# 基于Linux平台的网络分析程序

## 设计背景

随着网络时代不断的普及，网络已经成为人们生活中不可欠缺的重要部分。在人们使用网络的过程中，难免可能会出现一系列的网络问题，如ping不通，或者是网络不稳定开始掉包的问题，这一些问题通常一些精通网络的管理员才能解决。本项目通过设计类似于wireshark的网络抓包分析工具来帮助管理员解决网络流量分析问题，同时提高自己对于网络的理解。

## 设计思路

本程序打算在Linux上面设计，基于pcap网络抓包库进行网络数据包的捕获。从数据链路层分析到网络层，最后再到数据传输层，每一层分别对应相应的数据结构，将每个数据包的重要数据结构化的打印出来。由于Linux自带的的头文件已经为我们定义好了各种数据包的数据结构，所以我们只需要从相应结构体取出相应的值即可。如，netinet/ether.h提供了数据链里层的数据结构，netinet/ip.h提供了网络层的数据结构，netinet/tcp.h提供了传输层(tcp)的数据结构，netinet/udp.h提供了传输层udp的数据结构。

## 开发环境

### 操作系统:

➜ src cat /etc/issue

Ubuntu 20.04 LTS \n \l

### 编译器:

➜ src gcc --version

gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0

Copyright (C) 2019 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO

warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

### 编辑器

➜ src vim --version

VIM - Vi IMproved 8.1 (2018 May 18, compiled Feb 01 2022 09:16:32)

Included patches: 1-2269, 3612, 3625, 3669, 3741

### 项目管理器

➜ net-stat make --version

GNU Make 4.2.1

Built for x86\_64-pc-linux-gnu

Copyright (C) 1988-2016 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law.

## 关键代码

### 项目管理代码

# project name (generate executable with this name)

TARGET = netstat

CC = gcc

# compiling flags here

CFLAGS =-Wall -g

LINKER = gcc -o

# linking flags here

LFLAGS =

LDFLAGS =-lpcap -lpthread -g #Linker

# change these to set the proper directories where each files should be

SRCDIR = src

OBJDIR = build

BINDIR = bin

SOURCES := $(wildcard $(SRCDIR)/\*.c)

INCLUDES := $(wildcard $(SRCDIR)/\*.h)

OBJECTS := $(SOURCES:$(SRCDIR)/%.c=$(OBJDIR)/%.o)

rm = rm -f

$(BINDIR)/$(TARGET): $(OBJECTS)

@$(LINKER) $@ $(LFLAGS) $(OBJECTS) ${LDFLAGS}

@echo "Linking complete!"

$(OBJECTS): $(OBJDIR)/%.o : $(SRCDIR)/%.c

@$(CC) $(CFLAGS) -c $< -o $@ ${LDFLAGS}

@echo "Compiled "$<" successfully!"

.PHONEY: clean

clean:

@$(rm) $(OBJECTS)

@echo "Cleanup complete!"

.PHONEY: remove

remove: clean

@$(rm) $(BINDIR)/$(TARGET)

@echo "Executable removed!"

