一种支持多任务桌面环境的嵌入式系统解决方案

杨志义 回永利 李士宁 李志刚
(西北工业大学计算机学院 西安 710072)

摘 要 无论是在个人消费电子,还是在车载娱乐系统中,对支持多任务的嵌入式桌面环境的需求越来越迫切。友好的多任务支持为用户提供易用、方便的操作体验。一个健壮的嵌入式生态系统不仅要为用户的日常应用提供基本的桌面功能和应用,还要为开发者提供为该系统编写应用所需的工具和文档。通过守护线程和知名管道机制实现支持多任务的嵌入式桌面系统,并介绍了交互平台开发包的搭建。

关键词 嵌入式,桌面环境,多任务,Linux,Qt/Embedded
中图法分类号 TP316 **文献标识码** A

Implement Method of Desktop Environment Supporting Multitask Based on Embedded System

YANG Zhi-yi HUI Yong-li LI Shi-ning LI Zhi-gang
(School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract No matter personal consumer electronic products, or automobiles entertainment systems, they all cry for the desktop environment supporting multitask. User-friendly multitask support brings easy and convenient experience. A strong embedded ecosystem should provide basic desktop functions and applications for daily needs as well as tools and documentation for developers to write stand-alone applications for the system. With Daemon thread and well-known Pipe mechanism, a desktop environment supporting multitasks was implemented, and the paper introduced a method to develop the SDK for programmers.

Keywords Embedded, Desktop environment, Multitask, Linux, Qt/Embedded

1 引言

嵌入式应用从个人消费电子到车载娱乐系统终端,无一不需要有友好的、易用的图形用户界面。自从美国苹果公司在2007年推出iPhone之后,个人消费电子迎来了翻天覆地的变革,消费电子市场不断升温。嵌入式硬件技术的不断提高,则越来越模糊了嵌入式设备和个人PC的区别。有了硬件技术的支持,人们渴求类PC式的操作体验,而对多任务的支持则无疑是重中之重。另外,个人电子消费市场又催生了一种新的商业模式——应用商店。在这种商店中,充斥着第三方开发者提交的应用,从游戏类应用到地图应用,应有尽有。第三方开发者不需要强大的人力物力资源,可能一个程序员借助一台个人电脑,一个周末的时间就能编写出一款功能强大的应用,这就需要嵌入式解决方案为开发者提供交互平台开发包。

2 国内外几种主要嵌入式系统平台

2.1 iOS

iOS是美国苹果公司推出的嵌入式操作系统,有着华丽的界面、人性化的多点触控操作,为用户提供了完美的体验。另外,由于苹果公司在消费电子领域的强大号召力和创新性的App Store,因此有大量的开发者为其开发应用程序。然而,该操作系统是苹果公司专用系统,只能用于旗下的iPhone,iPad,iPod Touch等产品,不对外授权。

2.2 Android

Android是由美国Google公司主导的、开放手机联盟推出的、用于智能手机和平板电脑等移动设备的操作系统。它有和苹果公司iOS一样的完美用户体验及应用商店支持。又由于该系统大部分组件是开源的,因此它在全球范围内引起了OEM厂商、电信运营商及用户的追捧,大有赶超苹果的iOS之势。但该系统也有不可忽视的诟病:版本分裂。该问题造成了应用程序的不兼容,导致第三方开发者需要为不同的Android版本进行移植,带来了不必要的人力、物力浪费。其次,在2.2以后的版本中,Google公司将限制OEM厂商对系统界面进行差异性定制,限制了用户的个性化需求。

2.3 Windows CE

Windows CE是微软针对个人电脑以外的计算机设备所开发的嵌入式操作系统,在该平台上可以使用传统Windows上的编程工具,使用同样的函数和界面风格,使绝大多数应用软件只需简单地修改和移植就可在Windows CE平台上继续

到稿日期:2010-05-06 返修日期:2010-11-08 本文受国家科技支撑计划(2007BAD79B0),陕西省自然科学基金(SJ08-2T06),西北工业大学研究生创业种子基金(Z2011140)资助。

杨志义(1952—),男,教授,主要研究方向为分布计算、网络化嵌入式计算和测控技术等,E-mail:yangzy@nwpu.edu.cn;回永利(1985—),男,硕士生,主要研究方向为嵌入式计算技术。

使用。然而 Windows CE 是微软早年推出的嵌入式系统，没有华丽的界面和人性化的操作体验，尤其是在个人消费电子领域，落后于上述系统。另外，使用该系统需要向微软缴纳一定的授权费[1]。

