嵌入式Linux系统中GUI系统的研究与移植

■ 北京航空航天大学 徐广毅 张晓林 崔迎炜 杨欣昕 吴小伟

摘 要 针对嵌入式Linux系统中几种常见的GUI（Graphic User Interface）系统，讨论嵌入式GUI实现的底层支持方式；详细分析Microwindows、MiniGUI、Qt/Embedded等三种GUI的实现特点、体系结构、API接口。结合这三种嵌入式GUI在以Motorola i.MX1为核心的实际应用系统中移植开发的问题，讨论移植技术与中文化技术。

关键词 嵌入式Linux GUI 应用与移植 中文化

引 言

嵌入式GUI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。嵌入式GUI要求简单、直观、可靠、占用资源小且反应快速，以适应系统硬件资源有限的条件。另外，由于嵌入式系统硬件本身的特殊性，嵌入式GUI应具备高度可移植性与可裁减性，以适应不同的硬件条件和使用需求。总体来讲，嵌入式GUI具备以下特点：

- ◆ 体积小；
- ◆ 运行时耗用系统资源小；
- ◆ 上层接口与硬件无关，高度可移植；
- ◆ 高可靠性；
- ◆ 在某些应用场合应具备实时性。

1 基于嵌入式Linux的GUI系统底层实现基础

一个能够移植到多种硬件平台上的嵌入式GUI系统，应该至少抽象出两类设备：基于图形显示设备(如VGA卡)的图形抽象层GAL(Graphic Abstract Layer)，基于输入设备(如键盘，触摸屏等)的输入抽象层IAL(Input Abstract Layer)。GAL层完成系统对具体的显示硬件设备的操作，极大程度上隐藏各种不同硬件的技术实现细节，为应用程序开发人员提供统一的图形编程接口。IAL层则需要实现对于各类不同输入设备的控制操作，提供统一的调用接口。GAL层与IAL层的设计概念，可以极大程度地提高嵌入式GUI的可移植性，如图1所示。

目前应用于嵌入式Linux系统中比较成熟，功能也比较强大的GUI系统底层支持库有SVGA lib、LibGGI、X Window、framebuffer等。

图1 一种可移植嵌入式GUI的实现结构

2 三种嵌入式GUI系统的分析与比较

2.1 Microwindows

Microwindows是一个典型的基于Server/Client体系结构的GUI系统，基本分为三层，如图2所示。

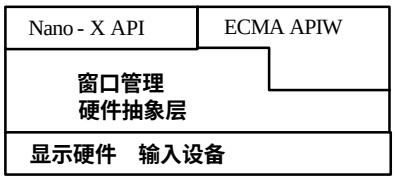

图2 Microwindows的体系结构

最底层是面向图形显示和键盘、鼠标或触摸屏的驱动程序；中间层提供底层硬件的抽象接口，并进行窗口管理；最高层分别提供兼容于X Window和ECMA APIW(Win32 子集)的API。其中使用Nano-X接口的API与X接口兼容，但是该接口没有提供窗口管理，如窗口移动和窗口剪切等高级功能，系统中需要首先启动nano-X的Server程序nanox-server和窗口管理程序nanowm。用户程序连接nano-X的Server获得自身的窗口绘制操作。使用ECMA APIW编写的应用程序无需nanox-server和nanowm，可直接运行。

Microwindows提供了相对完善的图形功能和一些高级的特性，如Alpha 混合、三维支持和TrueType字体支持等。该系统为了提高运行速度，也改进了基于Socket

套接字的X实现模式，采用了基于消息机制的Server/Client传输机制。Microwindows也有一些通用的窗口控件，但其图形引擎存在许多问题，可以归纳如下：

