# 基于 Linux 平台的多路视频采集系统的设计与实现

## Design and Create the video data collection system based on Linux OS

苏艳艳 （温州职业技术学院计算机系 325035）

**摘要**：本文讨论了一种以 Linux 操作系统为开发平台，基于 USB2.0 接口技术和 MPEG4 视频流硬件编码压缩技术实现的远程网络视频采集、存储和传输系统的设计和实现。全面介绍了系统软硬件构成，USB 客户端设备驱动设计和系统应用软件设计及采用的一些关键技术和策略，并对该系统应用前景做了一些初步探讨。

**关键词**：USB2.0 MPEG4 设备驱动程序 Linux 操作系统 C/S 模式 IP 多播

## 1 前言

本文所介绍的视频采集系统是建立在 Linux 操作系统上充分利用了以上两种技术，并同数据存储和网络传输相结合，实现了对视频数据的实时采集、存储和传输功能。系统由作为外设的视频图像采集卡和主机应用软件组成，二者之间通过 USB 接口来实现压缩后的视频图像数据传输。考虑到 Internet 的飞速发展，特别是远程视频图像服务在各个领域的日益广泛应用，系统中将主机应用软件设计成 C/S 模式中的服务器端，并采用 IP 多播方式来实现对视频图像数据的多点传送。这种系统架构的设计定位是网络视频图像服务器，可以用于远程视频图像监控，远程教育等相关应用领域。

## 2 系统硬件平台

### 2.1 采集卡硬件结构

视频图像采集卡通过对视频输入进行 MPEG4 压缩编码获取 MPEG4 格式的视频压缩数据，再通过 USB 总线将 MPEG4 码流传送给计算机，其硬件结构框图见图 1。

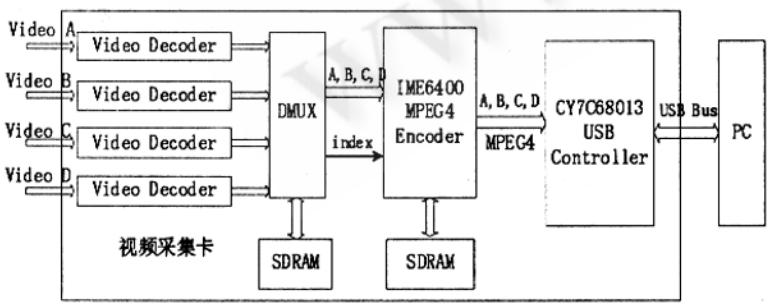


图 1 视频图像采集卡硬件框图

图像采集卡使用 Philips 公司制造的 SAA7114 视频芯片，用于接收 S－video(Y/C)或 CVBS 模拟视频信号，同时还有一个扩展端口可用于接收 MPEG1 格式的数字视频信号，芯片输出 8 位或 16 位宽的 NTSC 或 PAL 制式视频数据。解码输出通过数字多路复用器(DMUX)发送给视频图像压缩芯片。压缩芯片采用 InTime 公司制造的 IME6400 芯片，这是一款多路数字视频的 MPEG4 格式实时压缩芯片，NTSC 制式下，最大可支持 720×480 象素，30fps；PAL 制式下，最大可支持 720×576 象素，25fps，传输图像大小可以程序设置。USB 总线控制器采用的是 Cypress Semiconductor 公司制造的一款 USB2.0 控制芯片 CY7C68013，该芯片兼容 USB1.1 标准，支持全速和高速两种传输模式，内嵌有 8051 微控制器和 8Kbyte 的内部 RAM，用于运行固件程序处理复杂的 USB 低层协议，同时还具有一个 4KB 的 FIFO 可用于大规模数据透明传输。

### 2.2 系统结构

视频图像采集卡和服务器位于图像现场，共同构成前端视频图像服务器，完成图像采集和存储功能。客户通过网络来访问服务器，实现随时随地获取视频图像的功能。其中服务器系统软件由两部分组成：设备驱动程序和服务器应用软件。系统结构图见图 2。

