

移动富媒体客户端的研究与设计

香华冠, 冯 刚

(华南师范大学 计算机学院, 广东 广州 510631)

摘 要:通过分析当前富媒体业务的发展趋势,引出了富媒体客户端开发的应用前景,并介绍了一种提高移动富媒体客户端可移植性的设计思想及其实现方法。整个设计过程采用面向对象技术对系统进行建模,并根据模块化的设计思想对系统进行分块设计。整个系统围绕着移植性能进行优化设计,根据各模块的特点以及联系,在操作系统平台依赖性较大的环节采用了面向对象的方法重新定制接口,向上提供一个平台适配层,使得系统在移植时的工作量得到明显降低。

关键词:移动富媒体客户端;系统移植;接口封装

中图分类号: TN929.5

文献标识码: A

文章编号: 1673-629X(2010)07-0168-04

Research and Design of Mobile Rich Media Client

XIANG Hua-guan, FENG Gang

(School of Computer, South China Normal University, Guangzhou 510631, China)

Abstract: This article analyzes the current trend of rich media, and then leads to the development prospects of rich media client application. Here outline the design ideas and its implementation of the mobile rich media client. The client software is designed by using object-oriented technology and modular design concept. The software design is focus on the migration optimization, and the workload of the migration has been markedly reduced by providing a platform adaptation layer, and by using the object-oriented approach to re-customize the interface at the platform-dependent parts.

Key words: mobile rich media client; system migration; interface encapsulation

0 引 言

富媒体是为中、高端手机用户提供的一种以文字、图片为主,音视频为辅的移动通信业务,比起传统的彩信业务更具表现力,随着3G技术的推广,富媒体业务将会以更快的速度发展,具有惊人市场潜力。智能手机是富媒体业务的主要应用平台,目前,智能手机的操作系统种类繁多,体系各异,若客户端的设计没有经过优化,将带来沉重的系统移植负担。本设计旨在对各种手机操作系统求同存异,针对其中几款主流的智能手机操作系统,利用面向对象技术及组件化方法进行移动富媒体的设计与实现。

1 客户端的跨平台开发问题

富媒体客户端最终要运行于多款移动智能操作系统,而不同的操作系统提供的系统调用无论在功能上还是在形式上都会有各方面的差异,因此,客户端在跨

平台开发时会遇到以下问题:

(1)线程的实现机制:主要是指线程的管理(包括线程的创建及终止)以及同步(主要是线程的互斥和同步)处理上的差异。

(2)音视频接口:包括解码器和设备控制接口等,其中需要注意的是对于某些音视频格式并不是所有平台都会提供相应的编解码,同时对于音视频等硬件设备的操作,平台都具有一套自身特点的驱动,直接互操作性差。

(3)协议栈的实现:对于具体协议某些系统没有实现或者实现的细节不同,例如3G手机与2.5G手机之间的差别,前者提供3G网络协议,而后者仅提供2.5G如GPRS协议,除了系统对协议的支持与否,还需考虑到在协议各层次提供的系统API的功能与形式的差异^[1]。

(4)图形UI机制:包括图形元素管理以及消息循环机制的差异,主要涉及的功能包括图形组件如按钮、对话框等,组件的管理如组件的创建、销毁等以及消息循环如消息回调、消息定义等。

(5)设备管理:系统提供设备操作接口的不同,硬

收稿日期:2009-11-09;修回日期:2010-02-10

基金项目:国家高技术研究发展(863)计划项目(2006AA02Z346)

作者简介:香华冠(1983-),男,广东人,硕士研究生,研究方向为嵌入式系统。

件厂商针对各种操作系统的特性编写了各自的驱动^[2],而操作系统对设备操作所提供的 API 在功能和形式上都存在差异,特别是设备的管理上的差异,是不容忽视的。

传统的应用软件开发通过调用具体平台提供的系统调用完成以上各种功能,而通过分析系统调用存在的上面的一些弊端,单纯依靠系统调用开发的软件移植性很差,因此可以将各种平台的功能调用进行统一的封装,提供一个平台适配层,软件开发通过调用平台适配层提供的统一接口实现各种应用。