◆ 无任何硬件加速能力 ；

◆ 图形引擎中存在许多低效算法，如在圆弧绘图函数的逐点判断剪切的问题。

由于该项目缺乏一个强有力的核心代码维护人员，2003年Microwindows推出版本0.90后，该项目的发展开始陷于停滞状态。

2.2 MiniGUI

MiniGUI是由国内自由软件开发人员设计开发的，目标是为基于Linux的实时嵌入式系统提供一个轻量级的图形用户界面支持系统。MiniGUI的体系架构如图3所示。

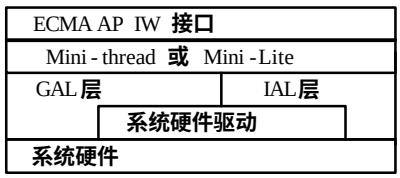

图3 MiniGUI的实现架构

MiniGUI分为最底层的GAL层和IAL层，向上为基于标准POSIX接口中pthread库的Mini-thread架构和基于Server/Client的Mini-Lite架构。其中前者受限于thread模式对于整个系统的可靠性影响——进程中某个thread的意外错误可能导致整个进程的崩溃，该架构应用于系统功能较为单一的场合。Mini-Lite应用于多进程的应用场合，采用多进程运行方式设计的Server/Client架构能够较好地解决各个进程之间的窗口管理、Z序剪切等问题。MiniGUI还有一种从Mini-Lite衍生出的standalone运行模式。与Lite架构不同的是，standalone模式一次只能以窗口最大化的方式显示一个窗口。这在显示屏尺寸较小的应用场合具有一定的应用意义。

MiniGUI的GAL层支持SVGA lib、LibGGI、基于framebuffer的native图形引擎以及哑图形引擎等，对于Trolltech公司的QVFB在X Window下也有较好的支持。IAL层则支持Linux标准控制台下的GPM鼠标服务、触摸屏、标准键盘等。

MiniGUI下丰富的控件资源也是MiniGUI的特点之一。当前MiniGUI的最新版本是1.3.3。该版本的控件中已经添加了窗口皮肤、工具条等桌面GUI中的高级控件支持。

2.3 QT/Embedded

Qt/Embedded是著名的Qt库开发商Trolltech公司开发的面向嵌入式系统的Qt版本。因为Qt是KDE等项目使用的GUI支持库，许多基于Qt的X Window程序因此可以非常方便地移植到Qt/Embedded上。Qt/Embedded同样是Server/Client结构。

Qt/Embedded延续了Qt在X上的强大功能，在底层摒弃了X lib，仅采用framebuffer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，底层接口支持键盘、GPM鼠标、触摸屏以及用户自定义的设备等。

Qt/Embedded类库完全采用C++封装。丰富的控件资源和较好的可移植性是Qt/Embedded最为优秀的一方面。它的类库接口完全兼容于同版本的Qt-X11，使用X下的开发工具可以直接开发基于Qt/Embedded的应用程序QUI界面。

与前两种GUI系统不同的是，Qt/Embedded的底层图形引擎只能采用framebuffer。这就注定了它是针对高端嵌入式图形领域的应用而设计的。由于该库的代码追求面面俱到，以增加它对多种硬件设备的支持，造成了其底层代码比较凌乱，各种补丁较多的问题。Qt/Embedded的结构也过于复杂臃肿，很难进行底层的扩充、定制和移植，尤其是用来实现signal/slot机制的moc文件。

Qt/Embedded当前的最新版本为3.3.2，能够支持Trolltech的手持应用套件Qtopia的Qt/Embedded最高版本为2.3.8。Trolltech公司将于2004年末推出以Qt/Embedded 3为基础的Qtopia 2应用套件。

3 三种嵌入式GUI的移植与中文化

在进行以上三种嵌入式GUI的研究和移植过程中，硬件平台采用自行设计的以Motorola MC9328 MX1为核心的开发系统。该系统采用CPU内部LCD控制器和320×240分辨率的16bpp TFT LCD作为显示设备，使用I^2C总线扩展出16按键的键盘，同时配置了9位A/D量化精度的电阻触摸屏作为鼠标类输入设备；同时移植了ARM Linux作为操作系统。以下分别讨论这三种嵌入式GUI的底层移植和中文化技术。