### 2.3 系统工作过程

位于现场服务器主机通过视频采集卡控制多摄像头之间的视频切换来对多个现场的图像进行高速捕获，图像采集卡对图像进行硬件实时压缩编码，然后通过 USB 总线传输给视频服务器，视频服务器接收压缩的 MPEG4 格式的视频图像，并按一定的时间粒度进行分组存储，对每组数据采用时间戳标记来达到分时段检索功能，同时服务器采用 IP 多播的方式将图像数据实时发送给多个客户端。服务器还将响应用户请求完成历史图像分时段检索服务，对网络传输进行简单的流量控制和用户身份

验证以及权限管理等功能。客户端则完成图像数据接收、解码和播放等人机交互功能，因此要求具有友好的用户界面，可以在多种系统平台上进行开发，本文则主要是讨论服务器端软件开发过程。

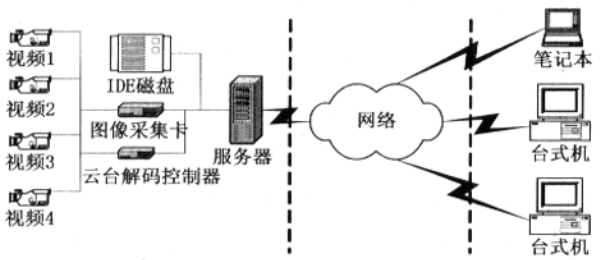


图 2　系统结构图

## 3　基于 Linux 平台的 USB 驱动程序设计

### 3.1　USB 主机软件系统结构

Linux 系统中设备驱动可以分为字符设备和块设备两类，字符设备是面向字符的 I/O 操作，只能顺序存取，而块设备则是面对数据块的 I/O 操作，所有的操作都是通过内核地址空间的 I/O 缓冲区来完成的，支持随机存取操作。系统加载设备驱动的方式有两种：模块加载和初始化加载，前者是通过 Root 用户使用 insmod 命令动态加载进内核中，通过 rmmod 命令卸载不需要的模块设备。USB 设备是通过快速串行通信的方式来读取设备的，是作为字符设备来处理的。

USB 主机软件是由主机控制器驱动程序（HC）、USB 驱动程序（USBD）和客户端驱动程序（Client Driver）3 个模块组成，其中 USBD 是整个 USB 主机软件的核心，负责管理所有的 HC、设备驱动程序和所有连接在 USB 总线上的设备。在 Linux 系统中 USB 子系统的体系结构采用了不对等、分层的方法，整个系统被分为 USB 接口层、USB 设备层和功能接口层，其中 USBD 和 HCD 被称作 USB 系统软件，它提供了支持客户端驱动程序开发的 API（应用编程接口），USB 客户端驱动程序不同于传统意义上的设备驱动程序在于不是通过 I/O 操作访问设备的，而是通过 USB 系统软件提供的标准接口与设备交互的，它和设备之间通信的基础是一组由管道组成的通道，客户端驱动程序利用 USBD 提供的功能创建所需的管道，并为它们提供数据传输的缓冲空间，使用 USBD 的功能进行数据传输。这种方式省去了客户端单独提供传输方式的繁杂，极大的简化了客户端驱动程序的设计。本系统中的 USB 驱动程序严格的讲是使用 USB 系统软件定义的数据结构，宏和函数接口来编写的图像采集卡客户端驱动程序。Linux 系统中 USB 子系统软件结构如图 3。

### 3.2　USB 客户端驱动程序的实现

USB 客户端驱动程序采用模块动态安装和卸载的方式，其实现功能是为应用层提供调用接口以便于应用层实现对设备进行访问操作，同时也为 USB 内核提供一个统一的接口来实现客户端驱动程序管理（安装、资源分配和卸载）、USB 设备接口的热插拔处理及对设备的控制和驱动等功能。

