Linux 环境下基于 TCP 的 Socket 编程浅析

吴佩贤
(绍兴文理学院 计算机系　浙江 绍兴　312000)

摘　要: Socket 适用于同一台计算机上的进程间通信，同时也适用于网络环境中的进程间通信。他已成为当前许多操作系统的网络 API，也是网络操作系统中必不可少的基础功能。因特网为网络中的应用提供了 2 种类型的服务：由 TCP 协议提供的面向连接服务和由 UDP 协议提供的无连接服务。随着 Linux 操作系统的不断推广，Linux 环境下的 Socket 开发和研究一直是人们关注的热点。文章介绍了 Linux 平台下的 Socket 及其在 TCP 协议下的编程原理，并通过一个用 Java 编写的基于 TCP 的客户/服务器程序，描述了网络中不同主机上的两个进程之间的 Socket 通信机制。

关键词: Socket; TCP/IP; 客户机/服务器; 进程; Java
中图分类号: TP316　　**文献标识码**: B　　**文章编号**: 1004　373X (2005) 16　053　03

Analysis of Socket Programming TCP based on Linux

WU Peixian
(Computer Science Department, Shaoxing University, Shaoxing, 312000, China)

Abstract: Socket is suitable for communication between two processes not only from one computer but also from network. Now Socket is an available network API of many OS, at the same time, it is one necessary part of network OS. Internet provides connection-oriented service (TCP based) and connectionless service (UDP based). Socket development and studying on Linux are always concerned while Linux is innovated on and on. The paper introduces the Socket and it's programming principle based on TCP in Linux environment. By using a TCP based client/server program which is edited in Java, it describes Socket communication mechanism between the two processes from the different computers in the network.

Keywords: Socket; TCP/IP; client/server; process; Java

Linux 是一个诞生于 Internet 和 WWW 的产品，他和网络密切相关。Linux 网络系统具有稳定、简易、高效、兼容性好等特点，并且支持多种网络协议，如 IP v4，IP v6，X.25，IPX，NETBIOS，DDP 等。套接字（Socket）是介于网络应用层和传输层之间的编程接口，套接字接口提供了访问下层通信协议的大量系统调用和相应的数据结构。在 Linux 中，套接字接口是应用程序访问下层的网络协议的惟一方法。具体讲，套接字在用户级实现了两个应用程序之间的网络连接和数据交换，所以 Linux 中的套接字意味着网络上的连接。套接字在 TCP/IP 网络模型中的地位如图 1 所示。

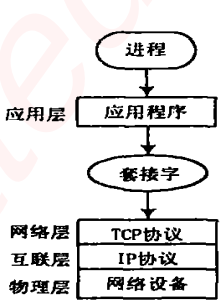

图 1　套接字

1　BSD 套接字接口

Socket 接口是为方便开发人员进行 TCP/IP 程序开发，而为 TCP/IP 协议所制定的一组应用程序接口。由于他最早应用于伯克莱大学的 BSD Unix 中，所以习惯上又称其为 BSD Socket（简称 BSD）。一个套接字描述为一个通信连接的一端，在一个通信连接中的两端通信程序应各自有一个套接字来描述他们自己那一端，不同主机中的两个进程通过各自的套接字发送和接收消息，从而实现进程间跨网络的通信。

Linux 的套接字支持多种网络协议，不同类型网络协议的工作方式不同，所使用的地址格式也完全不同。对于各种网络协议而言，使用相同地址格式的几个协议称为一个协议地址簇，表 1 列出了 BSD 套接字的常见地址簇。

表 1　BSD 套接字地址簇的主要类型

地址簇类型	对应的通信协议
AF_INET	TCP/IP 协议
AF_IPX	Novell IPX 协议
AF_UNIX	Unix 内部套接字
AF_AX25	AX.25 协议套接字
AF_APPLETALK	APPLETALK DDS (Macintosh 机器用)

Linux 将套接字地址簇抽象为统一的 BSD 套接字接

口，该接口是应用程序的开发接口，由各地址簇专有的软件支持。Linux BSD Socket 支持以下常见套接字类型:

(1) SOCK_STREAM（数据流套接口）：提供一个面向连接的双工顺序数据流传输和可靠的数据传输服务。这种套接字可以保证数据传输的可靠性，不会出现数据丢失、破损或重复出现等差错，而且通过流量控制避免发送的数据流超限。Internet 地址中的 TCP 协议支持流套接字。

(2) SOCK_DGRAM（数据报套接口）：提供一个无连接和不可靠的双工数据传输服务。数据包以独立包形式被发送和接收。不对数据的传输提供无错保证，即数据可能被丢失、破坏，也可能被重复接收。Internet 地址中的UDP 协议支持这种套接字。

