基于规则的IPv4源程序到IPv6源程序的移植方法

汪　浩，严　伟，任茂盛

（北京大学计算机科学技术系网络实验室，北京 100871）

摘　要：在分析IPv4 Sockets API编程接口与IPv6 Sockets API编程接口异同的基础上，提出了一组替换规则，在不必阅读理解IPv4源程序的前提下，能将支持IPv4的源程序移植成支持IPv6的源程序。利用该移植方法，已成功地将Linux下的流媒体库LIVE和流媒体播放器Mplayer从支持IPv4移植成支持IPv6。

关键词：IPv4；IPv6；程序移植

A Rule-based Method for Transformation of an IPv4 Program to an IPv6 Program

WANG Hao，YAN Wei，REN Maosheng

(Network Lab,Department of Computer Science and Technology,Peking University,Beijing 100871)

【Abstract】 This paper discusses the differences and the same aspects between IPv4 Sockets API and IPv6 Sockets API. After that, it gives a set of substitution rules which can be used to transform an IPv4 program to an IPv6 program. According to these rules, the authors have successfully ported a streaming media library called LIVE and a streaming media player called Mplayer from IPv4 to IPv6.

【Key words】 IPv4；IPv6；Program transformation

IPv6作为新一代网络互联协议,具有易扩展性、易用性、安全性、实时性及提供对QoS支持等特点。本文讨论如何把在IPv4协议栈上运行的用IPv4 Sockets API编写的软件方便地移植成能在IPv6协议栈上运行的用IPv6 Sockets API编写的软件。在分析IPv4 Sockets API编程接口与IPv6 Sockets API[3,4]编程接口异同的基础上，提出了一组替换规则，在不必阅读理解IPv4源程序的前提下，能将支持IPv4的源程序移植成支持IPv6的源程序。

1 IPv4 Sockets与IPv6 Sockets编程接口的差异

IPv4与IPv6 Sockets编程接口的差异主要体现在地址结构、地址转换函数、域名解析函数、加入和退出多播组、特殊地址等方面。

1.1 地址结构的差异

IPv4的地址结构通过struct in_addr和struct sockaddr_in定义，IPv6的地址结构通过struct in6_addr和struct sockaddr_in6定义。IPv4的Loopback地址定义为INADDR_LOOPBACK(主机字节序)，IPv4的wildcard地址定义为INADDR_ANY(主机字节序)。IPv6的Loopback地址定义为in6addr_loopback(主机字节序)，IPv6的wildcard地址定义为in6addr_any(主机字节序)[4]。

1.2 地址转换函数的差异

IPv4通过下列函数完成文本表示的十进制点分地址和32 bits的网络字节序地址之间的相互转换[4]:

```
int inet_aton(const char *strptr, struct in_addr *addrptr);
in_addr_t inet_addr(const char *strptr);
char *inet_ntoa(struct in_addr inaddr);
```

IPv6通过下列函数完成文本表示的IPv6地址和128 bits的网络字节序地址之间的相互转换：

```
int inet_pton(int family,const char *strptr,void *addrptr)
const char *inet_ntop(int family,const void *addrptr,char *strptr, size_t len)
```

在作移植时,inet_aton() 或 inet_addr()函数需要用inet_pton()函数来替换;inet_ntoa()函数需要用inet_ntop()函数来替换。

1.3 域名解析函数的差异

IPv4通过下列函数完成主机名或域名到IPv4地址的解析:

```
struct hostent *gethostbyname(const char *hostname)
```

其中入口参数hostname为主机名或域名，函数返回的结果存放在struct hostent[4]中。

而通过调用下列函数，可以完成主机名或域名到IPv6地址的解析：

```
struct hostent *gethostbyname2(const char *hostname,int family);
```

其中入口参数family指定为AF_INET6，入口参数hostname为主机名或域名, 函数通过struct hostent返回主机的IPv6地址。

1.4 加入和退出多播组的差异

表1　设置IPv4和IPv6多播参数

level	optname	optval的数据类型	说　明
IPPROTO_IP	IP_MULTICAST_IF	struct in_addr	optval参数指定发送多播包的网卡
	IP_MULTICAST_TTL	u_char	optval参数指定多播包能通过的路由器个数
	IP_MULTICAST_LOOP	u_char	optval参数指定多播包是否回送本机
	IP_ADD_MEMBERSHIP	struct ip_mreq	optval参数指定加入的多播组
	IP_DROP_MEMBERSHIP	struct ip_mreq	optval参数指定退出的多播组
IPPROTO_IPV6	IPV6_MULTICAST_IF	struct in6_addr	optval参数指定发送多播包的网卡
	IPV6_MULTICAST_HOPS	u_int	optval参数指定多播包能通过的路由器的个数
	IPV6_MULTICAST_LOOP	u_int	optval参数指定多播包是否回送本机
	IPV6_ADD_MEMBERSHIP	struct ipv6_mreq	optval参数指定加入的多播组
	IPV6_DROP_MEMBERSHIP	struct ipv6_mreq	optval参数指定退出的多播组