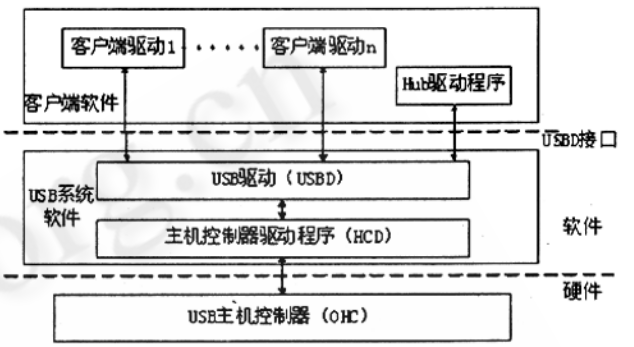


图 3　主机端 USB 软件结构

（1）图像采集卡设备驱动程序的初始化和卸载。USB 内核是通过一个双向链表 usb_driver_list 来维护所有安装在子系统中的客户端驱动程序，内核为客户端驱动程序的注册和卸载提供了两个接口函数：usb_register（struct usb_driver * new_driver）和 usb_deregister（struct usb_driver * driver），当驱动安装时需要调用 usb_register（）函数把驱动连接到 usb_driver_list 链表中，在本设备驱动初始化函数中完成驱动注册过程如下：

```
int __init usb_ime6400_init(void)
{
    if(usb_register(&ime6400_driver) <0)
        return -1;
        ⋮
    return 0;
}
```

当该设备驱动卸载的时候则调用 usb_deregister（）函数从 usb_driver_list 链表中摘下该设备驱动程序。其中 usb_driver 数据结构定义为：

```
static struct usb_driver ime6400_driver
{
    owner: THIS_MODULE,
    name: "IME6400",
    probe: probe_ime6400,/* 设备热插拔时接口配置函数 */
    disconnect: disconnect_ime6400,/* 卸载时释放占用资源 */
```

```
    fops: &usb_ime6400_fops,/* 设备操作的应用接口 */
    minor: IME6400_BASE_MNR,/* 定义的从设备号 */
    id_table: &ime6400_device_id,
};
```

(2) 对图像采集卡 USB 设备热插拔功能的处理。USB 设备支持热插拔功能,驱动程序中处理该功能的函数是:

```
static void * probe_ime6400(struct usb_device * dev,unsigned int ifnum,
const struct usb_device_id * id)和
static void disconnect_ime6400(struct usb_device * dev,void * ptr)
```

前者完成对图像采集卡设备接口的识别和配置功能,后者用于中止采集卡设备数据传输和释放其占用资源。

(3) 驱动程序提供的应用层接口。驱动程序提供的设备应用层操作接口结构如下:

```
static struct file_operations usb_ime6400_fops
{
    owner: THIS_MODULE,
    read: read_ime6400,
    ioctl: ioctl_ime6400,
    open: open_ime6400,
    release: close_ime6400,
};
```

open_ime6400() 用于打开图像采集卡设备文件,并完成设备初始化配置,压缩芯片的固件下载等操作,这里设置为输入为 PAL 制式,图像大小为 320×240,25fps。

read_ime6400() 主要用于图像采集卡的数据读操作。

ioctl_ime6400() 用于向设备发送各种控制命令。

close_ime6400() 用于停止设备传输工作,释放占用资源。

在这里考虑到视频数据的实时性,图像采集卡 USB 采用高速模式,传输类型为实时传输类型。

## 4 服务器软件设计

### 4.1 服务器软件设计思想

服务器软件为 C/S 模式下的服务器端软件,采用多线程机制。主线程主要负责用户管理和一些系统调度功能,多个子线程分别负责实现控制输出、数据采集、数据存储和网络通信等功能。视频数据交互则通过建立四块数据缓冲区的方式实现各线程之间的数据交互,四块数据缓冲区分别对应四路视频源。系统中的控制输出通过串口同云台解码器之间通过通信的方式来完成,通过自定义协议来控制对云台和摄像头的控制操作。