### 主函数

使用switch case语句判断程序的运行参数

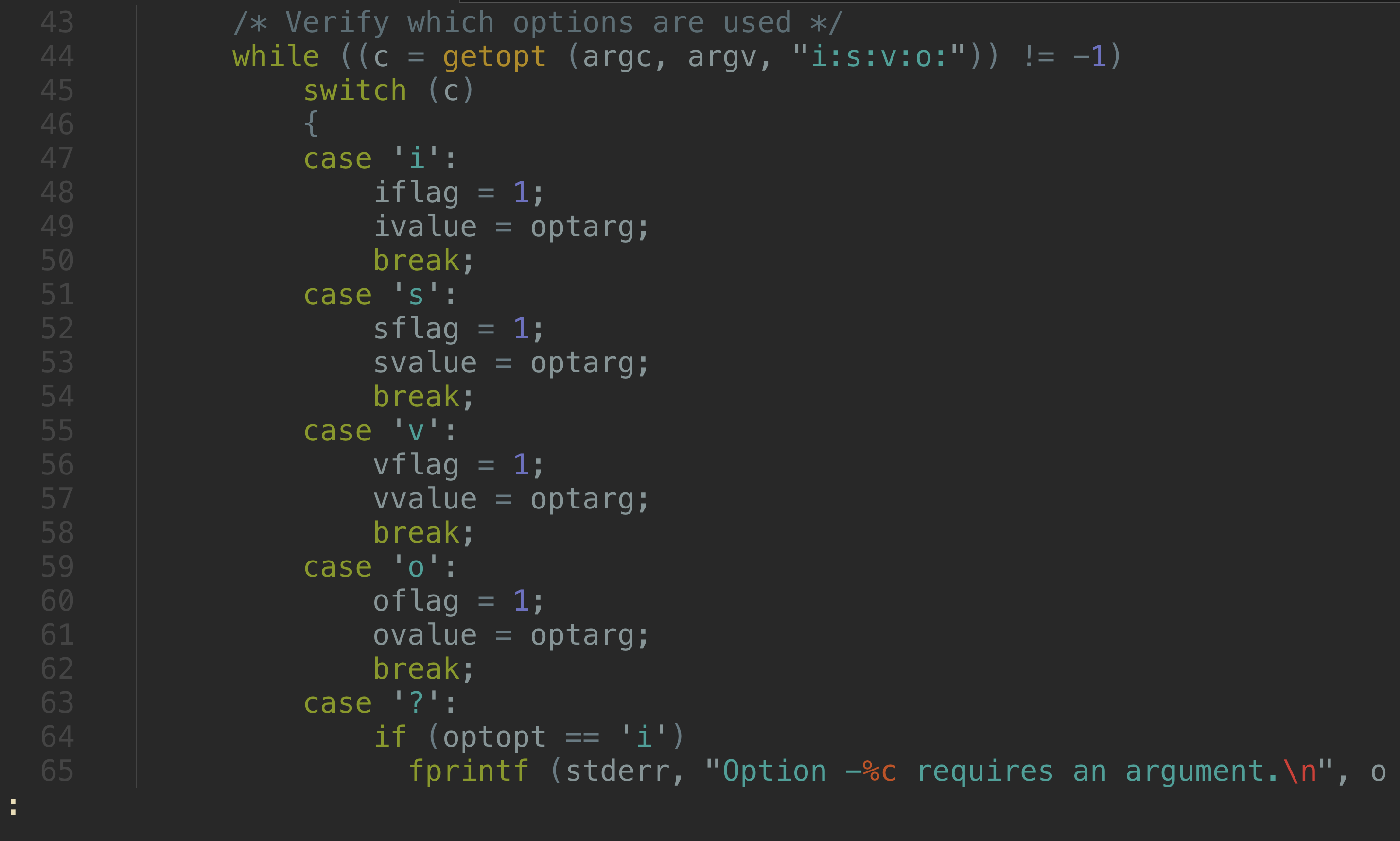