不管是IPv4还是IPv6多播组的加入和退出，都是通过下列函数完成的：

```
int setsockopt(int sockfd, int level,int optname, const void *optval, socklen_t optlen);
```

其中sockfd为Socket号，optlen为optval对应数据的长度，其余参数的设置方法见表1，有关结构的定义见文献[4]。

1.5 特殊地址的差异

IPv4的Loopback地址为127.0.0.1;IPv6的Loopback地址为::1。IPv4的多播地址范围为224.0.0.0 ~ 239.255.255.255；IPv6多播地址的定义见图1。其中，4-bit 标记的定义是：0表示知名；1表示临时。4-bit 范围的定义是：1表示本地节点；2表示本地链路；3表示本地站点；8表示本地机构；14表示全局。

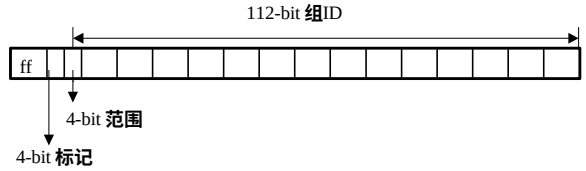

图1 IPv6的多播地址的结构

IPv4通过C语言的位运算判断IPv4地址的类型(Loopback地址，多播地址)；而IPv6的API提供了下述宏判断IPv6地址的类型（Loopback地址，多播地址）：

```
int  IN6_IS_ADDR_LOOPBACK(const struct in6_addr ipv6addr);
int  IN6_IS_ADDR_MULTICAST(const struct in6_addr ipv6addr);
```

2 IPv4 Sockets与IPv6 Sockets编程接口的相同之处

除了上节所述差异外，IPv4 Sockets和IPv6 Sockets编程接口的API函数原型都是相同的。例如:socket()、connect()、bind()、listen()、accept()、select()、recvfrom()、sendto()。

上述函数原型中，通过指针形式的struct sockaddr[4]传递IPv4和IPv6的地址及其长度。从而达到IPv4 Sockets API和IPv6 Sockets API的“兼容性”，在IPv4的程序移植到IPv6程序的过程中，就没有必要修改Sockets API调用的函数形式，而仅需修改传递给其入口参数的变量的数据类型。

例如，在IPv4中，调用connect ( ) 函数的过程如下(忽略出错处理等无关语句，下划线部分表示与IPv6的不同处)：

```
struct sockaddr_in servaddr；
int sockfd;
sockfd=socket(AF_INET , SOCK_STREAM,0);
servaddr.sin_family  =AF_INET;
servaddr.sin_port=htons(1289);
inet_aton(“ 18.18.18.18 ”,&servaddr.sin_addr);
connect(sockfd, (struct sockaddr*)&servaddr, sizeof(servaddr));
```

在IPv6中，调用connect（）函数的过程如下(忽略出错处理等无关语句，下划线部分表示与IPv4的不同处)：

```
struct sockaddr_in6 servaddr；
int sockfd;
sockfd=socket(AF_INET6 , SOCK_STREAM,0);
servaddr.sin6_family  =AF_INET6;
servaddr.sin6_port=htons(1289);
inet_pton(AF_INET6, “ fec0::5 ” , &servaddr.sin6_addr);
connect(sockfd, (struct sockaddr*)&servaddr, sizeof(servaddr));
```

IPv4和IPv6的connect（）调用形式都是一样的，差别仅在于其入口参数的定义方式不同。因此在移植过程中，涉及本小节所述的函数调用都不必修改。

另外，IPv4和IPv6的SOL_SOCKET选项和IPPROTO_TCP选项也是相同的，移植时不必修改。

3 源程序的移植规则

为了将IPv4的源程序移植成IPv6的源程序，需要做如下工作：

(1) 将IPv4的地址结构移植成IPv6的地址结构；
(2) 将IPv4的地址转换函数移植成IPv6的地址转换函数；
(3) 将IPv4的域名解析函数移植成IPv6的域名解析函数；
(4) 将IPv4的加入和退出多播组移植成 IPv6 的加入和退出多播组；
(5) 将IPv4的特殊地址移植成IPv6的特殊地址。

为了移植的方便性，并尽量少修改源程序，在移植过程中还要遵循如下原则：仅修改变量对应的数据类型，而不修改变量名和其存储类型。例如：如下IPv4的数据声明：

```
struct in_addr *addr1 , addr2[10] , addr3;
```

