课程编号：B080209020

信息安全工程实践 2
--网络程序设计实践报告

东北大学

学　　号	20155362	姓　　名	薛旗
班　　级	软信-1503	指 导 教 师	王学毅
程序实践名称	信息安全工程实践 2-网络程序设计		
程序实践内容	网 络 程 序 设 计		
开 设 学 期	2016-2017 第 二 学 期		
开 设 时 间	第 17 周 — — 第 19 周		
报 告 日 期	2017 年 7 月 7 日		
评 定 成 绩		评 定 人 签 字	
		评 定 日 期	

东北大学软件学院

实验 1 基于 Socket 的时间服务器

一、程序实践概述

1、题目名称：

基于 Socket 的时间服务器

2、时间进度：

2017.06.19 - 2017.06.20

3、开发环境：

Linux Deepin 15.4

二、问题分析

1、功能说明：

客户端向服务端发起连接，服务端给客户端返回时间。

2、解决方案：

将程序分为两个模块，一个模块负责通信，一个模块负责获取服务器时间。

三、方案设计

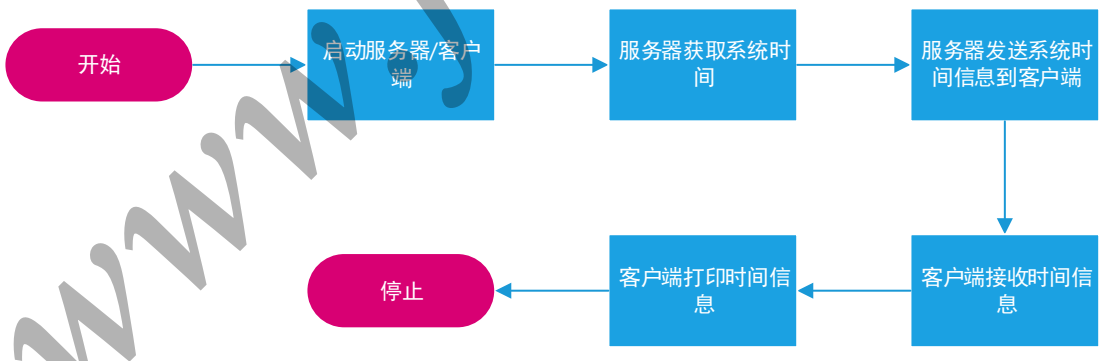
1、模块结构：

	调用函数	说明
1	int main(void)	主函数，负责通信
2	void getNowTime(char *)	获取服务器时间
3	void log_fun(FILE *, char *, char*)	日志

2、数据结构：

```
#define SERV_PORT 8080  //侦听端口地址
#define SERV_HOST "127.0.0.1"
```

3、总体流程：



4、关键算法：

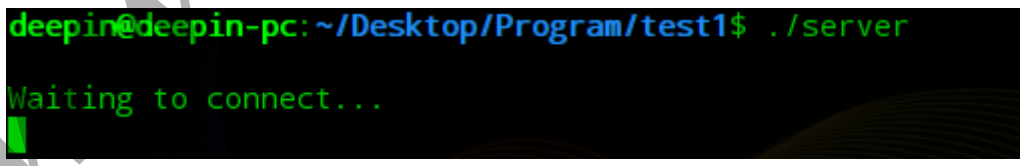
服务器获取系统时间

```
void getNowTime(char *nowTime)
{
    memset(nowTime, 0, sizeof(nowTime));
    char acYear[5] = { 0 };
    char acMonth[5] = { 0 };
    char acDay[5] = { 0 };
```