还有一些为嵌入式设备开发的操作系统，如国内中科红旗 Linux、Palm 公司(已被 HP 公司收购)的 WebOS、Nokia 和 Intel 公司合作开发的 Meego 系统、三星公司的 Bada 系统。这些系统都有良好的用户体验、完善的交互平台开发包以及为应用开发者开设的应用商店。

在分析了上述诸多嵌入式操作系统的优缺点之后，本文将介绍一种有别于上述操作系统的新的嵌入式系统解决方案——Linux+Qt/Embedded。

3 Linux+Qt/Embedded 解决方案

3.1 系统特点

该方案借鉴了以上各系统好的特征，突出了操作易用性，可为用户提供方便、快捷的操作体验，实现类 PC 式的交互式操作，并为开发者提供符合该框架的类及接口，便于扩展第三方应用，构建完善的生态系统。该方案的底层采用开源的 Linux，可支持 X86，PowerPC，ARM，XSCALE，MIPS，SH，68K，Alpha，SPARC 等多种体系结构，可以移植到多种硬件平台。Linux 采用一个统一的框架对硬件进行管理，同时对一个硬件平台到另一个硬件平台的改动与上层应用无关。Linux 也是开源的，所以可针对不同的应用领域进行适当的裁剪，以实现资源的有效利用。另外，Linux 有着广泛的社区支持，它的程序越来越多，其自身也逐渐发展壮大起来。

Qt/Embedded 移植了大量原来基于 Qt 的 XWindows 程序，提供了非常完整的嵌入式 GUI 解决方案。Qt 遵循 GPL 协议，开放主要的源代码，用户可以在 GPL 规定下自由添加新特性。与其他嵌入式 GUI 相比，Qt 不仅是一个完整的窗口系统，而且是一个应用程序框架，有利于应用程序的开发。Qt 具有丰富的 API，包括多达 250 个以上的 C++类，支持图形、网络、数据库、I/O 操作、各种控件和 XMI 等众多功能，满足大多数嵌入式应用系统开发的需要。Qt 是一个 GUI 仿真工具包，它使用各自平台上的低级绘图函数来仿真 MS Windows 和 Motif，因此程序运行速度快。Qt 有良好的封装机制，使得它的模块化程度非常高，可靠性好，易于程序开发[2]。

3.2 现有技术研究

Linux+Qt/Embedded 方案是一种比较成熟的嵌入式技术手段，它的两种主要的应用模式如下所述。

1)常见 Linux+Qt/Embedded 应用

嵌入式系统由于是以应用为中心，适用于应用系统对功能、可靠性、成本、体积、功耗有严格要求的专用计算机系统，因此在这些特殊应用环境中，底层是经过裁剪的 Linux 系统，上层应用是基于 Qt/Embedded 的应用程序，开机自动运行应用程序，完成单一的任务，不需要多任务的支持。这种单任务应用模式，常用于汽车自动化、医疗设备、POS 机等。其缺点是功能单一，系统不可扩展。

2)Linux+Qtopia

Qtopia 是一个构建于 Qt/Embedded 之上的类似桌面系统的应用环境，包含完整的应用层、灵活的用户界面、窗口操作系统、应用程序启动程序以及开发框架[3]。Qtopia 的特性如下：窗口操作系统、游戏和多媒体、工作辅助应用程序、同步框架、PIM 应用程序、Internet 应用程序、开发环境、输入法、Java 集成、本地化支持、个性化选项、无线支持等。Trolltech 公司(已被 Nokia 收购)提供 3 大 Qtopia 版本：Qtopia 手机版、Qtopia PDA 版和 Qtopia 消费电子产品平台。Qtopia 应用：从单一应用设备到具备领先功能的电话以及媒体产品等。Qtopia 最后一个版本是 4.3.2，它基于 Qt 4.3。然而 Qtopia 却不支持图形界面程序的多任务运行，使得用户只能运行单一任务，如果想切换到其他应用，必须从当前应用中退出。该系统没有为用户提供人性化的扩展接口，因此限制了第三方应用的扩展。

本文论述的解决方案，参照了 Qtopia 的基础架构，通过引入“守护线程”机制实现了支持多任务管理的嵌入式桌面系统，对窗口 UI 进行了重新设计，加入了无线升级技术，用于系统升级及应用程序更新；通过引入“知名管道”机制，开发了用于此解决方案的交互平台开发包。本解决方案主要应用于对实时性要求不高的嵌入式设备，如个人消费电子、车载娱乐系统、家庭安防的场合中。

