Qt/E 的嵌入式 Linux GUI 研究与实现

汤 伟 李 强
(杭州电子科技大学计算机学院 浙江 杭州 310018)

摘 要 嵌入式 GUI(Graphical User Interface)为嵌入式系统提供了一种应用于特殊场合的人机交互接口(Man-Machine Interface)。由于嵌入式系统本身的硬件资源有限,要求嵌入式 GUI 具有高度可移植性和可裁剪性,以适应不同的硬件条件和使用需求。首先介绍嵌入式 Linux GUI 目前的发展状况及各自的特点,然后针对目前主流的嵌入式 GUI 系统——Qt/Embedded,阐述其图形引擎的实现。最后,结合三星公司 S3C2410 的开发板,实现了一种嵌入式 GUI 系统在具体平台上的应用。

关键词 ARM9 Linux 交叉编译 嵌入式 GUI Qt Qt/Embedded Qtopia FrameBuffer Signals/Slots

中图分类号 TP316 **文献标识码** A

RESEARCH AND REALIZATION OF Qt/E EMBEDDED LINUX GUI

Tang Wei Li Qiang
(*School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, Zhejiang, China*)

Abstract Embedded GUI provides a man-machine interface for embedded systems that are applied to special occasions. The limitation of hardware resources of the embedded system demands its GUI to be highly portable and tailorable so as to adapt to various hardware platforms and practicalities. The article starts off by introducing the present development of embedded Linux GUIs and their respective characteristics. Next aiming at Qt/Embedded, a mainframe embedded GUI system, the article expounds how to realise its graphic engine. What's more, by integrating Samsung S3C2410 development board, the authors have achieved the application on a specific platform for an embedded GUI system.

Keywords ARM9 Linux cross-compiling Embedded GUI Qt Qt/Embedded Qtopia FrameBuffer Signal/Slots

0 引 言

由于嵌入式系统的特殊性,它一般不会建立在庞大的操作系统以及 GUI 之上,如 Windows 或 X Windows,它对实时性的要求非常高,对 GUI 的要求更高。

1 嵌入式 Linux GUI 概述

目前,以 Linux 为操作系统的嵌入式系统中,常用的 GUI 有 Qt/Embedded、MicroWindows、MiniGUI 及 OpenGUI 等。

1.1 Qt/Embedded

Qt/Embedded 是挪威的奇趣科技(TrollTech)公司(注:该公司已于 2008 年 2 月被诺基亚公司以 1.53 亿美元收购)推出的一个跨平台的 C++图形用户开发界面库。它的主要特点是界面美观、色彩配比好,使用与 Qt/Windows 和 Qt/X11 完全一样的 API 接口,许多基于 Qt 的程序可以非常方便地移植到嵌入式系统中;同时,它具有丰富的模块,用户可以根据需要选择它的特性集合。

1.2 MicroWindows

MicroWindows 是由美国 CenturySoftware 公司开发的开放源码的嵌入式 GUI 项目。它不需要其他图形系统的支持,可以充分利用 Linux 提供的 FrameBuffer 机制来进行图形显示。同时在底层提供了对多种芯片的支持,基本上用 C 语言实现,因此移植性较好。

1.3 MiniGUI

当然,我们国家的 MiniGUI 也是一个比较成熟的图形用户界面系统,面向基于 Linux 的实时嵌入式系统,使用现有成熟的图形引擎(SV2GALib/LibGGI),采用类似 Win32 的线程机制,集成了多字体和多字符集,支持硬件加速能力,充分利用显示内存。

1.4 OpenGUI

OpenGUI 在 Linux 上存在很长时间了。这个库是用 C++编写,提供 C++接口。OpenGUI 支持鼠标和键盘事件,在 Linux 上与 Qt/Embedded 一样,都是基于 FrameBuffer 实现绘图。但 OpenGUI 基于汇编实现内核并利用 MMX 指令进行了优化,因此运行速度快,从而影响了它的可移植性。