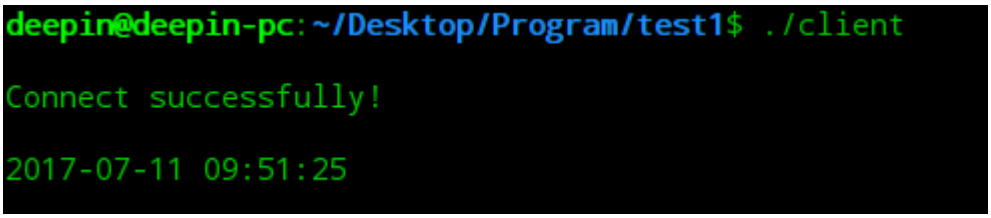
```
    char acHour[5] = { 0 };
    char acMin[5] = { 0 };
    char acSec[5] = { 0 };

    time(&now);
    timenow = localtime(&now);

    strftime(acYear, sizeof(acYear), "%Y", timenow);
    strftime(acMonth, sizeof(acMonth), "%m", timenow);
    strftime(acDay, sizeof(acDay), "%d", timenow);
    strftime(acHour, sizeof(acHour), "%H", timenow);
    strftime(acMin, sizeof(acMin), "%M", timenow);
    strftime(acSec, sizeof(acSec), "%S", timenow);

    strncat(nowTime, acYear, 4);
    strcat(nowTime, "-");
    strncat(nowTime, acMonth, 2);
    strcat(nowTime, "-");
    strncat(nowTime, acDay, 2);
    strcat(nowTime, " ");
    strncat(nowTime, acHour, 2);
    strcat(nowTime, ":");
    strncat(nowTime, acMin, 2);
    strcat(nowTime, ":");
    strncat(nowTime, acSec, 2);
    strcat(nowTime, "\t\t");

    return;
}
```

四、调试记录

1.启动服务器



2.启动客户端，成功接收时间信息并打印



3.服务器提示信息

```
deepin@deepin-pc:~/Desktop/Program/test1$ ./server

Waiting to connect...
Connect successfully!

Client IP:127.0.0.1, Client Port:37592

Sending time information to client successfully!
```

五、创新说明

增加服务端日志记录

```
1   2017-07-11 09:51:25     Connect successfully!
2   2017-07-11 09:51:25     Client IP:127.0.0.1, Client Port:37592
3   2017-07-11 09:51:25     Sending time information successfully!
4   2017-07-11 09:54:16     Connect successfully!
5   2017-07-11 09:54:16     Client IP:127.0.0.1, Client Port:37594
6   2017-07-11 09:54:16     Sending time information successfully!
```

实验 2 基于 Socket 的远程文件备份服务器

一、程序实践概述

1、题目名称：

Socket 远程文件备份服务器

2、时间进度：

2017.06.21 - 2017.06.22

3、开发环境：

Linux Deepin 15.4

二、问题分析

1、功能说明：

a）客户端指定一个文件并将文件名发送给服务端， 通过 socket 将文件发送给服务端。

b）服务端创建相应的文件， 并将服务端发送过来的文件保存。

2、解决方案：

a）客户端与服务器建立连接。

b）客户端把需要发送的文件名发送给服务端。

c）服务端根据文件名创建相应的可写文件。

d）客户端将文件通过套接字发送给服务器。

e）服务器保存文件。

三、方案设计

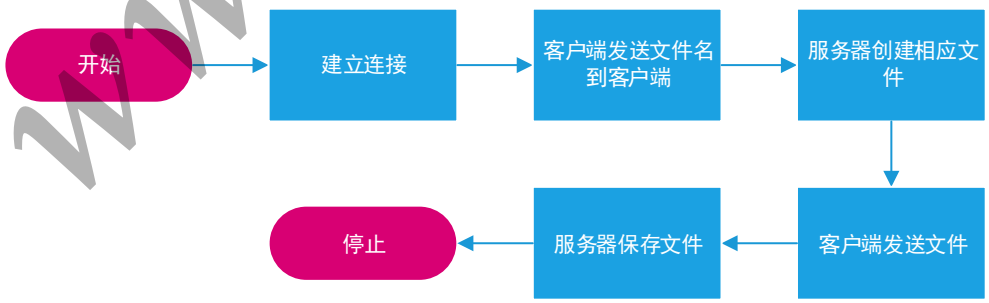
1、模块结构：

	调用函数	说明
1	int main(void)	主函数，负责通信
2	void get_time(char *)	获取当前时间字符串
3	void log_fun(FILE *,char *,char*)	日志

2、数据结构：

```
#define SERV_PORT 8080
#define SERV_HOST "127.0.0.1"
```

3、 总体流程：



4、关键算法：

服务端：

```
//从客户端接收数据到buf中
//每接收一段数据，便将其写入文件，循环直到文件接收完为止
bzero(buf, BUFSIZ);
```

```
len = 0;
while ((len = recv(cfd, buf, sizeof(buf), 0)) > 0) {
    if (fwrite(buf, sizeof(char), len, fp) < len) {
        fprintf(stderr, "ERROR: Write to file failed!\n\n");
        sprintf(temp, "ERROR: Write to file failed!\n");
        log_fun(log_fp, time_log, temp);
        exit(1);
    }
    bzero(buf, sizeof(buf));
    }
```

