt5.6/lib:$LD_LIBRARY_PATH
export QT_QPA_GENERIC_PLUGINS=tslib
export QT_QPA_FONTDIR=$QTDIR/lib/fonts
export QT_QPA_PLATFORM_PLUGIN_PATH=$QTDIR/plugins
export
QT_QPA_PLATFORM=linuxfb:fb=/dev/fb0:size=480x272:mmSize=480x272:offset=0x0:tty=/
dev/tty1
export QT_QPA_FB_TSLIB=1
```


此时，qt 已经移植完毕，你可以打包放入你的 nfs 目录启动进行测试了，至于制作 yaffs2 jffs2 等文件系统请参考：<http://blog.csdn.net/lizuobin2/article/details/52589215>，qt 库比较大，烧录的时候可能比较困难，可以先将 QT 去除，打包成文件系统大约之后 20M 不到，烧录到开发板之后，启动内核，通过 nfs tftp 等工具，再将打包好的 Qt 传到板子上展开即可。

在测试过程中我发现程序跑起来没问题，但是有以下两条错误信息：

QlconvCodec::convertFromUnicode: using Latin-1 for conversion, iconv_open failed

QlconvCodec::convertToUnicode: using Latin-1 for conversion, iconv_open failed

大概是缺少 libiconv

下载 链接：<http://pan.baidu.com/s/1c22xb4O> 密码：pbld

mkdir -p /usr/local/libiconv

