# 1 QT 5.5.1在ubuntu14中安装（32位系统）

## 1.1 安装qt

chmod +x qt-opensource-linux-x86-5.5.1.run

./qt-opensource-linux-x86-5.5.1.run

一路next就安装完成。

## 1.2 解决安装出现的问题

cannot resolve SSLv2\_server\_method

出错原因：

这是由于SSLv2由于某些漏洞导致的不安全原因， 在某些linux发行版里禁用掉了SSLv2。

解决办法：

是自己重新编译openssl，加上SSLv2的支持。

暂时不使用该功能，暂时不添加。

Cannot start '/usr/lib/i386-linux-gnu/qt4/bin/qmake': No such file or directory。

出错原因：

qtchooser默认选择路径中没有指向qmake路径，qtchooser默认路径在：/usr/lib/i386-linux-gnu/qt-default/qtchooser/default.conf

解决方法：

vim /usr/lib/i386-linux-gnu/qt-default/qtchooser/default.conf

第一行添加为qmake的路径（我的安装路径/home/chao/App/Qt5.5.1/5.5/gcc/bin）

测试：

qmake –v 查看是否安装成功

编译程序时提示： error: cannot find –lGL

是由于 Qt5.0 默认将OpenGL加入了工程，但是在机器上没有安装OpenGL

安装OpenGL Library:

接下来要把我们会用到的 Library 装上去，首先安装 OpenGL Library

$sudo apt-get install libgl1-mesa-dev

我运行到这一步再去编译，就不会出现这个错误了。

安装OpenGL Utilities (由于暂时不适用opengl ，故下面的步骤没有执行)

OpenGL Utilities 是一组建构于 OpenGL Library 之上的工具组，提供许多很方便的函式，使 OpenGL 更强大且更容易使用。接下来我们安装OpenGL Utilities

$sudo apt-get install libglu1-mesa-dev

注意在这一步的时候，可能会出现以下情况，

shell提示：正在读取软件包列表... 完成正在分析软件包的依赖关系树

正在读取状态信息... 完成

 E: 未发现软件包 libglut-dev如果出现上述情况

请输入将上述$sudo apt-get install libglut-dev命令改成$sudo apt-get install freeglut3-dev就行。

# 2 移植QT到开发板

## 2.1 安装arm-linux-gcc 4.4.3编译器

tar xzf arm-linux-gcc-4.4.3.tar.gz -C /

cp /opt/FriendlyARM/toolschain/4.4.3 /usr/local/arm/ -rf

vim /etc/environment

路径替换为/usr/local/arm/4.4.3/bin

重启测试是否安装成功

arm-linux-gcc –v

## 2.2 编译Linux-3.4.20内核

cp .config config\_backup

make clean

cp config \_backup .config

make uImage

## 2.3 制作跟文件系统

清理原来

make clean

编译

make

配置

安装

mkdir ../fs\_mini\_mdev\_4.4.3

make install CONFIG\_PREFIX=/home/chao/download/fs\_mini\_mdev\_4.4.3/

拷贝库文件

cd ../fs\_mini\_mdev\_4.4.3

mkdir lib

cp /usr/local/arm/4.4.3/arm-none-linux-gnueabi/lib/\*.so\* ./lib -rf -d

cp /usr/local/arm/4.4.3/lib/\*.so\* ./ lib -rf -d

制作etc 目录

以前的etc文件夹拷贝过来

cp ../fs\_mini\_mdev\_new/etc/ ./ -rf

创建设备节点

cd dev/

sudo mknod console c 5 1

sudo mknod null c 1 3

创建其他目录

mkdir proc tmp mnt sys root

制作映像文件

mkyaffs2image fs\_mini\_mdev\_4.4.3 fs\_mini\_mdev\_4.4.3.yaffs2

## 2.4移植tslib

1> 加载lcd和触摸屏驱动

把模块放在/lib/modules/3.4.20/目录下面，如lcd.ko

cp lcd.ko /lib/modules/3.4.20

cp s3c\_ts.ko /lib/modules/3.4.20

/ect/目录下面新建rc.modules。添加内容：

#!/bin/sh

insmod /lib/modules/3.4.20/lcd.ko

insmod /lib/modules/3.4.20/s3c\_ts.ko

在rcS文件里面添加如下：

if [ -f /etc/rc.modules ]; then

sh /etc/rc.modules

fi

2> 移植talib

修改编译器

cd /usr/local/arm/4.4.3

grep “EV\_VERSION” \* -nR

sudo vim arm-none-linux-gnueabi/sys-root/usr/include/linux/input.h +32

修改为 #define EV\_VERSION 0x010001(保持和内核一致)

