```
/*Write a program to analyze following packet formats captured through Wireshark
for wired
network. 1. Ethernet 2. IP 3.TCP 4. UDP*/

#include<stdio.h> //For standard things
#include<stdlib.h>    //malloc
#include<string.h>    //strlen
#include<netinet/ip_icmp.h>   //Provides declarations for icmp header
#include<netinet/udp.h>   //Provides declarations for udp header
#include<netinet/tcp.h>   //Provides declarations for tcp header
#include<netinet/ip.h>    //Provides declarations for ip header
#include<sys/socket.h>
#include<arpa/inet.h>

void ProcessPacket(unsigned char* , int);
void print_ip_header(unsigned char* , int);
void print_tcp_packet(unsigned char * , int );

void PrintData (unsigned char* , int);
int sock_raw;
FILE *logfile;
struct sockaddr_in source,dest;
int tcp=0,udp=0,icmp=0,others=0,igmp=0,total=0,i,j;

int main()
{
    int saddr_size , data_size;

     struct sockaddr saddr;
struct in_addr in;
    unsigned char *buffer = (unsigned char *) malloc(65536);

    logfile=fopen("log.txt","w");
    if(logfile==NULL)
    {
        printf("Unable to create log.txt file.");
    }
    printf("Starting...\n");

    int sock_raw = socket( AF_INET , SOCK_RAW , IPPROTO_TCP) ;


    if(sock_raw < 0)
    {
        //Print the error with proper message
        perror("Socket Error");
        return 1;
    }
    while(1)
    {
        saddr_size = sizeof saddr;
        //Receive a packet
        data_size = recvfrom(sock_raw , buffer , 65536 , 0 , &saddr ,
&saddr_size);
        if(data_size <0 )
        {
            printf("Recvfrom error , failed to get packets\n");
            return 1;
        }
        //Now process the packet
        ProcessPacket(buffer , data_size);

    }
    close(sock_raw);
    printf("\nFinished\n");
    return 0;
}

void ProcessPacket(unsigned char* buffer, int size)
{
    //Get the IP Header part of this packet , excluding the ethernet header
    struct iphdr *iph = (struct iphdr*)buffer;
    ++total;
    switch (iph->protocol) //Check the Protocol and do accordingly...
    {
        case 1:  //ICMP Protocol

           ++icmp;
            break;

        case 2:  //IGMP Protocol
            ++igmp;
            break;

       case 6:  //TCP Protocol
            ++tcp;
            print_tcp_packet(buffer , size);
            break;

        case 17: //UDP Protocol

            ++udp;
            break;

        default: //Some Other Protocol like ARP etc.
            ++others;
            break;
    }
    printf("TCP : %d   UDP : %d   ICMP : %d   IGMP : %d   Others : %d   Total :
%d\r", tcp , udp , icmp , igmp , others , total);
}



void print_ip_header(unsigned char* Buffer, int Size)
{


    unsigned short iphdrlen;

    struct iphdr *iph = (struct iphdr *)Buffer;
    iphdrlen =iph->ihl*4;

    memset(&source, 0, sizeof(source));
    source.sin_addr.s_addr = iph->saddr;

    memset(&dest, 0, sizeof(dest));
    dest.sin_addr.s_addr = iph->daddr;

    fprintf(logfile , "\n");
    fprintf(logfile , "IP Header\n");
    fprintf(logfile , "   |-IP Version        : %d\n",(unsigned int)iph-
>version);
    fprintf(logfile , "   |-IP Header Length  : %d DWORDS or %d Bytes\n",
(unsigned int)iph->ihl,((unsigned int)(iph->ihl))*4);
    fprintf(logfile , "   |-Type Of Service   : %d\n",(unsigned int)iph->tos);
    fprintf(logfile , "   |-IP Total Length   : %d  Bytes(Size of
Packet)\n",ntohs(iph->tot_len));
```

```
    fprintf(logfile , "   |-Identification    : %d\n",ntohs(iph->id));
    fprintf(logfile , "   |-TTL      : %d\n",(unsigned int)iph->ttl);
    fprintf(logfile , "   |-Protocol : %d\n",(unsigned int)iph->protocol);
    fprintf(logfile , "   |-Checksum : %d\n",ntohs(iph->check));
    fprintf(logfile , "   |-Source IP        :
%s\n",inet_ntoa(source.sin_addr));
    fprintf(logfile , "   |-Destination IP   : %s\n",inet_ntoa(dest.sin_addr));
}

void print_tcp_packet(unsigned char* Buffer, int Size)
{



    unsigned short iphdrlen;


    struct iphdr *iph = (struct iphdr *)Buffer;
    iphdrlen = iph->ihl*4;

    struct tcphdr *tcph=(struct tcphdr*)(Buffer+iphdrlen);


    fprintf(logfile , "\n\n***********************TCP
Packet*************************\n");

    print_ip_header(Buffer,Size);

    fprintf(logfile , "\n");



    fprintf(logfile , "IP Header\n");
    PrintData(Buffer,iphdrlen);

    fprintf(logfile , "TCP Header\n");
    PrintData(Buffer+iphdrlen,tcph->doff*4);

    fprintf(logfile , "Data Payload\n");
     PrintData(Buffer + iphdrlen + tcph->doff*4 , (Size - tcph->doff*4-iph-
>ihl*4));
    }


    //Move the pointer ahead and reduce the size of string*/


void PrintData (unsigned char* data , int Size)
{
    int i , j;
    for(i=0 ; i < Size ; i++)
    {
        if( i!=0 && i%16==0)   //if one line of hex printing is complete...
        {
            fprintf(logfile , "         ");
            for(j=i-16 ; j<i ; j++)
            {
                if(data[j]>=32 && data[j]<=128)
                    fprintf(logfile , "%c",(unsigned char)data[j]); //if its a
number or alphabet

                else fprintf(logfile , "."); //otherwise print a dot
            }
            fprintf(logfile , "\n");
        }

        if(i%16==0) fprintf(logfile , "   ");
            fprintf(logfile , " %02X",(unsigned int)data[i]);

        if( i==Size-1)  //print the last spaces
        {
            for(j =0;j<15-i%16;j++)
            {
              fprintf(logfile , "   "); //extra spaces
            }

            fprintf(logfile , "         ");

            for(j=i-i%16 ; j<=i ; j++)
            {
                if(data[j]>=32 && data[j]<=128)
                {
                  fprintf(logfile , "%c",(unsigned char)data[j]);
                }
                else
                {
                  fprintf(logfile , ".");
                }
            }

            fprintf(logfile ,  "\n" );
        }
    }
}
```

/*Output:

```
root@pl15:/home/pl15/Desktop# gcc Group-A-9.c

root@pl15:/home/pl15/Desktop# ./a.out

Starting...
TCP : 241   UDP : 0   ICMP : 0   IGMP : 0   Others : 0   Total : 240*/
```