

**实验报告**

**实 验（三）**

题 目 基于libnet的程序设计

专 业 计 算 机 类

学　　 号 1160300620

班　　 级 1603006

学 生 张 恒

实 验 地 点 格物214

实 验 日 期 2019.3.28

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 3 -](#_Toc5108659)

[1.1 实验目的 - 3 -](#_Toc5108660)

[1.2 实验内容 - 3 -](#_Toc5108661)

[1.3 实验环境 - 3 -](#_Toc5108662)

[第2章 实验原理 - 4 -](#_Toc5108663)

[2.1 libnet的安装 - 4 -](#_Toc5108664)

[2.2 libnet的原理 - 4 -](#_Toc5108665)

[2.3 本实验中用到的API - 5 -](#_Toc5108666)

[2.4 利用libnet构造数据包的流程 - 6 -](#_Toc5108667)

[第3章 程序流程设计 - 7 -](#_Toc5108668)

[第4章 代码实现 - 8 -](#_Toc5108669)

[4.1 main函数的实现 - 8 -](#_Toc5108670)

[4.2 init\_libnet函数的实现 - 8 -](#_Toc5108671)

[4.3 buildtcp\_libnet函数的实现 - 9 -](#_Toc5108672)

[4.4 buildudp\_libnet函数的实现 - 9 -](#_Toc5108673)

[4.5 buildip\_libnet函数的实现 - 9 -](#_Toc5108674)

[4.6 buildether\_libnet函数的实现 - 10 -](#_Toc5108675)

[第5章 运行结果 - 10 -](#_Toc5108676)

[第6章 总结 - 12 -](#_Toc5108677)

[6.1本次实验的收获 - 12 -](#_Toc5108678)

[6.2 本次实验的难点 - 12 -](#_Toc5108679)

[第7章 参考文献 - 12 -](#_Toc5108680)

# 第1章 实验基本信息

## 1.1 实验目的

（1）了解构造数据包的原理及流程;

（2）能够熟练使用libnet构造数据包;

（3）能结合前面实验验证构造的数据包；

## 1.2 实验内容

1. 编程实现基于libnet的数据包构造；
2. 结合前面实验给出验证过程；
3. 并能够对源码进行解释；

## 1.3 实验环境

本实验中使用的系统环境是Ubuntu 18.04，编程语言为C语言，使用了gcc编译器；

# 第2章 实验原理

## 2.1 libnet的安装

打开ubuntu的bash shell，输入如下的指令

等待安装成功即可。

## 2.2 libnet的原理

现代计算机广泛采用5层模型，不同的操作系统都抽象出了五个层次，每个曾对相邻的层通过API提供服务。如图2-1所示，实际上，5个层次中的物理层程序员是无法直接操控的，所以可以视为4个层次。Libnet实际上就是让程序员有一个更简单的方式调用操作系统中各层提供的API。

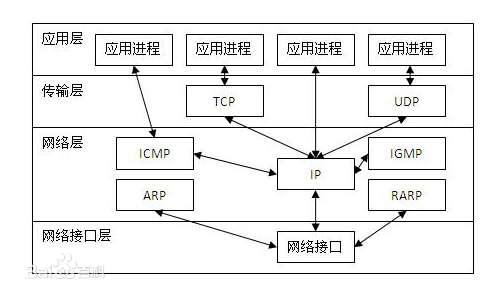


图2-1 操作系统中程序员可操纵的四个层

## 2.3 本实验中用到的API

// 数据包内存初始化及环境建立。函数返回一个 libnet \* 类型的指针，后面的操作// 都得使用这个指针

libnet\_t \*libnet\_init(int injection\_type, char \*device, char \*err\_buf);

// 释放资源

void libnet\_destroy(libnet\_t \*l);

// 将点分十进制数串转换为网络字节序 ip 地址

u\_int32\_t libnet\_name2addr4(libnet\_t \*l, char \*host\_name, u\_int8\_t use\_name);

// 获取接口设备硬件地址

struct libnet\_ether\_addr\* libnet\_get\_hwaddr(libnet\_t \*l);

