基于 IPv6 的 UDP 通信的实现

石炎生,刘利强
(湖南理工学院 计算机系,湖南 岳阳 414006)

摘 要:IPv6 是为了克服现在 IPv4 协议的缺陷而设计的下一代因特网协议。随着 Internet 的飞速发展,IPv6(Internet Protocol Version 6)必然会取代 IPv4,开发支持 IPv6 的网络应用程序已经迫在眉睫。文中首先对 IPv6 进行了简介;然后介绍了 UDP 的通信原理、IPv6 下 Socket 地址结构和 IPv6 下 Socket 函数,并给出了在 IPv6 下基于无连接的 Socket 的客户机/服务器模型和利用 UDP 协议进行网络通信的方法;最后实现了在 IPv6 协议下服务器端和客户端之间数据信息的传输,相信对今后开发基于 IPv6 的大型网络应用程序会有极大的帮助。

关键词:IPv6;UDP;Socket;网络编程
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2006)05-0191-03

Realization of UDP Communication Based on IPv6

SHI Yan-sheng, LIU Li-qiang
(Department of Computer, Hunan Institute of Science and Technology, Yueyang 414006, China)

Abstract: IPv6 is intended to be the next network layer protocol of the Internet and it is designed to overcome the limitations of the current IPv4 protocol. With the quick development of Internet, the IPv6(Internet protocol version 6) certainly can replace IPv4, developing and supporting network applications program of IPv6 extremely urgent. This article introduces IPv6, UDP communication principle, Socket address structure and Socket function, and provides the client/server model of Socket on non-connect and method of network communication on UDP protocol for IPv6. Believe developing based on IPv6 in the large network applications program can have great help.

Key words: IPv6; UDP; Socket; network programming

0 引 言

随着计算机技术的发展和网络的不断普及,目前的 IPv4 协议在许多方面已经显得不太适应,如 IPv4 地址严重不足、缺乏服务质量控制 QoS、安全性差和移动接入不方便等问题。IETF (Internet Engineer Task Force)组织已意识到:目前的 IPv4 网络协议已很不适用了,因此提出了下一代的网络协议 IPv6。IPv6 是为适应未来对于网络基础设施的数量和质量的需求而设计的下一代互联网协议,许多机构都已经开始了这方面的测试和研究,并且逐渐被主流的操作系统和网络硬件设备所支持。IPv6 协议作为目前热门的研究领域,相应的协议和标准还在不断的制订、讨论和变化中,这些都需要人们积极地参与研究、开发和实践。文中将探讨基于 IPv6 的 UDP 网络通信程序设计方法。

1 UDP 的通信原理

UDP(User Datagram Protocol,用户数据包协议)是一种无连接的通讯协议。UDP 协议将独立的数据包从一台计算机传输到另外一台计算机,但是并不保证接受方能够接收到该数据包,也不保证接收方所接收到的数据和发送方所发送的数据在内容和顺序上是完全一致的。因此,UDP 协议更类似于普通邮政服务,寄信人不能够保证所寄出去的信能够被收信人及时收到,后发出的信也许会比先发出的信更早到达。由于 UDP 协议比较简单,UDP 头包含很少的字节,比 TCP 负载消耗少,适用于对可靠性要求不高的应用环境。图 1 是使用无连接 Socket 的客户机/服务器模型。

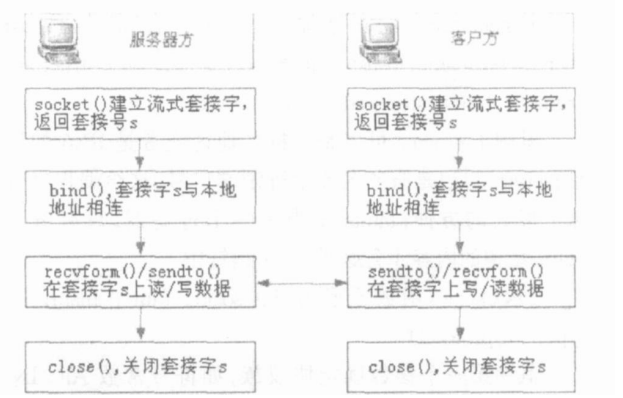

图 1 基于无连接的 Socket 的客户机/服务器模型

收稿日期:2005-11-17
基金项目:湖南省自然科学基金项目(05JJ40110)
作者简介:石炎生(1961-),男,湖南岳阳人,副教授,研究方向为计算机网络及虚拟现实技术。

