嵌入式系统

西安电子科技大学 白玉霞 刘旭辉 孙肖子

基于Qt/Embedded的GUI移植及应用程序开发

Transplantation of GUI and Implementation of Application Based on Embedded Linux

摘 要：介绍了系统平台构建；重点分别从主机环境和XSbase 255嵌入式开发板两部分介绍Qt/Embedded图形界面的移植，最后结合实例阐述了如何开发基于嵌入式Linux的应用程序以及文件系统镜像的制作。

关键词：嵌入式Linux；嵌入式GUI；Qt/Embedded；Qtopia

21世纪被誉为“移动之王”时代，与此同时，嵌入式Linux在手机、PDA等手持信息设备领域的应用出现了一个强劲的发展势头。手持设备的关键在于人机交互技术的体现，所以一个十分友好的图形用户界面(GUI)是必不可少的。

系统平台构建

本文构建的系统是实现一个掌上信息处理终端系统，集PDA应用、网络应用、多媒体应用于一身，并成功运行在XSbase255嵌入式开发板上。整个系统包括四部分：(1)引导装载程序(BootLoader)，这是一段驻留在开发板上的代码，系统上电后首先被执行，对CPU、内存等进行初始化，完成内核映像的装载和引导；(2)Linux内核，是在官方的2.4.18Linux内核基础上，加入了相应的硬件驱动和新的文件系统而构成的；(3)图形用户界面，采用基于Qt/Embedded的Qtopia桌面环境；(4)应用程序的编写与添加。本文重点介绍图形用户界面的移植和添加应用程序。表1给出了整个系统平台的结构。

表1 系统平台构建

GUI	Applications	Games	Settings	Documents		
	Qtopia					
	Qt/Embedded					
Kernel	Arm linux kernel					
				JFFS2		
	Driver			MTD	fb	
Board	CPU	SDRAM	Bootloader	FLASH	LCD	…

硬件平台的选择

采用XSBase255开发板，这是一款比较理想的PDA、手机等等应用的开发系统。采用高性能(400MHz主频)和低功耗的Intel PXA255处理器，64Mb SDRAM以及32MB的FLASH(闪存)组成。640×480分辨率的LG TFT LCD，和触摸屏驱动ADS7843。PXA255处理器是Intel公司新近推出的取代Strong ARM的新一代嵌入式应用处理器，它拥有Thumb压缩指令、64位长乘法指令、扩展型DSP指令等先进特性。PXA255具有众多的扩展接口与无线接口，可支持PCMCIA、Compact Flash、MMC/SD Card、USB、Bluetooth IF、IrDA等设备。

嵌入式GUI的移植

Qt/Embedded选取

嵌入式Linux系统的有代表性的GUI系统主要有MiniGUI、MicroWindows、Tiny X以及Qt/Embedded。这些GUI系统在接口定义、体系结构、功能特性等方面存在着很大的差别。

Tiny-X，是标准X-windows在嵌入式系统的小巧实现，作为一个图形环境，X-window是成功的，但由于在体系接口上的原因，限制了它对游戏、多媒体的支持能力。

MicroWindows，其主要特色在于提供了C/S体系结构，同时也提

本文于2005年4月30日收到。白玉霞：硕士生；刘旭辉：硕士生；孙肖子：教授。三人主要从事嵌入式Linux开发系统研究。

表2 Qt/Embedded与Qt/X11的比较

应用源代码	
Qt API	
Qt/Embedded	Qt/X11
	Qt/XLib
	X Window Server
帧缓冲	
Linux 内核	

供了相对完善的图形功能。但却无任何硬件加速能力，图形引擎中也存在着许多未经优化的低效算法。

MiniGUI，是建立在比较成熟的图形引擎之上，其特点是小巧精致。它尽量保持与Win32的兼容，这样在Win CE应用的场合，也可以使用MiniGUI。

Qt/Embedded，是一个专门为小型设备提供图形用户界面的应用框架和窗口系统。提供了丰富的窗口小部件(Widgets)，并且还支持窗口部件的定制，因此它可以为用户提供漂亮的图形界面。Qt是KDE等项目使用的GUI支持库，所以有许多基于Qt的X Window程序可以非常方便地移植到Qt/Embedded版本上。