// 构造UDP数据包

libnet\_ptag\_t libnet\_build\_udp(u\_int16\_t sp, u\_int16\_t dp, u\_int16\_t len, u\_int16\_t sum, u\_int8\_t \*payload, u\_int32\_t payload\_s, libnet\_t \*l, libnet\_ptag\_t ptag);

// 构造TCP数据包

libnet\_ptag\_t libnet\_build\_tcp(u\_int16\_t sp, u\_int16\_t dp, u\_int32\_t seq, u\_int32\_t ack, u\_int8\_t control, u\_int16\_t win, u\_int16\_t sum, u\_int16\_t urg, u\_int16\_t len, u\_int8\_t \*payload, u\_int32\_t payload\_s, libnet\_t \*l, libnet\_ptag\_t ptag );

// 构造ipv4数据包

libnet\_ptag\_t libnet\_build\_ipv4(u\_int16\_t ip\_len, u\_int8\_t tos, u\_int16\_t id, u\_int16\_t flag, u\_int8\_t ttl, u\_int8\_t prot, u\_int16 sum, u\_int32\_t src, u\_int32\_t dst, u\_int8\_t \*payload, u\_int32\_t payload\_s,libnet\_t \*l, libnet\_ptag\_t ptag );

// 构造以太网数据包

libnet\_ptag\_t libnet\_build\_ethernet(u\_int8\_t\*dst, u\_int8\_t \*src, u\_int16\_ttype, u\_int8\_t\*payload, u\_int32\_tpayload\_s, libnet\_t\*l, libnet\_ptag\_t ptag );

// 发送数据包

int libnet\_write(libnet\_t \* l);

## 2.4 利用libnet构造数据包的流程

如图2-2所示，使用libnet时，首先需要初始化，然后才能够构造数据包并发送，最后再释放初始化时申请的资源。其中，初始化之后，可以多次构造数据包，多次发送数据包。

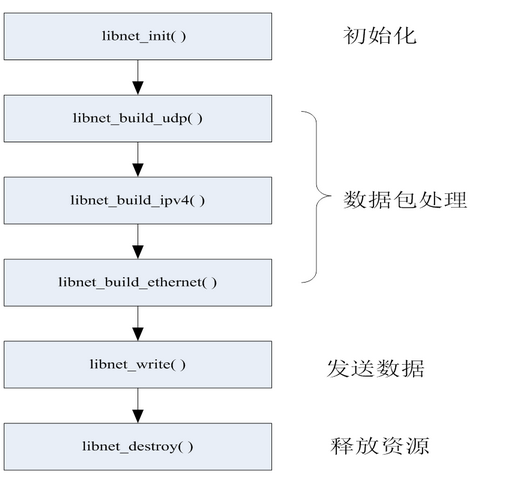


图2-2 使用libnet的流程

# 第3章 程序流程设计

程序的设计流程比较简单。如图3-1所示，用户打开程序后，程序自动进行初始化，并给用户展示一个菜单，可以选择：1）构造TCP包；2）构造UDP包；3）退出。

若是选择退出，程序会释放申请的资源并退出。

若是选择构造TCP包或者是UDP包，程序会要求用户输入相应的信息，并且发出。以构造一个UDP包为例：

1. 输入源端口和目的端口；
2. 输入需要携带的数据，允许为空；
3. 输入源ip，可以为空，默认为127.0.0.1；
4. 输入目的ip；
5. 输入源mac，可以为空，默认为当前网卡的mac；
6. 输入目的mac；