2 系统设计

2.1 系统总体结构

通过对系统关键模块的封装,一个应用软件的体系结构可以从三个层次描述:平台无关层、平台适配层以及操作系统层。最上层为平台无关层,通过调用平台适配层实现具体的业务逻辑,该层次完成的大部分工作是基于算法逻辑层面上,通过对平台适配层的接口调用获取系统提供的服务,因此在系统移植时该层次的代码基本上不作改动。平台适配层通过调用各种具体的系统调用,为上一层提供统一的接口。通过分析各种系统相应功能的系统调用,提取它们之间的共同点,然后重新定义接口函数的定义形式以及实现机制,实现统一封装,其主要功能如图 1 所示,包括以下方面:

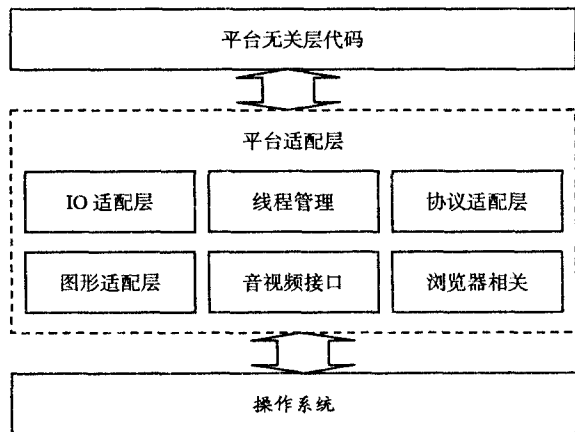


图 1 系统总体结构图

(1) 图形适配层:对各种操作系统在 UI 处理方面的不同进行封装,向上层提供统一的接口。目前提供图形界面运行库的框架有很多而且功能各异,但是它们主要的实现机制基本相同,可以对窗口元素的管理、消息循环机以及回调机制等进行封装。使用特殊的数据结构定义每一个图形组件,同时定义这些组件的管理函数如组件的创建以及销毁等^[3]。对于消息循环,统一定义一个循环核心处理函数,同时需要有一套标

准的消息类型定义,以标识消息的主题信息,最后在消息回调处理上需要对回调函数的形式进行统一的定义。通过对图形界面处理的关键部分进行统一的封装,软件在系统移植时只需要修改这部分功能,而对于上层所涉及的图形应用则无需改动,大大提高开发效率。

(2) 浏览器相关:主要用于解析 WAP 协议及基于该协议的网页代码,包括 WML, XHTML, Javascript, CSS 等^[4],对各种语言的语法、语义进行分析,最终向上层输出一种显示模板。该层调用系统的 WAP 处理 API,完成对 HTML、CSS 等这些语义的分析翻译,并组装成相应的数据结构,平台无关层根据这些结构在相应的图形界面上绘制组件,即平台无关层依赖的仅仅是该层所提供的数据结构^[5,6],因此可实现通过本层次的封装隔离,可以实现很好的移植。

(3) 音视频接口:包括各种标准的解码器,提供解码器的加载与卸载功能。主要对解码器提供的函数形式进行标准定义,提供一致的函数接口,同时提供一个合理的解码器集合,即对于每个系统都会提供的特定格式解码器。另外,对于音视频设备的管理通过调用具体的系统调用并向上层提供接口来实现。

(4) IO 适配层:是对一些基本的系统输入输出操作的封装,向上层提供统一接口,系统基本的输入输出网络 IO,文件 IO,设备 IO 等。通过分析各种系统的 IO 特性,以 Linux 为基础的 IO 体系更方便于管理的一致性,为此,使用文件形式对设备进行管理,并通过描述符标识每一种设备,对设备的操作都可以简单地通过调用 read 和 write 函数实现读写操作。

