**Linux 网络编程之ioctl函数 - minus01**

1.介绍

Linux网络程序与内核交互的方法是通过ioctl来实现的，ioctl与网络协议栈进行交互，可得到网络接口的信息，网卡设备的映射属性和配置网络接口.并且还能够查看，修改，删除ARP高速缓存的信息，所以，我们有必要了解一下ioctl函数的具体实现.

2.相关结构体与相关函数

#include <sys/ioctl.h>

int ioctl(int d,int request,....);

参数:

fd-文件描述符，这里是对网络套接字操作，显然是套接字描述符

request-请求码

省略的部分对应不同的内存缓冲区，而具体的内存缓冲区是由请求码request来决定的,下面看一下具体都有哪些相关缓冲区。

(1)网络接口请求结构ifreq

#define IFHWADDRLEN 6 //6个字节的硬件地址，即MAC

struct ifreq {

 union {

  char ifrn\_name[IFNAMESIZ]; //网络接口名称

 } ifr\_ifrn;

 union {

  struct sockaddr ifru\_addr; //本地IP地址

  struct sockaddr ifru\_dstaddr; //目标IP地址

  struct sockaddr ifru\_broadaddr; //广播IP地址

  struct sockaddr ifru\_netmask; //本地子网掩码地址

  struct sockaddr ifru\_hwaddr; //本地MAC地址

  short ifru\_flags; //网络接口标记

  int ifru\_ivalue; //不同的请求含义不同

  struct ifmap ifru\_map; //网卡地址映射

  int ifru\_mtu; //最大传输单元

  char ifru\_slave[IFNAMSIZ]; //占位符

  char ifru\_newname[IFNAMSIZE]; //新名称

  void \_\_user\* ifru\_data; //用户数据

  struct if\_settings ifru\_settings; //设备协议设置

 } ifr\_ifru;

};

#define ifr\_name ifr\_ifrn.ifrn\_name;//接口名称

#define ifr\_hwaddr ifr\_ifru.ifru\_hwaddr;//MAC

#define ifr\_addr ifr\_ifru.ifru\_addr;//本地IP

#define ifr\_dstaddr ifr\_ifru.dstaddr;//目标IP

#define ifr\_broadaddr ifr\_ifru.broadaddr;//广播IP

#define ifr\_netmask ifr\_ifru.ifru\_netmask;//子网掩码

#define ifr\_flags ifr\_ifru.ifru\_flags;//标志

#define ifr\_metric ifr\_ifru.ifru\_ivalue;//接口侧度

#define ifr\_mtu ifr\_ifru.ifru\_mtu;//最大传输单元

#define ifr\_map ifr\_ifru.ifru\_map;//设备地址映射

#define ifr\_slave ifr\_ifru.ifru\_slave;//副设备

#define ifr\_data ifr\_ifru.ifru\_data;//接口使用

#define ifr\_ifrindex ifr\_ifru.ifru\_ivalue;//网络接口序号

#define ifr\_bandwidth ifr\_ifru.ifru\_ivalue;//连接带宽

#define ifr\_qlen ifr\_ifru.ifru\_ivalue;//传输单元长度

#define ifr\_newname ifr\_ifru.ifru\_newname;//新名称

#define ifr\_seeting ifr\_ifru.ifru\_settings;//设备协议设置

如果想获得网络接口的相关信息，就传入ifreq结构体.

(2)网卡设备属性ifmap

struct ifmap{//网卡设备的映射属性   
 unsigned long mem\_start;//开始地址   
 unsigned long mem\_end;//结束地址   
 unsigned short base\_addr;//基地址   
 unsigned char irq;//中断号   
 unsigned char dma;//DMA   
 unsigned char port;//端口   
}

(3)网络配置接口ifconf

struct ifconf{//网络配置结构体是一种缓冲区   
 int ifc\_len;//缓冲区ifr\_buf的大小   
 union{   
  char\_\_user \*ifcu\_buf;//绘冲区指针   
 struct ifreq\_\_user\* ifcu\_req;//指向ifreq指针   
}ifc\_ifcu;   
};   
#define ifc\_buf ifc\_ifcu.ifcu\_buf;//缓冲区地址   
#define ifc\_req ifc\_ifcu.ifcu\_req;//ifc\_req地址