./configure --host=arm-none-linux-gnueabi --prefix=/usr/local/libiconv CC=arm-none-linux-gnueabi-gcc LDFLAGS="-L /work/qt/FriendlyARM-gcc-4.4.3/arm-none-linux-gnueabi/sys-root/lib" --enable-static （根据自己的实际路径）

make

make install（前面不要加 sudo，这是与之前博主分享的区别）

不然会出现 arm-none-linux-gnueabi-gcc: command not found

最后查找 su 和 sudo 的区别，终于理解了，原来是工作环境的问题。

su 和 sudo 的区别：

1.共同点：都是 root 用户的权限；

2.不同点：su 仅仅取得 root 权限，工作环境不变，还是在切换之前用户的工作环境；

sudo 是完全取得 root 的权限和 root 的工作环境。以上来自网络。

把安装目录 /lib 下的 preloadable_libiconv.so 文件系统的 /lib 下，在 /work/my_rootfs/etc /etc/profile 中添加

export LD_PRELOAD=/lib/preloadable_libiconv.so

8. 测试 Qt 应用程序

下载 qt 工程例子，代码中 Ui 尺寸写死为 320*240，如果不适合你的板子请修改。

链接：<http://pan.baidu.com/s/1qXV4C1Y> 密码：908q

下载字库 链接：<http://pan.baidu.com/s/1bp9QFQv> 密码：2u81

将 DroidSansFallback.ttf 放到文件系统 /usr/local/Qt5.6/lib/fonts 目录下，虽然 Qt 自带了很多字库了，但是都没有中文的。

为 ubuntu 添加 qmake，不然输入 qmake 会找不到命令