客户端：

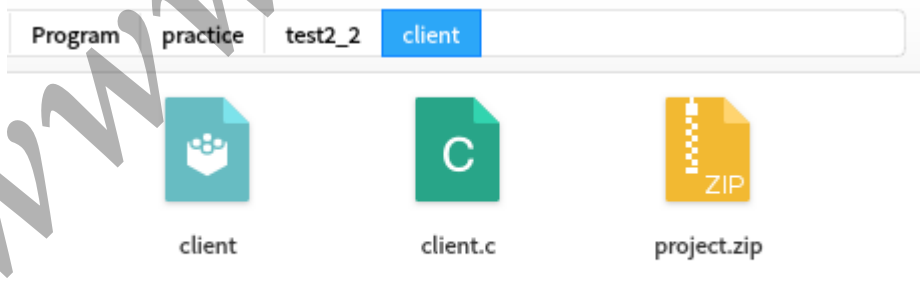
```
//发送数据到服务器
//每读取一段数据，便将其发送给服务器，直到文件读完为止
bzero(buf, BUFSIZ);
len = 0;
while ((len = fread(buf, sizeof(char), sizeof(buf), fp)) > 0) {
    if (send(cfd, buf, len, 0) < 0) {
        printf("ERROR: Send file: %s failed!\n", file_name);
        exit(1);
    }
    bzero(buf, BUFSIZ);
    }
```

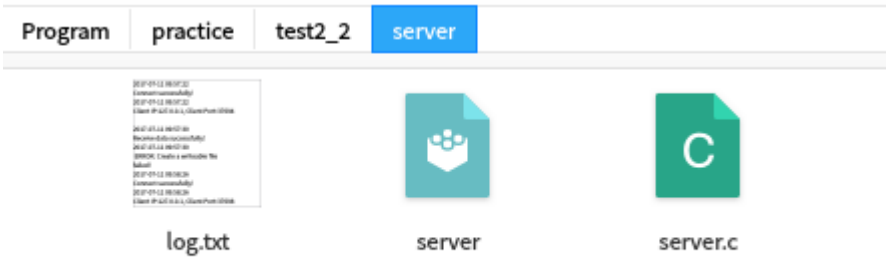
5、界面设计：

默认界面

四、调试记录

1.确定发送文件：客户端发送 project.zip 文件到服务端





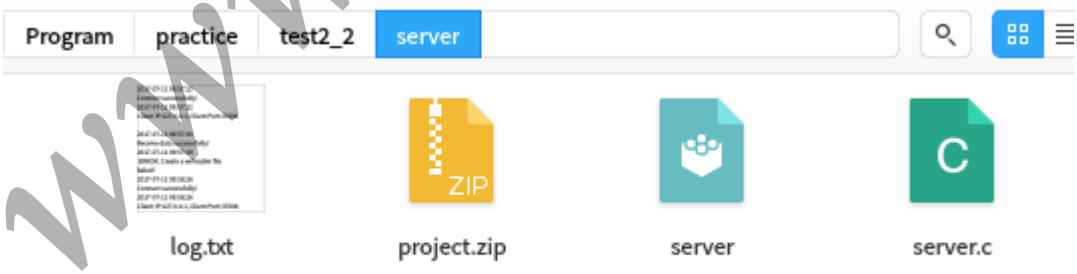
2. 启动服务器和客户端，客户端输入将要发送的文件

```
deepin@deepin-pc:~/Documents/Program/practice/test2_2/client$ ./client

登录时间： 2017年07月11日 10:00:16

Connect successfully!

Please input the file name:  project.zip

Sending data...
File: project.zip transfer successfully!
```

3. 服务器提示信息

```
deepin@deepin-pc:~/Documents/Program/practice/test2_2/server$ ./server

Waiting to connect...
Connect successfully!

Client IP:127.0.0.1, Client Port:37700

Waiting to receive data...
Receive data successfully!
Waiting to download project.zip...
Download project.zip from the client...
Receive project.zip from client successfully!
```