2 嵌入式 Linux GUI 软硬件环境

2.1 硬件环境

本文的嵌入式系统是针对三星公司的 S3C2410,它集成

收稿日期:2010-08-10。汤伟,硕士生,主研领域:嵌入式 Linux。

ARM920T 内核,采用 0.18 微米 CMOS 工艺,并在 ARM 核基础上集成了丰富的外围接口,如图 1 所示。

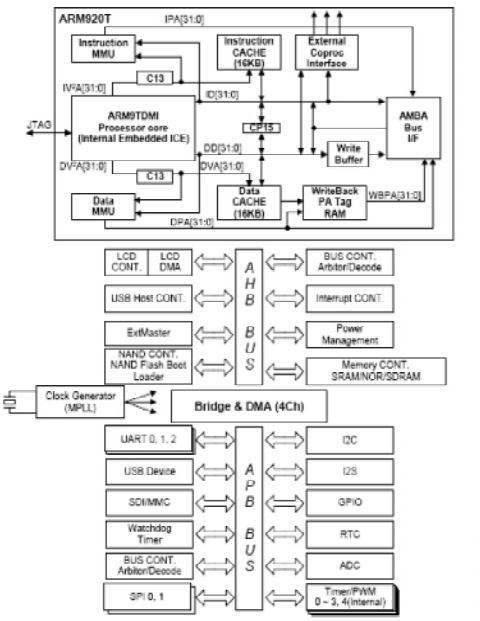

图 1　S3C2410 内部结构

主要功能: 1 个 MMU(实现 Linux 内核的虚拟内存管理)、4 个 DMA 通道、1 个 LCD 控制器(结合 DMA 实现 Linux 内核的 mmap()系统调用)、3 个 UART、2 个 SPI、2 个 USB 主设备端口、1 个看门狗、4 个 PWM 定时器、117 个 I/O 端口、24 个外部中断源、8 路 10 位 ADC 以及触摸屏接口、外部扩展存储器控制器(达 1G,足够 Linux 内核与上层软件使用)等[1]。

外围器件的扩展和电路板的设计主要由硬件工程师完成,不是本文的主要讨论范围,在此只是阐述嵌入式 Linux GUI 的硬件环境。

2.2　软件环境

在完成开发板的设计制作之后,在其上运行 GUI 程序之前,必须要有一定的底层软件环境。如下:

1) 采用宿主机/目标机的开发模式,在宿主机上搭建交叉编译环境[2]

2) ViVi(Linux 的引导程序)的移植

ViVi 是三星公司专门为 S3C2410 加载 Linux 内核。

3) Linux 内核移植和文件系统的制作

内核的移植比较复杂,这也是采用 Linux 的繁琐之处。移植采用 2.4.x 的内核源码,针对相应的硬件平台(S3C2410),修改部分体系结构相关的源码,主要是汇编语言部分,添加相应的驱动(FB"帧缓冲"驱动程序、鼠标、键盘类设备驱动程序等),再裁剪内核,去掉无关选项,将配置好的内核交叉编译后,生成映像文件,烧写到开发板的 flash 中。文件系统采用 jffs2,通过 busybox 制作 Linux 需要的命令,建立基本目录和设备文件等,最后通过 mkfs.jffs2 工具生成映像文件,烧写到 flash 中[3]。

3　Qt/Embedded 图形引擎的实现

Qt/Embedded 图形引擎基于 FrameBuffer,FrameBuffer 是在 Linux 2.2 版本以后推出的标准显示设备驱动接口,采用 mmap()系统调用,可将它的显示缓存映射为可连续访问的一段内存指针[4]。FrameBuffer 的驱动包括两个方面: 一是 LCD 的初始化(ARM9 中集成了 LCD 控制模块)。二是对画面缓冲区的读写,如 read、write 和 lseek 的系统调用。而将画面缓冲区的内容输出到 LCD 上,则由硬件自动完成。DMA 通道和画面缓冲区设置完成后,DMA 开始正常工作,并将缓冲区的内容不断发送到 LCD 上。这个过程是基于 DMA 对 LCD 的不断刷性。基于该特性,FrameBuffer 驱动程序必须将画面缓冲区的存储空间重新映射到一个不加高缓存和写缓存的虚拟地址空间中,这样才能保证应用程序通过 mmap()系统调用将该缓存映射到用户空间后,该画面缓存的写操作能够实时地体现到 LCD 上。

Qt/Embedded 中,Qscreen 类为抽象出的底层显示设备基类,其中声明了对于显示设备的基本描述和操作方式。另一个重要的基类是 QGfx 类,它抽象出对于显示设备的具体操作接口,如选择画线、画矩形和 alpha 操作等。这两个基类是 Qt/Embedded 图形引擎的底层抽象。其中具体函数基本都是虚函数,Qt/Embedded 对具体的显示设备,如 Linux FrameBuffer、Qt Virtual FrameBuffer 做的抽象接口类全都由此继承并重载基类中的虚函数来实现。图 2 给出了 Qt/Embedded 中图形引擎实现的结构框图。