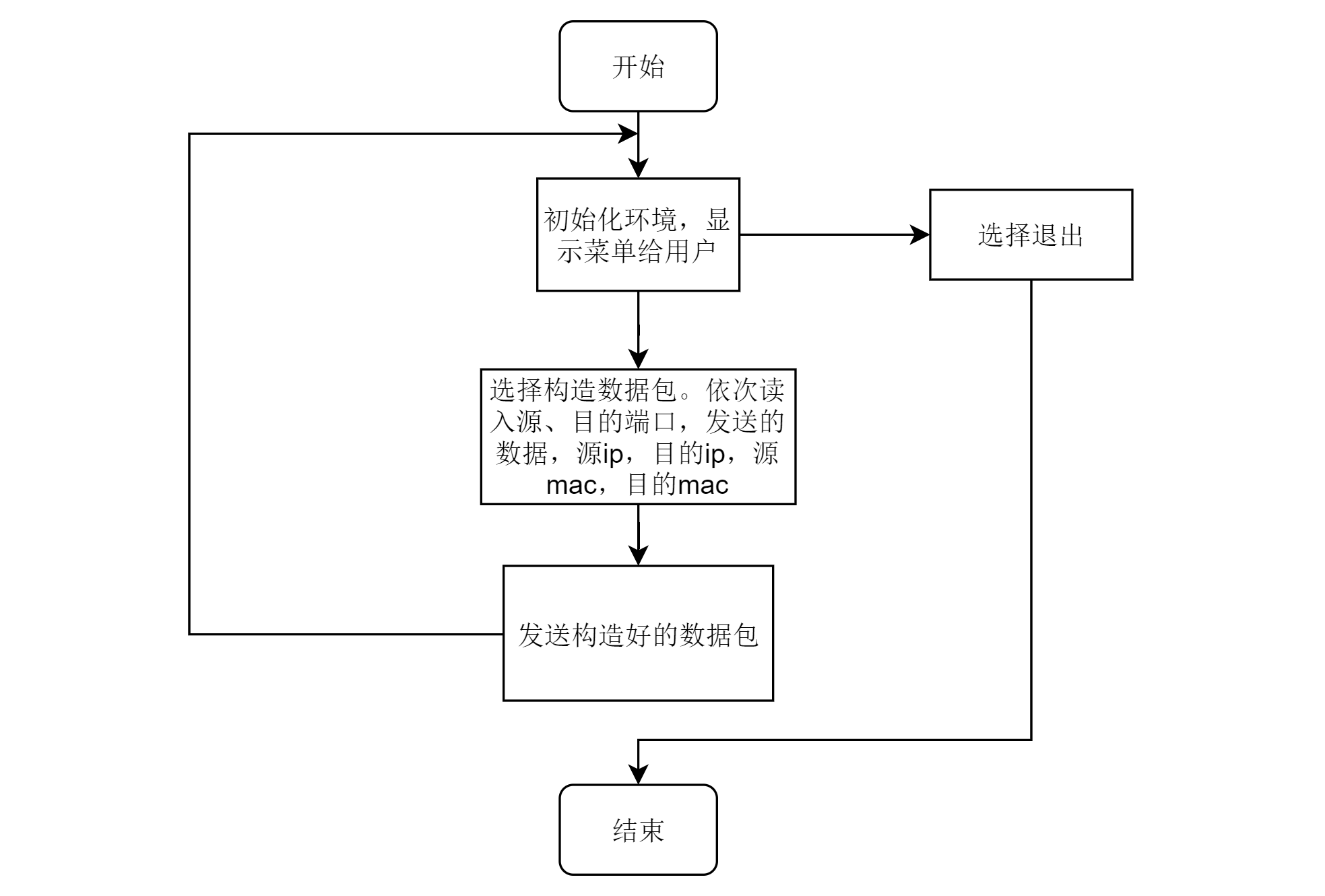


图3-1 程序的整体流程

# 第4章 代码实现

## 4.1 main函数的实现

main函数有一个无线循环执行代码块。在进入循环代码块前，会初始化libent，在离开循环代码块前，会释放初始化时申请的资源。在循环代码块中，每一次循环都能够构造一个tcp包或者是udp包并发送。

值得注意的是，在每次循环中，程序是否上一次构造udp（tcp）包，而这次构造tcp（udp）包。当构造第 i+1 个 udp（tcp）包时，libnet\_build\_udp（libnet\_build\_tcp）、libnet\_build\_ip、libnet\_build\_ether 函数中参数 ptag 需要是构造第 i 个 udp（tcp）时的相应函数返回值。因此，在从构造 udp（tcp）转换为构造 tcp（udp）时，需要将上一次构造 udp（tcp） 时的返回值缓存下来，并设置为上一次构造 tcp（udp） 时的返回值。具体实现的代码如图4-1所示。