移植以上三种嵌入式GUI系统，需要首先实现Linux内核中的framebuffer驱动。对应于开发系统为MC9328中的LCD控制器，该部分驱动程序必须以静态方式编译进内核，在系统启动时由传递进内核的启动参数激活该设备。I^2C键盘的驱动程序和触摸屏的驱动程序实现后，作为Linux内核模块在使用时动态加载。

3.1 Microwindows的移植

Microwindows驱动层相应的源码目录为src/drivers/。其中以scr*开头的源码是针对显示设备的驱动接口，以mou*开头的源码文件为鼠标设备(包括触摸屏)的驱动接口，以kbd*开头的源码文件针对键盘设备的驱动接口。移植过程中需要实现自己的设备驱动接口提供给Microwindows使用，就必须按照指定的接口格式编写相

应的scr、mou、kbd的底层支持。这种方式实现简单，条理也很清晰。

显示设备驱动接口：Microwindows的图形发生引擎支持framebuffer，修改src/中的config文件指定使用framebuffer作为底层图形支持引擎；但需要注意嵌入式Linux的framebuffer较少支持控制台字符模式，需要修改Microwindows中对framebuffer的操作部分以关闭显示模式的转换。

鼠标驱动接口移植说明如表1所列；键盘驱动接口移植说明如表2所列。

表1 MicroWindows Mouse驱动接口说明

函数原型指针	说 明
Int (*Open)(struct _mousedevice *)	打开鼠标类设备
void (*Close)(void)	关闭鼠标类设备
Int (*GetButtonInfo)(void)	获得该鼠标设备支持的按键类型
void (*GetDefaultAccel)(int *pscale,int *pthresh)	获得该类鼠标非线性的移动行为参数
Int (*Read)(MWCOORD *dx, MWCOORD *dy, MWCOORD *dz, int *bp)	获取鼠标位置信息和点击信息
Int (*Poll)(void)	针对某些无select调用的系统实现的poll调用替代

表2 Microwindows keyboard驱动接口说明

函数原型指针	说 明
Int (*Open) (struct _kbddevice *pkd)	打开键盘设备
void (*Close) (void)	关闭键盘设备
void (*GetModifierInfo) (MWKEYMOD *modifiers, MWKEYMOD *curmodifiers)	获取组合键相应状态信息
int (*Read) (MWKEY *buf, MWKEYMOD *modifiers, MWSCANCODE *scancode)	获取键盘按键状态信息，buf参数返回系统虚拟键编号，scancode返回键盘编号
Int (*Poll)(void)	针对某些无 select 调用的系统实现的poll调用替代

在应用程序开发移植中需要注意的是：使用ECMA APIW接口设计的程序无需nano-X的Server程序和nanowm，如图2所示。系统中可以直接启动使用该接口编写的用户程序；但需要注意的是，一个系统中如同时存在使用两种不同API接口编写的进程，会造成nano-X的Server与ECMA APIW的进程对系统硬件资源的使用竞争，双方的程序将无法正常显示或响应用户输入。

在为Microwindows增加中文显示的支持时，主要工作包括两个部分。一部分是系统字体的中文支持。此处使用等宽光栅字体，主要负责窗口标题和内置控件的中文绘制，将字体编译进Microwindows内核中，光栅信息作为一维数组，显示时按照字符偏移量从该数组中调出相应的光栅信息显示即可。除此之外，当程序调用CreateFont时，需要在内部实现为打开文件系统中的字体文件。通过修改src/engine/devfont.c中的GdCreateFont部分，添加相应的hzk(汉字库)支持，便可以实现在CreateFont时创建出一个支持GB2312字符集的逻辑字体，并使用外部字体进行显示。在应用程序设计时，如果没有调用SelectObject将外部字体选入，中文显示时将默认使用系统字体。