在移植成IPv6时将变成：

```
struct in6_addr *addr1 , addr2[10] , addr3;
```

从中可以看出，不但变量名不变，变量的存储类型也不变，即addr1仍为指针，addr2仍为数组，addr3仍为一般的变量。

在详述移植工作的具体细节前，首先要对“替换规则”作形式化的定义，然后采用替换规则来严格描述IPv4到IPv6的移植方法。

定义1 由变量、常数和字符串组成的集合称为字母表。

约定用希腊字母η，ξ…表示字母表中的元素。

定义2 由字母表中若干元素经有限次联结运算后的结果称为字。字中所含元素的个数称为字的长度。

约定用大写字母A，B，C…表示字。

定义3 称字A匹配于字B，如果满足如下条件：

(1) 字A和字B长度相等。不妨设$A=\xi_1\xi_2\cdots\xi_n$，$B=\eta_1\eta_2\cdots\eta_n$

(2) 对任意的i(i＝1，2，…,n), 如果η_i是字符串，则ξ_i必须是同η_i相等的字符串；如果η_i是变量或常数，则ξ_i必须也是变量，且同时将η_i当作字符串赋值给ξ_i。

(3) 匹配时忽略字A和字B元素间的空白字符。

定义4 假设A为字，不妨设$A=\xi_1\xi_2\cdots\xi_n$，字A的值定义如下：

(1) 对任意的i(i＝1，2，…,n), 如果ξ_i是字符串，则ξ_i的值是字符串本身。

(2) 对任意的i(i＝1，2，…,n), 如果ξ_i是变量，则ξ_i的值是变量的值。

(3) 字A的值由$\xi_1, \xi_2, \cdots, \xi_n$经联结运算联结而成。

定义5 假设A, B为字, A→B, 称为替换规则, 如果满足:

(1) 字A必须匹配于源程序中的某字(不妨用字C表示)。
(2) 字B中出现的变量必须在字A中出现。
(3) 用字B的值覆盖源程序中字C的值。

3.1 IPv4的地址结构移植成IPv6的地址结构

为了将IPv4的地址结构移植成IPv6的地址结构，需要运用如下替换规则：

规则1“ AF_INET ”→“ AF_INET6 ”
规则2“ INADDR_LOOPBACK ”→“ in6addr_loopback ”
规则3“ INADDR_ANY ”→“ in6addr_any ”
规则4“ in_addr ”→“ in6_addr ”
规则5“ s_addr ”→“ s6_addr ”
规则6“ sockaddr_in ”→“ sockaddr_in6 ”
规则7“ sin_len ”→“ sin6_len ”
规则8“ sin_family ”→“ sin6_family ”
规则9“ sin_port ”→“ sin6_port ”

规则10“ sin_addr ”→“ sin6_addr ”

3.2 IPv4的地址转换函数移植成IPv6的地址转换函数

由于存放文本表示的IPv4十进制点分地址所需空间为16byte，而存放文本表示的IPv6地址，所需空间为INET6_ADDRSTRLEN字节，因此在移植地址转换函数时，要用如下两规则：

规则11 strptr“ [“ arraysize ”] ”→ strptr“ [“ “ INET6_ADDRSTRLEN ” ”] ”

规则12 strptr“ = ”“ malloc ”“ (“ size ”) ”→ strptr“ = ”“ malloc ”“ (“ “ INET6_ADDRSTRLEN ” ”) ”

规则11的作用是为了将存储文本表示的IP地址的数组空间扩大至INET6_ADDRSTRLEN。规则12的作用是为了将存储文本表示的IP地址的由malloc分配的空间扩大至INET6_ADDRSTRLEN。

(1) inet_aton（）函数的移植方法

为了将inet_aton（）移植成inet_pton()函数，需要如下替换规则：

规则13 inet_aton (strptr , addrptr→ inet_pton(AF_INET6, strptr , addrptr)

(2) inet_addr（）函数的移植方法

为了将inet_aton（）移植成inet_pton()函数，需要如下替换规则：

规则14“ in_addr_t ”→“ struct in6_addr ”

规则15 ipaddr“ = ”“ inet_addr ”“ (“ strptr ”) ”→ “ inet_pton(AF_INET6, “ strptr ” , & “ ipaddr ”) ”

规则16 “ * ” addrptr “ = ”“ inet_addr ”“ (“ strptr ”) ”→ “ inet_pton(AF_INET6, “ strptr ” , “ addrptr ”) ”

其中规则15用于ipaddr为非指针的情况，规则16用于addrptr为指针的情况。

(3) inet_ntoa（）函数的移植方法

为了将inet_ntoa（）移植成inet_ntop()函数，需要如下替换规则：

规则16 strptr“ = ”“ inet_ntoa ”“ (“ ipaddr ”) ”→ “ inet_pton(AF_INET6,& ” ipaddr “ , ” strptr , “ INET6ADDRSTRLEN) ”