4. 文件已接收



五、创新说明

(1)错误校验并提示

```
deepin@deepin-pc:~/Documents/Program/practice/test2_2/client$ ./client

登录时间： 2017年07月11日  10:01:18

Connect successfully!

Please input the file name:  test
ERROR: File test is not found
```

```
deepin@deepin-pc:~/Documents/Program/practice/test2_2/server$ ./server

Waiting to connect...
Connect successfully!

Client IP:127.0.0.1, Client Port:37702

Waiting to receive data...
Receive data successfully!
Waiting to download ...
ERROR: Create a writeable file failed!
```

(2)服务端日志记录

```
 1  2017-07-11 09:57:22     Connect successfully!
 2  2017-07-11 09:57:22     Client IP:127.0.0.1, Client Port:37694
 3  2017-07-11 09:57:30     Receive data successfully!
 4  2017-07-11 09:57:30     ERROR: Create a writeable file failed!
 5  2017-07-11 09:58:26     Connect successfully!
 6  2017-07-11 09:58:26     Client IP:127.0.0.1, Client Port:37698
 7  2017-07-11 10:00:16     Connect successfully!
 8  2017-07-11 10:00:16     Client IP:127.0.0.1, Client Port:37700
 9  2017-07-11 10:00:22     Receive data successfully!
10  2017-07-11 10:00:22     Download files from the client...
11  2017-07-11 10:00:22     Receive files successfully!
12  2017-07-11 10:01:18     Connect successfully!
13  2017-07-11 10:01:18     Client IP:127.0.0.1, Client Port:37702
14  2017-07-11 10:01:53     Receive data successfully!
15  2017-07-11 10:01:53     ERROR: Create a writeable file failed!
```

实验 3 基于 DES 加密的 TCP 聊天程序/基于 RSA 算法自动分配密钥的加密聊天程序

一、程序实践概述

1、题目名称：

(1)基于 DES 加密的 TCP 聊天程序

(2)基于 RSA 算法自动分配密钥的加密聊天程序

2、时间进度：

2017.06.26 - 2017.06.29

3、开发环境：

Linux Deepin 15.4

二、问题分析

1、功能说明：

实现加密聊天

2、解决方案：

(1)将套接字设置为非阻塞状态

(2)分组加密数据填充预处理

(3)非对称加密公私密钥管理

三、方案设计

1、模块结构：

(1)DES

	调用函数	说明
1	int main(void)	主函数,负责通信
2	void log_fun(FILE*,char*, char*)	日志
3	void get_time(char *)	获取当前系统时间
4	void my_decryption(unsigned char *, unsigned char *)	解密
5	void my_encryption(unsigned char *, unsigned char *)	加密

(2)RSA

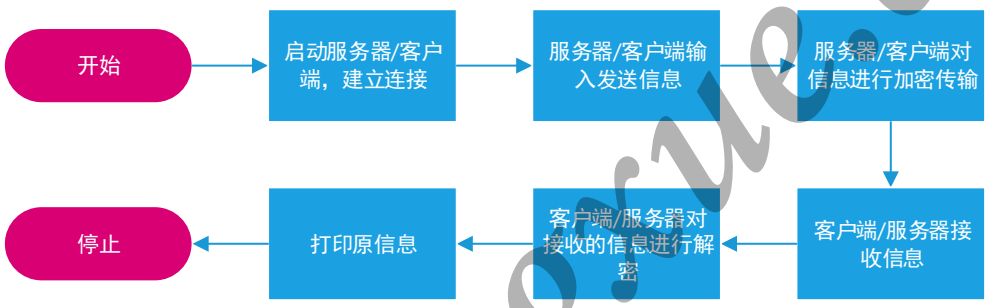
	调用函数	说明
1	int main(void)	主函数，负责通信
2	int private_decrypt(unsigned char *, int, unsigned char *)	使用私钥对数据进行解密
3	int public_encrypt(unsigned char *, int, unsigned char *)	使用公钥对数据进行加密
4	RSA * createRSA(unsigned char *key, int public)	通过密钥字符串生成 RSA*
5	RSA * createRSAWithFile(char *, int public)	通过密钥文件生成 RSA*