cd /usr/lib/x86_64-linux-gnu/qt-default/qtchooser/会有一个 default.conf 配置文件，可以修改默认配置文件为你编译好的 qt 库路径。

将

/usr/lib/x86_64-linux-gnu/qt4/bin

/usr/lib/x86_64-linux-gnu

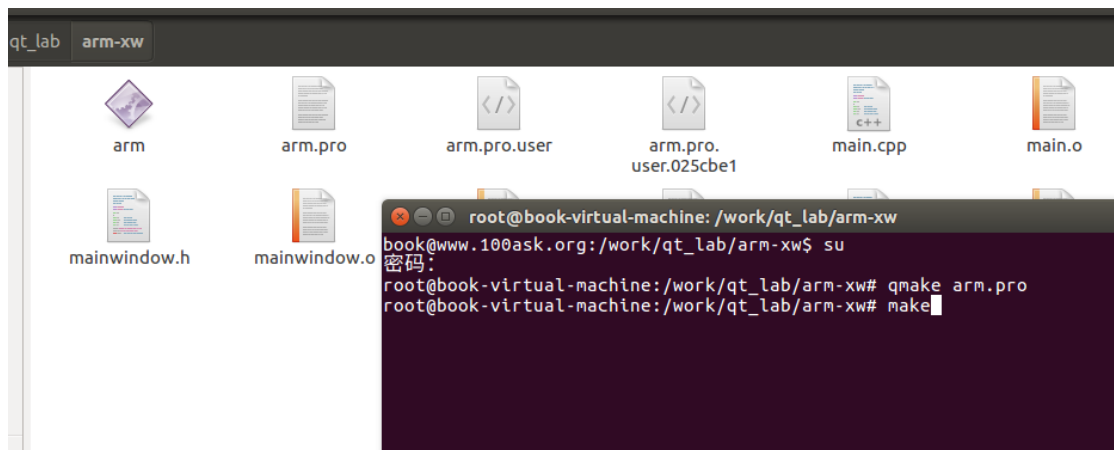
改为

/usr/local/Qt5.6/bin（我们自己生成的 qt 交叉编译环境）

/usr/local/Qt5.6

打开你的 qt 工程目录首先进行 qmake arm.pro 生成 Makefile 文件

然后 make 生成可执行文件



将生成的 arm 文件放在 nfs 目录去测试。

9.整个工程测试：

a.首先三机（笔记本、虚拟机、开发板）互通，新一期加强版免费视频里面有讲解（[https://eyun.baidu.com/s/3miEaDza#sharelink/path=%2F%27%99%BE%E9%97%AE%E7%BD%91%E5%88%86%E4%BA%AB%E7%9A%84%E6%89%80%E6%9C%89%E6%96%87%E4%BB%B6%2F005_ARM%E8%A3%B8%E6%9C%BA1%E6%9C%9F%E5%8A%A0%E5%BC%BA%E7%89%88%2F%E8%A7%86%E9%A2%91%2F%E7%AC%AC006%E8%AF%BE_%E5%BC%80%E5%8F%91%E6%9D%BF%E7%86%9F%E6%82%89%E4%B8%8E%E4%BD%93%E9%AA%8C\(%E5%85%8D%E8%B4%B9\)&parent_path=%2F%E5%88%86%E4%BA%AB%E7%9A%84%E6%89%80%E6%9C%89%E6%96%87%E4%BB%B6](https://eyun.baidu.com/s