2 IPv6 下 Socket 套接字

2.1 IPv6 的 Socket 地址结构

IPv6 使用 128 位地址,定义了本身的专用 Socket 地址结构[1]。

⑴sockaddr_in6 结构。

```
struct sockaddr_in6 {
sa_family_t sin6_family; //必须是 AF_INET6
in_port_t sin6_port; //传输层端口号
uint32_t sin6_flowinfo; //IPv6 拥塞级别与流信息
struct in6_addr sin6_addr; //128 位的 IPv6 地址
uint32_t sin6_scope_id; //IPv6 的接口范围
};
```

其中,sin_family 表示地址所属的协议族,对于 IPv6 地址来说,始终为 AF_INET6;sin6_flowinfo 包含 IPv6 流信息;sin6_addr 标识一个 128 位的 IPv6 地址,in_addr6 结构如下所示:

```
struct in6_addr{
uint8_t s6_addr[16]; //IPv6 地址
};
```

⑵addrinfo 结构。

```
struct addrinfo {
int ai_flags; //AI_PASSIVE, AI_CANONNAME,AI_NUMERICHOST
int ai_family; //必须是 AF_INET6
int ai_socktype; //字节流用 SOCK_STREAM,数据报用 SOCK_DGRAM
int ai_protocol; //TCP 协议用 IPPROTO_TCP,UDP 协议用 IPPROTO_UDP
socklen_t ai_addrlen; //ai_addr 地址长度
char *ai_canonname; // 规范名
struct sockaddr *ai_addr; // sockaddr 地址结构
struct addrinfo *ai_next; // 指向下一个 addrinfo 结构
};
```

2.2 IPv6 的 Socket 函数

IPv6 的 Socket API 函数中一部分沿用了 IPv4 的 Socket 函数,也新增了一些 IPv6 专用的 Socket 函数,IPv6 的 Socket 函数如表 1 所示[2]。

1) getaddrinfo() 函数原形为:int getaddrinfo (const char * hostname, const char * servname, const struct addrinfo * hints, struct addrinfo * * res)

其中 hostname 可以是主机名地址或者是 IPv6 的地址;service 可以是服务名或十进制端口号,这给编程也带来了很大的方便;Hints 相当于一个过滤器,只有符合 Hints 结构的内容才会返回到 res 指针中。

2) socket() 函数原形为:int socket (int domain, int type, int protocol)

其中第一个参数确定协议族,如符号常数 AF_INET6 表示用 IPV6 协议;第二个参数指明了想使用的通信服务类型,如符号常数 SOCK_DGRAM 表示数据报,用 SOCK_STREAM 表示字节流;第三个参数指明此 Socket 请求使用的协议,如符号常数 IPPROTO_TCP 表示用 TCP 协议,用 IPPROTO_UDP 表示用 UDP 协议[3]。

表 1 IPv6 的 Socket 函数

IPv6 函数	功能说明
inet_ntop()	字符串地址转为 IP 地址
inet_pton()	IP 地址转为字符串地址
getipnodebyname()	由名字获得 IP 地址
getaddrinfo()	获得全部地址信息
getnameinfo()	获得全部名字信息
socket()	建立 Socket
bind()	Socket 与地址绑定
listen()	网络监听
accept()	接收 TCP 连接
connect()	建立 TCP 连接
send()	发送数据(TCP)
sendto()	发送数据(UDP)
recv()	接收数据(TCP)
recvfrom()	接收数据(UDP)
close()	关闭连接

