IPv6 新型套接字的网络编程剖析

袁德明

(徐州建筑职业技术学院 电子信息工程系，江苏 徐州 221116)

【摘　要】在分析 RFC3493、RFC3542 和 RFC4038 等参考文档的基础上，结合有连接的 IPv6 套接字编程实验，文中介绍了 IPv6 新型套接字、地址结构、地址转换与解析处理，阐述了用 IPv6 新型套接字实现网络通信的编程方法，并给出一个在普通局域网环境下实现的有连接通信程序示例和运行结果。试验表明文中所提出的 IPv6 网络编程方法是正确的和可行的。

【关键词】IPv6；地址结构；套接字；网络通信

【中图分类号】TP393.09　　**【文献标识码】**A　　**【文章编号】**1002-0802(2007)12-0363-04

IPv6 New Socket Network Programming Analysis

YUAN De-ming

(Department of Electronic and Information Engineering, Xuzhou Institute of Architectural Technology, Xuzhou, Jiangsu 221116, China)

【Abstract】Based on RFC3493, RFC3542, RFC4038 and in combination of connection-oriented programming, with IPv6 new socket, this paper expounds network programming, method with IPv6 new socket, discusses its address structure, address translation & address analysis, and describes example and running result of connection-oriented communication program in LAN. The experiment results indicate that the network programming method with IPv6 is correct and practicable.

【Key words】IPv6；address structure；socket；networking communication

0 引言

IPv6 以其提供 128 位的地址空间、高安全性、实时性和支持 QOS 等突出特性，即将取代 IPv4，正逐步进入商用阶段。随着中国互联网和通信领域从 IPv4 向 IPv6 的快速过渡，基于 IPv6 的应用需求日渐迫切，应用 IPv6 实现网络通信将成为趋势。目前，大多数网络程序都是基于 IPv4 套接字标准的，如何编写适应 IPv6 发展趋势的网络程序是推广和普及 IPv6 应用所面临的一个重要问题。

2003 年正式推出的关于 IPv6 的新型套接字标准[1]完全取代了 1999 年起草的 IPv6 套接字标准[2]，同年还推出了 IPv6 的高级套接字标准[3]，文献[1]和文献[3]是 IPv6 新型套接字通信编程的基础。

1 IPv6 套接字

与 IPv4 套接字[4]相比，IPv6 新型套接字主要在地址结构和地址解析方面进行了扩展，文献[1]中对所有扩展内容进行了详细描述，文献[3]中进一步描述了 IPv6 数据报头结构、IPv6 套接层新增选项、IPv6 原始套接字等高级套接字的编程内容。

1.1 IPv6 的地址结构

IPv6 的地址结构是通过 struct in6_addr 和 struct sockaddr_in6 描述的，其定义形式分别如下：

```
struct in6_addr{ u_int8_t  s6_addr[16];  /* IPv6 address */  };

struct sockaddr_in6{ u_char  sin6_len; /*length of this struct*/ u_char  sin6_family; /*AF_INET6*/
    u_int16_t  sin6_port; /*transport layer port# */ u_int32_t sin6_flowinfo; /*IPv6 flow information */
    struct in6_addr  sin6_addr;  /*IPv6 address */ u_int32_t sin6_scope_id; /*set of interfaces for a
    scope */ };
```

在 sockaddr_in6 结构中，新增两个字段项 sin6_flowinfo 和 sin6_scope_id。sin6_flowinfo 是与 IPv6 的新增流标（fiow label）和流量类（traffic Class）字段类相对应的一个选项，在编程时，通常可置为零。sin6_scope_id 字段是一个 32bit 的整数，用来指明字段

地址族的宏名定义为：

#define PF_INET6 AF_INET6

为了便于开发与协议无关的网络应用程序，文献[1]中定义了新的独立于协议的地址结构 sockaddr_storage。该结构拥有足够大的地址空间来容纳 IPv4 或 IPv6 的地址信息，它还具有针对 64 位对齐问题的填充项，并可方便的转换成 sockaddr_in 和 sockaddr_in6 形式，其结构定义如下：

```
struct sockaddr_storage { short ss_family; /*Address family.*/ char _ss_pad1[_SS_PAD1SIZE];
    /* 6 byte pad, this is to make implementation specific pad up to alignment field that follows explicit
    in the data structure.*/ _int64 _ss_align; /*Field to force desired structure.*/
    char _ss_pad2[_SS_PAD2SIZE]; /* 112 byte pad to achieve desired size _SS_MAXSIZE value
    minus size of ss_family, _ss_pad1, and _ss_align fields is 112. */ };
```