/3miEaDza#sharelink/path=%2F%27%99%BE%E9%97%AE%E7%BD%91%E5%88%86%E4%BA%AB%E7%9A%84%E6%89%80%E6%9C%89%E6%96%87%E4%BB%B6%2F005_ARM%E8%A3%B8%E6%9C%BA1%E6%9C%9F%E5%8A%A0%E5%BC%BA%E7%89%88%2F%E8%A7%86%E9%A2%91%2F%E7%AC%AC006%E8%AF%BE_%E5%BC%80%E5%8F%91%E6%9D%BF%E7%86%9F%E6%82%89%E4%B8%8E%E4%BD%93%E9%AA%8C(%E5%85%8D%E8%B4%B9)&parent_path=%2F%E5%88%86%E4%BA%AB%E7%9A%84%E6%89%80%E6%9C%89%E6%96%87%E4%BB%B6) 第 6 节)，核心思想就是哪个 ip 端口与开发板相连你就要使用哪个 ip，最新的 uboot 1.1.6 说是开机就使能网络，但是我测试了效果差强人意，建议还是使用一个路由器，其实 3 机不能互通主要问题就出现在我们笔记本上面有两个网卡，一个无线一个有线，你得确定好虚拟机使用哪个网卡与主机通信（可以无线也可以有线），但是我们的开发板与虚拟机之间是通过有线网卡进行通信的！所以我们虚拟机也要使用有线网卡与笔记本进行通信，如果看视频实在听不明白，很简单先将无线网卡禁用，配置好以后，再打开无线上网即可。



b.设置 nfs 目录：

sudo gedit /etc/exports，在里面增加以下内容，以后将通过网络文件系统访问 /work/nfs_root 目录：

添加

/work/nfs_root *(rw,sync,no_root_squash)

修改完毕后，执行以下命令重启 nfs 服务：

```
sudo /etc/init.d/nfs-kernel-server restart