(4)ARP高速缓存操作arpreq

/\*\*   
ARP高速缓存操作，包含IP地址和硬件地址的映射表   
操作ARP高速缓存的命令字 SIOCDARP,SIOCGARP,SIOCSARP分别是删除ARP高速缓存的一条记录，获得ARP高速缓存的一条记录和修改ARP高速缓存的一条记录   
struct arpreq{   
  struct sockaddr arp\_pa;//协议地址   
  struct sockaddr arp\_ha;//硬件地址   
  int arp\_flags;//标记   
  struct sockaddr arp\_netmask;//协议地址的子网掩码   
  char arp\_dev[16];//查询网络接口的名称   
}

3. 请求码request

|  |  |  |  |
| --- | --- | --- | --- |
| 类别 | Request | 说明 | 数据类型 |
| **套**  **接**  **口** | SIOCATMARK  SIOCSPGRP  SIOCGPGRP | 是否位于带外标记  设置套接口的进程ID或进程组ID  获取套接口的进程ID或进程组ID | int  int  int |
| **文**  **件** | FIONBIN  FIOASYNC  FIONREAD  FIOSETOWN  FIOGETOWN | 设置/清除非阻塞I/O标志  设置/清除信号驱动异步I/O标志  获取接收缓存区中的字节数  设置文件的进程ID或进程组ID  获取文件的进程ID或进程组ID | int  int  int  int  int |
| **接**  **口** | SIOCGIFCONF  SIOCSIFADDR  SIOCGIFADDR  SIOCSIFFLAGS  SIOCGIFFLAGS  SIOCSIFDSTADDR  SIOCGIFDSTADDR  SIOCGIFBRDADDR  SIOCSIFBRDADDR  SIOCGIFNETMASK  SIOCSIFNETMASK  SIOCGIFMETRIC  SIOCSIFMETRIC  SIOCGIFMTU  SIOCxxx | 获取所有接口的清单  设置接口地址  获取接口地址  设置接口标志  获取接口标志  设置点到点地址  获取点到点地址  获取广播地址  设置广播地址  获取子网掩码  设置子网掩码  获取接口的测度  设置接口的测度  获取接口MTU  （还有很多取决于系统的实现） | struct ifconf  struct ifreq  struct ifreq  struct ifreq  struct ifreq  struct ifreq  struct ifreq  struct ifreq  struct ifreq  struct ifreq  struct ifreq  struct ifreq  struct ifreq  struct ifreq |
| **ARP** | SIOCSARP  SIOCGARP  SIOCDARP | 创建/修改ARP表项  获取ARP表项  删除ARP表项 | struct arpreq  struct arpreq  struct arpreq |
| **路**  **由** | SIOCADDRT  SIOCDELRT | 增加路径  删除路径 | struct rtentry  struct rtentry |
| **流** | I\_xxx |  |  |

4. 相关例子

(1)网络接口信息   
选项获取填充struct ifreq的ifr\_name

1 #include <stdio.h>

2 #include <stdlib.h>

3 #include <string.h>

4 #include <sys/types.h>

5 #include <sys/socket.h>

6 #include <netinet/in.h>

7 #include <linux/if.h>

8 #include <arpa/inet.h>

9 #include <linux/sockios.h>

10 /\*\*

11 ioctl函数是与内核交互的一种方法，使用ioctl函数与内核协议栈进行交互

12 ioctl函数可操作I/O请求，文件请求与网络接口请求

13 网络接口请求的几个结构体:

14 struct ifreq{

15 #define IFHWADDRLEN 6 //6个字节的硬件地址，即MAC

16 union{

17 char ifrn\_name[IFNAMESIZ];//网络接口名称

18 }ifr\_ifrn;