4 解决方案的实现

4.1 系统框架

本解决方案的系统框架[4]如图 1 所示。底层由 Linux 内核、驱动程序和电源管理模块构成。不同硬件平台需要开发者做出不同的定制，这里不做特别说明。中间层是基于 Qt/Embedded 的嵌入式 Qt 库，它精简和优化了各种图形操作，程序运行时无需额外系统的支持，可以有效减少内存消耗和 CPU 负担。Qt/Embedded 本身是可扩展的，并能不断地升级。开发人员可以根据自己所面对的嵌入式设备的实际需要，对其进行适当的裁减，参照 Qt/Embedded 的移植参考文献[5]。最上层是用户应用程序，包括桌面环境所必需的一些基本功能组件，以及第三方开发者为使用符合该框架的交互平台开发包编写的应用程序。桌面系统位于应用程序之下，不仅提供了一个基于桌面配置的简易的人机对话机制，还提供了通用的应用程序开发框架。下面将详细介绍桌面环境的设计思想、实现方法以及交互平台开发包的构建。

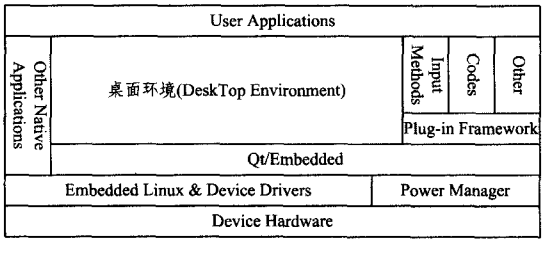

图 1 嵌入式系统框架

4.2 桌面环境组件

桌面环境是为嵌入式系统提供一个简单、易用的用户界面，集成了任务管理器、资源管理器、文件管理器、配置系统、桌面工具、应用程序管理器、无线升级管理器等功能部件[6]，其框图如图 2 所示。

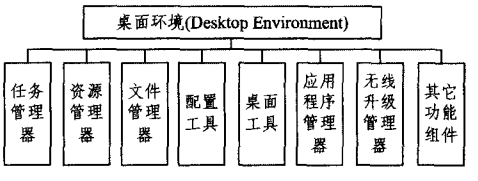

图 2 桌面系统组件框图

在这些功能组件中，任务管理器是最重要的部件，它不仅可以提供类似桌面 PC 系统的任务栏功能，还可以显示正在运行客户任务的详细信息，并可以切换当前任务窗口、关闭指定的任务、显示任务的进程号、父进程、进程运行时间、程序所在路径等信息。

此桌面环境中，GUI 窗口系统采用的工作模式是客户/服务器方式，进程之间有消息采集、交互和同步的需求。服务器负责为客户和自身分配显示区域，生成鼠标、键盘或者触摸屏事件，通常包含启动客户的用户界面。而客户则通过与服务器通信来申请显示区域，接受鼠标或触摸屏事件。客户可以直接访问所分配的显示区域，以便为用户提供 GUI 服务。服务器和客户通过管道的方式来传递所分配显示区域上的信息。服务器进程通过内部进程通信机制 QCopChannel 类与客户进程进行交互，切换任务窗口以及关闭应用程序。

4.3 交互平台开发包

本桌面系统采用了特殊的客户/服务器结构，需要为开发者提供必需的类及服务接口。为了搭建健壮的生态系统，便于第三方开发者迅速搭建起适用于该平台的开发环境，有以下两种解决方法用于构建交互平台开发包。

1)在源代码级进行修改：利用管道机制进行内部进程通信[7]，修改 Qt/Embedded 源码包中的 qwindowsystem_qws.cpp，加入该框架所需的类及服务接口，然后交叉编译，提供给第三方。缺点是桌面系统底层是基于 Qt 的。为了支持 Qt 最新版本，每当有新的版本出现时，都要做相应的跟进修改，这样会限制第三方开发者的自由。

2)将该框架所需类及服务接口编译成独立的库文件，只要第三方程序员熟悉该接口及所需的类，无需了解具体的实现细节，就能自由编写适用于该平台的应用程序。这种方法实现起来简单、灵活，对于系统升级无限制，所以本文采用此种方法。