1.2 IPv6 的地址转换与地址解析函数

1.2.1 IPV6 的地址转换

用文本形式表示的 IPv6 地址串与 IPv6 网络字节顺序地址间的转换可通过 inet_pton()和 inet_ntop()两个函数完成。前者能将 IPv6 地址串转换成网络字节顺序地址，后者与前者的功能相反。这两个函数既适用于 IPv4，也适用于 IPv6，但由于它们对协议具有依附性，所以在文献[5]中建议不使用这两个地址转换函数。

1.2.2 IPv6 的地址解析

在 IPv6 新型套接字中，getaddrinfo()和 getnameinfo()是两个独立于协议的重要解析函数，前者是通过主机名捕获地址信息，而后者是通过地址捕获主机名字信息。以下介绍的是它们在 Windows 环境下扩展的定义形式。

（1）由名字获取主机地址信息 getaddrinfo()

```
int WSAAPI getaddrinfo( IN const char FAR *nodename, IN const char FAR *servname,
    IN const struct addrinfo FAR * hints, OUT struct addrinfo FAR *FAR * res );
```

（2）由地址获取名字信息 getnameinfo()

```
int WSAAPI getnameinfo( IN const struct sockaddr FAR *sa, IN socklen_t salen, OUT char FAR *host,
    IN DWORD hostlen, OUT char FAR *serv, IN DWORD servlen, IN int flags );
```

使用 getaddrinfo()和 getnameinfo()可以捕获所有与主机名字和地址相关的网络编程信息。在这两个函数中的 IN 表示该项为输入参数，OUT 表示该项为输出参数（即保存捕获结果的变量）。其中有个关键的地址信息结构 addrinfo 的定义如下：

```
typedef struct addrinfo { int ai_flags; int ai_family; /*PF_xxx. */ int ai_socktype; /* SOCK_xxx.*/ int ai_protocol;
    /* 0 or IPPROTO_xxx for IPv4 and IPv6.*/ size_t ai_addrlen; /* Length of ai_addr.*/
    char *ai_canonname; /* Canonical name for nodename.*/
    struct sockaddr *ai_addr; /* Binary address.*/
    struct addrinfo *ai_next; /* Next structure in linked list.*/ } ADDRINFO, FAR * LPADDRINFO;
```

在 getaddrinfo()函数中有两个地址信息结构变量，一个是 hints，另一个是 res。hints 相当于一个过滤器，只有与 hints 指向的参数相符的内容，才能被输出到由 res 指向的链表中，可直接将这些返回的地址值传递给 bind()、connect()和 sendto()等函数，通过 res 返回的地址信息是动态分配的，使用完这些信息后，应该调用 freeaddrinfo()函数释放其所占的空间。当 hints 为空时，getaddrinfo()函数默认 ai_flag、ai_socktype 和 ai_protocol 为零，ai_familly 为 AF_UNSPEC（表示 IP 版本不定）。

2 IPv6 网络编程试验

2.1 试验环境

本试验的硬件由装有 XP 操作系统的两台电脑（一个是 COMPAQ M2010AP，另一个是 IBM ThinkPad R50e）和一台普通交换机(TaiLink TES-3008)组成，其互连结构和地址配置如图 1 所示。

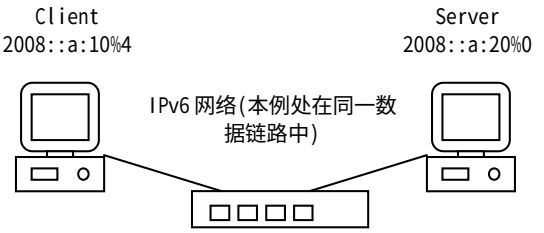

图 1 IPv6 编程的硬件环境配置

2.2 试验目标说明

建立面向连接的 IPv6 网络通信程序。基本要求：首先启动 IPv6 服务器，使其进入侦听等待接受客户端连接请求的状态；当客户端向服务器发出连接请求，并得到服务器允许建立链接时，向服务器发送指定信息；服务器收到信息后，向客户端发送响应信息。