(3) SOCK_RAW（原始套接口）：这种类型的套接字允许对低层协议如 IP 或 ICMP 直接访问，可以直接填充 IP，TCP，UDP 或者 ICMP 的包头，发送用户自己定义的 IP 包或者 ICMP 包。主要用于一些协议的开发，在网络安全的抓包中有重要的应用。

2 Linux 平台下的 Socket 编程原理

在Linux 中，Socket 属于文件系统的一部分，网络通信可以被看作是对文件的读取。这使得用户对网络的控制极为方便，Linux 的许多特性都非常有助于网络编程：首先，Linux 拥有 POSIX 标准库函数，Socket()，bind()，listen()，send()，accept() 这几个库函数可以方便地实现客户机/服务器模型中数据的传送与接收。其次，Linux 的进程管理策略也非常适合服务器的工作环境，Linux 中的每个进程都对应一个父进程，同时他也能创建多个子进程。在服务器端可以用父进程去侦听客户机的连接请求，当有客户机的连接请求发生时，父进程创建一个子进程去建立连接线路并与该客户机通信，而他本身可继续侦听其他客户机的连接请求，这样就可避免当有一个客户机与服务器建立连接后服务器就不能再与其他客户机通信的问题。最后，Linux 传承了Unix 设备无关性这一优秀特性，即通过文件描述符实现了统一的设备接口：磁盘、显示终端、音频设备、打印设备甚至网络通信都使用统一的 I/O 调用。这 3 个特性大大方便和简化了Linux 环境下的网络程序设计。

(1) Socket 函数

Socket()用来创建套接口描述符，其格式声明为:

int Socket(int domain, int type, int protocol)

参数 domain 说明网络程序所在的主机采用的通信协议，如AF_INET（IP v4 协议），AF_INET6（IP v6 协议），AF_LOCAL（Unix 域协议）；参数 type 指定套接口类型（数据流套接口、数据报套接口、原始套接口），即相当于指明了网络程序所采用的通信协议（TCP 还是 UDP）；参数protocol 由于指定了 type，一般取 0 即可。

(2) bind 函数

bind()用来将套接口绑定到本地计算机的某一端口，其格式声明为:

int bind(int sockfd, struct sockaddr * my_addr, int addrlen)

sockfd 为套接口描述字；指针my_addr 指向 sockaddr 结构，该结构包含了远程服务程序的 IP 地址与端口号；addrlen 指明 sockaddr 结构的长度。

(3) listen 与 accept 函数

服务器端通过 listen()等待一个进入的连接请求，接收到的请求存于队列中。当有多个客户端和服务器相连时，通过参数 backlog 指定队列的最大长度。accept()则从已接收的连接请求队列中取出请求并分析这个请求，若队列为空，则使服务器端程序阻塞。

本文主要阐述的是面向连接的数据流通信的 Socket 编程过程。客户机/服务器主要编写客户程序和服务器程序，以下是主要的编程框架。

面向连接的数据流通信中，服务器端程序基本的编写步骤如图 2 所示。首先，在服务器端使用 Socket()建立一个通信端口，再用 bind()命令把一个地址绑定到这个端口上。然后，服务器端使用 listen()侦听连接请求，当远程的客户机试图通过 connect()连接 listen()正在侦听的端口时，连接将会在队列中等待，直到使用 accept()处理他。在 accept()处理了连接请求后，将会生成一个新的描述这个连接端口的套接字，利用这个套接字就可以发送和接收数据了。如果 listen()一直没有侦听到连接请求，那么服务器任务就会在 accept()处阻塞(在阻塞模式下)，一直到有连接请求到来。不论何种编程语言，和 Socket 打交道都是这一组调用，只是在格式上有所区别。Visual Basic 采用 WinSock 控件，C++ Builder 采用 TclientSocket 元件和 TserverSocket 元件，Visual C++ 采用 MFC 类中的 CAsyncSocket 类和 CSocket 类，Java 中通过 Java.net.Socket 和 Java.net.ServerSocket 类库来实现网络之间的通讯。

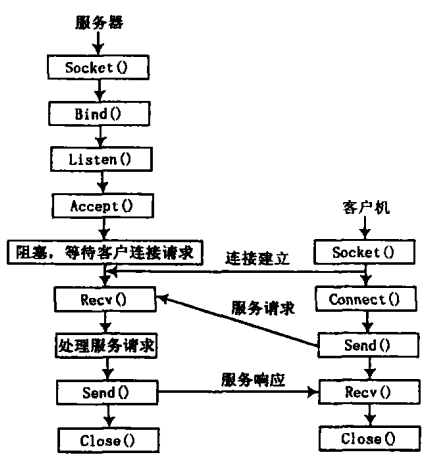

图 2 TCP 客户机/服务器程序的工作流程

对于客户机任务,他也需要先用 Socket()建立一个通信端口,但是不必用 bind()把一个本地地址绑定到这个端口上,而是直接使用 connect()向指定的服务器发送连接请求。函数 connect()调用成功后,套接口描述字就与远程服务程序建立好了连接,建立连接后通过 send()和 recv()实现数据的发送与接收。send()返回实际发送的字节数,如果返回的字节数比要发送的字节数少,则在以后必须发送剩下的数据。套接口读写完毕后,可通过 close()调用关闭连接的套接口文件描述符。