规则17 strptr“ = ”“ inet_ntoa ”“ (“ “ * ” addrptr ”) ”→ “ inet_pton(AF_INET6, “ addrptr ” , “ strptr ” , INET6ADDRSTRLEN) ”

其中规则16用于ipaddr为非指针的情况，规则17用于addrptr为指针的情况。

3.3 IPv4的域名解析函数移植成IPv6的域名解析函数

为了将gethostbyname()移植成gethostbyname2()函数，需要如下替换规则：

规则18“ gethostbyname ”“ (“ hostname ”) ”→ “ gethostbyname2(“ hostname ” ,AF_INET6) ”

3.4 IPv4的加入和退出多播组移植成IPv6的加入和退出多 播组

为了将IPv4的加入和退出多播组移植成IPv6的加入和退出多播组，需要如下替换规则：

规则18“ IPPROTO_IP ”→“ IPPROTO_IPV6 ”

规则19“ IP_MULTICAST_LOOP ”→“ IPV6_MULTICAST_LOOP ”

规则20“ IP_MULTICAST_TTL ”→“ IPV6_MULTICAST_HOPS ”

规则21“ IP_MULTICAST_IF ”→“ IPV6_MULTICAST_IF ”

规则22“ IP_ADD_MEMBERSHIP ”→“ IPV6_ADD_MEMBERSHIP ”

规则23“ IP_DROP_MEMBERSHIP ”→“ IPV6_DROP_MEMBERSHIP ”

规则24“ ip_mreq ”→“ ipv6_mreq ”

规则25“ imr_multiaddr ”→“ ipv6mr_multiaddr ”

规则26“ imr_interface ”“ = ”“ INADDR_ANY ”→“ ipv6mr_interface = 0 ”

规则27“ imr_interface ”“ = ” ipaddr→ “ ipv6mr_interface = if_ipaddrtoindex （“ ipaddr ” ） ”

规则26、规则27分别对应IP地址分别是INADDR_ANY和某特定IP地址的情形，对后一情形，要用if_ipaddrtoindex（）将IP地址转换为网卡的索引号。

3.5 IPv4的特殊地址移植成IPv6的特殊地址

如果程序中出现了IPv4的Loopback地址“ 127.0.0.1 ”，则将其替换成“ ：：1 ”。如果出现了IPv4的多播地址，如“ 232. 255. 42. 42 ”，则将其替换成 IPv6 的多播地址，如“ ff03：：1 ”等。如果程序中出现了通过位运算判断某IPv4地址是否为loopback地址或多播地址，则可替换成用宏IN6_IS_ADDR_LOOPBACK() 或 IN6_IS_ADDR_MULTICAST()判断某IPv6地址是否是loopback地址或多播地址。规则如下：

规则28“ 127.0.0.1 ”→“ ::1 ”

规则29 IPv4的多播地址→IPv6的多播地址

规则30 用位运算判断某IP地址addr是否为loopback地址→ IN6_IS_ADDR_LOOPBACK(addr)

规则31用位运算判断某IP地址addr是否为多播地址→ IN6_IS_ADDR_MULTICAST(addr)

由于IPv4的多播地址同IPv6的多播地址没有一一对应关系，因此规则29无法写得太具体；由于“ 用位运算判断某IP地址addr是否为loopback地址 ”和“ 用位运算判断某IP地址addr是否为多播地址 ”的情况也较复杂，因此规则30、规则31左端也无法写得太具体，但右端却是很明确的。规则29～规则31不满足替换规则的严格定义，仅是一种“ 伪 ” 替换规则。

4 结论

对于建立在TCP、UDP协议上的IPv4程序，利用本文给出的规则，一定能成功地将其移植成IPv6的程序，但对于利用Raw Sockets编写的程序，除了使用本文给出的方法进行移植外，还有其他更进一步的工作要做，因篇幅所限，拟另文讨论。

参考文献

1 马 严,赵晓宇.IPv4向IPv6过渡技术综述.北京邮电大学学报，2002,25(4):1-5

2 伍海桑,李 星. 基于代理的针对WWW服务的IPv4/IPv6过渡机制. 计算机工程与应用, 2001,37(18)

3 Gilligan R, Thomson S, Bound J. Basic Socket Interface Extensions for IPv6. RFC 3493, 2003-03

4 Stevens R. Unix Network Programming Networking APIs:Sockets and XTI. Prentice Hall,Inc.,1998

5 莫绍揆. 算法论. 北京: 科学出版社,1982

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
88. IPv6 新型套接字的网络编程剖析

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.