在图 1 所示的环境下，采用 VS2005 开发 MFC 应用项目。通信应用程序界面及通信结果如图 2 和图 3 所示。

图 2 IPv6 服务器端应用界面

图 3 IPv6 客户端应用界面

2.3 试验程序

2.3.1 编译预处理程序

```
/* ipv6_test.h */ #include "winsock2.h"
#include "ws2tcpip.h"
#include "tpipv6.h" /*IPv6 编程接口，可从网上下载的 tpipv6-001205-sp3-ie6 压缩包中获得*/
#pragma comment(lib,"ws2_32.lib") #define BUFFER_SIZE 64
```

2.3.2 IPv6 服务器主要程序

```
#include "ipv6_test.h"
void Cserver6mDlg::OnStartServer6()
{ char Buffer[BUFFER_SIZE]; CString st; int RetVal, FromLen, AmountRead,bd;
int Family = AF_INET6; /*Internet 地址域*/ int SocketType = SOCK_STREAM; /*流式套接字类型*/
char *Port = "2008"; //服务端口号：2008
char *Address = "2008::a:20%0"; /*服务器地址：2008::a:20 ，地址接口识别号 0 */
char *ms="客户你好！我是 IPv6 服务器，这是发给你的应答信息。"; //服务器响应客户的信息
SOCKADDR_STORAGE From; WSADATA wsaData; ADDRINFO Hints, *AddrInfo; SOCKET ss,cs;
WSAStartup(0x0202,&wsaData); memset(&Hints, 0, sizeof(Hints));
Hints.ai_family = Family; Hints.ai_socktype = SocketType; Hints.ai_protocol=IPPROTO_TCP;
Hints.ai_flags = AI_NUMERICHOST|AI_PASSIVE;
/*用 getaddrinfo 函数解析本机地址和端口，并将结果存至 AddrInfo 结构中*/
RetVal = getaddrinfo(Address, Port, &Hints, &AddrInfo);
UpdateData(true); ipv6addr=Address; ServerPort=Port; FromLen=sizeof(From);
ss = socket( AddrInfo->ai_family, AddrInfo->ai_socktype, AddrInfo->ai_protocol ); //创建套接字 ss
bd = bind( ss, AddrInfo->ai_addr , (int)AddrInfo->ai_addrlen ); //将 ss 与本机地址和端口绑定
listen(ss, 10); //在套接字 ss 上建立侦听队列
cs = accept(ss, (LPSOCKADDR)&From, &FromLen); /*在 ss 上等待接收客户端的链接请求*/
/*在建立连接的 cs 上接收客户端发来的信息*/
//处理收到的的客户信息，并显示在相应文本框中
st=Buffer;clientinfo=st.Left(AmountRead); SendClientInfo=ms; UpdateData(false);
RetVal = send(cs, ms, 64, 0); /*向客户端发送响应信息*/
closesocket(cs); closesocket(ss); WSACleanup(); }
```

2.3.3 IPv6 客户端主要程序

```
#include "ipv6_test.h"
void Cclient6mDlg::OnConnect()
{ ClientButton.EnableWindow(0); int RetVal, AmountRead; char Buffer[BUFFER_SIZE];
CString st; int Family = AF_INET6; int SocketType = SOCK_STREAM;
char *Port = "2008"; //服务端口号：2008
char *Address ="2008::a:20%0"; /*指定服务器地址：2008::a:20 ，地址接口识别号 0 */
char *ms="你好！是 IPv6 服务器吗？我是 IPv6 客户。"; //客户端发给服务器的指定信息
WSADATA wsaData; ADDRINFO Hints, *AddrInfo; SOCKET ccs; WSAStartup(0x0202,&wsaData);
memset(&Hints, 0, sizeof(Hints)); Hints.ai_family = Family; Hints.ai_socktype = SocketType;
Hints.ai_protocol=IPPROTO_TCP; Hints.ai_flags = AI_NUMERICHOST|AI_PASSIVE;
RetVal = getaddrinfo(Address, Port, &Hints, &AddrInfo);
ccs = socket( AddrInfo->ai_family, AddrInfo -> ai_socktype, AddrInfo->ai_protocol );
connect(ccs, AddrInfo->ai_addr, (int) AddrInfo -> ai_addrlen ); //向指定服务器发送链接请求
RetVal = send(ccs,ms, 64, 0); /*服务器响应后，向服务器发送指定信息*/
UpdateData(true); SendToServerInfo=ms; AmountRead = recv(ccs, Buffer, sizeof(Buffer),0);
st=Buffer;serverinfo=st.Left(AmountRead); UpdateData(false);
closesocket(ccs); WSACleanup(); ClientButton.EnableWindow(1); }
```