(5) 线程管理:包括线程的基本操作如创建与销毁,同时实现线程的互斥与同步,线程的实现至今没有一个统一的标准,不过各种平台在具体实现上或多或少遵循了相同的机制,为此可以 Posix 标准为前提,实现线程的一些基本管理函数定义,并提供相应同步机制,这些功能已能够满足应用程序实现日常应用,并且可以很好地实现平台隔离。

(6) 协议适配层:提供多种协议的编程接口,按照协议的层次以及类别如 TCP/IP 协议的网络层以及传输层等,在本层向上提供统一的调用^[7];同时,在跨平台移植时需要解决目标平台不支持的协议的问题。

2.2 平台适配层设计

通过对平台适配层的功能分析,该层次的各部分功能需提供接口,使用 C++ 语言的抽象类,在抽象类上对接口实现的具体功能进行约束。从图 2 可知,平台适配层一共实现了 6 种接口,分别是(从左上角开始顺时针)图形操作、音视频数据处理、WAP 数据处理、

协议接口、系统 I/O 以及线程操作等。其中,系统 I/O 是最基本的一种接口,提供了对文件和设备的管理,包括打开及关闭操作、读写操作以及命令控制等。根据文件和设备管理的区别,分别有文件管理和设备管理两个实现了系统 I/O 接口的功能类实现。

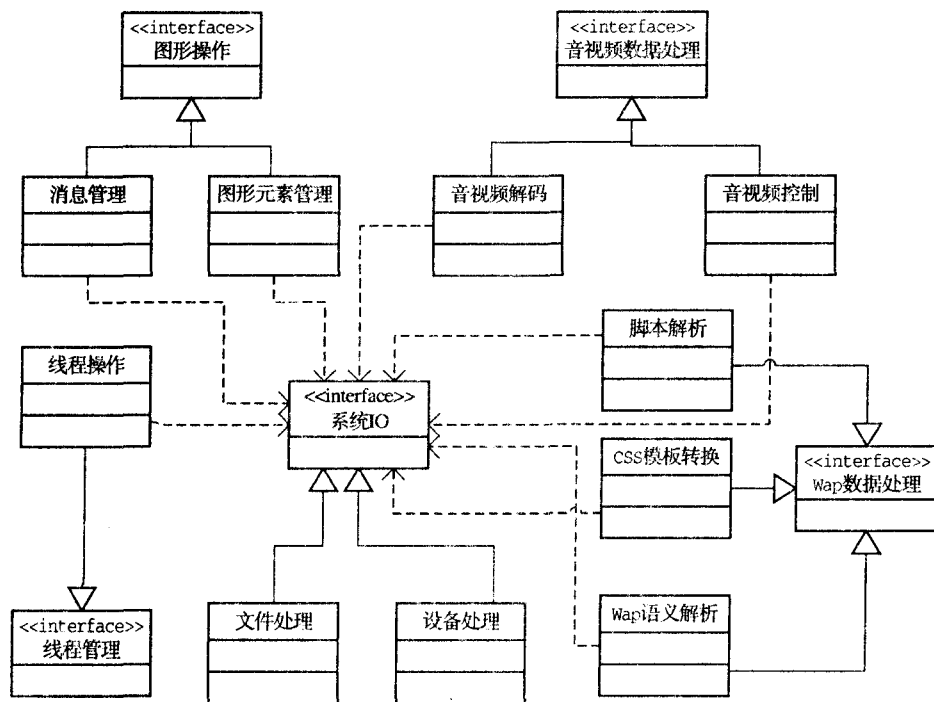


图2 平台适配层类结构图

考虑到不同平台使用不同的线程管理机制,例如 Windows 平台的线程机制与 Linux 平台的类 Posix 线程机制就存在差异,为此,平台适配层提供了统一的线程管理接口。接口的具体实现通过对调用各种平台提供的线程库,并作统一封装,实现了线程的创建、线程属性的处理、线程互斥以及同步的管理等。