3.2 MiniGUI

由于MiniGUI较好地将硬件设备抽象为GAL层和IAL层，移植时只需要针对自身的硬件特点按照GAL层调用接口和IAL层调用接口来做内部实现即可。图4为MiniGUI的GAL层结构示意，IAL层结构类似。

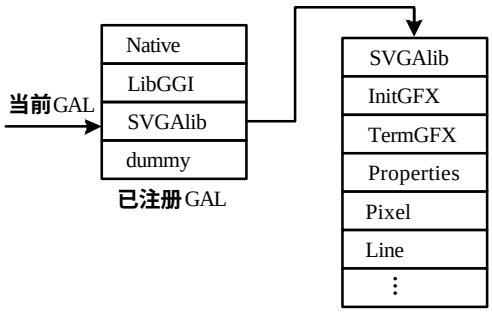

图4 MiniGUI的GAL层结构示意

实现了framebuffer的Linux驱动后，配置MiniGUI选择Native的GAL引擎，便可以使用framebuffer作为MiniGUI的图形发生引擎。其中与GAL层相关的源码说明如表3所列。

MiniGUI的IAL层将输入设备的输入事件最终映射

表3 MiniGUI的GAL层源码文件

文 件	说 明
src/include/gal.h	根据NEWGAL定制与否选择相应GAL支持的头文件（如下）
src/include/newgal.h	NEWGAL支持的头文件，声明NEWGAL接口
src/include/oldgal.h	一般GAL支持的头文件,声明OLDGAL接口
src/gal/gal.c	初始化GFX，根据GFX数组和配置文件所指定的GAL初始化cur_gfx的GAL接口
src/gal/libggi.h & libggi.c	LibGGI实现的GAL
src/gal/svgalib.h & svgalib.c	SVGALib实现的GAL
src/gal/vga16.h & vga16.c	VGA16色模式下的GAL（需SVGALib支持）
src/gal/native/native.h & native.c	Native GAL的GFX初始化实现等，依赖于具体的设备显示能力，调用src_fb.c中打开fb设备时,调用具体的fb.h中指定的子设备psd驱动模块
src/gal/src_fb.c	打开fb设备，调用fb.h中接口选择子设备
src/gal/fb.h & fb.c	子设备选择操作接口
src/gal/fblin1.c … …	具体的子设备psd操作接口

为GUI系统API层的消息事件。IAL层默认处理两种设备的输入操作：键盘设备和鼠标设备。键盘设备向上层提供不同的按键输入信息，鼠标设备提供点击、抬起和落笔坐标等的信息。在实现MiniGUI与输入设备驱动的接口时，采用Select的方式获得输入设备的动作，并转换为消息队列中的消息。消息参数按照Win32接口定义为点击键编号或鼠标当前的坐标(其中触摸屏事件与鼠标事件类似)。通过编写针对硬件开发系统的IAL支持代码，实现了IAL层的移植。其中MiniGUI的IAL层代码说明如表4所列。

表4 IAL层相关接口说明

接口函数	说 明
static int mouse_update(void)	更新鼠标状态
static void mouse_getxy(int *x, int* y)	获取鼠标当前所在位置
static int mouse_getbutton(void)	获取鼠标当前所按下按钮编号
static int keyboard_update(void)	更新键盘状态
static const char* keyboard_getstate(void)	获取键盘当前状态
static int wait_event (int which, fd_set *in, fd_set *out, fd_set *except, struct timeval *timeout)	事件等待。该函数采用select方式监测键盘和鼠标设备，并从相关事件的设备中读取数据
BOOL InitIPAQInput (INPUT* input, const char* mdev, const char* mtype)	初始化IAL设备
void TermIPAQInput (void)	关闭IAL设备