安装必须软件

sudo apt-get install autoconf

sudo apt-get install automake

sudo apt-get install libtool

配置编译tslib

cd tslib

./autogen.sh

mkdir tmp

echo "ac\_cv\_func\_malloc\_0\_nonnull=yes" >arm-linux.cache

./configure --host=arm-linux --cache-file=arm-linux.cache --prefix=$(pwd)/tmp

make clean

make

make install

拷贝到开发板

cd mnt/myderivers342/11th\_ts/tslib/tmp/

cp \* -rf /

修改 /etc/ts.conf第1行(去掉#号和第一个空格，否则会产生段错误Segmentation fault)：

# module\_raw input

改为：

module\_raw input

开发板依次执行

vi /etc/profile 添加如下

(event0还是event1根据触摸屏是哪个设备决定)

#!bin/sh  
  export TSLIB\_TSDEVICE=/dev/event0

export TSLIB\_CALIBFILE=/etc/pointercal

export TSLIB\_CONFFILE=/etc/ts.conf

export TSLIB\_PLUGINDIR=/lib/ts

export TSLIB\_CONSOLEDEVICE=none

export TSLIB\_FBDEVICE=/dev/fb0

校准触摸屏

ts\_calibrate//执行命令后点击屏幕校准,最好用稍尖的东西会准确。手指不准确。

测试触摸屏

必须执行才能测试

ts\_test //可以在屏幕画线

## 2.5 移植qt5.6

解压qt源码

tar xzf qt-everywhere-opensource-src-5.5.1.tar.gz

cd qt-everywhere-opensource-src-5.5.1/

修改qmake.conf

cd qtbase/mkspecs/linux-arm-gnueabi-g++

vi 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

配置后后自动生成makefile

chao@chao:~/work/myderivers342/11th\_ts/tslib/tmp$

sudo cp bin etc include lib /usr/local/tslib/ -rf //拷贝tslib

sudo mkdir -p /usr/local/Qt5.6 //安装文件夹

./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 -no-iconv -I/usr/local/tslib/include -L/usr/local/tslib/lib

make //使用4个作业编译提高编译速度，需要四线程cpu

  make install //sudo su切换到root.不要用sudo 命令，会出错。

cp lib/ /usr/local/Qt5.6/ -rf -d //将生成的lib拷贝到开发板

开发板vi /etc/profile 添加如下

export QTEDIR=/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=$QTEDIR/lib/fonts

export QT\_QPA\_PLATFORM\_PLUGIN\_PATH=$QTEDIR/plugins

export QT\_QPA\_PLATFORM=linuxfb:fb=/dev/fb0:size=480x272:mmSize=480x272:offset=0x0:tty=/dev/tty1

exportQT\_QPA\_FB\_TSLIB=1

拷贝中文库

cp /mnt/Qt5.6/lib/fonts/DroidSansFallback.ttf Qt5.6/lib/fonts/

# 3 应用测试

## 3.1 测试工程hello world

ubuntu 测试成功的程序。拷贝到目录hello。拷贝文件.h;.cpp;.ui文件

/opt/fsl-imx-x11/4.1.15-2.1.0/sysroots/cortexa9hf-neon-poky-linux-gnueabi/usr/lib/cmake

自动生成工程文件和makefile

/usr/local/Qt5.6/bin/qmake -project //使用交叉编译工具,make install的目录

/usr/local/Qt5.6/bin/qmake -makefile

需要修改arm.pro

QT += core gui

QT += gui

greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets

TEMPLATE = app

TARGET = arm

INCLUDEPATH += .

# Input

HEADERS += mainwindow.h

SOURCES += main.cpp mainwindow.cpp

FORMS += mainwindow.ui

make

生成hello可执行程序

./hello

运行时出现错误提示：（需要重新编译，已经测试是该原因）

QIconvCodec::convertToUnicode: using Latin-1 for conversion,iconv\_open failed  
QIconvCodec::convertFromUnicode: using Latin-1 for conversion,iconv\_open fail

解决：./configure 后添加编译 -no-iconv

## 3.2 如何调用外部程序

void HelloDialog::on\_pushButton\_clicked()

{

ui->label->setText(tr("arm"));

QString program = "/app/arm"; //调用程序的路径

QStringList arguments;

arguments << ""; //传递给新程序的参数

myProcess.start(program, arguments);

}

## 3.3 如何给对话框添加背景图片

HelloDialog::HelloDialog(QWidget \*parent) :

QDialog(parent),

ui(new Ui::HelloDialog)

{

ui->setupUi(this);

setStyleSheet("QDialog{background-image:url(:/images/back.jpg)}");

}

## 3.4 如何调用c程序

如果调用的c源代码为firstdrvtest.c。那么对应的头文件firstdrvtest.h如下：

#ifndef FIRSTDRVTEST\_H

#define FIRSTDRVTEST\_H

extern "C" {

int led\_ctrl(char \*str);

}

#endif // FIRSTDRVTEST\_H

这样声明后就可以在c++函数中调用c函数了。而c程序中处理与驱动交互。

对应linux系统调用，只需添加相应头文件就可以了。

## 3.5 如何解决不能显示在左上角问题

int main(int argc, char \*argv[])

{

TextEdit w;

w.setGeometry(0,0,480,272);

w.show();

return a.exec();

}