图 1参数判断

使用pcap库中的pcap\_lookupnet函数进行网卡的查找

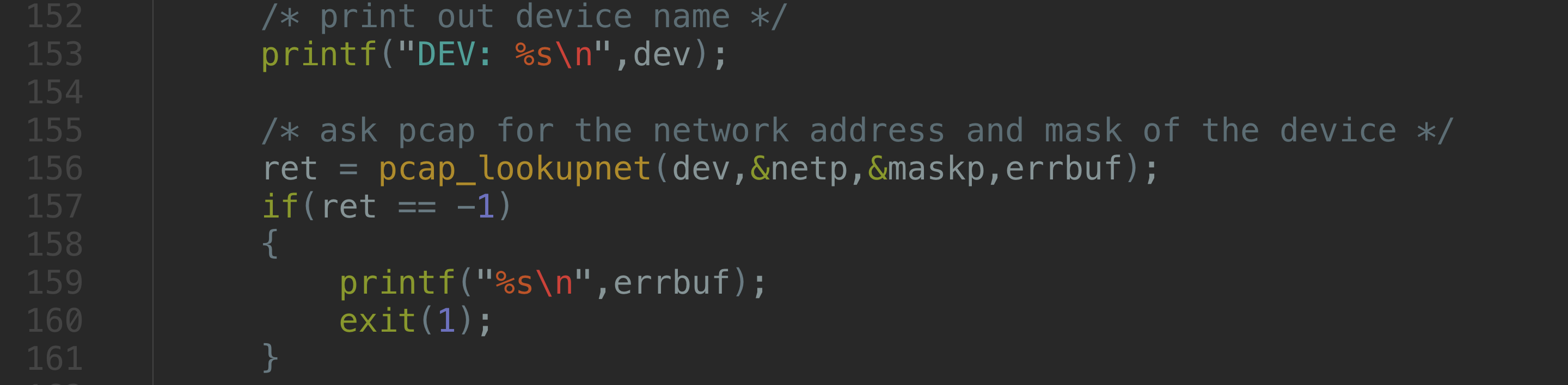