MiniGUI中多字体和多字符集支持是通过设备上下文(DC)的逻辑字体(LOGFONT)实现的，创建逻辑字体时指定相应的字符集，其内部实现为对于所需显示字符的所属字符集的识别处理，最终调用相应字符集的处理函数族。应用程序在启动时，可切换系统字符集，如GB2312、BIG5、EUCKR、UJIS。MiniGUI的这种字符集支持方式不同于采用UNICODE的解决方案。在节省系统资源的意义上讲，这种实现更加适合于嵌入式系统应用，是MiniGUI的一大创新点。MiniGUI同时支持包括ttf、bdf、type 1、vbf等多种字体格式，可以根据需要配置MiniGUI来支持相应字体的显示。

3.3 Qt/Embedded的移植

Qt/Embedded的底层图形引擎完全依赖于framebuffer，因此在移植时需考虑目标平台的Linux内核版本和framebuffer驱动程序的实现情况，包括分辨率和颜色深度等在内的信息。当前嵌入式CPU大多内部集成LCD控制器，并支持多种配置方式。除少数CPU低色彩配置时的endian问题外，Qt/Embedded能够较好地根据系统已有的framebuffer驱动接口构建上层的图形引擎。

Qt/Embedded图形发生引擎中的图形绘制操作函数都是由源码目录src/kernel/中的src/kernel/qgfxraster_qws.cpp中所定义的QGfxRasterBase类发起声明的。对于设备更加底层的抽象描述，则在src/kernel目录中的qgfx_qws.cpp中的QScreen类中给予相应定义。这些是对framebuffer设备直接操作的基础，包括点、线、区域填充、alpha混合、屏幕绘制等函数均在其中定义实现。在framebuffer驱动程序调试通过后，配置Qt/Embedded的编译选项，可以保证Qt/Embedded的图形引擎正常工作。

Qt/Embedded中的输入设备，同样分为鼠标类与键盘类。其中鼠标设备在源码目录中的src/kernel/qwsmouse_qws.cpp中实现，从该类又重新派生出一些特殊鼠标类设备的实现类，其派生结构如图5所示。

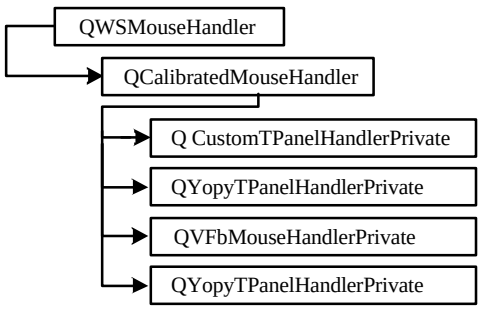

图5 QWSMouseHandler类派生结构

根据具体的硬件驱动程序实现的接口，可以实现类似的接口函数。其中相关的接口函数如表5所列。

表5 Qt/Embedded中触摸屏类接口

函数接口	描 述
QcustomTPanelHandlerPrivate ::QCustomTPanelHandlerPrivate (MouseProtocol, QString)	打开触摸屏类设备，连接触摸屏点击事件与处理函数
QcustomTPanelHandlerPrivate ::~QCustomTPanelHandlerPrivate()	关闭触摸屏类设备
struct CustomTPdata	驱动程序返回的点击数据结构
void QcustomTPanelHandlerPrivate ::readMouseData()	触摸屏点击数据获取

Qt/Embedded中对于键盘响应的实现函数位于src/kernel/qkeyboard_qws.cpp中，在qkeyboard_qws.h中，定义了键盘类设备接口的基类QWSKeyboardHandler。具体的键盘硬件接口依然要建立在键盘驱动程序基础上，移植时需要根据键盘驱动程序从该类派生出实现类，实现键盘事件处理函数processKeyEvent()即可。

Qt/Embedded内部对于字符集的处理采用了UNICODE编码标准。Qt/Embedded同时支持两种对于其它编码标准(如GB2312和GBK)的支持方式：静态编译和动态插件装载。通过配置config.h文件添加相应的编码支持宏定义，可以获得其它编码标准向UNICODE

