基于 Linux 的嵌入式图形用户界面的研究与应用①

Research and Application of Embedded Graphic User Interface Based on Linux

陈孝文[1]　陈蜀宇[2]　马同杰[1]

(1.重庆大学 计算机学院 重庆 400030; 2.重庆大学 软件学院 重庆 400030)

摘　要： 基于 Linux 操作系统和嵌入式系统两个热点领域，分析对比了当前嵌入式 GUI 的特点之后，深入研究了 Linux 平台下的嵌入式 GUI 的分层模块，研究了消息驱动、窗口管理、输入输出和窗口裁剪等几个重要 GUI 模块的原理，设计实现了一个基于 Linux 的 GUI 原型系统。

关键词： 嵌入式 Linux 图形用户界面 GUI

1　引言

嵌入式图形用户界面(GUI)不仅具有相当好的应用前景，同时对计算机图形学、操作系统、面向对象软件理论、数字图像和汉字信息处理技术等具有重要的理论价值[1]。嵌入式 GUI 有两层含义:一是指嵌入式系统中以图形方式工作的 UI 模块或者任务;二是指为设计和实现嵌入式系统 UI 模块或者任务而采用的应用开发组件。在具体上下文中，两者仅有其一恰当。为遵循习惯，默认情况下嵌入式 GUI 是指应用开发组件，即第二层含义。

目前国内外在 GUI 的研究开发中已经取得了很大的进步，已经在商业上得到广泛的应用，在国内也有不少高校、科研单位以及公司参与到 GUI 的研发中来，并且已经取得了初步的成果。目前嵌入式系统上的 GUI 的实现方法各有不同，大致可划分为如下几类:

(1) 某些大型厂商有能力开发满足自身需要的 GUI 系统。如 Microsoft 的 WinCE 和 SUN 的 Personal Java 等。

(2) 某些厂商没有将 GUI 作为一个软件层从应用程序中剥离，GUI 的支持由应用程序自己负责;这是一种相对临时的解决方案，利用这种手段编写的程序，无法将显示逻辑和数据处理逻辑划分开来，从而导致程序结构不好，不便于调试，并导致大量的代码重复。

(3) 还有些厂商采用目前比较成熟的图形用户界面系统,如 MiniGUI, MicroWindows,Qt/Embedded 或者其他的 GUI 系统。

目前看来，在 Linux 上进行嵌入式系统开发的厂商，一般选择如下几种 GUI 系统:紧缩的 X Window 系统、MiniGUI, MicroWindows, OpenGUI, Qt/Embedded 等[2]。

随着嵌入式设备的硬件条件提高，对于嵌入式系统中轻量级图形用户界面的需求也就越来越迫切。这些系统一般不希望建立在庞大累赘的、非常消耗资源的操作系统和图形用户界面之上，比如 Windows 或 X Window。同时，嵌入式系统对图形用户界面轻型和可定制方面有较高的要求，它们希望图形用户界面占用资源少、高性能、高可靠性、易移植、可配置。

本文的选题正是结合 Linux 操作系统和嵌入式系统研究这两个热点领域，在研究对比了现有的几种嵌入式用户界面后，分析出各种系统的优势和劣势;在此基础之上，扬长避短，开发出一个更适合于嵌入式特点的图形用户界面原型。

2　GUI的分层模块结构

GUI 系统应用的总体层次包括上层应用、窗口系统和操作系统，本文讨论的是处于中间的窗口系统，其结构图如图 1 所示。

① 基金项目:重庆市信息产业发展基金(200721003)
收稿时间:2009-03-17

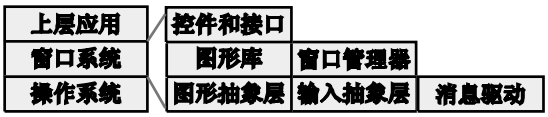

图 1 GUI 系统与分层模块结构

总体来说，窗口系统可以分为三层：基础层、展现层和扩展层。基础层包括图形抽象层(Graphics Abstract Layer,GAL)、输入抽象层(Input Abstract Layer，IAL)和消息驱动和消息驱动，展现层包括窗口管理器和图形库，扩展层包括控件和其他上层接口。