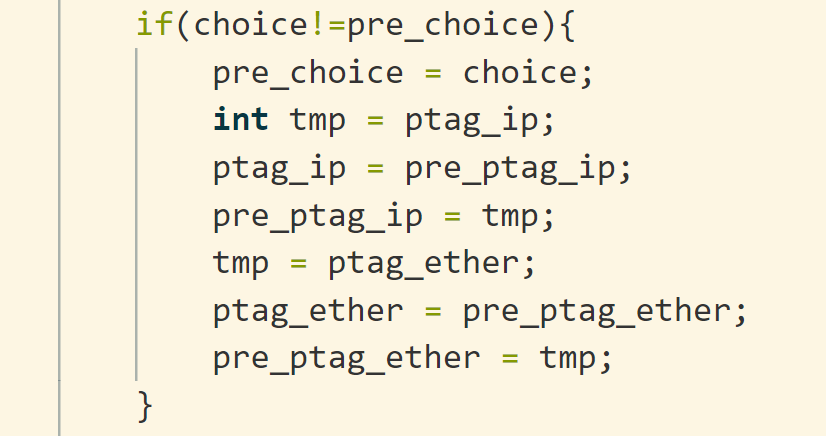


图4-1 检查构造包的类型是否发生变化，并进行相应处理

## 4.2 init\_libnet函数的实现

该函数的定义如下

int init\_libnet(libnet\_t \*\*device);

1. libnet\_t \*\*device：用以存储打开的设备的句柄；
2. 返回值为1表示成功，0表示失败；

该函数完成环境的初始化。

## 4.3 buildtcp\_libnet函数的实现

该函数的定义如下：

libnet\_ptag\_t buildtcp\_libnet(libnet\_t \*device, libnet\_ptag\_t ptag, u\_int16\_t \*len);

1. libnet\_t \*device：用以指明网卡设备；
2. libnet\_ptag\_t ptag：指明本次构造tcp包的ptag；
3. u\_int16\_t \*len：用以存储tcp包的总长度
4. 返回本次构造tcp包返回的ptag；

该函数的功能是让用户输入源端口、目的端口和需要携带的信息，并构建tcp包。

## 4.4 buildudp\_libnet函数的实现

该函数的定义如下：

libnet\_ptag\_t buildudp\_libnet(libnet\_t \*device, libnet\_ptag\_t ptag, u\_int16\_t \*len);

1. libnet\_t \*device：用以指明网卡设备；
2. libnet\_ptag\_t ptag：指明本次构造udp包的ptag；
3. u\_int16\_t \*len：用以存储udp包的总长度
4. 返回本次构造udp包返回的ptag；

该函数的功能是让用户输入源端口、目的端口和需要携带的信息，并构建udp包。

## 4.5 buildip\_libnet函数的实现

该函数的定义如下：

libnet\_ptag\_t buildip\_libnet(libnet\_t \*device, libnet\_ptag\_t ptag, u\_int16\_t \*len, int proc);

1. libnet\_t \*device：用以指明网卡设备；
2. libnet\_ptag\_t ptag：指明本次构造ip包的ptag；
3. u\_int16\_t \*len：用以存储ip包的总长度；
4. int proc：用以指明上层协议的类型；
5. 返回本次构造ip包返回的ptag；

该函数的功能是让用户输入源ip（允许为空，为空时使用默认值127.0.0.1）、目的ip，并构建ip包。

## 4.6 buildether\_libnet函数的实现

该函数的定义如下：

libnet\_ptag\_t buildlink\_libnet(libnet\_t \*device, libnet\_ptag\_t ptag, int proc);

1. libnet\_t \*device：用以指明网卡设备；
2. libnet\_ptag\_t ptag：指明本次构造以太网包的ptag；
3. int proc：用以指明上层协议的类型；
4. 返回本次构造以太网包返回的ptag；

该函数的功能是让用户输入源mac、目的mac，并构建以太网包。

# 第5章 运行结果

函数的结果比较简单。

打开程序，会出现一个菜单，可以选择1或者2进行组包。如图5-1所示是选择1的组包的过程。为求方便，这里面省略了一些输入过程，默认源端口和目的端口均为8080，源ip和目的ip均为127.0.0.1，目的mac和源mac均为当前网卡的mac地址。