17

的比较。

4.3 任务切换时间和中断延迟时间

任务切换时间和中断延迟时间是评估RTOS性能的两个重要指标。任务切换时间可以反映出RTOS执行任务的速度，而中断延迟时间可以反映出RTOS对外界变化的反应速度。表3为这两种操作系统任务切换时间和中断延迟时间的比较。

表3 任务切换时间和中断延迟时间的比较

	任务切换时间/μs	中断延迟时间/μs	测试环境
μC/OS-II	29.7~34.2	78.8	Inte180186(33MHz)
eCos	15.84	19.2	MPC860A3(33MHz)

4.4 对硬件的支持

μC/OS-II和eCos支持当前流行的大部分嵌入式CPU，都具有很好的可移植特性。μC/OS-II支持从8位到32位的CPU；而eCos可以在16位、32位和64位等不同体系结构之间移植。μC/OS-II和eCos由于本身内核就很小，经过裁剪后的代码最小可以分别为小于2KB和10KB，所需的最小数据RAM空间可以为4KB和10KB，因此它们对硬件的要求很低，具有极高的经济性。

结 语

通过比较可以看到：μC/OS-II相对eCos来说，源代码量小很多，特别适合学习和研究。它最大的特点是小巧，适合应用在一些RAM和ROM有限的小型嵌入式系统中，如单片机系统。eCos最大的特点是配置灵活，适合于用在一些商业级或工业级的嵌入式系统，如一些消费电子、汽车领域等等。总之，选用什么样的操作系统，要根据目标系统的硬件条件和用户应用程序的复杂程度来确定。

参考文献

1 Labrosse Jean J. μC/OS-II源码公开的实时嵌入式操作系统[M]. 邵贝贝译. 北京: 中国电力出版社, 2001

2 蒋句平. 嵌入式可配置实时操作系统eCos开发与应用[M]. 北京: 机械工业出版社, 2003

3 桑楠. 嵌入式系统原理及应用开发技术[M]. 北京: 北京航空航天大学出版社, 2002

4 黄玉东, 朱华杰. 浅论嵌入式系统[J]. 沈阳电力高等专科学校学报, 2003(10)

(收稿日期：2004-06-07)

14 的转换支持，从而在QFont类中得以转换与显示。由于UNICODE涵盖了中文部分，Qt/Embedded对中文支持也非常好。

Qt/Embedded能够支持TTF、PFA/PFB、BDF和QPF字体格式。由于自身采用UNICODE编码方式对字符进行处理，在一定程度上导致了所能够使用的字体文件体积的增大。为了解决这一问题，Qt/Embedded采用了QPF格式，使用makeqpf等工具可以将TTF等格式的字体转换至QPF格式。图6为笔者在自行设计的MC9328系统上移植Qt/Embedded和Qtopia套件后，增加中文支持后的显示截图。Qt/Embedded版本为2.3.7，Qtopia版本为1.7.0。

图6 Qt/Embedded和Qtopia中文化显示

4 结 论

综上所述，一个具备良好移植性的嵌入式GUI系统，其底层接口应该在很大程度上隐藏具体硬件的实现细节，抽象出GAL与IAL层。对多字符集的支持，也可以从MiniGUI的字符集支持方式和Qt/Embedded的UNICODE支持方式上获得启发。

参考文献

1 Greg Haerr. Microwindows 0.89 Architecture. Censoft Inc. 1999

2 北京飞漫软件技术公司. MiniGUI用户手册, 2003

3 罗从难, 耿增强, 李小群, 等. 嵌入式的图形用户界面. 测控技术, 2000

4 陈泓, 毛洋林, 潘志浩. 基于嵌入式Linux的图形界面显示系统的设计. 微计算机信息, 2004

徐广毅：硕士研究生，主要研究方向为嵌入式Linux系统、实时嵌入式电子飞行信息系统等。张晓林：教授、博士生导师，主要研究方向为通信与信息系统、嵌入式系统、无人飞行器遥控遥测、集成电路设计。

(收稿日期：2004-06-03)