2、数据结构：

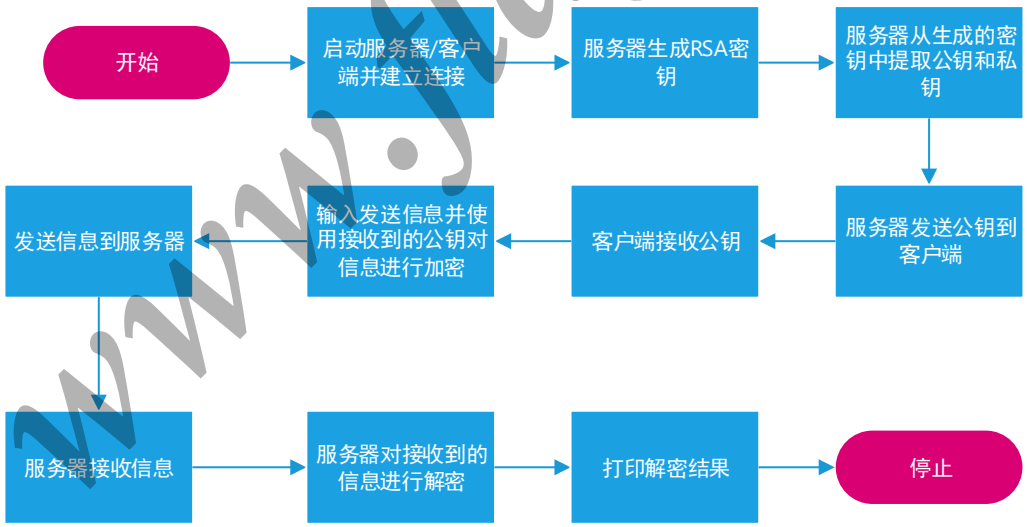
```
#define SERV_PORT 8080
#define SERV_HOST "127.0.0.1"
typedef struct encry {
  int len;
  unsigned char encrypted[4098];
} trans;

//填充方式
int padding = RSA_PKCS1_PADDING; //RSA_NO_PADDING
char *pubkey = "public.pem";
char *prikey = "private.pem";
```

3、总体流程：

(1)DES



(2)RSA



4、关键算法：

(1)DES

加密函数:

```
void my_encryption(unsigned char *data, unsigned char *encryption)
{
```

```
//定义openssl的密钥实体
DES_cblock key;
//定义轮密钥结构
DES_key_schedule key_schedule;

//根据指定字符串生成密钥
DES_string_to_key(keystring, &key);
//生成密钥轮结构
DES_set_key_unchecked(&key, &key_schedule);

unsigned char *src;
unsigned char *test;

//对原始明文进行处理
//判断是否需要填充
int data_len = strlen(data);
//printf("data_len:%d\n", data_len);

int data_rest = data_len % 8;
int len;
//printf("data_rest:%d\n", data_rest);
if (data_rest != 0)
{
    len = data_len + (8 - data_rest);
}
else
{
    len = data_len;
}

//printf("len:%d\n", len);

unsigned char tmp[8];
unsigned char in[8];

int count, i;//计数变量;

src = (unsigned char *)malloc(len);
test = (unsigned char *)malloc(len);

//printf("%d\n", len);

if (src == NULL)
```

```
    {
        fprintf(stderr, "malloc failed\n");
        exit(1);
    }
    else
    {
        //填充初始明文
        if (data_rest != 0)
        {
            memset(src, 0, len);
            memcpy(src, data, data_len);
            memset(src + data_len, 0, 8 - data_rest);
        }
        else
        {
            memset(src, 0, len);
            memcpy(src, data, data_len);
        }

        //根据明文长度，进行分组加密
        count = len / 8;
        for (i = 0; i < count; i++)
        {
            memset(tmp, 0, 8);
            memset(in, 0, 8);
            memcpy(tmp, src + (8 * i), 8);

            //加密
            DES_ecb_encrypt((const_DES_cblock*)tmp, (DES_cblock*)in, &key_schedule,
DES_ENCRYPT);
            memcpy(test + (8 * i), in, 8);
        }
    }

    memcpy(encryption, test, len);

}
```

解密函数：

```
void my_decryption(unsigned char *encryption, unsigned char *dst)
{

    DES_cblock key;
    DES_key_schedule key_schedule;
```

```
    DES_string_to_key(keystring, &key);
    DES_set_key_unchecked(&key, &key_schedule);

    int len;
    len = strlen(encryption);
    unsigned char tmp[8];
    unsigned char in[8];
    int count, i;//计数变量
    unsigned char *test = NULL;
    test = (unsigned char *)malloc(len);

    if (test == NULL)
    {
        fprintf(stderr, "malloc fail\n");
        exit(1);
    }
    else
    {
        //分组解密
        memset(test, 0, len);
        count = len / 8;
        for (i = 0; i < count; i++)
        {
            memset(tmp, 0, 8);
            memset(in, 0, 8);
            memcpy(tmp, encryption + (8 * i), 8);

            DES_ecb_encrypt((const_DES_cblock*)tmp, (DES_cblock*)in, &key_schedule,
DES_DECRYPT);
            memcpy(test + (8 * i), in, 8);
        }
    }

    memcpy(dst, test, len);

}
```