作为与用户直接交互的应用程序,图形界面是必不可少的组成部分。平台适配层提供了图形界面的标准接口,这一类接口是图形操作中最经常使用的也是最主要的。应用程序可以根据系统的主题特性设置图形显示样式,同时应用程序涉及的一系列图形操作可以通过该层次实现高效的隔离,提高移植效率。

2.3 系统实现

2.3.1 系统 I/O 的实现

对于文件操作,不同平台提供的系统调用存在差异,但是几乎各种平台都支持 C/C++ 函数库,可以使用 C/C++ 语言提供的文件操作接口操作文件^[8]。另一方面,对于系统管理设备也存在平台的差异性,例如 Windows 对设备的管理有异于对文件的管理,而 Linux 对设备的管理则跟文件管理相同(设备也是文件而且是特殊文件)^[9]。为解决这种差异性,提出了在系统

I/O 接口中对文件和设备操作进行统一管理,接口内部使用各种平台特有的文件或者设备操作,而对于上层程序,则提供一个统一的调用接口,只需要使用相同的读写接口,便可以实现文件与设备透明操作。

系统 I/O 接口规定了一系列的 I/O 操作,包括打

开、关闭、读取、写入以及发送命令等功能,一个对象绑定一个待操作的文件或者设备。系统 I/O 参照 Linux 系统的文件 I/O 系统调用,主要机制包括文件描述符和标准读写操作。对于文件操作,实现了一个文件管理类(从系统 I/O 派生),该类调用 C++ 提供的文件操作 ifstream 以及 ofstream 实现具体的文件操作,对于每个文件都提供一个描述符,对文件的读写操作需要通过 read 和 write 标准调用,将描述符作为参数传入,然后在函数体内部调用 C/C++ 的标准函数完成具体的操作。对于设备的

操作,同样从系统 I/O 派生一个设备管理类,其成员调用具体平台的系统调用,完成对设备的管理。设备在打开以后也会相应地分配一个描述符,而对设备的读写也是提供一个 read 和 write 函数,需要传入描述符,在函数体里面是具体的设备操作。客户端软件的业务逻辑部分通过调用系统 I/O 提供的接口,无需关心系统具体的实现细节,大大提高了软件移植性。

2.3.2 线程管理的统一处理

Linux 系统使用的 pthread 线程库是对 Posix 标准的修改,其 API 与 Windows 系统提供的线程操作接口存在差异,但是对于实现机制如线程的创建与维护,线程的互斥与同步等都有相似之处,因此可以对它们进行统一的封装^[10]。

线程管理类实现了对线程管理的一些基本功能,包括线程的创建、属性的修改以及互斥与同步等。线程管理提供了线程生命周期中涉及的基本操作,并提供了重要机制的实现,该类提供互斥和同步接口,方便应用程序获取相应的信号量,同时也实现了同步操作的具体实现。一个对象代表了一个线程,记录了线程的运行信息。对于线程的创建,Linux 与 Windows 都提供了创建 API,其中都需要传递线程函数体以及函

数参数等信息,为此可以用以下函数实现两个系统中的线程创建:

```
int createTread(pid_t& pid, //线程 ID
void * func, //线程函数
void * param, //传递给线程函数的参数
void * attr, //线程属性结构体
void * append = NULL); //附加结构体
```

该函数前面四个参数是 Linux 与 Windows 都需要的,而第五个参数是一个用户定义结构体,对于 Windows,创建线程时需要额外附加线程的栈长度以及线程标志等,用户可以在此处传入这些参数。可以使用宏定义实现根据具体平台而调用相应的线程创建函数,如定义宏 LINUX 和 WINDOWS,然后使用 #ifdef 语句实现相应的线程创建函数的调用。