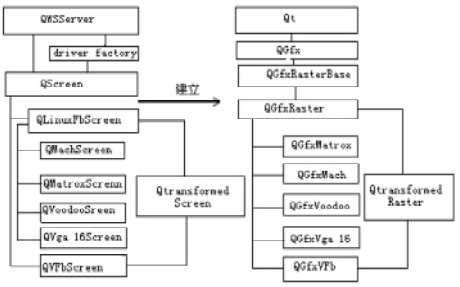

图 2　Qt/Embedded 图形引擎的结构框图

Qt/Embedded 用 QLinuxFbScreen 来处理 FB 设备,针对具体显示硬件(如 Mach 卡和 Voodoo 卡)的加速特性,Qt/Embedded 从 QLinuxFbScreen 和图形设备环境模板类 QGfxRaster 类继承出子类,并针对相应硬件重载相关虚函数。

4　Qt/Embedded 和 Qtopia 的移植

4.1　在宿主机上搭建 Qt/Embedded 环境

本过程需要 qt-x11.2.3.2.tar.gz、e2fsprogs-1.37.tar.gz、qt-embedded-2.3.7.tar.gz、tmake-1.13.tar.gz、qtopia-free-1.7.0.tar.gz 等软件包。qt-x11.2.3.2.tar.gz 提供 uic、qvfb、designer 等开发工具; e2fsprogs-1.37.tar.gz 提供 uuid 头文件和 libuuid.a; qt-Embedded-2.3.7.tar.gz 主要提供 libqte.so; tmake-1.13.tar.gz 生成和管理 Makefile; qtopia-free-1.7.0.tar.gz 实现应用程序的桌面。

由于 Qt/Embedded 和 Qt/X11 有一样的 API,所以我们在开发嵌入式环境的应用程序时,可以先在宿主机上开发,利用强大的 qvfb 工具模拟嵌入式的 GUI 运行环境,通过调试后,再经过重新编译,移植到开发板上,这样就不用每次都刷性开发板上的内容,大大提高了开发 GUI 的速度。

进入宿主机 kde 的图形化界面,打开终端,进入 root 用户模式。将上述软件包拷贝到自己的目录下的 qte 目录,用 vi 编辑器建立 build 安装脚本(设置 QTDIR、QPEDIR、TMAKEDIR、TMAKEPATH、PATH 等环境变量,通过./configure-help 命令分

别配置各个软件包的 Makefile 文件)。有几步需要注意,cp-r ../e2fsprogs-1.37/lib/uuid ./qtopia/include 和 cp ../e2fsprogs-1.37/lib/libuuid.a ./qtopia/lib 帮助 Qtopia 建立时所需要的头文件库文件,如果不加,在编译 qtopia 时会找不到 libuuid.a,编译出错;在编译 qt-embedded 时,根据./configure -help 配置 qt/embedded 库,比如加入对 jpeg、gif 的选项,当然,如果选择它们,将增强应用程序的功能,但也增加了 qt/embedded 库的大小。

运行 build 脚本./build,根据宿主机的处理速度不同,该安装脚本运行时间会有长短,本人用 1G 内存的宿主机,大概四十几分钟即可完成。安装完成之后,就可以开发嵌入式 GUI 程序了,先来测试一下,在终端下输入 qvfb[5] &,让它在后台运行,然后执行 qpe -qws,即可看到 qtopia 的运行界面,如图 3 所示。

图 3 qtopia 的运行界面

4.2 Qt/Embedded 编程核心

在宿主机上建立好开发环境后,就可以在上面开发应用程序了,开发后的程序先在 qvfb 上运行调试,通过之后,再移植到开发板上,方便快捷。

Qt/Embedded 软件体系结构如图 4 所示。该图比较了 Qt/Embedded 和 X11 的体系结构。Qt/Embedded 摒弃了 X lib 库,采用帧缓冲(FrameBuffer)作为底层图形接口。同时将外部输入设备抽象为键盘和鼠标事件。Qt/Embedded 底层图形接口采用 mmap()系统调用,将显示设备抽象为帧缓冲,从而应用程序可以直接写内核 FrameBuffer,避免了使用繁琐的 X lib/Server 机制,这正是资源相对紧张的嵌入式系统所需要的。

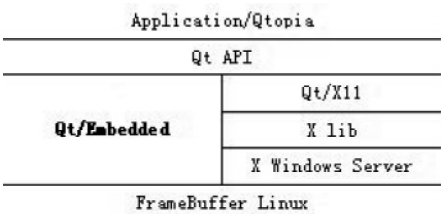

图 4 Qt 的软件层次结构

Signals/Slots 是 Qt 编程的基础,也是核心,如果没有它,就失去了 Qt 的活力。

Signals/Slots 是对象间通信的手段,它不同于传统的回调函数,增加了应用程序的可靠性和安全性。当某个对象的状态发生改变的时候,它发出信号,让另一个或者多个对象的插槽接收,尽管它并不知道哪些函数定义了插槽,而插槽也不知道要怎样接收。Signals/Slots 真正实现了封装的概念,信号与插槽不必一一对应[6]。

