轻松搭建 Qt/Embedded

开发环境

--------By LastRitter
QQ:314665354
E-mail:superyongzhe@163.com
2009 年 8 月 12 日

一、 准备工作

1. 硬件环境

A. 主机：x86 系列 PC 机。

B. 开发板：友善之臂 mini2440。

C. 通讯连接：串口和 USB。

2. 软件环境

A. 主机操作系统：VMware 虚拟机下的 Redhat linux 9.0(完全安装）。

B. 开发板操作系统：嵌入式 Linux，内核版本为 2.6.13。

C. 交叉编译器：arm-linux-gcc-3.3.2。

D. 通讯方式：虚拟机与主机（Windows XP）使用共享文件夹通讯，主机与开发板使用 DNW 和 Secure CRT 通讯。

3. 相关说明

A. 命令：所有以#开头的均为在 shell 中执行的命令。

B. 交叉编译器位置：/usr/local/arm/3.3.2。

C. 工作目录：默认为/opt/qt。

D. 脚本注释：均放在命令下一行的圆括号内。

4. 下载源码包

A. 工程管理 tmake-1.1.1.tar.gz，用于生成 Makefile。

B. Qt/X11 软件包 qt-x11-2.3.2.tar.gz，用于生成 qvfb 等开发工具。

C. Qt/Embedded 软件包 qt-embedded-2.3.7.tar.gz，Qt/Embedded 图形库。

D. Qt 的 PDA 应用框架 qtopia-free-1.7.0.tar.gz，可以生成一个 Qt 的 PDA 程序。

5. 编译前准备

A. 建立工作目录

#mkdir /root/qt

#cd /root/qt

#mkdir x86-qt

（仿真开发目录）

#mkdir arm-qt

（交叉编译目录）

#cp …

（复制所有源码包到此目录）

B. 解压源码包

#cd /root/qt/

#tar zxvf tmake-1.1.1.tar.gz －C x86-qt

#tar zxvf tmake-1.1.1.tar.gz －C arm-qt

#tar zxvf qt-x11-2.3.2.tar.gz －C x86-qt

#tar zxvf qt-x11-2.3.2.tar.gz －C arm-qt

#tar zxvf qt-embedded-2.3.7.tar.gz －C x86-qt

#tar zxvf qt-embedded-2.3.7.tar.gz －C arm-qt

#tar zxvf qtopia-free-1.7.0.tar.gz －C x86-qt

#tar zxvf qtopia-free-1.7.0.tar.gz －C arm-qt

（加压源码包分别到两个目录）

#cd x86-qt

#mv tmake-1.1.1 tmake

#mv qt-x11-2.3.2 qt-x11

#mv qt-embedded-2.3.7 qt

#mv qtopia-free-1.7.0 qtopia

#cd ../arm-qt

#mv tmake-1.1.1 tmake

#mv qt-x11-2.3.2 qt-x11

#mv qt-embedded-2.3.7 qt

#mv qtopia-free-1.7.0 qtopia

（为了方便，给目录改名）

二、 搭建 Qt/Embedded 仿真开发环境

1. 安装 tmake

#cd /root/qt/x86-qt

#export TMAKEDIR=$PWD/tmake

#export TMAKEPATH = $PWD/tmake/ lib/qws/linux-generic-g++

（只用注册好 tmake 的环境变量皆可使用）

2. 安装 Qt/X11

#cd qt-x11

#export QTDIR=$PWD

（设置环境变量）

#./configure －static －no-xft －no-opengl －no-sm

(配置，回答 yes)

#make －C /src/moc

#cp src/moc/moc bin

（编译并复制 moc 工具到 bin 目录）

#make -C src

（编译 Qt/X11 库）

#make -C tools/designer

#cp tools/designer/designer bin

（编译 Designer，用于可视化界面设计）

#make -C tools/qvfb

#cp tools/qvfb/qvfb bin

（编译 qvfb，用于在 PC 机上仿真 Qt 程序）

3. 编译 Qt/Embedded

#export QTDIR=$PWD/qt

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

#cd qt

（设置环境变量）

#cp /qtopia/src/qt/qconfig-qpe.h src/tools/

（从 Qtopia 源码中复制配置文件）

#./configure -system-jpeg -gif -system-libpng -system-zlib -platform linux-generic-g++ -qconfig qpe -depths 16,24,32

(配置 Qt/Embedded 图形库，然后回答两个 yes)