最终，在分析和比较了各种GUI的特点后，我们选用Qt/Embedded作为移植对象。

Qt/Embedded底层支持分析

Qt/Embedded以原始Qt为基础，并做了许多出色的调整以适用于嵌入式环境。Qt/Embedded通过Qt API与Linux I/O设施直接交互，成为嵌入式Linux端口。同Qt/X11相比，Qt/Embedded很省内存，因为它不需要一个X服务器或是Xlib库，它在底层摒弃了X lib，采用framebuffer(帧缓冲)作为底层图形接口。同时，将外部输入设备抽象为keyboard和mouse输入事件。Qt/Embedde的应用程序可以直接写内核缓冲帧，这避免开发者使用繁琐的Xlib/Server系统。

Qt/Embedded的底层图形引擎基于framebuffer，framebuffer出现在2.2.x以上内核的版本当中的一种驱动程序接口。这种接口采用mmap系统调用，将显示设备抽象为帧缓冲区。用户可以将它看成是显示内存的一个映象，将其映射到进程地址空间之后，就可以直接进行读写操作了，而写操作可以立即反映在屏幕上。framebuffer驱动程序是最重要的驱动程序之一，正是这个驱动程序才能使系统屏幕显示内容。其实现分为两个方面：一是对LCD及其相关部件的初始化，包括画面缓冲区的创建和对DMA通道的设置；二是对画面缓冲区的读写，具体到代码为read、write等系统调用接口。

表3 宿主机移植所需工具及环境变量声明

tmake1.11	生成和管理Makefile
	TMAKEDIR TMAKEPATH PATH
Qt-X11-2.3.2	qvfb虚拟缓冲帧工具 uic用户界面编译器 Designer Qt应用程序设计工具
	PATH LD LIBRARY_PATH
Qt/Embedded-2.3.7	Qt库支持 libqte.so
	QTEDIR PATH LD_LIBRARY_PATH
Qtopia1.7.0	应用程序开发包桌面环境
	QPEDIR PATH LD LIBRARY PATH

Qt/Embedded和Qtopia的移植

移植过程中我们采取了宿主机和目标板的开发模式。宿主机是一台运行Linux的PC机，目标板即hybus开发板。先在宿主机上调试通过后，再移植到目标板上。

宿主机上的移植

前面介绍过Qt/Embedded直接写入帧缓冲，在宿主机上则是通过qvfb(vitural framebuffer)来模拟帧缓冲。qvfb是X窗口用来运行和测试Qtopia应用程序的系统程序，允许我们在桌面及其上开发Qt嵌入式程序，而不需要在命令台和X11之间来回切换。qvfb使用了共享存储区域(虚拟的帧缓冲)来模拟帧缓冲并且在一个窗口中(qvfb)模拟一个应用来显示帧缓冲，显示的区域被周期性的改变和更新。通过指定显示设备的宽度和颜色深度，虚拟出来的缓冲帧和物理的显示设备在每个像素上保持一致。这样我们在每次调试应用时不需要总是刷新嵌入式设备的FLASH存储空间，从而加速了应用的编译、连接和运行周期。