2.3.3 统一图形操作的实现

图形操作标准具有明显的平台异性,因此需要分析各种图形标准的共同特征,而它们的不同点主要集中在外观以及函数的原型,其主体结构是相似的^[3]。

(1) 初始化以及消息循环。

无论各种标准图形之间存在着多少的区别,它们都是以事件驱动为核心运行模式,为此,统一的图形操作接口设计需要对消息循环做一个统一的定义。首先需要做的是提取图形设备的初始化信息,如屏幕的分辨率、大小以及颜色深浅等等。将设备的信息提取成功后,根据具体的设备设置图形库的操作特性。消息循环使用一个 while 循环,在循环内不断地对图形界面收到的消息事件进行处理,例如可以处理控件的操作、键盘的输入以及图形交互等;另外,还可以处理其他非核心消息如网络事件、时钟事件等等。

(2) 界面组件。

图形界面组件包括按钮、文本框、对话框等,在不同的图形库里面它们有不同的表现形式如颜色以及大小的不同等。这里考虑到各种差异性,使用了一些数据结构对这些组件进行定义。该数据结构提供了一个组件链表用于链接所有的同类型组件,同时将组件的具体实现类作为结构的元素,结构体还包括其他操作信息等。对图形组件定义后,还需要定义图形的创建、销毁以及组件处理函数,这些函数负责一个组件生命周期的各种操作。

(3) 回调函数。

消息回调是消息循环的核心机制,但是这种机制在具体实现时具有平台异性,主要表现在函数的定义上。要解决这个差异性,首先要有一个统一的回调函

数定义。该函数与事件受体(图形组件)相联,同时提供参数数据输入接口,方便用户自定义的数据传入函数体;同时还需一套对应各种消息类型的宏定义以标识消息主体。之后无论是系统中的何种消息回调处理,其回调函数都应该严格按照该函数定义。

3 结束语

本研究针对移动富媒体客户端的可移植性进行了分析,提出了基于操作系统的平台适配层的软件设计方法。通过对平台适配层的分析与设计,可以大大减少软件在跨平台开发时产生的负担,提高了软件开发效率。

系统从客户端简要使用的各种功能出发,分析了这些功能的平台依赖性,使用面向对象的方法以及模块化的设计思路,通过调用具体平台提供的系统调用向上层提供了统一的功能接口。系统对平台各大功能做了统一封装,对于模块实现机制更复杂的模块如图形接口以及协议层次等还需进一步的设计与优化,这将能够更加大幅度地提高软件开发效率。

参考文献:

- [1] 周正勇,阳富民,胡贯荣.一种嵌入式浏览器的核心技术及特色[J].计算机工程与设计,2003(3):21-23
- [2] Coulson G. A Configurable Multimedia Middleware Platform[J]. IEEE Multimedia,1999(6):62-76.
- [3] 刘 曙,汤伟华.motif 图形用户界面编程接口的可移植性封装[J].空军工程大学学报:自然科学版,2003(4):78-80.
- [4] 李小群,郑良辰,耿增强,等.浅析嵌入式系统中的浏览器[J].测控技术,2000(4):9-11.
- [5] 柳 洋,蒋泽军.基于分层结构的可移植 WAP 浏览器设计[J].科学技术与工程,2006(8):1072-1074.
- [6] 阳富民,李 俊,周正勇,等.嵌入式浏览器的设计与实现[J].计算机工程与科学,2003(4):39-41.
- [7] 侯 波.嵌入式 LINUX 中套接字组件的设计与实现[J].科学技术与工程,2007(1):142-144.
- [8] 钱树人,朱怀宏,王静英.源级移植的分析和易移植量度[J].计算机研究与发展,1992(12):33-38.
- [9] Coulson G, Blair G S, Davies N, et al. Supporting Mobile Multimedia Applications through Adaptive Middleware[J]. IEEE Journal,1999(17):1651-1659.
- [10] Fitzpatrick T, Blair G S, Coulson G, et al. A Software Architecture for Adaptive Distributed Multimedia Systems[J]. IEEE Proceedings - Software,1998,145(5):163-171.