2.4 试验程序中关键问题剖析

上述通信试验程序的逻辑结构与文献[4]中介绍的网络通信编程结构基本是一致的，所不同的就是在通信协议及地址处理方面。从试验程序中可以看出：IPv6 套接字的创建、绑定、请求连接等操作均是采用独立于协议的地址解析函数捕获的结果实现的。在服务器端还采用了独立于协议的地址结构 SOCKADDR_STORAGE。

程序中利用 getaddrinfo()函数对 IPv6 地址处理的主要

(1)定义 IPv6 的地址串和端口串参数 Addresst 和 Port

char *Port = "2008"; char *Address = "2008::a:20%0";

(2) 定义 IPv6 地址信息结构参数 AddrInfo 和 Hints

ADDRINFO Hints, *AddrInfo; memset(&Hints, 0, sizeof(Hints)); /*将初始结构参数 Hints 置 0 */

(3) 设置结构参数 Hints 的地址域、套接字类型、使用的协议和标志值

Hints.ai_family = Family; Hints.ai_socktype=SocketType;

Hints.ai_protocol=IPPROTO_TCP; Hints.ai_flags = AI_NUMERICHOST|AI_PASSIVE;

(4) 对 Address、Port 和 Hints 解析获取 IPv6 网络地址，并存入 AddrInfo 结构中

RetVal = getaddrinfo(Address, Port, &Hints, &AddrInfo);

3 结语

在网络编程结构上，IPv6 与 IPv4 没有本质区别，文献[4]IPv6 与 IPv4 的地址结构与地址解析处理上存在的区别，另外在原始套接字和套接字选项设置方面 IPv6 与 IPv4 也存在较多差异。文中探讨的 IPv6 编程方法，着重强调使用独立于协议的接口函数来实现 IPv6 套接字编程，其目的在于实现独立于协议的网络应用程序，使编写的 IPv6 网络程序支持双栈共享和能将 IPv4 网络程序顺利迁移到 IPv6 上。

参考文献

1 Thomson S, Bound J, McCann J, et al. Basic Socket Interface Extensions for IPv6, February 2003. RFC3493[S].

2 Gilligan R, Thomson S, Bound J, et al. Basic Socket Interface Extensions for IPv6, March 1999. RFC2553[S].

3 Thomas M, Sun T. Jinmei. Advanced Sockets Application Program Interface (API) for IPv6, May 2003. RFC3542[S].

4 Yuan De ming. Windows Network Programming [M]. Beijing, Science Press, 2004.

5 Savola P, Castro E M. Application Aspects of IPv6 Transition, March 2005. RFC4038[S].

（上接第 362 页）

(1) 新路由区域的界定。节点（可以是 sinks 损坏的传感器节点的上游节点等）首先标注自己为 in-region ，然后广播一个初始 TTL（Time To Live 生存时间）等于 1 或 2 的寻找路由区域数据包 RDR（route discovery region），TTL 的值取决于它能否沿着出问题的路由到达下一跳，若能，则取 2。当旧路由上的一个中继节点收到 RDR 包时，它把 RDR 包的 TTL 值恢复为 1，再把该 RDR 包广播出去；当不在原路由上的一个中继节点收到 RDR 包时，仅仅当 RDR 包的 TTL>1 时，才把它广播出去，而且在它广播出去之前，要把 RDR 包的 TTL 值减 1，所有收到 RDR 包的节点都标注自己为 in_region，最终，RDR 包就沿着旧路由传到了路由发现的目的地。这样，由旧路由上的节点连同它们的邻居节点就形成了新的路由区域。