3 用 Java 编写的客户/服务器应用程序示例

网络应用程序的核心由一对程序组成——一个客户程序和一个服务器程序。当这 2 个程序执行的时候,系统就会创建一个客户端进程和一个服务器端进程,并且这 2 个进程通过对 Socket 的读写来互相通信。下面就客户/服务器模型来对 TCP 的 Socket 编程进行示范,程序功能如下:客户端从其键盘读入一行并将该行发送到通往服务器的套接字上;服务器从其连接套接字读入一行并将该行转换成大写;服务器将改写后的这一行输出到通往客户端的连接套接字上;客户端从他的套接字中读出修改后的行,并将该行在显示器上打印输出。

(1) 应用程序的客户端代码如下:

```
import java.io.*;
import java.net.*;
class TCPClient {
    public static void main(String argv[]) throws Exception
    {
      String sentence;
       //sentence 是用户在键盘打出并发送给服务器的字符串
      String modifiedSentence;
      BufferedReader inFromUser = new BufferedReader(
      new InputStreamReader(System.in));
      //创建BufferedReader 类型的流对象 inFromUser
      Socket clientSocket = new Socket("hostname", 1234);
      //创建 Socket 类型的对象 clientSocket,还对客户和服务
        器间的 TCP 连接进行初始化。字符串"hostname"必须
        用服务器的主机名,1234 是假设的端口号。
      DataOutputStream outToserver = new
      DataOutputStream (clientSocket.getOutputStream());
       BufferedReader inFromServer = new BufferedReader
(new InputStreamReader(clientSocket.getInputStream()));
    //创建了两个连接在套接字上的流对象:outToServer 流为
      进程提供了到套接字的输出;inFromServer 流为进程提
      供了从套接字的输入
      sentence = inFromUser.readLine();
      outToServer.writeBytes(sentence + '\n');
      //将字符串 sentence 加上一个回车,传送到 outToServer
        流中
      modifiedSentence = inFromServer.readLine();
      //从服务器接收字符,经过流 inFromServer 放入字符串
        modifiedSentence 中
      System.out.println("String From Server:" + modified
Sentence);
      ClientSocket.close();
                          //关闭客户与服务器间的 TCP 连接
    }
}
```

java.io 和 java.net 是 Java 的包,java.io 包包含了输入和输出流的类,java.net 包提供了网络支持类。流 inFromUser 是该程序的一个输入流,当用户在键盘上打出字符的时候,字符就进入了 inFromUser 流中。InFromServer 流是程序的另一个输入流,他被加入到套接字 clientSocket 上。从网络中到来的字符进入流 InFromServer 中。最后,流 outToServer 是该程序的输出流,他也被加入在套接字上。客户端发送到网络中的字符流入 outToServer 流中。

(2) 应用程序的服务器端代码如下:

```
import java.io.*;
import java.net.*;
class TCPServer {
    public start void main(String argv[]) throws Exception
    {
      String clientSentence;
      String resultSentence;          //希望得到的结果字符串
       ServerSocket welcomeSocket = new ServerSocket
       (1234);
      //创建对象 welcomeSocket,他是等待客户端到来的敲击
        的门。端口号 1234 对服务器进程进行标识
      while(true) {
       Socket connectionSocket = welcomeSocket.accept();
      //当有客户敲击 welcomeSocket 时,该行创建一个新的套接
        字 connectionSocket,然后 TCP 建立了一个在客户
        clientSocket 和服务器 connectionSocket 间的连接,客户端
        和服务器端就可以通过该连接相互传送字节了。需要说明
        的是,connectionSocket 建立后,服务器应该可以为使用
        welcomeSocket 的应用程序继续侦听来自其他客户端的请
        求,为简化阐述,本程序并未侦听更多的连接请求。
    BufferedReader inFromClient = new BufferedReader (new
InputStreamReader(ConnectionSocket.getInputStream()));
    DataOutputStream outToClient = new DataOutputStream
(connectionSocket.getOutputStream());
     clientSentence = inFromClient.readLine();
     resultSentence = clientSentence.toUpperCase() + '\n';
     outToClient.writeBytes(resultSentence);
                                  //输出结果,并等待下一输出
    }
  }
}
```

一旦两个程序在各自的主机上被编译了,服务器程序就首先在服务器上执行,他在服务器上创建了一个进程。如前所述,服务器进程等待着一个客户进程与他进行联系。当客户程序执行时,客户端创建一个进程,并且这个进程与服务器进行联系并建立一个到服务器的 TCP 连接。然后客户端的用户可以使用该应用程序来发送一行并接收该行的转换成大写之后的字符串。