2.1 基础层

基础层涉及窗口系统与操作系统相关联的功能，包括图形抽象层、输入抽象层和消息驱动三大模块。

建立在系统硬件驱动之上的是图形抽象层和输入抽象层，是 GUI 和系统硬件相连的接口，通过这两套接口，分离了窗口系统与特定硬件驱动，增强了 GUI 的扩展性。消息驱动是系统各部分之间通信的机制，它是保证系统效率的关键。

2.2 展现层

展现层涉及窗口关系处理与显示的功能，包括窗口管理器和图形库两大模块，窗口管理其是 GUI 的核心，负责协调各个窗口的叠放关系和响应窗口命令。图形库是对高级图形绘制功能的封装。它位于窗口管理器和图形抽象层之间，提供一些基本的画点、划线、矩形填充以及显示文字和图像的函数。

2.3 扩展层

扩展层包括系统控件以及其他上层接口，控件是一组预先定义的窗口元素，使得用户可以通过输入设备向系统提供特定形式的数据，用于简化上层应用开发。

总体来说，采用上述分层模块化设计可以在满足 GUI 功能的同时，支持系统设计最大限度的个性化，达到可靠性高，扩展性好等指标。

3 GUI的系统结构设计与实现

3.1 系统总体结构

对于嵌入式 GUI，客户端/服务器模型是系统设计的首选。由于客户端和服务器处于同一运行环境中，可以同时对输出设备进行操作，使得客户端分担了部分屏幕显示任务，减轻了服务器的负担，同时减少了通信开销。系统的总体架构如图 2 所示。

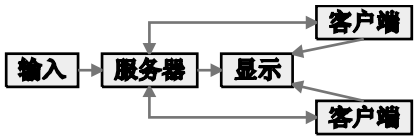

图 2 GUI 系统架构图

这种体系结构具有两个特性，分别对应与嵌入式系统高可靠性和高效率的设计要求：

(1) 基于多进程的 C/S 模式

对于功能单一的运用环境，如实时控制系统，往往采用单进程多线程模式的 GUI，所有数据和操作共享同一进程空间，具有较小的系统开销。这样做也同时导致了系统的脆弱性，基于多进程的多客户端单服务器的结构，系统比较健壮。

(2) 客户端与服务器的显示同步

采用 C/S 结构的 GUI，客户端窗口的建立、激活、移动和销毁等一般通过套接字传送到服务器，由服务器完成实质性工作，这样的系统非常依赖于套接字通讯，而套接字通讯要经过内核，这样浪费很多系统资源，造成系统整体效率低下。为了减轻服务器端的负荷，本系统设计为的为只是把设备输入收集和窗口管理完全交给服务器，而消息事件的响应和图形输出则由服务器和客户端各自独立完成，双方只保持必要的反馈和同步。这样的设计把进程间通信限制在较低的水平，从而减少系统的资源消耗，达到整体的高效率。

3.2 主要模块的功能

如图 3 所示，系统根据功能主要分为进程间通信模块、消息驱动模块、窗口剪裁模块、输入模块和显示模块。其中进程间通信模块负责其他模块的数据交流，是最基础的部分。

(1) 进程间通信模块

进程间通信模块负责建立并维护客户端和服务器端的数据连接，处理服务器与客户端之间的消息传递。它主要包括消息发送和消息接收两个子模块。

服务器响应来自客户端的请求并建立连接之后，消息接收模块根据对方的请求产生特定的消息转发给消息处理模块和窗口剪裁模块，由后两者做响应处理。处理的结果如果涉及到其他客户端进程，由消息发送模块通过与指定进程的连接向目标进程发送消息数据。由于数据连接是基于流式的字节序传输，需要同时监听各个连接，一旦一个连接有效数据流到达，立

即进行消息的转换。

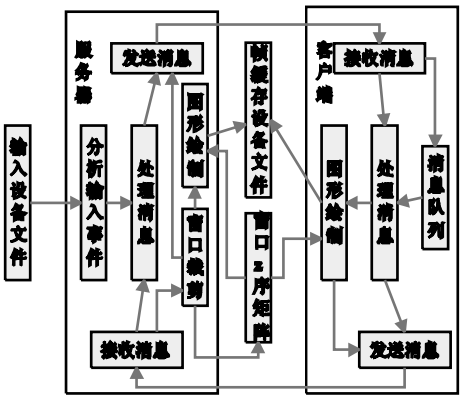

图 3　系统功能模块

(2) 消息驱动模块

消息驱动[3]模块在整个系统中具有非常重要的作用，它采用基于消息分类和消息队列的机制，主要包含消息产生、传递和处理三个部分。

对于客户端进程，消息主要由进程本身和接收到的服务器通信数据转化而来。客户端通过进程间通信模块向服务器发送消息，服务器对于消息进行分拣，将事件类消息交由响应的处理函数处理，将套接字类消息交由窗口裁剪模块处理。服务器对消息的处理涉及到其它客户端时，会通过进程间通信模块向对应的客户端发送消息，由客户端分拣处理。