(2)RSA

通过密钥文件生成RSA*

```
RSA * createRSAWithFile(char *filename, int public)
{
    FILE *fp = fopen(filename, "rb");
```

```
    if (fp == NULL)
    {
        perror("open key file Error");
        return NULL;
    }
    RSA *rsa = NULL;

    if (public)
    {
        rsa = PEM_read_RSA_PUBKEY(fp, &rsa, NULL, NULL);
    }
    else
    {
        rsa = PEM_read_RSAPrivateKey(fp, &rsa, NULL, NULL);
    }

    return rsa;
}
```

通过密钥字符串生成RSA*

```
RSA * createRSA(unsigned char *key, int public)
{
    //密钥实体
    RSA *rsa = NULL;
    //大数串
    BIO *keybio;
    keybio = BIO_new_mem_buf(key, -1);

    if (keybio == NULL)
    {
        printf("Failed to create key BIO");
        return 0;
    }

    if (public)
    {
        rsa = PEM_read_bio_RSA_PUBKEY(keybio, &rsa, NULL, NULL);
    }
    else
    {
        rsa = PEM_read_bio_RSAPrivateKey(keybio, &rsa, NULL, NULL);
    }
```

```
    if (rsa == NULL)
    {
        printf("Failed to create RSA");
    }

    return rsa;

}
```