为了实现对多任务的管理，客户进程必须向系统注册第三方应用程序的相关信息，例如主窗口句柄、客户任务 ID 等信息。服务器进程将通过这些信息建立客户进程节点，通过“知名管道”进行进程间通信。“知名管道”是公开的服务器进程管道，这里所说的“公开”类似于 TCP/IP 协议中的端口号，用于同客户进程进行通信。第三方应用程序中不能出现和该管道相同的名字，通过继承 QCopChannel 类来实现。简单起见，这里仅介绍窗口类 HMainWindow 的实现，其他 GUI 控件的实现方法类似。

```
HMainWindow::HMainWindow(QWidget * parent)
    :QMainWindow(parent)
{
  ……
  //仅需客户进程的主窗口向服务器进程注册
  if(! this->parent()){
      //向服务器进程注册客户进程
      registId(winId());
      //建立客户进程管道，同服务器进程通信
      QCopChannel * guestChannel=new QCopChannel(QString::
  number(winId(),10),this);
      connect(guestChannel, SIGNAL(received(…)), this, SLOT
(doGuestMessage(…)));
    }
    ……
}
void HMainWindow::registId(int id)
{
    ……
    QDataStream out(&data,QIODevice::WriteOnly);
    //向服务器进程注册主窗口句柄、客户任务 ID
    out<<QString::number(id,10)<<getpid();
    QCopChannel::send("deskTop","regist",data);
    ……
}
```

该类中还有其他成员函数，例如槽函数 doGuestMessage(…)，用于处理服务器进程向该客户进程发送的指令信息，如改变窗口大小、位置、显隐等 GUI 操作。这里不详细介绍这些成员函数。

把必要的控件类及服务接口编译成独立的库文件，提供给开发者。在第三方开发者建立应用程序时，用到的控件类必须是交互平台开发包中的控件类。开发者根据实际系统需要自行定制底层的 Qt/Embedded。

4.4 任务管理器核心思想

当有新的客户任务开启时，任务管理器开启一个线程。该线程处于服务器进程所在空间，用于同客户进程交互，既可以控制客户进程，实现多任务的切换，又可以在客户进程退出时通知服务器进程刷新任务列表信息，最后释放守护线程所占用的资源。这里把用于管理客户任务的线程称之为“守护线程”。同时，在服务器进程空间中维持 windowList 和 threadList 两个链表，用于记录正在运行客户进程的相关信息。当有新的客户进程启动时，分别向这两个链表中添加指向该任务的节点。当有客户进程结束时，在链表中删除指向该任务的节点[8]。

实现多任务管理的客户/服务器模型如图 3 所示。为了实现客户任务的同步操作，任务管理器组件中用到了一个重要的机制——Qt 的信号/槽机制，它是一种同步机制。当有新的客户任务开始或客户进程终止时，采用该机制，能够及时更新 windowList，threadList 两个链表。

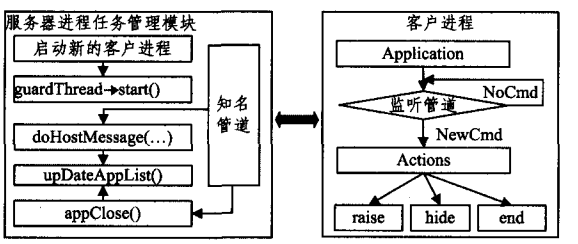

图 3 多任务管理的客户/服务器模型

4.5 任务管理器的实现

windowList 链表记录的是正在运行的客户进程信息，如客户进程 ID、主窗口句柄以及其它用于描述客户进程的信息。由任务管理器组件操控该链表，其节点的数据结构定义如下：

```
struct{
    int pid;//记录任务的进程 ID
    char appInfo[100];//记录程序绝对路径等信息
    int winId;//记录客户任务的主窗口句柄
    windowNode * next;//指向链表中下一个节点
```

```
}windowNode;
```

threadList，该链表维护了守护线程的一系列信息，由管道处理函数和守护线程操作该链表。该链表节点的数据结构定义如下：

```
struct{
    Qt::HANDLE id;//记录线程 ID
    int pid;//记录任务的进程 ID
    threadNode * next;//指向链表中的下一个节点
}threadNode;
```