对于服务器进程，除了进程本身和其他进程的通信数据之外，还有一个重要的消息的来源，就是输入事件。所有的输入事件由服务器进行解析，把鼠标点击、键盘输入抓换成双击、按键消息，将其发送到当前激活的窗口。

(3) 窗口剪裁模块

窗口裁剪模块是系统的核心，，其主要的任务是维护窗口 z 序矩阵和进行窗口裁剪，更新剪切域。在客户端连接/断开时，服务器将该窗口的 id 加入/移出窗口 z 序，并将整个会话过程中根据窗口命令更新该窗口的 z 序，上述改变同时也导致窗口 z 序矩阵的更新。窗口裁剪是在窗口 z 序的基础上，使用窗口裁剪算法重新计算各个窗口的剪切域，使之正确反映变化之后的各个窗口显示效果。对于一个窗口 z 序的改变，裁剪算法由最上层窗口，以此裁剪下面的窗口，分割各窗口的剪切域，之后从第二层开始重复刚才的操作，直至迭代到这个窗口为止。

(4) 显示模块和输入模块

这两个模块主要实现了窗口操作在显示设备上的更新和从输入事件到消息的转化。显示模块包括了图形绘制和显示同步两部分。其中最关键的部分是显示同步的实现，即各个进程使用共享的窗口 z 序矩阵判断自己的无效区，使用帧缓存映射机制自行完成这部分区域的显示更新。

3.3 系统工作流程

整个系统由多进程组成，一个服务器进程和多个客户端进程，这两种进程在初始化、通信和消息处理各个方面有一些不同之处。下面就服务器和客户端两个方面介绍系统的工作流程。

在本系统中，服务器负责初始化显示和输入设备文件，等待并维护客户端的连接，完成窗口裁剪的工作，同时也承担部分的输入显示和消息响应任务。工作流程图如图 4 所示。

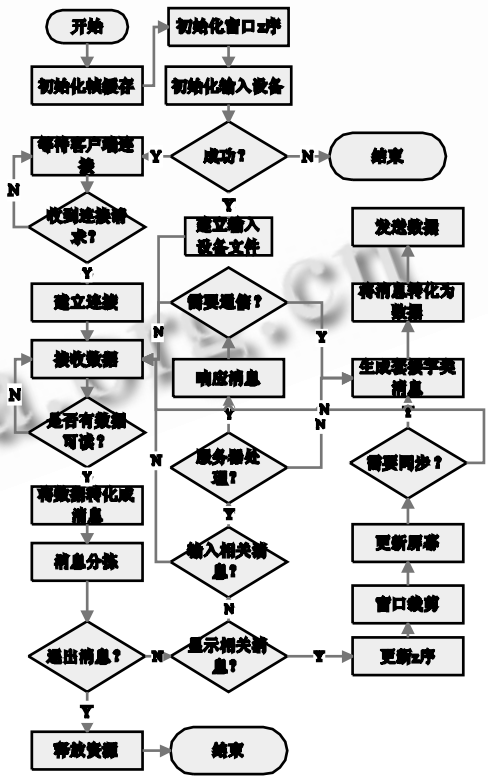

图 4　服务器端工作流程图

具体分为以下几个阶段：

(1) 初始化

服务器首先初始化帧缓存设备文件，设定屏

幕的分辨率、色深、刷新率像素字节等参数，得到帧缓存设备文件描述符；随后初始化相应的数据结构，申请内存资源，包括客户端标识数组，窗口 z 序矩阵，以及消息和消息处理器的链表等；接着是初始化鼠标和键盘，得到设备文件描述符，开始等待输入事件，如果鼠标和键盘初始化失败，进程终止。