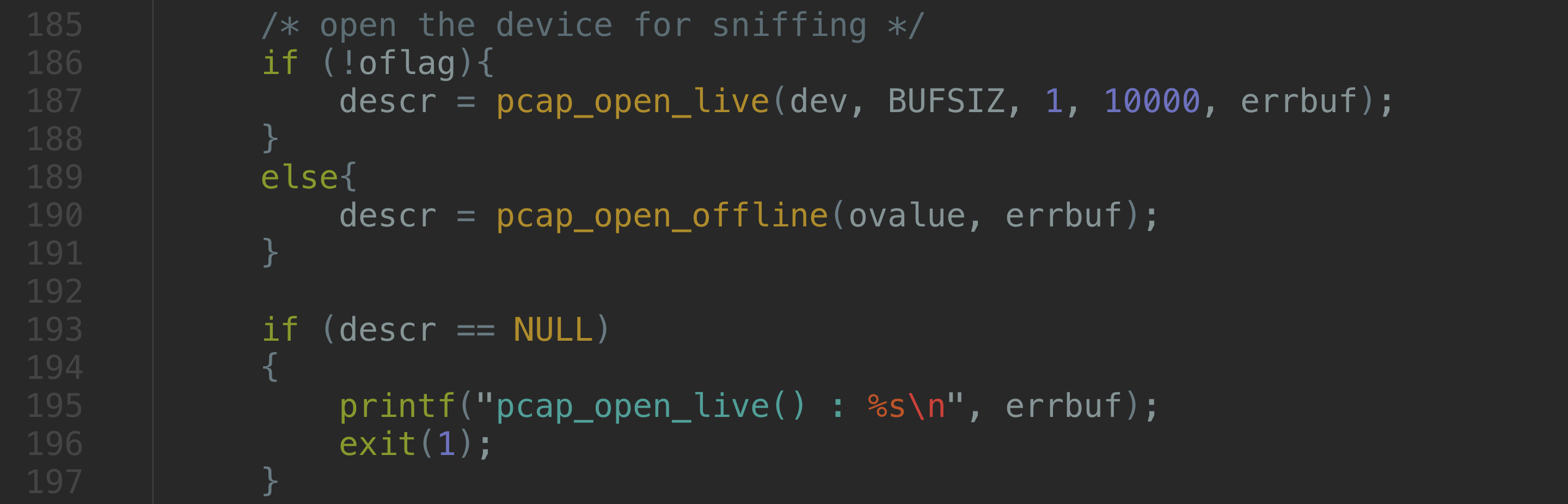
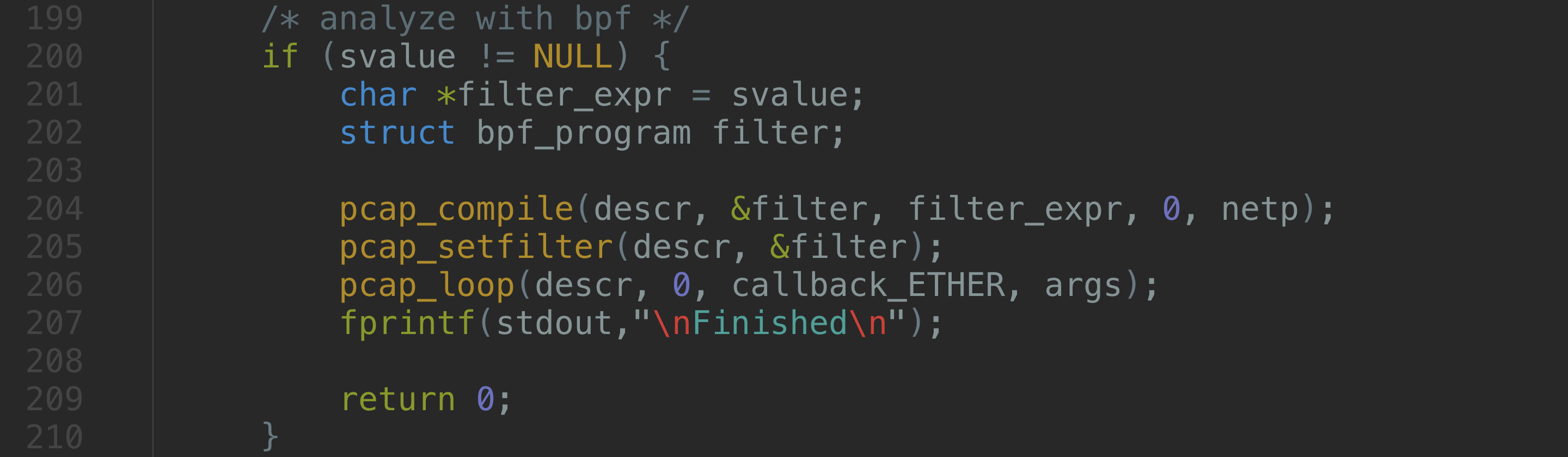