19 union{

20 struct sockaddr ifru\_addr;//本地IP地址

21 struct sockaddr ifru\_dstaddr;//目标IP地址

22 struct sockaddr ifru\_broadaddr;//广播IP地址

23 struct sockaddr ifru\_netmask;//本地子网掩码地址

24 struct sockaddr ifru\_hwaddr;//本地MAC地址

25 short ifru\_flags;//网络接口标记

26 int ifru\_ivalue;//不同的请求含义不同

27 struct ifmap ifru\_map;//网卡地址映射

28 int ifru\_mtu;//最大传输单元

29 char ifru\_slave[IFNAMSIZ];//占位符

30 char ifru\_newname[IFNAMSIZE];//新名称

31 void \_\_user\* ifru\_data;//用户数据

32 struct if\_settings ifru\_settings;//设备协议设置

33

34 }ifr\_ifru;

35 }

36 #define ifr\_name ifr\_ifrn.ifrn\_name;//接口名称

37 #define ifr\_hwaddr ifr\_ifru.ifru\_hwaddr;//MAC

38 #define ifr\_addr ifr\_ifru.ifru\_addr;//本地IP

39 #define ifr\_dstaddr ifr\_ifru.dstaddr;//目标IP

40 #define ifr\_broadaddr ifr\_ifru.broadaddr;//广播IP

41 #define ifr\_netmask ifr\_ifru.ifru\_netmask;//子网掩码

42 #define ifr\_flags ifr\_ifru.ifru\_flags;//标志

43 #define ifr\_metric ifr\_ifru.ifru\_ivalue;//接口侧度

44 #define ifr\_mtu ifr\_ifru.ifru\_mtu;//最大传输单元

45 #define ifr\_map ifr\_ifru.ifru\_map;//设备地址映射

46 #define ifr\_slave ifr\_ifru.ifru\_slave;//副设备

47 #define ifr\_data ifr\_ifru.ifru\_data;//接口使用

48 #define ifr\_ifrindex ifr\_ifru.ifru\_ivalue;//网络接口序号

49 #define ifr\_bandwidth ifr\_ifru.ifru\_ivalue;//连接带宽

50 #define ifr\_qlen ifr\_ifru.ifru\_ivalue;//传输单元长度

51 #define ifr\_newname ifr\_ifru.ifru\_newname;//新名称

52 #define ifr\_seeting ifr\_ifru.ifru\_settings;//设备协议设置

53

54 struct ifmap{//网卡设备的映射属性

55 unsigned long mem\_start;//开始地址

56 unsigned long mem\_end;//结束地址

57 unsigned short base\_addr;//基地址

58 unsigned char irq;//中断号

59 unsigned char dma;//DMA

60 unsigned char port;//端口

61 }

62

63 struct ifconf{//网络配置结构体是一种缓冲区

64 int ifc\_len;//缓冲区ifr\_buf的大小

65 union{

66 char\_\_user \*ifcu\_buf;//绘冲区指针

67 struct ifreq\_\_user\* ifcu\_req;//指向ifreq指针

68 }ifc\_ifcu;

69

70 };

71 #define ifc\_buf ifc\_ifcu.ifcu\_buf;//缓冲区地址

72 #define ifc\_req ifc\_ifcu.ifcu\_req;//ifc\_req地址

73

74 (1)获得配置选项SIOCGIFCONF获得网络接口的配置情况 需要填充struct ifreq中ifr\_name变量

75 (2)其它选项获取填充struct ifreq的ifr\_name

76 \*\*/

77

78 int

79 main(int argc, char\*argv[])

80 {

81 int s;

82 int err;

83 s = socket(AF\_INET, SOCK\_DGRAM, 0);

84 if (s < 0)

85 {

86 perror("socket error");

87 return;

88 }

89

90 //传入网络接口序号，获得网络接口的名称

91 struct ifreq ifr;

92

93 ifr.ifr\_ifindex = 2; //获得第2个网络接口的名称

94 err = ioctl(s, SIOCGIFNAME, &ifr);

95

96 if (err)

97 {

98 perror("index error");

99 }

100 else

101 {

102

103 printf("the %dst interface is:%s\n", ifr.ifr\_ifindex, ifr.ifr\_name);

104

105 }

106 //传入网络接口名称，获得标志

107 memcpy(ifr.ifr\_name, "eth0", 5);

108 err = ioctl(s, SIOCGIFFLAGS, &ifr);

109 if (!err)