(2) 发现新路由。新路由区域被界定后，启动路由更新协议的发现者向目的节点广播一个路由请求（router request）包 RRQ 去发现新路由，仅仅被标注为 in_region 的节点且剩余能量超过规定显示才能继续广播该 RRQ 包。那些接收到 RRQ 包但不能继续广播的节点保留一个临时路由请求纪录，以便在可能的情况下扩展路由区域，如果旧路由周围的传感器节点比较密，且大多数节点有足够的能量作为中继设备，则 RRQ 包就会从多个新路由被传播到目的节点，目的节点在这些新路由中选择出最优的一条，沿着该路由发回一个路由请求回答包 RRP（route request reply），至此，就完成了路由更新的过程。

2 结语

文中在介绍当前研究热门 WSN 的基本信息之后着重对 WSN 的关键技术——路由协议问题进行了分析。不同于其他关于 WSN 路由协议概述性文章的是，作者提出可以将 WSN 的整个路由过程分为路由建立和路由更新两步，并通过对目前应用较广的几个路由协议来进一步阐述这两个子过程的工作机制。在查阅有关 WSN 路由协议的大量资料文献中，作者发现目前提出的许多路由协议都只是在路由建立这一层次进行探讨，而并未深入涉及路由更新问题，在这一方面还需要进一步的工作以完善适用于 WSN 的路由协议。

参考文献

1 Intanagonwiwat C, Govindan R, Estrin D, et al. Directed diffusion for wireless sensor networking. IEEE/ACM Trans. on Networking, 2003, 11(01):2-16.

2 Braginsky D, Estrin D. Rumor routing algorithm for wireless networks[M]. In: Proc. of the 1st workshop on sensor networks and applications. Atlanta: ACM Press,2002.22-31.

3 Shah R, Rabaey J. Energy aware routing for low energy ad hoc sensor networks[C]. In: Proc. of the IEEE Wireless Communica- tions and Networking Conf. Orlando: IEEE Communications Society, 2002. 350-355.

4 Karp B, Kung H. GPSR: Greedy perimeter stateless routing for wireless networks[C]. In: Proc. of the 6th Annual Int ' l Conf. on Mobile Computing and Networking. Boston: ACM Press, 2000. 243-254.

5 Heinzelman W R, Chandrakasan A, Balakrishnan H. An application-specific protocol architecture for wireless microsensor networks[J]. IEEE Transactions on Wireless Communications, 2002,1 (04):660-670.

6 Karp B, Kung H T. GPSR: Greedy perimeter stateless routing for wireless networks[C]. In: Proc 6th Annual Int ' l Conf on Mobile Computing and Networking, Boston, MA.Aug, 2000:6-11.

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
56. 基于 IPv6 的 UDP 通信的实现
57. 一种基于 IPv6 的流媒体传送方案研究与实现
58. 基于 IPv4-IPv6 双栈的 MODBUS-TCP 协议实现
59. RS485CAN 网关设计与实现
60. MVB 周期信息的实时调度
61. RS485 和 PROFINET 网关设计
62. 基于 IPv6 的 Socket 通信的实现
63. MVB 网络重复器的设计
64. 一种新型 MVB 通信板的探究
65. 具有 MVB 接口的输入输出设备的分析
66. 基于 STM32 的 GSM 模块综合应用
67. 基于 ARM7 的 MVB CAN 网关设计
68. 机车车辆的 MVB CAN 总线网关设计
69. 智能变电站冗余网络中 IEEE1588 协议的应用
70. CAN 总线的浅析 CANopen 协议
71. 基于 CANopen 协议实现多电机系统实时控制
72. 以太网时钟同步协议的研究
73. 基于 CANopen 的列车通信网络实现研究
74. 基于 SJA1000 的 CAN 总线智能控制系统设计
75. 基于 CANopen 的运动控制单元的设计
76. 基于 STM32F107VC 的 IEEE 1588 精密时钟同步分析与实现