3) listen() 函数的函数原型为:int listen(int sockfd, int backlog)

其中 sockfd 是 Socket 函数返回的 Socket 描述符;backlog 指定在请求队列中允许的最大请求数。

4) sendto() 函数原型为:int sendto (int sockfd, const void *buf, int len, unsigned int flags, struct sockaddr *ai_addr, socklen_t ai_addrlen);

其中 sockfd 是你想用来传输数据的 Socket 描述符;buf 是存放发送数据的缓冲区;len 是以字节为单位的数据的长度;flags 一般情况下置为 0,struct sockaddr *ai_addr 确定地址,socklen_t ai_addrlen 指明目的地址的大小。

5) recvfrom() 函数原型为:int recvfrom(int sockfd, void *buf, int len, unsigned int flags, struct sockaddr *addr, int *addrlen);

其中 sockfd 是接受数据的 Socket 描述符;buf 是存放接收数据的缓冲区;len 是缓冲的长度;flags 也被置为 0,addr 通常是一个指向 sockaddr 变量的指针;addrten 通常为一个指向值为 sizeof (struct sockaddr) 的整型指针变量[4]。

3 编程实现

3.1 服务器端程序设计

服务器端首先启动,通过调用 Socket() 建立一个 Socket,然后调用 bind() 将该 Socket 和本地网络地址联系在一起,接着服务器端就可以通过 recvfrom() 和 sendto() 来接收和发送数据。最后,待数据传送结束后,调用 close

() 关闭 Socket[5]。服务器端设计过程如下。

⑴创建服务器端套接字。

```
memset ( &hints , 0 , sizeof (hints)) ;
hints. ai _ family = AF _ INET6 ; / / 指定用 IPV6 协议
hints. ai _ socktype = SOCK _ DGRAM ; / / 指定用数据报
hints. ai _ protocol = IPPROTO _ UDP ; / / 指定用 UDP 协议
hints. ai _ flags = AI _ NUMERICHOST ; / / IP 用数字表示
rc = getaddrinfo (“: :1 ”,“5001 ”, &hints , &res) ; / / 解析本机地址
s _ send = socket (res - > ai _ family , res - > ai _ socktype , res - > ai _ protocol) ;/ / 创建 socket
```

⑵绑定本机监听端口。

```
rc = bind(s _ send ,res - > ai _ addr ,res - > ai _ addrlen) ;
```

⑶接收数据。

```
recvfrom( s _ send , buf , sizeof ( buf) , 0 , ( struct sockaddr * ) &sin , &sin _ len)) ; / / 接受数据到 buf
printf (“recvfrom : %s ”,buf) ; / / 打印接受的数据
```

⑷关闭套接字。

```
close(s _ send ) ;
```

3.2 客户端程序设计

客户端创建一个 Socket ,然后调用 bind() 将该 Socket 和远地网络地址联系在一起,客户端就可以通过 recvfrom () 和 sendto () 来接收和发送数据。最后,待数据传送结束后,调用 close() 关闭 Socket。客户端设计过程如下。

⑴创建客户端套接字。

```
memset ( &hints , 0 , sizeof (hints)) ;
hints. ai _ family = AF _ INET6 ; / / 指定用 IPV6 协议
hints. ai _ socktype = SOCK _ DGRAM ; / / 指定用数据报
hints. ai _ protocol = IPPROTO _ UDP ; / / 指定用 UDP 协议
hints. ai _ flags = AI _ NUMERICHOST ; / / IP 用数字表示
rc = getaddrinfo (“3ffe :3211 : :1 ”,“5001 ”, &hints , &res) ; / / 解析服务器地址
s _ send = socket (res - > ai _ family , res - > ai _ socktype , res - > ai _ protocol) ; / / 创建 socket
```