110 {

111 printf("SIOCGIFFLAGS:%d\n", ifr.ifr\_flags);

112 }

113 //获得MTU和MAC

114 err = ioctl(s, SIOCGIFMTU, &ifr);

115 if (!err)

116 {

117 printf("SIOCGIFMTU:%d\n", ifr.ifr\_mtu);

118 }

119 //获得MAC地址

120 err = ioctl(s, SIOCGIFHWADDR, &ifr);

121 if (!err)

122 {

123 unsigned char\* hw = ifr.ifr\_hwaddr.sa\_data;

124 printf("SIOCGIFHWADDR:%02x:%02x:%02x:%02x:%02x:%02x\n", hw[0], hw[1],

125 hw[2], hw[3], hw[4], hw[5]);

126 }

127 //获得网卡映射参数 命令字SIOCGIFMAP

128 err = ioctl(s, SIOCGIFMAP, &ifr);

129 if (!err)

130 {

131 printf(

132 "SIOCGIFMAP,mem\_start:%d,mem\_end:%d,base\_addr:%d,ifr\_map:%d,dma:%d,port:%d\n",

133 ifr.ifr\_map.mem\_start, ifr.ifr\_map.mem\_end, ifr.ifr\_map.base\_addr,

134 ifr.ifr\_map.irq, ifr.ifr\_map.dma, ifr.ifr\_map.port);

135 }

136

137 //获得网卡序号

138 err = ioctl(s, SIOCGIFINDEX, &ifr);

139 if (!err)

140 {

141 printf("SIOCGIFINDEX:%d\n", ifr.ifr\_ifindex);

142 }

143 //获取发送队列的长度

144 err = ioctl(s, SIOCGIFTXQLEN, &ifr);

145 if (!err)

146 {

147 printf("SIOCGIFTXQLEN:%d\n", ifr.ifr\_qlen);

148

149 }

150 //获取网络接口IP

151

152 struct sockaddr\_in \*sin = (struct sockaddr\_in\*) &ifr.ifr\_addr; //保存的是二进制IP

153 char ip[16]; //字符数组，存放字符串

154 memset(ip, 0, 16);

155 err = ioctl(s, SIOCGIFADDR, &ifr);

156 if (!err)

157 {

158 inet\_ntop(AF\_INET, &sin->sin\_addr.s\_addr, ip, 16); //转换的字符串保存到ip数组中，第二个参数是要转换的二进制IP指针，第三个参数是转换完成存放IP的缓冲区，最后一个参数是缓冲区的长度

159 printf("SIOCGIFADDR:%s\n", ip);

160

161 }

162

163 //查询目标IP地址

164 err = ioctl(s, SIOCGIFDSTADDR, &ifr);

165 if (!err)

166 {

167 inet\_ntop(AF\_INET, &sin->sin\_addr.s\_addr, ip, 16);

168 printf("SIOCGIFDSTADDR:%s\n", ip);

169 }

170

171 //查询子网掩码

172 err = ioctl(s, SIOCGIFNETMASK, &ifr);

173 if (!err)

174 {

175 inet\_ntop(AF\_INET, &sin->sin\_addr.s\_addr, ip, 16);

176 printf("SIOCGIFNETMASK:%s\n", ip);

177 }

178

179 //设置IP地址，设置网络接口

180 inet\_pton(AF\_INET, "222.27.253.108", &sin->sin\_addr.s\_addr); //将字符串IP转换成二进制

181 err = ioctl(s, SIOCSIFADDR, &ifr); //发送设置本机ip地址请求命令

182 if (!err)

183 {

184 printf("check IP-----");

185 memset(&ifr, 0, sizeof(ifr));

186 memcpy(ifr.ifr\_name, "eth0", 5);

187 ioctl(s, SIOCGIFADDR, &ifr);

188 inet\_ntop(AF\_INET, &sin->sin\_addr.s\_addr, ip, 16);

189 printf("%s\n", ip);

190 }

191 //得到接口的广播地址

192 memset(&ifr, 0, sizeof(ifr));

193 memcpy(ifr.ifr\_name, "eth0", 5);

194 ioctl(s, SIOCGIFBRDADDR, &ifr);