77. 分布式控制系统精确时钟同步技术
78. 基于 IEEE 1588 的时钟同步技术在分布式系统中应用
79. 基于 SJA1000 的 CAN 总线通讯模块的实现
80. 嵌入式设备的精确时钟同步技术的研究与实现
81. 基于 SJA1000 的 CAN 网桥设计
82. 基于 CAN 总线分布式温室监控系统的设计与实现
83. 基于 DSP 的 CANopen 通讯协议的实现
84. 基于 PCI9656 控制芯片的高速网卡 DMA 设计
85. 基于以太网及串口的数据采集模块设计
86. MVB1 类设备控制器的 FPGA 设计
87. MVB 接口彩色液晶显示诊断单元的显示应用软件设计

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
35. 基于 GPU FPGA 芯片原型的 VxWorks 下驱动软件开发
36. VxWorks 下的多串口卡设计
37. VxWorks 内存管理机制的研究
38. T9 输入法在 Tilcon 下的实现
39. 基于 VxWorks 的 WindML 图形界面开发方法
40. 基于 Tilcon 的 IO 控制板可视化测试软件的设计和实现
41. 基于 VxWorks 的通信服务器实时多任务软件设计
42. 基于 VXWORKS 的 RS485MVB 网关的设计与实现
43. 实时操作系统 VxWorks 在微机保护中的应用
44. 基于 VxWorks 的多任务程序设计及通信管理
45. 基于 Tilcon 的 VxWorks 图形界面开发技术
46. 嵌入式图形系统 Tilcon 及应用研究
47. 基于 VxWorks 的数据采集与重演软件的图形界面的设计与实现
48. 基于嵌入式的 Tilcon 用户图形界面设计与开发
49. 基于 Tilcon 的交互式多页面的设计
50. 基于 Tilcon 的嵌入式系统人机界面开发技术
51. 基于 Tilcon 的指控系统多任务人机交互软件设计
52. 基于 Tilcon 航海标绘台界面设计
53. 基于 Tornado 和 Tilcon 的嵌入式 GIS 图形编辑软件的开发
54. VxWorks 环境下内存文件系统的应用
55. VxWorks 下的多重定时器设计
56. Freescale 的 MPC8641D 的 VxWorks BSP
57. VxWorks 实验五[时间片轮转调度]
58. 解决 VmWare 下下载大型工程.out 出现 WTX Error 0x100de 的问题
59. 基于 VxWorks 系统的 MiniGUI 图形界面开发
60. VxWorks BSP 开发中的 PCI 配置方法
61. VxWorks 在 S3C2410 上的 BSP 设计
62. VxWorks 操作系统中 PCI 总线驱动程序的设计与实现
63. VxWorks 概述
64. 基于 AT91RM9200 的 VxWorks END 网络驱动开发
65. 基于 EBD9200 的 VxWorks BSP 设计和实现
66. 基于 VxWorks 的 BSP 技术分析

67. ARM LPC2210 的 VxWorks BSP 源码
68. 基于 LPC2210 的 VxWorks BSP 移植
69. 基于 VxWorks 平台的 SCTP 协议软件设计实现

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
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
42. Linux 下以太网的 IPv6 隧道技术的实现
43. Research and design of mobile learning platform based on Android
44. 基于 linux 和 Qt 的串口通信调试器调的设计及应用
45. 在 Linux 平台上基于 QT 的动态图像采集系统的设计
46. 基于 Android 平台的医护查房系统的研究与设计
47. 基于 Android 平台的软件自动化监控工具的设计开发
48. 基于 Android 的视频软硬解码及渲染的对比研究与实现
49. 基于 Android 移动设备的加速度传感器技术研究
50. 基于 Android 系统振动测试仪研究
51. 基于缓存竞争优化的 Linux 进程调度策略
52. Linux 基于 W83697 和 W83977 的 UART 串口驱动开发文档
53. 基于 AT91RM9200 的嵌入式 Linux 系统的移植与实现
54. 路由信息协议在 Linux 平台上的实现

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
21. DCOM 协议在网络冗余环境下的应用
22. Windows XP Embedded 在变电站通信管理机中的应用
23. XPE 在多功能显控台上的开发与应用
24. 基于 Windows XP Embedded 的 LKJ2000 仿真系统设计与实现
25. 虚拟仪器的 Windows XP Embedded 操作系统开发
26. 基于 EVC 的嵌入式导航电子地图设计
27. 基于 XPEmbedded 的警务区 SMS 指挥平台的设计与实现
28. 基于 XPE 的数字残币兑换工具开发
29. Windows CENET 下 ADC 驱动开发设计
30. Windows CE 下 USB 设备流驱动开发与设计
31. Windows 驱动程序设计
32. 基于 Windows CE 的 GPS 应用
33. 基于 Windows CE 下大像素图像分块显示算法的研究
34. 基于 Windows CE 的数控软件开发与实现