⑵绑定远地监听端口。

```
bind (s _ send ,res - > ai _ addr ,res - > ai _ addrlen) ;
```

⑶发送数据。

```
sendto(s _ send ,buf ,sizeof(buf) ,0 ,res - > ai _ addr ,res - > ai _ addrlen) ;
```

⑷关闭套接字。

```
close (s _ send) ;
```

4 结束语

随着 Internet 技术的不断发展, IPv6 作为一种新的 Internet 协议必定会取代 IPv4 成为下一代的互联网协议,开发支持 IPv6 的网络应用程序的问题会变得越来越重要。开发在 IPv6 下用 UDP 协议的服务器和客户机应用程序,能够较好地实现服务器端和客户端之间数据信息的传输,对开发基于 IPv6 的大型网络应用程序会有极大的帮助。

参考文献:

[1] RFC2553. Basic Socket Interface Extensions for IPv6[S]. 1999.

[2] RFC2292. Advanced Sockets API for IPv6[S]. 1998.

[3] RFC3493. Basic Socket Interface Extensions for IPv6[S]. 2003.

[4] Davies J. 理解 IPv6[M]. 张晓彤等译. 北京:清华大学出版社,2004.

[5] Stevens W R. TCP/ IP 详解(卷 3)[M]. 胡谷雨等译. 北京:机械工业出版社,2000.

(上接第 190 页)

据。

(2) 技术特色:

a. 采用先进的分布式多层架构,应用了 COM + , VS. NET 等领先技术。

b. 解决了远程大型数据库数据共享的难题。

c. 基于 Web 的数据库通用查询和对比分析。

d. 基于 Web 的报表定制技术。

e. 提供了报表、曲线、图形等丰富的表现形式。

7 结束语

永宁钻采公司管理信息系统的建立,可以充分利用各种“信息资源为生产服务;实现整个公司数据共享”[3],提高了永宁钻采公司的生产管理透明度,使相关工序、相关部门可随时了解彼此间生产状况,及时进行生产调整。有关人员可以从繁重的数据处理、统计计算、报表加工工作中解放出来,多花一些时间去考虑生产中存在的问题,多做一些生产分析、质量分析,去寻找提高产品产量和产品质量的技术改进方法和管理措施。“系统的建立,有助于带动和提高管理人员的素质[4]。”可以肯定地讲,系统的建立将给生产管理带来可观的社会效益和经济效益,同时也将永宁钻采公司的生产管理推上一个新台阶。

参考文献:

[1] 刘 渝,李 涛,蒋红红. 油田生产运行信息管理系统[J]. 油气田地面工程,1999,22(7) :13 - 14.

[2] 张培宏,陈武新. 油田生产实时监控系统设计方案探讨[J]. 测绘通报,2004(5) :26 - 28.

[3] 汤 军. 采油厂静态信息 GIS 综合管理系统设计与实现[J]. 江汉石油学报,2004,26(3) :33 - 34.

[4] 陈 虎,孙 鹏. 地理信息系统技术在油田开发方案设计中的应用[J]. 电脑开发与应用,2000,16(9) :29 - 31.

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较
42. 基于 PCI Express 总线系统的热插拔设计
43. 基于 RAID5 的磁盘阵列 Cache 的研究与实现
44. 基于 PCI 总线的 MPEG2 码流播放卡驱动程序开发
45. 基于磁盘异或引擎的 RAID5 小写性能优化
46. 基于 IEEE1588 的时钟同步技术研究
47. 基于 Davinci 平台的 SD 卡读写优化
48. 基于 PCI 总线的图像处理及传输系统的设计
49. 串口和以太网通信技术在油液在线监测系统中的应用
50. USB30 数据传输协议分析及实现
51. IEEE 1588 协议在工业以太网中的实现
52. 基于 USB30 的设备自定义请求实现方法
53. IEEE1588 协议在网络测控系统中的应用
54. USB30 物理层中弹性缓冲的设计与实现
55. USB30 的高速信息传输瓶颈研究