195 struct sockaddr\_in \*broadcast = (struct sockaddr\_in\*) &ifr.ifr\_broadaddr;

196 //转换成字符串

197 inet\_ntop(AF\_INET, &broadcast->sin\_addr.s\_addr, ip, 16); //inet\_ntop将二进制IP转换成点分十进制的字符串

198 printf("BROADCAST IP:%s\n", ip);

199 close(s);

200

201 }

View Code

运行结果:

[root@localhost ~]# ./ioctl-test   
the 2st interface is:eth0   
SIOCGIFFLAGS:4163   
SIOCGIFMTU:1500   
SIOCGIFHWADDR:00:13:d4:36:98:34   
SIOCGIFMAP,mem\_start:0,mem\_end:0,base\_addr:60416,ifr\_map:201,dma:0,port:0   
SIOCGIFINDEX:2   
SIOCGIFTXQLEN:1000   
SIOCGIFADDR:222.27.253.108   
SIOCGIFDSTADDR:222.27.253.108   
SIOCGIFNETMASK:255.255.255.0   
check IP-----222.27.253.108   
BROADCAST IP:222.27.253.255

(2)查看arp高速缓存信息

1 #include <stdio.h>

2 #include <sys/types.h>

3 #include <sys/socket.h>

4 #include <netinet/in.h>

5 #include <arpa/inet.h>

6 #include <net/if\_arp.h>

7 #include <string.h>

8 #include <stdlib.h>

9 #include <linux/sockios.h>

10 /\*\*

11 ARP高速缓存操作，包含IP地址和硬件地址的映射表

12 操作ARP高速缓存的命令字 SIOCDARP,SIOCGARP,SIOCSARP分别是删除ARP高速缓存的一条记录，获得ARP高速缓存的一条记录和修改ARP高速缓存的一条记录

13

14 struct arpreq

15 {

16 struct sockaddr arp\_pa; //协议地址

17 struct sockaddr arp\_ha;//硬件地址

18 int arp\_flags;//标记

19 struct sockaddr arp\_netmask;//协议地址的子网掩码

20 char arp\_dev[16];//查询网络接口的名称

21 }

22

23 \*\*/

24 //根据IP地址查找硬件地址

25 int main(int argc,char\*argv[])

26 {

27 int s;

28 int err;

29 struct arpreq arpreq;

30 struct sockaddr\_in \*addr=(struct sockaddr\_in\*)&arpreq.arp\_pa; //IP地址

31 s=socket(AF\_INET,SOCK\_DGRAM,0);

32 if(s<0)

33 {

34 perror("socket error");

35 }

36

37 addr->sin\_family=AF\_INET;

38 addr->sin\_addr.s\_addr=inet\_addr(argv[1]); //转换成二进制IP

39 if(addr->sin\_addr.s\_addr==INADDR\_NONE)

40 {

41 printf("IP地址格式错误\n");

42 }

43 strcpy(arpreq.arp\_dev,"eth0");

44 err=ioctl(s,SIOCGARP,&arpreq);

45 if(err==-1)

46 {

47 perror("arp");

48 return;

49 }

50

51 unsigned char\* hw=(unsigned char\*)&arpreq.arp\_ha.sa\_data; //硬件地址

52 printf("%s\n",argv[1]);

53 printf("%02x:%02x:%02x:%02x:%02x:%02x\n",hw[0],hw[1],hw[2],hw[3],hw[4],hw[5]);

54 close(s);

55 return 0;

56

57 }

View Code

运行结果:

[root@localhost ~]# ./ioctl-arp 222.27.253.1   
222.27.253.1   
00:0f:e2:5f:3c:8c   
查看网关的MAC.在查看ARP高速缓存时要传入IP地址与接口信息.而获得接口信息要传入接口名ifr\_name,如eth0.

总结:

本文主要介绍了获得网络接口请求信息，获得网卡设备映射属性,配置网络接口,获得ARP高速缓存等.其它ioctl函数还能对操作文件，操作I/O,操作路由等。最后,对于网络接口的操作与ARP高速缓存的操作分别给出了实例.

转自 ： <http://blog.csdn.net/chenjin_zhong/article/details/7269558>