(2) 等待连接

服务器建立监听套接字，等待客户端连接，一旦接收到连接请求，则建立服务器与客户端的专用连接，准备接收该连接上传输的数据。如果未接收到连接请求，则服务器继续等待。

(3) 接收数据

至此，服务器已经拥有了输入设备文件和连接套接字文件，接下来就是读取并处理这些文件上的数据。对于鼠标和键盘上的设备输入，转化成鼠标点击，拖拽，键盘按键的输入类消息，对于客户端连接上的数据，则根据数据头类型将数据流封装为相应类型的消息，必要时进行消息类型转化。当等待的各个文件上没有可以读取的数据时，系统接入下一轮循环继续等待。

(4) 响应

对于上一步产生的消息，需要进行分拣，按照一定的顺序予以响应。最先响应的是退出消息，其次是显示相关消息，最后是输入相关消息。消息响应模块接收到一条消息后，首先按照优先级顺序判断当前的消息的种类。消息主要分为退出消息、显示相关消息和输入相关消息，对于不用的消息，采取不同的响应方式。

在完成后两种情况下的消息处理之后，系统会重新回到步骤 3 等待数据的输入。至此，整个服务器进程的工作流程全部介绍完成。

与服务器类似，客户端也承担部分的输出显示和消息响应任务，不同之处在于：客户端无需设立监听套接字，无需初始化和监听输入设备，只需在建立连接后通过哦初始化显示消息获得显示同步所需的帧缓存和窗口 z 序矩阵的访问权。其工作流程如图 5 所示。

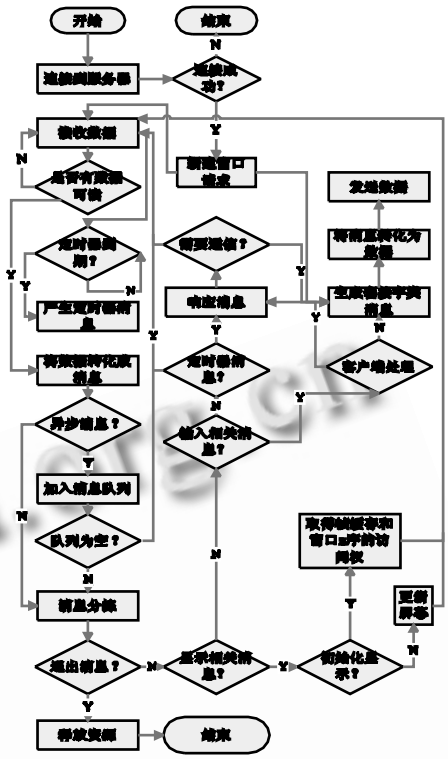

图 5　客户端工作流程图

(1) 初始化

客户端发起向服务器的连接，如果连接失败则进程结束。连接建立之后，首先向服务器发送新建窗口请求，在第三阶段接收并响应初始化显示消息之后，客户端获得帧缓存设备文件描述符和对窗口 z 序矩阵的访问权限，初始化工作完成。

(2) 接收数据

在服务器的接收数据阶段，所有消息都作为同步消息直接进入到下一阶段等待分拣。而在客户端进程中，这一阶段对消息做了同步/异步处理，对于同步消息，直接进入下一阶段；对于异步消息，则按照优先级，将其送入消息队列。客户端在每一轮循环等待的同时，还检查窗口定时器是否到期，对于到期的定时器进行复位重新计时，并生成一条该定时器的消息，送入消息队列。

以上是客户端相对于服务器增加的工作步骤，同时，客户端直接接收服务器发送的输入相关消息，省去了对输入设备文件数据的读取和转化。

(3) 响应

想对于服务器进程，客户端增加了一种消息，这就是定时器消息，对于定时器消息，客户端调用该消息制定的定时器处理函数，执行定时的例行工作。对于显示相关消息，需要响应的第一个是初始化显示消息，取得帧缓存和窗口 z 序矩阵的访问权，此后客户端才可以响应其他消息。

以上介绍服务器和客户端进程从初始化、进入循环再到终止的运行过程，从控件和时间上展现了原型系统的运作机制。

该 GUI 原型系统借鉴 X window 的客户端/服务

器体系结构，采用 UNIX 域套接字(UNIX Domain Socket，UDS)[4]在服务器端和客户端之间传送数据。服务器首先要监听、接受来自客户端的 UDS 连接请求，建立与维护各个客户进程的 UDS 连接。另一方面，服务器也需要监听各个连接、各种输入设备上的输入事件，将他们转换成相应的消息。任何消息产生后，服务器判断该条消息应由服务器还是某个客户端处理，将消息发给相应的进程。对于具体的实现技术细节，由于篇幅关系，本文仅给出客户端数据发送等待机制的实现部分，这种机制也适用于服务器端。