图 2查找网卡信息

使用pcap库中提供的pcap\_open\_live实现实时网卡数据包的捕获，使用pcap\_open\_offline实现打开离线数据包，也就是之前捕获的pcap后缀的数据包文件



使用pcap库中自带的pcap\_compile、pcap\_setfilter函数进行网络数据包规则的过滤（BPF规则），最后使用pcap\_loop函数将回掉函数传入pcap\_loop  


### 头文件依赖情况

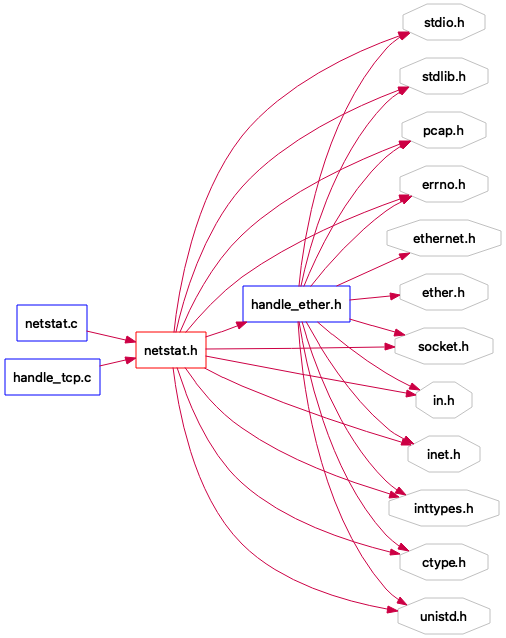


图 3主函数头文件依赖

