QT环境配置手册

源码包介绍：

- 工程管理 tmake-1.11-9315.tar，用于生成 Makefile。
- Qt/X11 软件包 qt-2.3.2-9315.tar，用于生成 qvfb 等开发工具。
- Qt/Embedded 软件包 qt-2.3.10-9315.tar，Qt/Embedded 图形库。
- Qt/Embedded 软件包 qt-2.3.10-x86.tar，Qt/Embedded 图形仿真库。

假设 tmake、qt2.3.2-9315、qt-2.3.10-9315和 qt-2.3.10-x86解压后的源码都放在 /home/share/tangh/qt/work 目录下，并且文件夹名称分别重命名为 tmake-1.11、qt-2.3.2、qt-2.3.10-arm、qt-2.3.10-host。

一、准备工作

1. 安装 tmake

只需指定上一步解压后 tmake-1.11文件夹的目录即可，如：

```
export TMAKEDIR=/home/share/tangh/qt/work/tmake-1.11
export TMAKEPATH=$TMAKEDIR/lib/qws/linux-arm-g++
```

2. 安装 Qt/X11

```
cd qt-2.3.2
```

（设置环境变量）

```
export TMAKEDIR=/home/share/tangh/qt/work/tmake-1.11
export TMAKEPATH=$TMAKEDIR/lib/qws/linux-generic-g++
export QT2DIR=/home/share/tangh/qt/work/qt-2.3.2
```

```
export QTDIR=$QT2DIR
export PATH=${TMAKEDIR}:${TMAKEPATH}:${QTDIR}/bin:${PATH}

（配置，回答 yes）
./configure -static -no-xft -no-opengl -no-sm

（编译并复制 moc 工具到 bin 目录）
make -C src/moc
cp src/moc/moc bin

（编译 Qt/X11 库）
make -C src

（编译 Designer，用于可视化界面设计）
make -C tools/designer
cp tools/designer/designer bin

（编译 qvfb，用于在 PC 机上仿真 Qt 程序）
make -C tools/qvfb
cp tools/qvfb/qvfb bin
```

qvfb 工具用来生成 Virtual framebuffer，这是一个非常有用的工具，它可以模拟在开发板上的显示情况，如果在 Virtual framebuffer 中运行没有问题的话，可以直接通过交叉编译在开发板上运行。

二、交叉编译 Qt/Embedded 图形库

1. 交叉编译器的安装

2.3.10在 9315的 2.6内核下用的是 3.4的编译器，在光盘里找到

arm-linux-gcc-3.4.3-1.0.1.tar.bz2，用如下命令安装：

```
# tar jxvf arm-linux-gcc-3.4.tar.bz2
```

解压后有两个目录 usr和 opt，然后分别把这两个移动对应的目录下

```
# mv usr/local/arm/ /usr/local
# mv opt/buildroot /opt
```

2. 添加触摸屏库的支持

```
tar xvf tslib-0.1.1.tar.gz
cd tslib-0.1.1
make clean
./autogen-clean.sh
export CC=/usr/local/arm/3.4/bin/arm-linux-gcc
./autogen.sh
/建一个目录 ,准备将编译好的文件全放在这个目录下面
mkdir /usr/local/TslibBuil
./configure --host=arm-linux  --prefix=/usr/local/TslibBuil
make
make install
```

这一步把编译 tslib生成的库文件加入到编译器的 lib里面去：

```
cd /usr/local/TslibBuil/lib
cp -vrf libt* /usr/local/arm/3.4/lib/
```

3. 设置环境变量

cd qt-2.3.10-arm

export PATH=/usr/local/arm/3.4/bin:$PATH

export TMAKEDIR=/home/share/qt/work/tmake-1.11

export TMAKEPATH=/home/share/qt/work/tmake-1.11/lib/qws/linux-arm-g++

export QTEDIR=/home/share/qt/work/qt-2.3.10-arm

export QTDIR=$QTEDIR

export LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH

export PATH=$QTDIR/bin:$TMAKEDIR/bin:$PATH

4. 配置编译规则

./configure -xplatform linux-arm-g++ -no-qvfb -accel-ep93xx -depths 16 -tslib -thread -I$QTDIR/include -L$QTDIR/lib

配置时输入 yes,5

5. 编译

Make

这步完成后，我们会在 /$QTEDIR/lib/目录下面看到 libqte.so，libqte.so.2，libqte.so.2.3，libqte.so.2.3.10 这四个文件，我们可以使用 file 命令来查看这个库文件是否是我们需要的在开发板上跑的库。

file libqte.so.2.3.10

libqte.so.2.3.10: ELF 32-bit LSB shared object, ARM, version 1 (ARM), not stripped

有了这个库以后我们就可以把它拷贝到我们的开发板中相应的库目录下面，这里我们选择了开发板上的 /lib 目录，将 /$QTEDIR/lib/下的 libqte.so* 复制到 /lib 目录下。

三、搭建 Qt/Embedded 仿真开发环境

1. 设置环境变量

```
cd qt-2.3.10-host

export  TMAKEDIR=/home/share/tangh/qt/work/tmake-1.11
export  TMAKEPATH=$TMAKEDIR/lib/qws/linux-x86-g++
export  QTEDIR=/home/share/tangh/qt/work/qt-2.3.10-host
export  QTDIR=$QTEDIR
export  LD_LIBRARY_PATH=$QTDIR/lib:$LD_LIBRARY_PATH
export PATH=${TMAKEDIR}/bin:${TMAKEPATH}:${QTDIR}/bin:${PATH}
```

3. 配置编译规则

```
./configure -no-xft -qvfb -depths 4,8,16,32
```

4. 编译

```
Make
```

5. 查看运行结果

如果上面各步都能够成功的编译通过，下面就可以通过运行 Qt/Embedded 自带的 demo 来查看运行结果。

在 Virtual framebuffer 上运行：

```
export  QTEDIR=/home/share/tangh/qt/work/qt-2.3.10-host
export QTDIR=$QTEDIR
export QT2DIR=/home/share/tangh/qt/work/qt-2.3.2
export PATH=$QTEDIR/bin:$QT2DIR/bin:$PATH
```

```
export LD_LIBRARY_PATH=$QTEDIR/lib:$QT2DIR/lib:$LD_LIBRARY_PATH
cd examples/launcher
qvfb -width 640 -height 480 &
sleep 10
./launcher -qws
```