#make –C src

（编译 Qt/Embedded）

4. 编译 Qtopia

#cd ../qtopia/src

#./configure -platform linux-generic-g++

#make

5. 编写环境变量脚本

脚本内容如下，在/opt/x86 -qt/下保存为 set-env，在编译或者运行 Qt 程序之前进入该目录执行此脚本。

export QTDIR=$PWD/qt

export QPEDIR=$PWD/qtopia

export TMAKEDIR=$PWD/tmake

export TMAKEPATH=$TMAKEDIR/lib/qws/ linux-arm-g++

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

三、 交叉编译 Qt/Embedded 图形库

首先要确保我们安装好了交叉编译器 arm-linux-gcc，可以使用如下命令测试：

#which arm-linux-gcc

（如果所显示的 arm-linux-gcc 的版本和路径与你所安装的一致，那么就可以用它来交叉编译 Qt 库和应用程序了。）

1. 安装 tmake

#cd /root/qt/arm-qt

#export TMAKEDIR=$PWD/tmake

#export TMAKEPATH=$PWD/tmake/lib/qws /linux-arm-g++

（设置 tmake 环境变量，即可直接使用）

2. 交叉编译 Qt/Embedded

#export QTDIR=$PWD/qt

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

（设置环境变量）

#cd qt

#cp ../qtopia/src/qt/qconfig-qpe.h ./src/tools

（从 Qtopia 源码中复制配置文件）

#make clean

#./configure ./configure -system-jpeg -gif -system-libpng -system-zlib -platform linux-arm-g++ -qconfig qpe -depths 16,24,32

(配置,然后回答两个yes)

make -C src

（编译 Qt/Embeddesd）

3. 交叉编译 Qtopia

cd ../qtopia/src

./configure -platform linux-arm-g++

make

4. 编写环境变量脚本

脚本内容如下，在/opt/arm -qt/下保存为 set-env，在编译或者执行 Qt 程序之前在该目录执行此脚本。

```
export QTDIR=$PWD/qt
export QPEDIR=$PWD/qtopia
export TMAKEDIR=$PWD/tmake
export TMAKEPATH=$TMAKEDIR/lib/qws/ linux-arm-g++
export PATH=$QTDIR/bin:$ QPEDIR/bin:$TMAKEDIR/bin:
$PATH
```

四、 仿真 Qt/Embedded 应用程序

1. 注册环境变量

#cd/root/qt/x86-qt

#source set-env

（或者#. set-env，.和 set-env 之间有一个空格）

#mkdir hello

#cd hello

2. 设计界面

使用 Qt/X11 的可视化界面设计工具 Qt Designer 设计界面。

#designer hello.ui

然后设计好界面，保存并退出。

3. 生成界面代码

使用 Qt 的代码自动生成工具 uic 把设计好的界面 hello.ui 生成相应的 c++代码：hello.h 和 hello.cpp。

#uic -o hello.h hello.ui

#uic -o hello.cpp -impl hello.h hello.ui

生成的源文件中把我们设计好的界面定义为一个类，在我们的程序中使用这个类创建对象，并设为主控件就可以了。

4. 编写代码

#vi main.cpp

（编辑 main.cpp，内容如下）

```
#include<qapplication.h>
#include"hello.h"
//包含我们设计的界面 MainWindow 类的声明

int main(int argc,char **argv)
{
    QApplication app(argc,argv);
    MainWindow *mainwindow=new MainWindow(0,"MainWindow");
//实例化 MainWindow，MainWindow 为在使用 Designer 设计界面时给主窗口的名称。

    app.setMainWidget(mainwindow);
//把 mainwindow 设定为程序的主控件，在程序执行时就可以显示这个控件。

    mainwindow->show();
//显示 mainwindow，这样就为我们的界面显示做好准备了。

    return app.exec();
}
```

5．建立工程

可以使用 tmake 中的 progen 工具产生一个工程模板，然后自己修改。也可以自己写工程文件，个人觉得这样方便，下面我们就自己写一个吧！

#vi hello.pro

（编辑内容如下）

```
TEMPLATE=app
#表明这是一个应用程序
CONFIG+=qtopia warn_on release
#qtopia：生成 qtopia 应用程序，warn_on：所有的警告全部打开，release：不带调试信息
SOURCES=main.cpp hello.cpp
#指明工程中包含的源程序
HEADERS=hello.h
#指明源程序中包含的头文件
TARGET=hello
#生成的程序名为 hello
```

6．编译并仿真运行程序

#tmake -o Makefile panel.pro

#make

（这样就会在当前目录下生成 hello 可执行文件）

#qvfb &

（执行 Qt 虚拟缓冲帧，用于仿真）

#./hello -qws

（运行编译好的程序，在 qvfb 上就可以看到我们设计的界面了）

五、 交叉编译程序

#cd /root/qt/arm-qt

#. set-env

cp ../x86-qt/hello . -r

#cd hello

#tmake -o Makefile panel.pro

#make

（把生成的 hello 下载到带有 Qt/Embedded 图形库的开发板上执行即可）