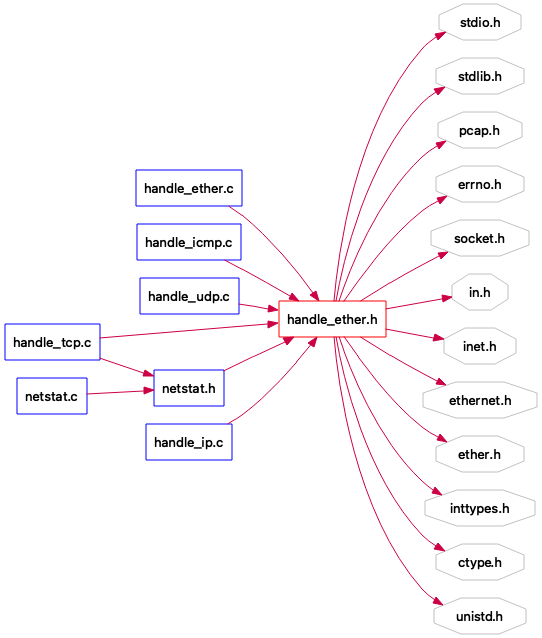


图 4数据链路层头文件依赖

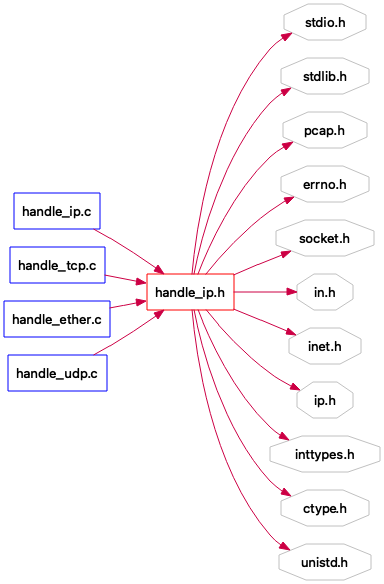


图 5网络层依赖情况

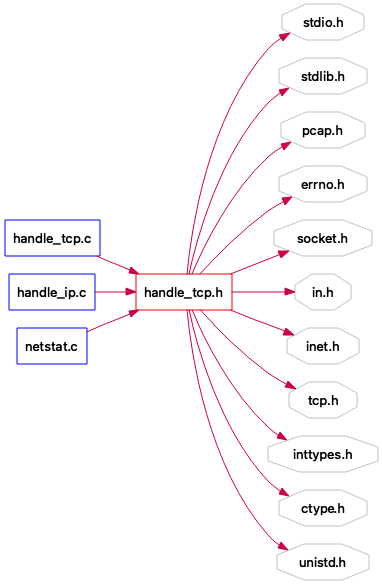


图 6传输层依赖情况

### 函数声明情况

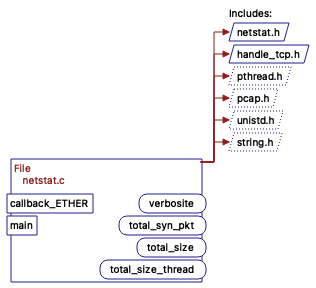


图 7主函数声明

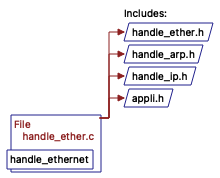