### 4.2 软件设计中采用的关键问题

#### 4.2.1 视频数据传递中的进程调度问题

系统需要实现 4 路 MPEG4 码流的采集、存储和网络实时输出,因而大量的视频数据流在不同线程之间的有效传递就成为影响系统性能的一个关键点,这里使用建立数据缓冲区的方式使数据采集线程、存储线程和网络通信线程之间实现数据交互,其中数据采集线程是数据生产者,存储和通信线程是数据消费者,线程之间的关系是互斥关系。由于 linux 操作系统进程调度是采用时间片轮转方式,而网络传输采用 IP 多播方式,数据在网络上传输存在着一定的时延,为了尽可能的提高系统实时性和系统资源利用率,我们把缓冲区设计成环行缓冲区,并用一个标示缓冲区状态信息的数据结构来维护,该数据结构主要包括一个写标示和两个读标示以及当前缓冲区数据量、缓冲区大小等其他状态信息。同时设定线程为不同的访问优先级,数据采集线程优先级最高,存储线程优先级最低,对环行缓冲区的互斥访问则通过互斥锁的方式来保证。为保证数据传输的高效性,我们设定了一个缓冲区容量阈值和一个标示位,当缓冲区中数据大于该阈值且标示位为 0 时,生产者以信号灯的方式唤醒睡眠中的消费者,然后将标示位置 1,当消费者读取完数据后将标示位重新置为 0 并进入睡眠等待状态。

访问环行缓冲区遵循如下原则:

(1) 当前优先级高的线程优先访问缓冲区。

(2) 缓冲区数据结构由所有访问者共同维护。生产者写入数据后移动写标示,并重新计算缓冲区数据量,消费者读取数据后,移动自己的读标示,并根据其他读标示来决定是否修改缓冲区数据量,只有当前读取的数据全部或部分已经被其他消费者读取才能修改缓冲区数据量。

(3) 消费者每次读取数据量不能超过一定的阈值,以保证每个线程访问时间的合理性,以及在出错情况下系统不至于产生大的数据量。由于优先级的引入,有可能因为高优先级线程的频繁访问导致低优先级线程长期得不到访问权限,对每个访问者引入一个计数器,当访问者进入等待状态时,启动计数器,每个高优先级的来访者都将低优先级的等待者的计数器加 1,当计数器值到一定阈值的时候,低优先级线程自动提高自身优先级,完成访问后则重置。我们在实现过程中发现确定缓冲区大小在系统设计中比较关键,如果太小则可能导致上溢或下溢,反之则导致系统处理时延增加,增大了系统耗费。

#### 4.2.2 视频数据流的存储和分时段检索

视频数据的存储一般有三种方式:完全存储,流水存储和报警存储。完全存储是将全部视频数据保存在硬盘中,流水存储是选择只保存 MPEG4 编码中的基本帧(即 I 帧和 P 帧或者只保存 I 帧),这样的图像质量略低,只适用于一般场合,报警存储是在特定条件下发生的存储,如报警信号触发

存储或预置时间段存储等。本系统采用两块 160G 容量的 IDE 硬盘作为数据存储介质，存储方式可以根据用户设置选取具体的存储方式。在数据存储中考虑到主机相对于用户来说是作为远端服务器应用，对存储空间使用循环存储的利用策略，当存储空间即将耗尽时用新数据覆盖最早一段时间的数据。

为了便于分时段随机检索系统将数据流按时间段分成一系列文件进行存储，以数据开始时间为文件名，并建立一个系统文件来记录整个硬盘存储情况，如视频数据存储方式，数据记录开始时间点，各视频源所属目录结构，视频数据操作历史记录等内容，这种机制可以极大的缩短数据随机检索时间和简化数据存储操作过程。

4.2.3 IP 多播中的流量控制

系统传输数据可以分为控制数据和视频图像数据流两类，其中控制信息有用户信息验证，控制指令，查询请求等，要求使用 TCP 方式进行点对点的可靠传输。实时视频数据传输为点对多点，少量的数据包丢失并不会影响用户接收效果，因而这里使用建立在 UDP 协议上的 IP 多播方式。由于历史数据查询一般情况下是点对点的数据传输，则采用 UDP 方式进行点对点传输。