下面是一个简单的信号与插槽的部分代码示例:

```
QApplication app(argc,argv);
QPushButton quit("click me",0);
quit.resize(100,30);
quit.setFont(QFont("Times",18,QFont::Bold));
QObject::connect(&quit,SIGNAL(clicked()),&app,SLOT(quit()));
app.setMainWidget(&quit);
quit.show();
return app.exec();
```

QApplication 类负责应用程序的控制流和主要设置,它包括系统设置函数(如 fontMetrics())、事件处理函数(如 exec())、文本处理(如 translate())、高级光标处理(如 overrideCursor())等,处理和调度所有来自窗口系统和其他资源的事件,管理程序的开始和结束。

QObject 类是所有能够处理信号、插槽和事件的 Qt 对象的基类,能够创建带有父对象及其名字的对象,对象的父对象可以看作该对象的所有者。

QObject::connect(&quit,SIGNAL(clicked()),&app,SLOT(quit()))实现了 quit 对象和 app 对象的通信,只要前者一产生 clicked 信号,后者马上作出响应。

编译上述代码,输出 clickme 可执行文件,在 qvfb 模拟环境下,有两种方式执行它。方法一,启动 qvfb,在命令行中直接输入./clickme -qws,即可看到单个 clickme 应用程序在 qvfb 中运行,点击按钮,立即退出;方法二,将该应用程序加到 Qtopia 桌面环境下运行,在执行 clickme 之前,在$ QPEDIR/apps/Applications 目录下新建 clickme.desktop 文件,编辑该文件:

```
[Desktop Entry]
Comment = A click Program
Exec = clickme
Icon =
Type = Application
Name = clickme
Name[no] = Klokke
Name[de] = Uhr
```

然后把可执行文件 clickme 拷贝到$ QPEDIR/bin 下,在后台启动 qvfb,运行 qpe -qws,这是在 Qtopia 界面上就能看到 clickme 应用程序图标,运行之,如图 5 所示,应用程序界面上出现了 clickme 图标,点击它即可运行。

图 5 clickme 应用程序示例

4.3 开发板上的移植

交叉编译 Qt/Embedded,方法同宿主机上编译类似,将编译器换成 arm-linux-gcc,修改 qt-2.3.7/configs/linux-arm-g++-shared 文件,将其中的相应行改为:

SYSCONF_CXX = /home/bull/2.95.3/bin/arm-linux-g++; SYSCONF_CC = /home/bull/2.95.3/bin/arm-linux-gcc; SYSCONF_LINK = /home/bull/2.95.3/bin/arm-linux-gcc; SYSCONF_LINK_SHLIB = /home/bull/2.

95.3/bin/arm-linux-gcc; SYSCONF_AR = /home/bull/2.95.3/bin/arm-linux-ar。

至此,可以编写 Qt 应用程序,在宿主机上通过 qvfb 模拟嵌入式环境下运行,调试通过后,方便地移植到开发板上,做到"一次编写,多次编译"。

5 结　语

本文针对嵌入式 Linux GUI,介绍了目前流行的几种 GUI,阐述了各自的特点说明了嵌入式 GUI 的软硬件环境,然后着重研究了 Qt/Embedded 图形引擎的实现,并通过对该库的裁剪配置,使其尽可能精简,以适应嵌入式系统资源有限的束缚,最后通过一个实例,测试移植效果。Qtopia 作为一款成功的嵌入式 GUI 桌面环境,是 PDA、消费电子等嵌入式产品的优秀解决方案,本文通过实例说明了它的良好性能。

参 考 文 献

[1] 三星. S3C2410 数据手册(arm920T)[M]. 2003.

[2] Karim, Yaghmour. Building Embedded Linux Systems [M]. O'Reilly, 2003.

[3] 周绪宏,梁阿磊,戚正伟. 基于嵌入式 Linux 的智能手机系统软件的设计与实现[J]. 计算机应用与软件,2008,25(3):60-61.

[4] W Richard Stevens, Stephen A Rago. Advanced Programming in the UNIX Environment: Second Edition [M]. Addison Wesley Professional, 2005.

[5] 谭大鹏,李培玉,潘晓弘. 基于 Qt/E 的嵌入式工业监测轻型图形用户界面构件库开发[J]. 计算机集成制造系统,2009,15(2):401.