```

具体目录根据自己的目录进行设置。

开发环境 IP 说明

路由器 IP 192.168.1.1

虚拟机 IP 192.168.1.2

开发板 IP 192.168.1.3

笔记本 IP 192.168.1.4

记得将 nfs 文件里面 etc/init.d/rcs 的 IP 地址，改为开发板的 ip

在 uboot 里面设置

```
set serverip 192.168.1.4 //笔记本
```

```
set ipaddr 192.168.1.3 //开发板
```

```
set gatewayip 192.168.1.1 //网关
```

```
set bootargs noinitrd root=/dev/nfs console=ttySAC0,115200
```

```
nfsroot=192.168.1.2:/work/my_rootfs
```

```
ip=192.168.1.3:192.168.1.2:192.168.1.1:255.255.255.0::eth0:off init=/linuxrc
```

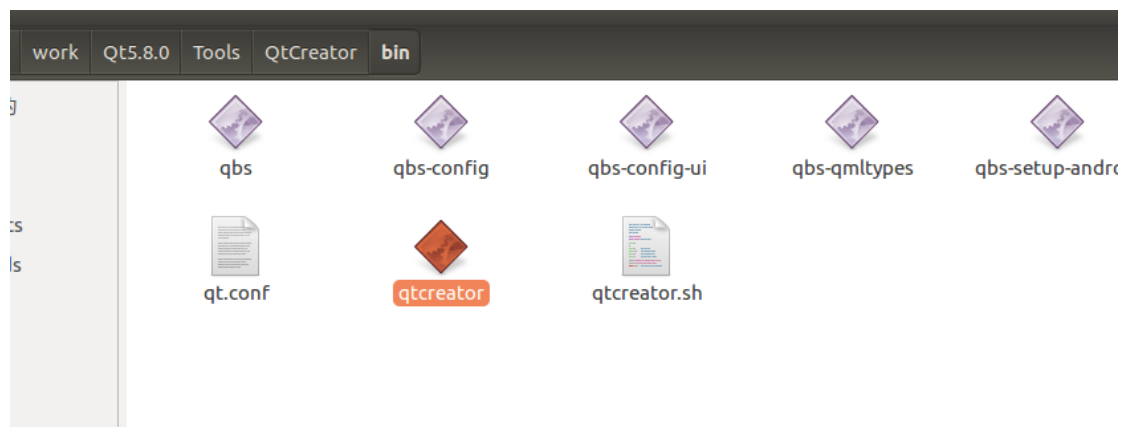
```
save
```

10.ubuntu 下的 Qt 的安装和配置

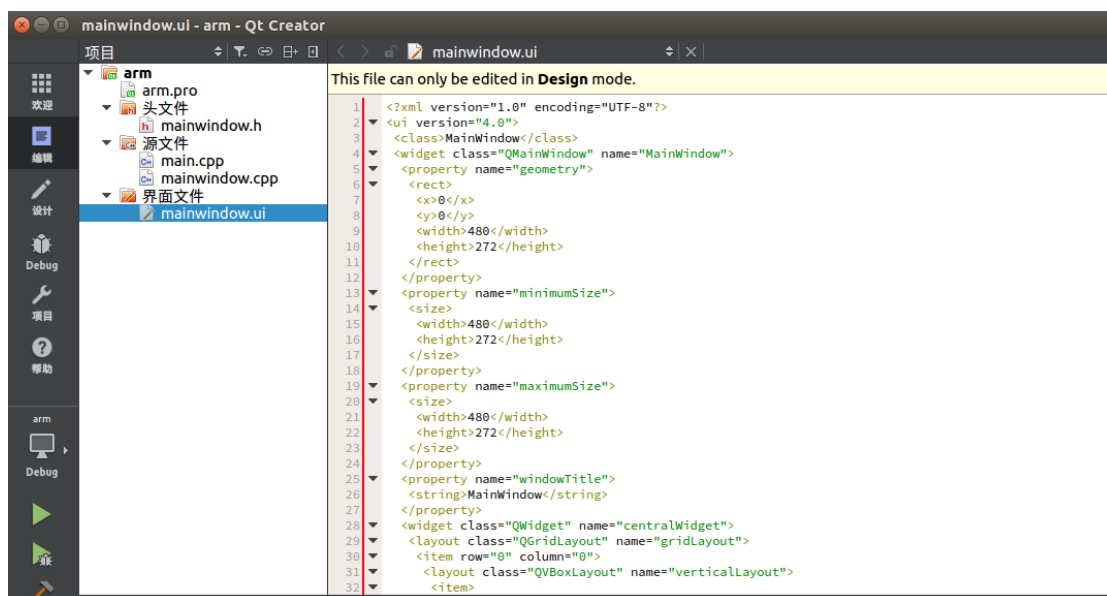
个人感觉 linux 下安装 qt 不仅体积小，而且 linux 本来就带 gcc 也不需要安装其他编译器，我安装的 qt-opensource-linux-x64-5.8.0.run，如果在 Windows 下面个人建议安装直接包括 mingw530 的，这样就不需要自己再安装 VS 了，仅限入门而言。

./ qt-opensource-linux-x64-5.8.0.run 即可实现安装。

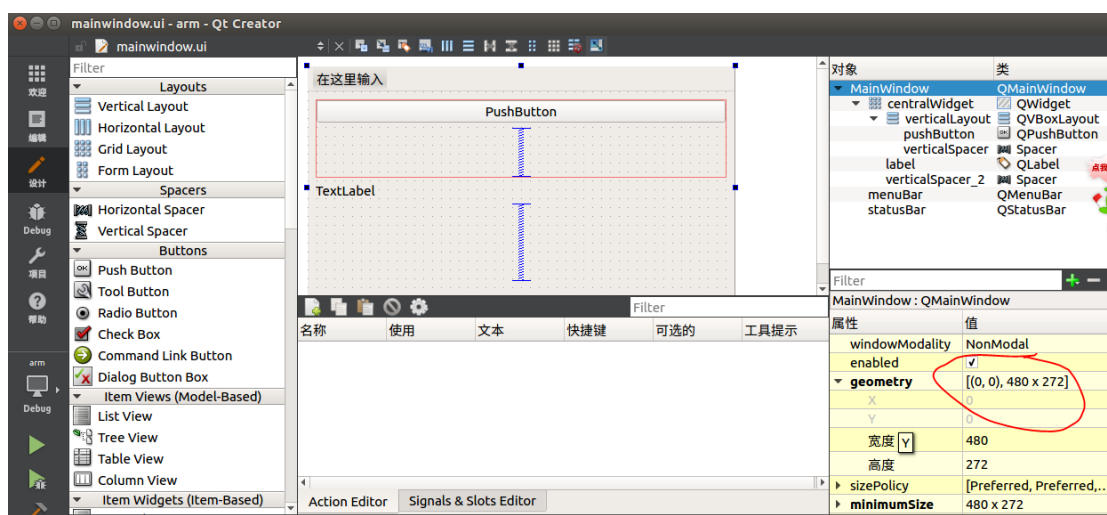