服务器进程任务管理模块中的 guardThread→start()用于启动守护线程。该线程类的 run()函数的实现代码如下所示：

```
void guardThread::run()
{
    ......
    //获取当前任务守护线程的 ID
    threadId=currentThreadId();
    QByteArray xApplication=appName.toLatin1();
    If(! program(xApplication))
    {//如果客户进程不能正常执行，发送运行错误信号，在关联的槽函数中处理
        emit runAppError();
        return();
    }
    appNo=locateElem_L(threadL,threadId);
    //通过 threadId，在 threadList 中找到对应节点
    pidNo=searchNode_L(threadL,appNo);
    //释放 threadId 对应节点的内存空间
    freeNode_L(threadL,threadId);
    //发送信号，表明客户进程结束，在槽函数中进行有关处理工作
    emit threadClose(pidNo);
}
```

当有新的客户进程启动后，该客户进程通过知名管道向服务器注册相关信息，包括进程号、主窗口句柄等其它相关信息。知名管道位于服务器进程中，槽函数 doHostMessage(…)用于处理知名管道接收到的客户进程信息，向 threadList 中添加客户进程节点，并发送信号 newPidSignal(pid,pidInfo,registId.toInt(&ok,10))以更新任务管理器列表。

```
void DesktopWindow::doHostMessage(const QString &message,
const QByteArray &data)
{
    ......
    QDataStream in(data);
    MakeNode_L(s,tempId);
    s->next=NULL;
    //这里判断是否是客户进程注册信息
    if(message=="regist") {
    in >> registId>>pid;
    s->pid=pid;
    //仅注册客户进程的主窗口
    if(LocateElem_P(threadL,pid)==0){
        // 向 threadL 添加进程节点
        Append_L(threadL,s);
        //更新任务管理器
        emit newPidSignal(pid,pidInfo,registId.toInt(&ok,10));
    }
    }
    ......
}
```

任务管理器模块中还包括一些其它常用功能，即在任务管理器中切换任务窗口以及关闭应用程序，这些动作只涉及到 windowList 链表的插入、删除操作和与客户进程的通信操作。

该嵌入式桌面环境的其它功能组件都是针对具体应用进行量身定制的，所以没有给出一个通用的解决方案。

5 系统测试

对本文讨论的嵌入式桌面环境，进行了功能实现及系统健壮性的模拟测试。

硬件平台：采用飞凌公司的 OK2440-III 型开发板，CPU 为 Samsung 公司的 S3C2440A 处理器，主频 400MHz，内存 64M，配置 128M 的 NAND Flash。

系统软件清单：Linux 2.6.12 内核，qt-embedded-linux-opensource-src-4.5.1。

结束语 嵌入式桌面环境是嵌入式 Linux 不可缺少的组成部分。本文通过分析、比较目前流行的几种嵌入式 OS，选择了 Linux＋嵌入式 Qt 作为研究对象并对其进行了深入讨论。通过“守护线程”和“知名管道”机制，完成了多任务桌面环境系统的设计和实现，可为其他嵌入式系统软件设计提供参考价值。

本系统利用服务器进程中的线程来管理客户进程。这种方式能够有效管理客户进程，但缺点是当用户开启多个任务时，就会产生对应数量的守护线程。在嵌入式环境中，硬件资源的分配尤为重要，过多的线程会导致系统健壮性不够健全，占用过多的硬件计算资源。为此，需要进一步优化任务管理机制，这是解决硬件利用率所面临的重要问题。

参 考 文 献

[1] 飞凌嵌入式. 飞凌开发板配套教程[R]. 保定：飞凌嵌入式技术有限公司，2008：99

[2] 刘军锋，朱洪雷，熊邦宏，等. 基于嵌入式 Qt 的车载 GUI 平台的设计[J]. 自动化与信息工程，2008(3)：29

[3] 苗忠良，宛斌. Qtopia 编程之道[M]. 北京：清华大学出版社，2009：35

[4] Griffith A. KDE 2/Qt Programming Bible [M]. IDG Books Worldwide, Inc., 2001：3

[5] QT 4.5.2 嵌入式开发平台的搭建[EB/OL]. http://hi.baidu.com/a263238386/blog/item/6ff055dda933033e5882dd32.html

[6] 詹瑾瑜，熊光泽，孙明. 一种嵌入式 GUI 软件结构实现方案[J]. 电子科技大学学报，2003，32(1)：89

[7] Chen A. QT/Embedded 窗体事件是如何派发的[EB/OL]. http://www.dzjs.net/html/qianrushixitong/2009/0225/3663.html

[8] Danesh A. Making Linux work：essential tips & techniques [M]. ONWORDE Press，2002：363