[6] Jasmin Blanchette, Mark Summerfield. C++ GUI Programming with Qt 4 [M]. Prentice Hall, 2006.

(上接第 197 页)

4) B 将由 CB 盲签名了的新电子现金发送给 C。B 向 C 转发电子现金合并应答(ID_C, ID_B, β^o, T_B, Sgn_B),Sgn_B 为 B 对数据 $H(ID_C, ID_B, \beta^o, T_B)$ 的签名。

5) C 接收合并的新电子现金。C 检查合并的电子现金无误后,计算 $\beta^3 = \beta^o + \gamma^3$,得到电子现金($\beta^3$, α^3, v^3, Q_t^3)。

3 新协议的安全性分析

对本文提出的新电子现金框架,着重在以下三方面进行分析。

3.1 匿名性

本文通过盲签名和匿名的临时公钥密码来实现电子现金的匿名性。因为电子现金的持有者以盲签名的方式将临时公钥密码嵌入到电子现金中,而电子现金基本信息 v 不含敏感的身份信息,所以银行和网上商店不能追踪电子现金的持有者。银行只知道顾客兑换了多少电子现金,网上商店也只记录了顾客的电子现金基本信息,这就达到了电子现金的匿名性效果。

3.2 不可抵赖性

本文提出的框架协议,通过在每个通信消息中嵌入签名的方式来保证不可抵赖性。例如在电子现金发售过程中,顾客向银行发送的请求消息中加入了顾客的签名 Sgn_C。如果顾客拒绝承认此行为,银行可以向法院出示此顾客的签名信息来证实此顾客确实有此行为。另一方面如果顾客没有发送此请求消息,银行也就没有证据可以证明顾客的行为。

另外顾客在购物时,要用电子现金的临时私钥对电子现金进行签名。这样就使此电子现金的持有者不能否认其签了名的消息。另一方面这也曾加了电子现金的安全性,因为只有电子现金的持有者拥有电子现金的临时私钥。

3.3 抗攻击性

首先,本协议中所有的通信消息都是在安全信道(*SSL* 或 *STL*)上进行的。除了通信双方知道消息内容,其他人是无法获得通信内容的。

其次,本协议也可抵抗重发和篡改等主动攻击。每次通信消息中都包含有时间戳,接收方可以十分容易地发现重发的消息。如果不怀好意者篡改或是假冒电子现金参与的某一方时,接收者通过验证消息中的数字签名可以很容易地辨别出来。

4 结　语

本文提出的电子现金方案的特点是更接近现实中的真实货币。电子现金由中央银行统一定制,而普通银行的职责类似于现实生活中的银行职责,只是对从中央银行得到的电子现金进行发售。顾客可在任何一家网上商店使用电子现金,并可找回电子现金余额。如果电子现金是由政府的金融部门统一发售,各银行建立相应的电子现金业务部门,就可以保证电子现金更广的流通性,同时也方便了政府对电子现金的管理。

参 考 文 献

[1] Despoina Palaka, Petros Daras, Kosmas Petridis, et al. A novel peer-to-peer payment protocol [J]. International Journal of Network Security, 2007, 4(1):107-120.

[2] Dimitrios Lekkas, Diomidis Spinellis. Implementing regular cash with blind fixed-value electronic coins [J]. Computer Standards & Interfaces, 2007, 29(3):277-288.

[3] Rivest R, Shamir A. Payword and micromint: Two simple micropay-ment schemes [C]//Proceedings of Security Protocols Workshop, LNCS 1189, Springer-Verlag, Springer-Verlag. 1997:69-87.

[4] Sangjin Kim, Heekuck Oh. Efficient anonymous cash using the hash chain [J]. IEICE Transactions on Communications, 2003, E86-B(3): 1140-1143.

[5] Ronggong Song, Larry Korba. How to make ecash with non-repudiation and anonymity [C]//International Conference on Information Technology: Coding and Computing (ITCC04), 2004, 2: 167-172.

[6] Alan O Freier, Philip Karlton, Paul C Kocher. SSL 3.0 Transport Layer Security Working Group [M/OL]. November 1996. http://tools.ietf.org/html/draft-ietf-tls-ssl-version3-00.

[7] Dierks T, Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.2 [M/OL]. RFC5246, August 2008. http://tools.ietf.org/html/rfc5246.

[8] Darrel Hankerson, Alfred Menezes, Scott Vanstone. 椭圆曲线密码学导论[M]. 张焕国,等译. 电子工业出版社,2005:13-18,165-177.

[9] 张方国,王常杰,王育民. 基于椭圆曲线的数字签名与盲签名[J]. 通信学报,2001,22(8):26-27.