图 8数据链路层声明

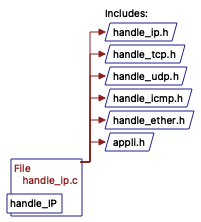


图 9网络层声明

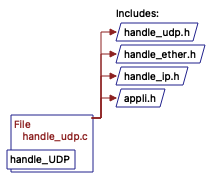


图 10传输层声明

## 运行步骤

### 程序编译

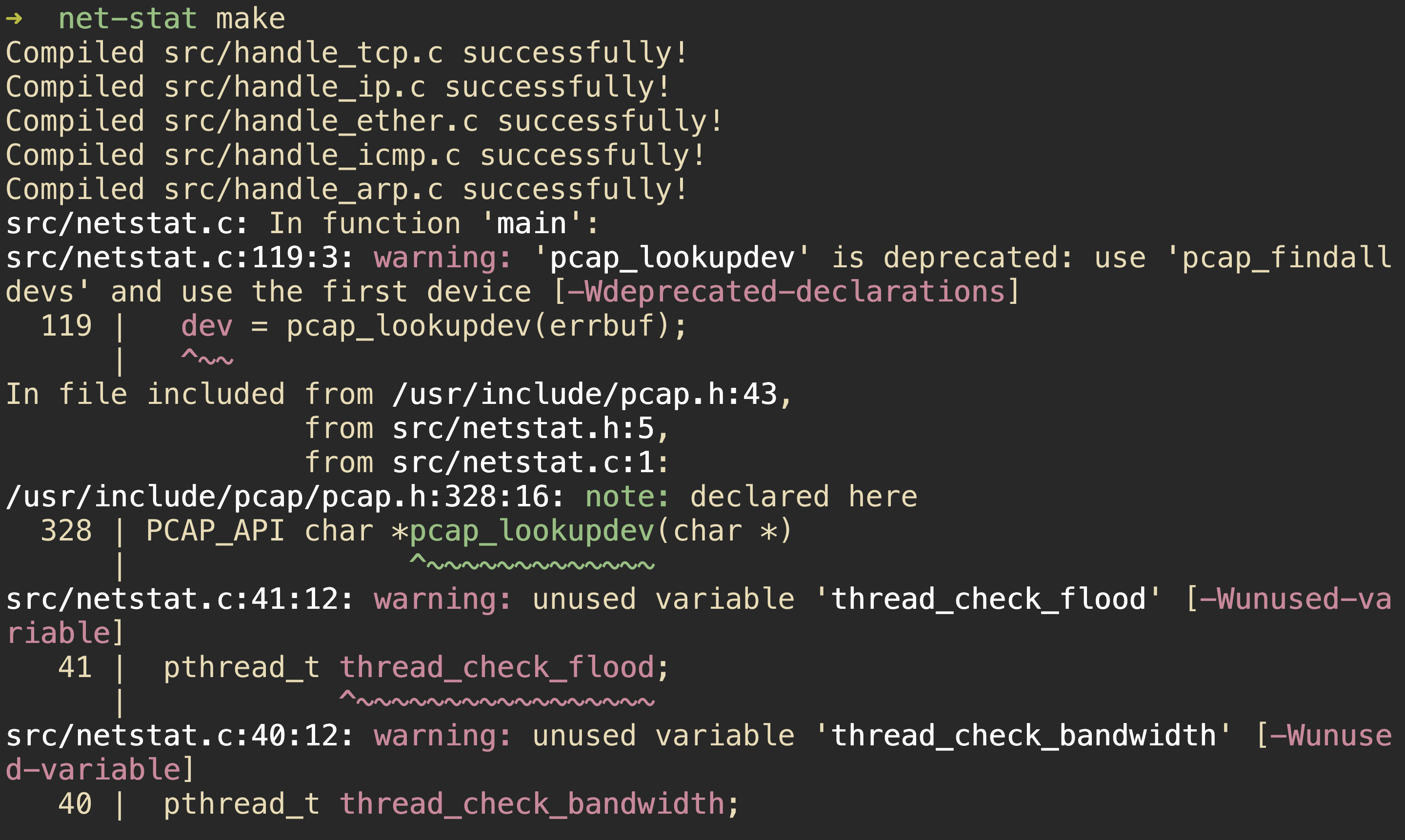


图 11编译程序

### 编译结果

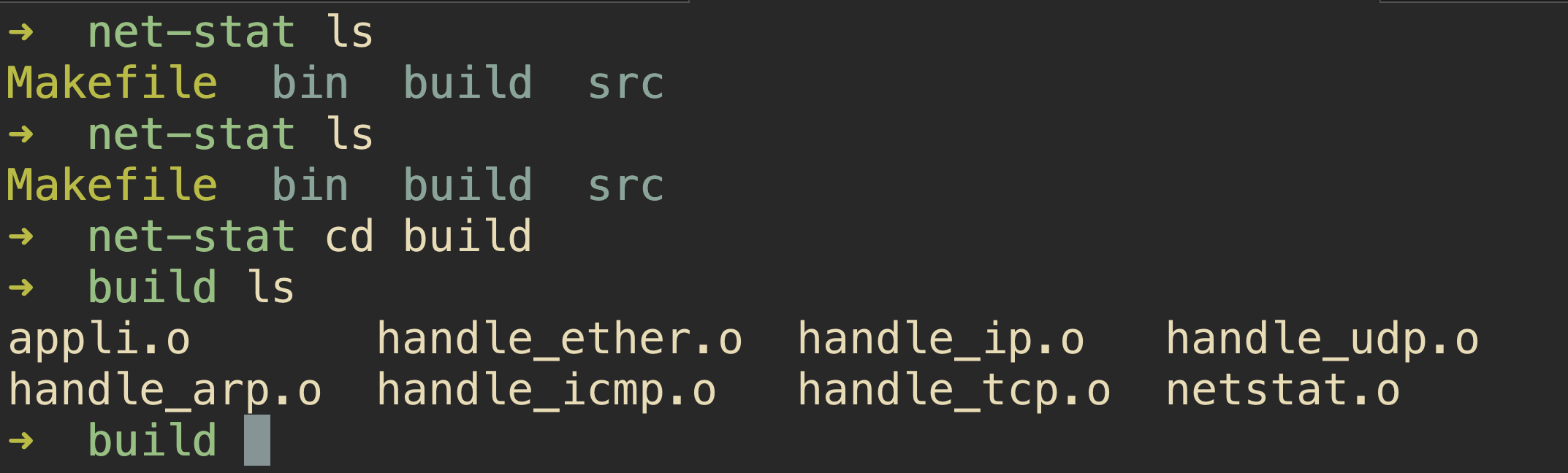


图 12目标文件