用 Linux 套接字编程实现 IP 多播的方法很简单，发送方只要采用 D 类 IP 地址作为数据报分组接收地址，接收方要想接收组播数据报只需要通过 setsockopt( ) 函数发送 IP_ADD_MEMBERSHIP 套接字选项，然后就可以接收发往该组播地址的数据报了，该套接字选项类型是 ip_mreq 结构，定义如下：

```
struct ip_mreq {
    struct in_addr imr_multiaddr; /* IP multicast address of group */
    struct in_addr imr_interface; /* local IP address of interface */
};
```

退出组播组则用 setsockopt( ) 函数发送选项 IP_ROP_MEMBERSHIP 。

采用 IP 组播方式实现实时视频数据传送可以避免不必要的数据重复发送，减轻系统和网络负担，提高网络带宽利用率，改善数据传送实时性和数据传输效率。

在采用 IP 组播的基础上，为进一步提高系统对不同网络负荷的适应性，减少由于网络拥塞造成的数据分组丢失，我们采用了一个简单的流量控制策略。发送方将每个发送的分组进行编号，接收方统计各分组接收数目，如果分组丢失率达到一定的阈值(15%)，发送方则改变图像采集卡设置，降低 MPEG4 编码码流以达到减少分组丢失率。考虑到用户重要级别的不同，分组丢失率的计算如下：对相同级别的用户：

$$\text{分组丢失率} = \text{MAX}\left(1 - \frac{\text{分组到达数目}}{\text{分组预期到达数}}\right)$$

若存在不同级别的用户，则以高级别用户的分组丢失率为当前分组丢失率。

试验证明通过这种简单的码流控制策略，能够较好的提高系统在不同网络负荷情况下的适应能力。

4.2.4 用户管理和控制

用户权限管理在系统设计中是比较重要的，在这里我们把用户权限分为四级：0 级(设备级)，1 级(系统级)，2 级(控制级)，3 级(使用级)。其中 0 级级别最高，用于对系统进行远程调试和升级，该级别不提供给用户。1 级具有对系统管理进行配置的权限，可以添加用户或修改用户级别。2 级具有对视频设备进行远端控制的权限。3 级只具有接收视频数据的权限，属于最低权限。

系统对云台和摄像头的控制主要通过串口通信方式来完成，主要通过自定义协议的方式完成云台的上、下、左、右、自动扫描和摄像头的变焦等控制操作输出，并可通过增加类型定义来达到扩充的目的。

## 5 结论

本文全面描述了基于 Linux 平台搭建远程视频采集服务器的硬件和软件构成，并详细说明了系统设计过程，介绍了 Linux 操作系统下的 USB 客户端驱动程序设计和实现服务器系统软件过程中涉及的一些关键技术，如多线程编程中的线程调度问题，视频数据存储和分时段检索问题以及网络多播中采用的一些流量控制策略等。这些策略的使用使系统具有良好的网络适应能力，提高了系统性能，使之能够适用于多种场合下的远程视频采集监控等领域。

目前，Linux 操作系统在嵌入式领域中已经得到广泛的应用，本系统的设计可以通过极少的修改，即可移植到使用嵌入式 Linux 操作系统的嵌入式设备中构成嵌入式视频网络服务器，可见本系统具有良好的可扩展性。

### 参考文献


1 IME6400 MPEG4 Encoder Hardware Reference Manual, In-Time Corporation, April 30, 2004.

2 lessandro Rubini & Jonathan, Linux Device Drivers, 2nd Edition, O'Reilly & Associates, Inc ,June 2001.

3 Andrew S. Tanenbaum, Albert S. Woodhull, 操作系统：设计与实现(上册)，电子工业出版社，1998。

4 肖骁雄、翁铁成、宋中庆等，USB 技术及应用设计，清华大学出版社，2003。

5 张斌、高波等，Linux 网络编程，清华大学出版社，2000。