VxWorks:

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈

16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发
23. Vxworks 下的冗余 CAN 通讯模块设计
24. WindML 工业平台下开发 S1d13506 驱动及显示功能的实现
25. WindML 中 Mesa 的应用
26. VxWorks 下图形用户界面开发中双缓冲技术应用
27. VxWorks 上的一种 GUI 系统的设计与实现
28. VxWorks 环境下 socket 的实现
29. VxWorks 的 WindML 图形界面程序的框架分析
30. VxWorks 实时操作系统及其在 PC104 下以太网编程的应用
31. 实时操作系统任务调度策略的研究与设计
32. 军事指挥系统中 VxWorks 下汉字显示技术
33. 基于 VxWorks 实时控制系统中文交互界面开发平台
34. 基于 VxWorks 操作系统的 WindML 图形操控界面实现方法

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核

18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings
25. Android 应用逆向分析方法研究
26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版
30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现
33. LINUX ARM 下的 USB 驱动开发
34. Linux 下基于 I2C 协议的 RTC 驱动开发
35. 嵌入式下 Linux 系统设备驱动程序的开发
36. 基于嵌入式 Linux 的 SD 卡驱动程序的设计与实现
37. Linux 系统中进程调度策略
38. 嵌入式 Linux 实时性方法
39. 基于实时 Linux 计算机联锁系统实时性分析与改进
40. 基于嵌入式 Linux 下的 USB30 驱动程序开发方法研究
41. Android 手机应用开发之音乐资源播放器

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法

13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现
19. 基于 WinCE 的嵌入式图像采集系统设计
20. 基于 ARM 与 WinCE 的掌纹鉴别系统
21. DCOM 协议在网络冗余环境下的应用

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植
13. 基于 PowerPC 的雷达通用处理机设计
14. PowerPC 平台引导加载程序的移植
15. 基于 PowerPC 嵌入式内核的多串口通信扩展设计
16. 基于 PowerPC 的多网口系统抗干扰设计
17. 基于 MPC860T 与 VxWorks 的图形界面设计
18. 基于 MPC8260 处理器的 PPMC 系统
19. 基于 PowerPC 的控制器研究与设计
20. 基于 PowerPC 的模拟量输入接口扩展
21. 基于 PowerPC 的车载通信系统设计

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计
22. ARM CortexM3 处理器故障的分析与处理
23. ARM 微处理器启动和调试浅析
24. 基于 ARM 系统下映像文件的执行与中断运行机制的实现
25. 中断调用方式的 ARM 二次开发接口设计
26. ARM11 嵌入式系统 Linux 下 LCD 的驱动设计
27. Uboot 在 S3C2440 上的移植
28. 基于 ARM11 的嵌入式无线视频终端的设计
29. 基于 S3C6410 的 Uboot 分析与移植

Hardware:

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计

7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究
15. 基于 UEFI 固件的攻击验证技术研究
16. 基于 UEFI 的 Application 和 Driver 的分析与开发
17. 基于 UEFI 的可信 BIOS 研究与实现
18. 基于 UEFI 的国产计算机平台 BIOS 研究
19. 基于 UEFI 的安全模块设计分析
20. 基于 FPGA NiosⅡ的等精度频率计设计
21. 基于 FPGA 的 SOPC 设计
22. 基于 SOPC 基本信号产生器的设计与实现
23. 基于龙芯平台的 PMON 研究与开发
24. 基于 X86 平台的嵌入式 BIOS 可配置设计
25. 基于龙芯 2F 架构的 PMON 分析与优化

Programming:

1. 计算机软件基础数据结构 - 算法
2. 高级数据结构对算法的优化
3. 零基础学算法
4. Linux 环境下基于 TCP 的 Socket 编程浅析
5. Linux 环境下基于 UDP 的 socket 编程浅析
6. 基于 Socket 的网络编程技术及其实现