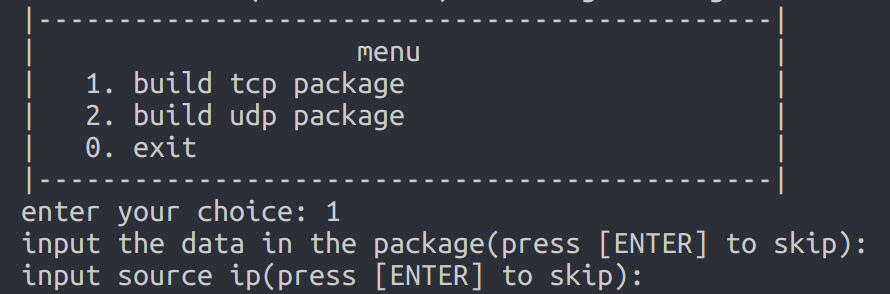


图5-1 构造tcp包的过程

如图5-2所示，所示是选择2的组包的过程，这是在组了若干个tcp包后切换组包的类型。同样，为求方便，这里面省略了一些输入过程，默认源端口和目的端口均为8080，源ip和目的ip均为127.0.0.1，目的mac和源mac均为当前网卡的mac地址。

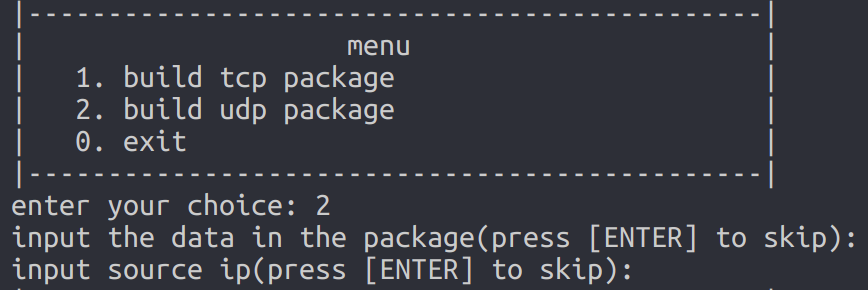


图5-2 构造udp包的过程

如图5-3所示，是利用上次实验中的程序进行抓包，将源ip、源端口、目的ip、目的端口存储在文件中的结果。

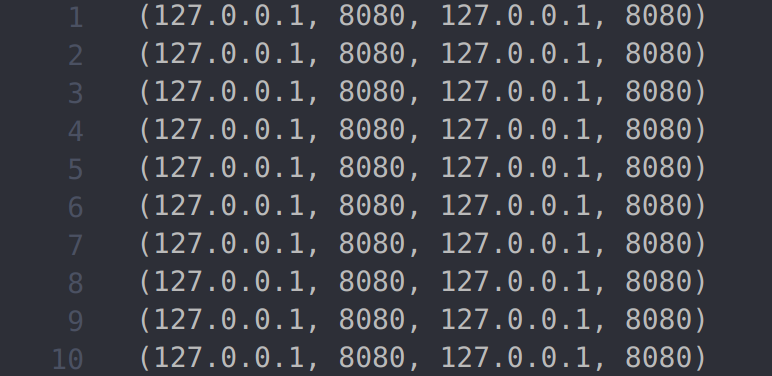


图5-3 抓包结果

# 第6章 总结

## 6.1本次实验的收获

（1）了解并掌握了使用libnet提供的API编写抓包程序技能；

（2）理解了构造数据包并发送的原理；

## 6.2 本次实验的难点

libnet单次初始化后，允许多次组包并发送。初始化后组第一个包的ptag为0，之后每次组包该参数是相应函数上一次组包的返回值。难点就在于，这里所谓的上一次需要区分上层协议的类型。以构造tcp包和udp包为例。

当构造第 i+1 个 udp（tcp）包时，libnet\_build\_udp（libnet\_build\_tcp）、libnet\_build\_ip、libnet\_build\_ether 函数中参数 ptag 需要是构造第 i 个 udp（tcp）时的相应函数返回值。因此，在从构造 udp（tcp）转换为构造 tcp（udp）时，需要将上一次构造 udp（tcp） 时的返回值缓存下来，并设置为上一次构造 tcp（udp） 时的返回值。

# 第7章 参考文献

[1] https://blog.csdn.net/tennysonsky/article/details/44944849.