因此在最初编译配置嵌入式Linux内核时必须使其支持帧缓冲。宿主机上的移植需要的工具及环境变量见表3。其中环境变量可以直

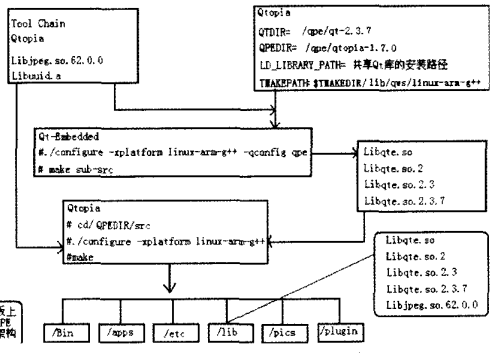

图1 Qtopia配置编译及其架构

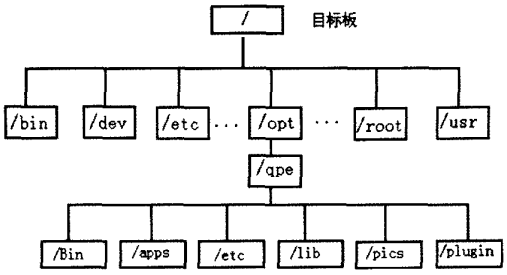

图2 文件系统组织图

接用 export 来声明，也可以在 ~/.bash_profile脚本文件中进行设置。

特别指出，在配置qt-2.3.7时，./configure-qconfig -qvfb -depths 4，8，16，32就是指定Qt嵌入式开发包生成虚拟缓冲帧工具qvfb，并支持4，8，16，32位的显示颜色深度。运行Qt的虚拟缓冲帧工具的方法是：在Linux图形模式下运行命令：./qvfb &。如果要把Qt嵌入式应用程序的显示结果输出到虚拟缓冲帧时，运行时需在程序名后加上－qws选项。如./canvas-qws。

Qtopia即QPE(Qt PalmtopEnvironment)是由著名的Trolltech公司基于Qt的嵌入式版本Qt/Embedded库的基础上，专门针对PDA、智能手机这类运行嵌入式Linux的移动计算设备和手持设备所开发的开放源码的一套应用程序包和开发库。Qtopia是基于qt/embedded程序库编写的应用程序环境(Qtopia是Qt在Linux/embedded Linux版本里的一个application实现。)，界面优美。主要应用于高端手机、PDA等嵌入式系统，具有广阔的发展前景。

目标板上的移植

目标板上的移植与宿主机类似，只需将编译参数做一定的修改即可。图1列出了qtopia移植中qtembedded共享库的支持，环境变量声明和关键的编译配置命令，以及最后目标板上qpe的架构。

添加应用程序到qtopia

如前所述，我们已经安装好了Qtopia应用环境。下面介绍如何在Qtopia里添加我们编写的应用程序(camera)例子，具体Qt程序的编写不在本文内容之内。

1) 建立camera程序的图标文件

制作一个32 × 32大小的PNG格式的图标文件，将此文件存放在Qtopia/pic/inline目录下，然后我们要用到qt-x11-free-3.3.3里的一个工具qembed，将Qtopia/pics/inline下所有的图形文件转换成一个C语言的头文件，此头文件包含了该目录下的图形文件的rgb信息。

2)重新交叉编译qtopia

3)建立.desktop文件，将其保存在qtopia/apps/applications目录下，具体内容可参考qtopia自带应用的.desktop文件。

4)制作文件系统映像

表4 脚本文件的改写

```
.........
PATH:/qpe/bin:$PATH
LD_LIBRARY_PATH=$LD_LIBRARY_PATH=/qpe/lib
QTDIR=/qpe
QPEDIR=/qpe
QWS_MOUSE_PROTO=TPanel:/dev/ts

cd /qpe/bin
export PATH LD_LIBRARY_PATH QTDIR QPEDIR QWS_MOUSE_PROTO
qpe
.........
```

我们需要利用原有的文件系统映像，把新建的应用程序的相关文件加入其中。图2为我们下载到Flash中的JFFS2的文件系统结构。根目录下除opt以外的文件目录都来自原有文件系统。我们首先需要把新建的应用程序的相关文件(包括启动器文件，包含了图标的库文件libqte.so.*和应用程序的可执行文件)复制到qpe的对应的目录下。接下来通过JFFS2工具mkfs.jffs2创建生成新的文件系统映像。利用bootloader将生成的文件系统映像下载后写入flash，从而为内核启动作好了根文件挂载的准备。

5)自动运行

我们对嵌入式系统上的linux启动过程进行了研究，若要使qpe能够自动运行，我们需要改写其脚本文件(表4)，在etc/profile脚本中，做如下添加。

重新运行qtopia，就可以看到我们添加的应用的图标，点击此图标就可以运行此应用程序了。图3是我们编写的Camera程序在Qtopia下的截图。

参考文献：

1. 'Linux Device Driver', Alessandro Rubini & Jonathan Corbet.

图3 添加camera程序后的Qtopia