客户端与服务器端建立连接后，需要获得显示屏幕的分辨率，色深，帧缓存映射长度，这是通过发送初始化屏幕消息完成的，具体流程如下：

(1) 等待数据

由于 UDS 是可靠的流连接，同一条连接一次只能存在一个会话，要发送数据必须等待上一次的流通信结束。因此客户端也需要等待对 UDS 文件的写操作就绪。fds[0]保存的是当前 UDS 文件的 poll_s 结构体，则将 fds[0]的 fd 成员设置为当天 UDS 描述符 soc，fds[i]的 events 标志设置为 POLLIN|POLLOUT，表示等待该描述符上的读和写操作就绪。同样进程阻塞在 poll(fds，1，-1)函数处监听数目为 1 表示只监听一个文件，就是客户端 UDS 连接。

(2) 检查是否写就绪，并进行处理

poll 函数返回时检查 poll 函数的返回值，如果是正数，表明有 I/O 操作发生，进一步判断 fds[0]的 revents 标志。如果 fds[0]的标志为 POLLIN，说明文件已经被写入数据，需要先接收数据，将其转化为消息，再重复步骤 1，如果 revents 标志为 POLLOUT，说明文件已经写就绪，于是向对应文件发送数据。

4 系统功能测试

本文使用基于嵌入式硬件平台架构，基于 linux 操作系统进行模拟测试，测试环境：CPU：VIA C3-800，OS：Ubuntu 8.04，kernel version：2.6.24-21.

在功能测试中，通过编写不同的客户端实例程序，展示窗口操作，鼠标键盘响应，图形显示等方面的功能。

对所有的功能测试用力，都是在后台先运行服务器程序，再运行测试程序。测试的结果如表 1 所示。

从上面的测试结果可以看出，系统正确的实现窗口管理，输入响应和图形显示方面的功能，达到了预期的设计目标，验证了预期的设计目标是有效可行的。

表 1　功能测试结果统计表

测试项	测试结果
新建窗口	成功创建窗口。
关闭窗口	关闭窗口同时。
窗口切换	窗口焦点切换正常，窗口内容和窗口间层叠关系显示正常。窗口功能正常。
移动窗口、改变窗口大小	操作后窗口显示正常，可以继续响应消息。
鼠标键盘响应	客户端能够响应鼠标键盘消息，并做出正确响应。
定时器	定时调用已经注册的处理函数，定时间隔准确
文字	能够正确显示窗口标题
颜色	各种颜色正常显示
基本绘图操作	能够正确的完成画点、画线、矩形显示和填充。
图像显示	能够正确的显示图片，并进行缩放。

本文所做的只是一个原型系统的设计，在实际应用中具有一定的理论参考价值，但是，由于嵌入式 linux 应用的迅速发展，要求不断提高，使得系统与实际应用还有一定的差距，将在以后的研究工作中加以改进。

5 结语

本文首先阐述了 GUI 分层模块结构各层次模块的划分和实现策略，对 GUI 原型系统的总体结构进行了详细的功能描述，设计了 GUI 系统的工作流程。在本文提出的 GUI 的系统架构的基础上，设计实现了一个 GUI 原型系统，满足了嵌入式 GUI 所要求的高可靠性、低资源占用率和扩展性好等要求的同时，提高了系统的整体效率，在实际应用中具有很高的参考价值。

参考文献

1 王文启.嵌入式 Linux 图形用户界面的研究与开发[硕士学位论文].上海:东华大学, 2007.

2 王淑华,周定康.嵌入式 GUI 系统 Microwindows 的研究与分析.计算机与现代化, 2007,147(11):10－15.

3 李升亮,徐剑锋,李峻林.嵌入式系统中的多窗口 GUI 系统的研究.计算机数字与工程,2008,36(10):126－128.

4 Stevens WR, Rago SA,著;尤晋元,张亚英,戚正伟,译. UNIX 环境高级编程.第 2 版,北京:人民邮电出版社. 2006,8:384－385,390－392,427－428, 476－479.