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
22. 基于 PowerPC 的嵌入式系统中通用 IO 口的扩展方法
23. 基于 PowerPC440GP 型微控制器的嵌入式系统设计与研究
24. 基于双 PowerPC 7447A 处理器的嵌入式系统硬件设计
25. 基于 PowerPC603e 通用处理模块的设计与实现
26. 嵌入式微机 MPC555 驻留片内监控器的开发与实现
27. 基于 PowerPC 和 DSP 的电能质量在线监测装置的研制
28. 基于 PowerPC 架构多核处理器嵌入式系统硬件设计
29. 基于 PowerPC 的多屏系统设计
30. 基于 PowerPC 的嵌入式 SMP 系统设计
31. 基于 MPC850 的多功能通信管理器

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
30. 基于 ARM 嵌入式系统的高保真无损音乐播放器设计
31. UBoot 在 Mini6410 上的移植
32. 基于 ARM11 的嵌入式 Linux NAND FLASH 模拟 U 盘挂载分析与实现
33. 基于 ARM11 的电源完整性分析
34. 基于 ARM S3C6410 的 uboot 分析与移植
35. 基于 S5PC100 移动视频监控终端的设计与实现
36. UBoot 在 AT91RM9200 上的移植简析
37. 基于工控级 AT91RM9200 开发板的 UBoot 移植分析

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
26. CPU 与 GPU 之间接口电路的设计与实现
27. 基于龙芯 1A 平台的 PMON 源码编译和启动分析
28. 基于 PC104 工控机的嵌入式直流监控装置的设计
29. GPGPU 技术研究与发展
30. GPU 实现的高速 FIR 数字滤波算法
31. 一种基于 CPUGPU 异构计算的混合编程模型
32. 面向 OpenCL 模型的 GPU 性能优化
33. 基于 GPU 的 FDTD 算法
34. 基于 GPU 的瑕疵检测
35. 基于 GPU 通用计算的分析与研究
36. 面向 OpenCL 架构的 GPGPU 量化性能模型
37. 基于 OpenCL 的图像积分图算法优化研究
38. 基于 OpenCL 的均值平移算法在多个众核平台的性能优化研究
39. 基于 OpenCL 的异构系统并行编程
40. 嵌入式系统中热备份双机切换技术研究

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
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
7. 数据结构考题 - 第 1 章 绪论
8. 数据结构考题 - 第 2 章 线性表
9. 数据结构考题 - 第 2 章 线性表 - 答案
10. 基于小波变换与偏微分方程的图像分解及边缘检测
11. 基于图像能量的布匹瑕疵检测方法
12. 基于 OpenCL 的拉普拉斯图像增强算法优化研究
13. 异构平台上基于 OpenCL 的 FFT 实现与优化
14. 数据结构考题 - 第 4 章 串

15. 数据结构考题 - 第 4 章 串答案
16.

FPGA / CPLD:

1. 一种基于并行处理器的快速车道线检测系统及 FPGA 实现
2. 基于 FPGA 和 DSP 的 DBF 实现
3. 高速浮点运算单元的 FPGA 实现
4. DLMS 算法的脉动阵结构设计及 FPGA 实现
5. 一种基于 FPGA 的 3DES 加密算法实现
6. 可编程 FIR 滤波器的 FPGA 实现
7. 基于 FPGA 的 AES 加密算法的高速实现
8. 基于 FPGA 的精确时钟同步方法
9. 应用分布式算法在 FPGA 平台实现 FIR 低通滤波器
10. 流水线技术在用 FPGA 实现高速 DSP 运算中的应用
11. 基于 FPGA 的 CAN 总线通信节点设计
12. 基于 FPGA 的高速时钟数据恢复电路的实现
13. 基于 FPGA 的高阶高速 FIR 滤波器设计与实现
14. 基于 FPGA 高效实现 FIR 滤波器的研究
15.