4 结 语

Java 是平台无关的,且编写的程序简洁,因而用 Java

核。在此过程中，FPGA Architect - Xilinx XC4000/Spartan 功能将对设计进行校验，检验参数是否符合 Xilinx DSP LogicCORE 的要求。符合要求后，将设计转换到 Xilinx 的设计环境中，其中包括 VHDL 的行为级仿真模型和网表文件（NetList）。

多电平判决模块的判决门限应该能区分 4 个电平，所以可以采用数值为 1，2 和 3 的 3 个门限，将 0～4 的区间等分，从而可以将信号量化为 0，1，2，3 四个电平。

5 结 语

软件无线电的提出和发展，标志着无线通信从硬件到软件的质的飞跃，是未来无线电通信的发展方向。近年来 FPGA 技术迅速发展，他既具有 ASIC 的高速处理能力，又拥有很好的可重构性能，而且开发成本低、开发周期短，使其成为实现软件无线电的数字信号处理的一种非常有效的选择。本文对其中较为关键的 16QAM 调制解调系统提出了 FPGA 的实现方法，其他调制解调方式也可类似得到。

参 考 文 献

[1] 徐以涛，沈良，王金龙. FPGA 技术在软件无线电中的应用[J]. 电信科学，2001，11（17）.

[2] 曾艺城. 可编程逻辑技术及其在软件无线电中的应用- 数字调制解调系统的 FPGA 实现[D]. 北京：北京航空航天大学，2001.

[3] 汪东艳. 软件无线电技术与可重配置计算体系结构[J]. 今日电子，2002，(7)：12 13.

作者简介 苏小妹 2002 级电路与系统专业硕士研究生。主要研究方向为软件无线电在第三代移动通信系统中的应用。

黎福海 硕士生导师，副教授，湖南大学电气与信息工程学院副院长。主要研究方向为现代网络与通信技术、电子系统与智能化系统设计、集成电路设计与测试技术。

汪 浩 2002 级电路与系统专业硕士研究生。

（上接第 52 页）

据方案的核心对象。Dataset 是数据的内存驻留表示形式，无论数据源是什么，他都会提供一致的关系编程模型。他可以用于多个不同的数据源，用于 XML 数据，或用于管理应用程序本地的数据。Dataset 表示包括相关表、约束和表间关系在内的整个数据集。

Dataset 中的方法和对象与关系数据库模型中的方法和对象一致。Dataset 也可以按 XML 的形式来保持和重新加载其内容，并按 XML 架构定义语言（XSD）架构的形式来保持和重新加载其架构。

4 结 语

ADO.NET 是当前数据库技术处理的标准，集合了所有用于数据处理的类和程序化接口。这些类和接口代表了数据容器对象，他们以典型的数据库功能为特色。同时 ADO.NET 是 .NET 数据库应用程序的最终解决方案。他以整体设计为特色，是一个新的数据访问编程模型，能够帮助程序员以更加有效的方式构建高效数据库应用程序。

参 考 文 献

[1] 杨志娟，李朋朋，马云艳，等. 用 Visual Basic·NET 和 Visaul C# 开发 XML Web 服务与服务组件[M]. 北京：清华大学出版社，2002.

[2] 孙三才，许薰尹. 程序设计 C# &ASP.NET [M]. 北京：中国青年出版社，2001.

[3] 康际科技. ASP.NET 行家设计实务[M]. 北京：中国铁道出版社，2001.

[4] 石磊. VB.NET 与数据库开发[M]. 北京：人民邮电出版社，2001.

[5] FredBorell.VB.NET 高级编程[M]. 康博译. 北京：清华大学出版社，2001.

[6] Douglas Reilly J.ASP.NET Web 应用程序开发新思维[M]. 北京：清华大学出版社，2002.

作者简介 孔延香 女，1974 年出生，西北民族大学计算机科学与信息工程学院 2003 级计算机应用技术硕士。研究方向为计算机网络数据库应用。

（上接第 55 页）

进行客户/服务器编程越来越流行。以上介绍了 Socket 的编程机制及在 Linux 系统下的一个套接口程序，能够发现用 Socket 描述网络程序比较直观，且框架清晰、明确。

参 考 文 献

[1] 毛德操，胡希明. Linux 内核源代码情景分析（下）[M]. 杭州：浙江大学出版社. 2001.

[2] 张威. Linux 网络编程教程[M]. 北京：北京希望电子出版社，2002.

[3] Douglas Comer，David L Stevens. TCP/IP 网络互联技术（卷 3）：客户- 服务器编程与应用[M]. 北京：清华大学出版社，2004.

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

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

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

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

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

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

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

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