我安装在 work/Qt5.8.0 目录下，安装完毕以后双击/work/Qt5.8.0/Tools/QtCreator/bin 的 qtcreator 启动 Qte。



可以直接打开配套下载的 arm qt 工程。



打开mainwindow.ui 修改为 4.3 寸 480 272

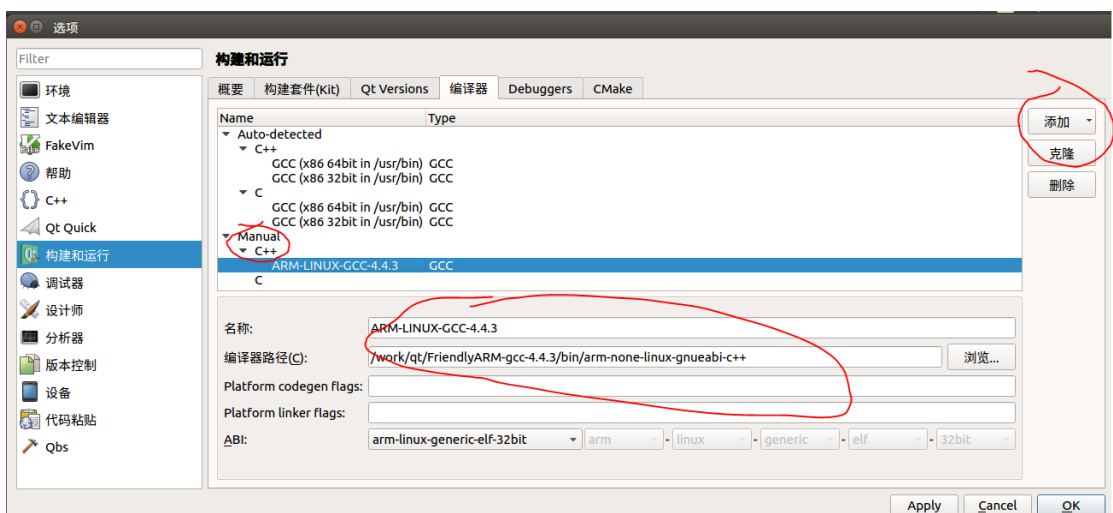


对 QtE 设置交叉编译环境

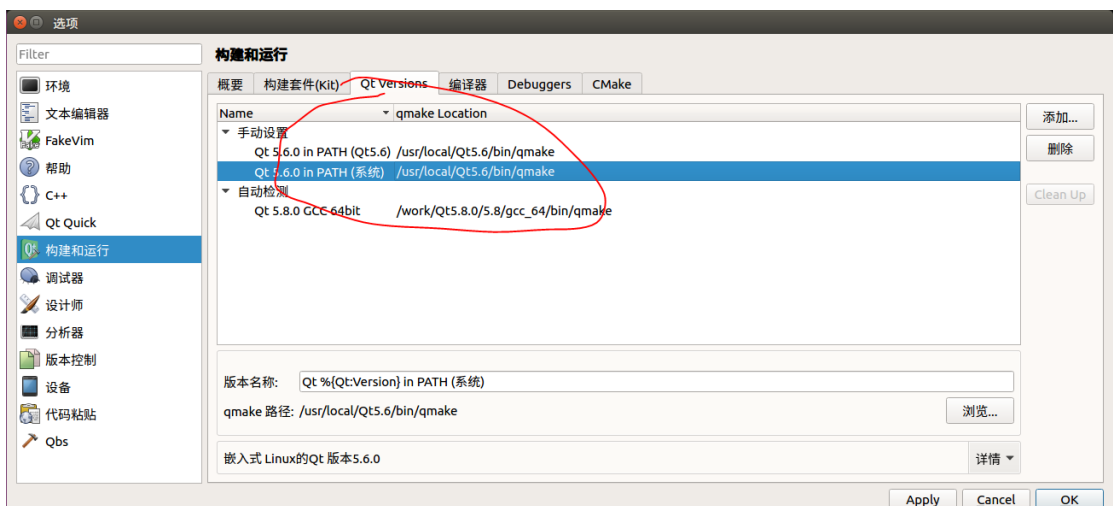
Step1:选择 Manage Kits



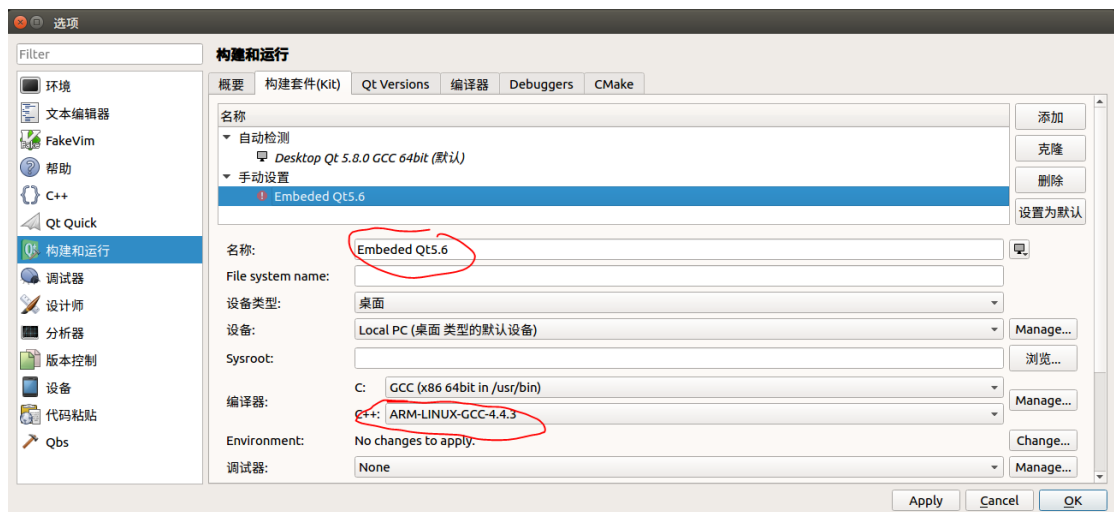
Step2:设置 C++ 编译器



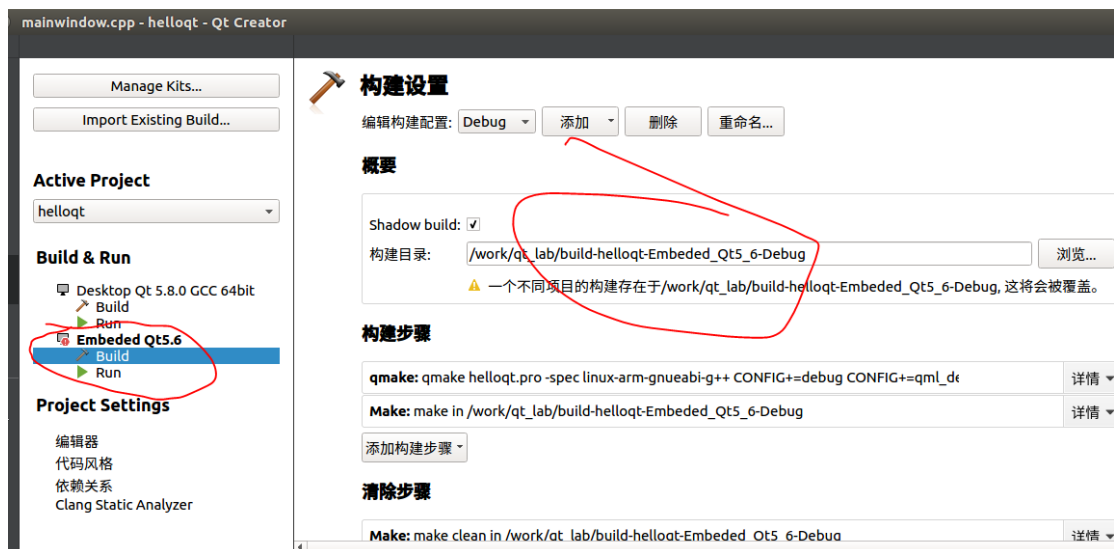
Step3: 设置 Qt Versions (之前只要设置过 /usr/lib/x86_64-linux-gnu/qt-default/qtchooser/会有一个 default.conf 配置文件就会系统直接生成)



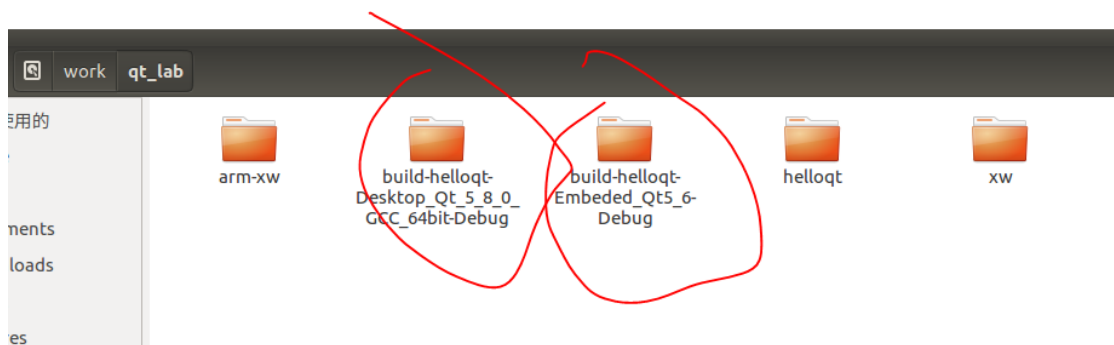
Step4:设置构建套件 (step2 设置完需要 apply 一下, 这边就可以选择那个名字)

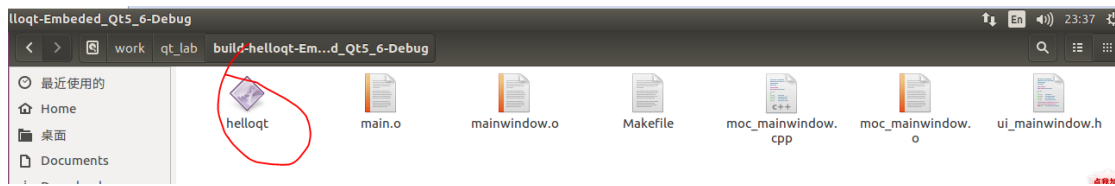


Step5:设置 Debug 目录, 自己根据生成的名字新建文件夹 (感觉是个小 bug, 难道就不能软件直接生成嘛 !!!)



Step6:右击工程再次编译后可以看到如下文件夹下增加了基于嵌入式 LINUX 系统的可执行文件.





这样就不用使用 qmake 自己去 qmake 生成 makefile 了。

联系方式：微信 Startingray,可以一起交流,我一定热心,只要我会的,坚信高手也是从小白成长过来的！