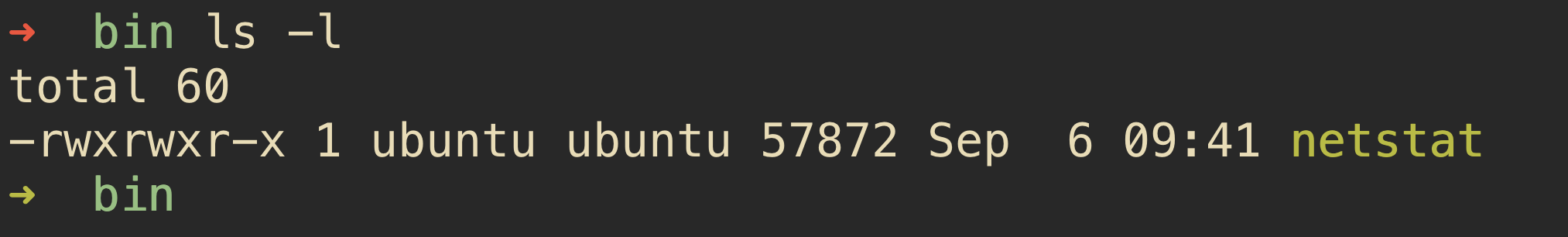


图 13可执行文件

### 运行程序

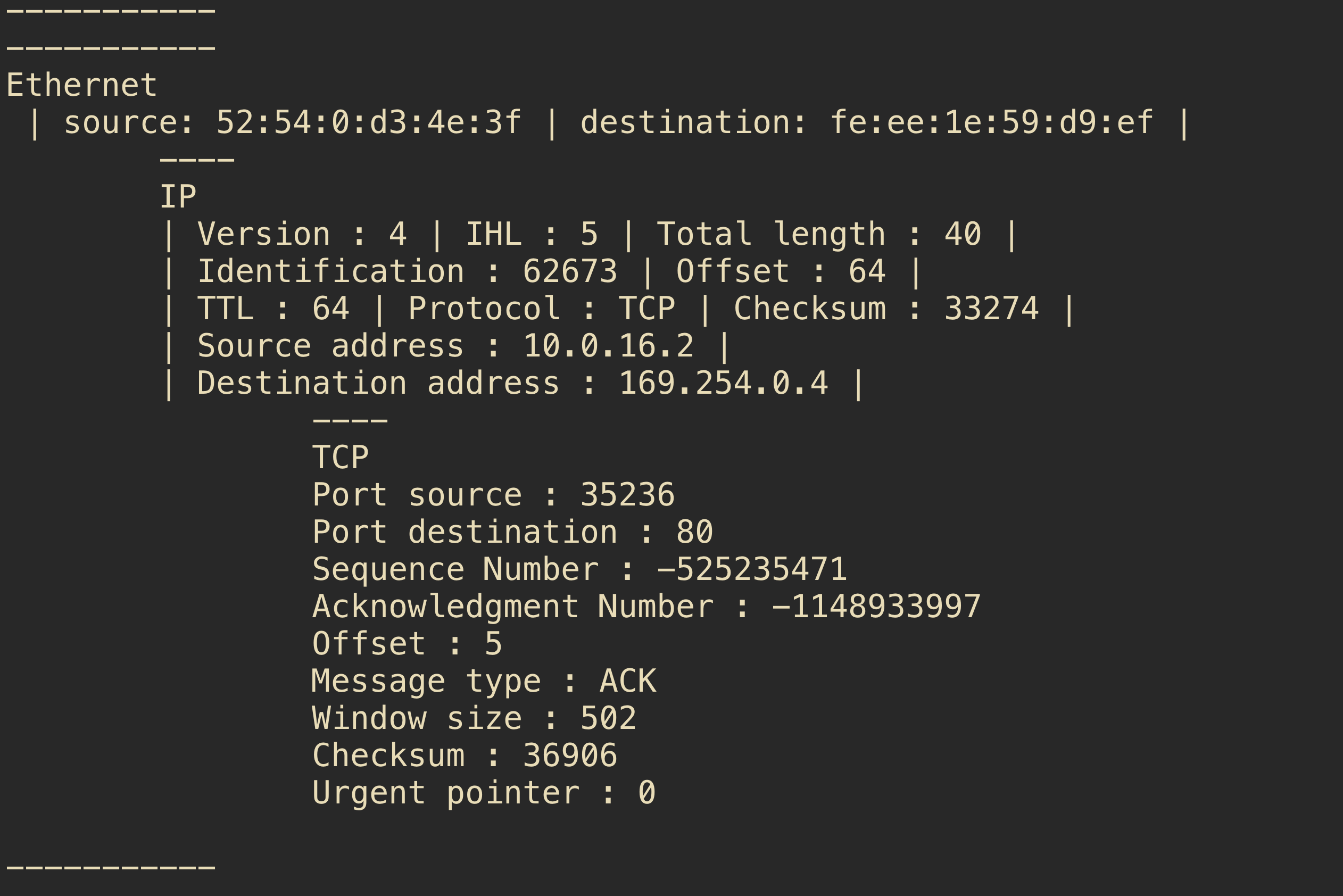


图 14数据链路层->网络层->传输层

## 结论

经历了本次项目，我更加巩固了C语言的基础，熟练地掌握了各种数据结构的分析和处理。通过在Linux上编程更加了解了Linux对于网络数据包的处理方式，学习了Pcap这个强大的网络数据包库。通过网上查找资料学会了使用Makefile对项目进行管理，减少了项目编译和调试的步骤，同时还学习了vim编程的各种技巧。