5、界面设计：

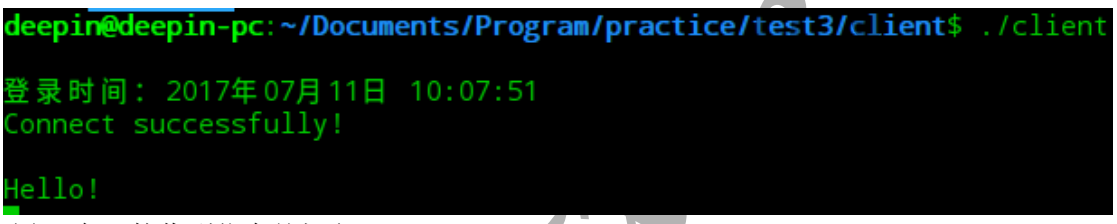
默认界面

四、调试记录

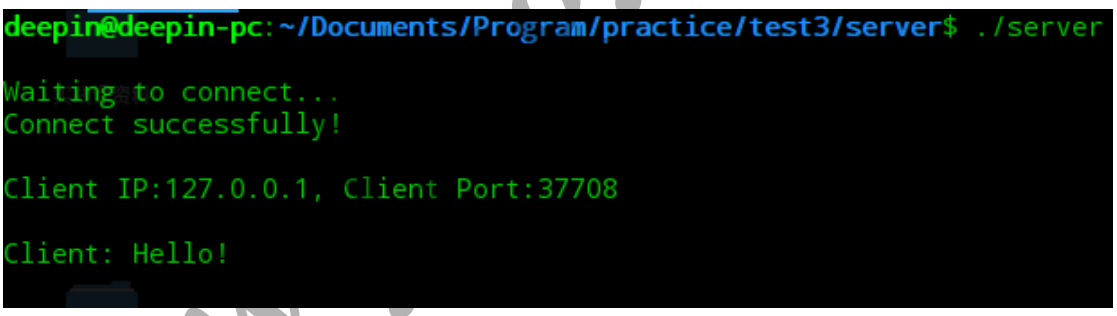
1.DES

通信示例

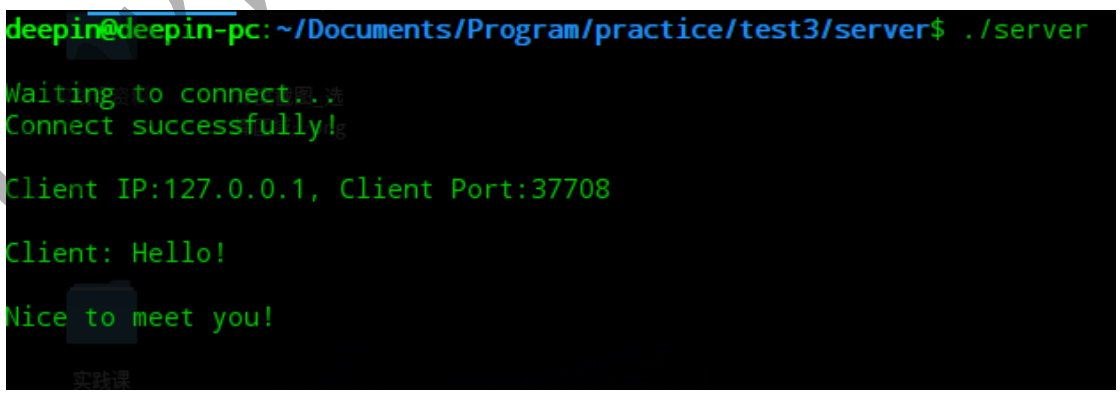
(1)启动服务器/客户端，客户端向服务器发送”Hello!”



(2)服务器接收到信息并打印



(3)服务器向客户端发送”Nice to meet you!”



(4)客户端接收信息并打印

```
deepin@deepin-pc:~/Documents/Program/practice/test3/client$ ./client

登录时间： 2017年07月11日  10:07:51
Connect successfully!

Hello!
Server: Nice to meet you!
```

加密/解密细节

```
deepin@deepin-pc:~/Documents/Program/practice/test3_2$ ./client

登录时间： 2017年07月11日  10:13:20
Connect successfully!

Hello!

Befor Encrypt:Hello!
After Encrypt:0x04 0x36 0xf2 0xd7 0xb3 0x3e 0xfa 0x4c
Send infomation successfully!

Receive information: ♦OWM♦
Waiting to decryption...

Server: Nice!
```

```
deepin@deepin-pc:~/Documents/Program/practice/test3_2$ ./server

Waiting to connect...
Connect successfully!

Client IP:127.0.0.1, Client Port:37712

Receive information: ♦♦'>♦L
Waiting to decryption...

Client: Hello!


Nice!

Befor Encrypt:Nice!
After Encrypt:0xdf 0x4f 0x57 0x4d 0xa5 0x16 0x05 0x50
Send infomation successfully!
```

2.RSA：客户端向服务端发送加密后的信息，服务端接受加密信息并解密，打印输出。

```
deepin@deepin-pc:~/Documents/Program/practice/test4_3/client$ ./client

Connecting successfully!
Waiting to receive data...
Receive data successfully!
Waiting to download public.pem...
Receive public.pem from server successfully!


Please input messages that you want to send...
Hello!
Befor Encrypt: Hello!

Length of encrypted: 256
String Length of encrypted: 127
After Encrypt:
 0x61 0xD6 0x16 0x18 0x94 0x93 0xAE 0x4E 0x91 0x98 0x17 0x44 0x46 0xB6
 0x17 0x2D 0xFA 0x78 0x92 0xCD 0x6D 0x29 0x4C 0x20 0x8D 0xEF 0x48 0x9A
 0x0E 0xF2 0xE6 0x55 0xDF 0xD1 0x2C 0x88 0x67 0x24 0xDC 0x24 0xDC 0xDD
 0xFE 0xC5 0x3B 0x81 0xDF 0xF4 0x9D 0x6A 0xDB 0x7D 0x52 0xF4 0xF4 0x2E
 0x72 0x51 0x92 0xD7 0x18 0x50 0x1F 0x24 0x7C 0x9F 0x99 0x8C 0x19 0xA7
 0xE6 0xDF 0xA1 0x3A 0x6C 0xB4 0x61 0x92 0x96 0x15 0xE4 0xF3 0xD5 0x78
 0x02 0x58 0x5D 0x0D 0x87 0x8F 0x6D 0xA4 0x07 0x95 0x35 0x11 0x5F 0xFD
 0x87 0x92 0x9D 0x6B 0x5F 0x34 0x10 0xDD 0x89 0x64 0x95 0x59 0xC4 0xB1
 0xE1 0x47 0x64 0xDD 0x56 0xAF 0xC9 0xB4 0x0C 0x19 0x73 0x30 0x08 0xF4
 0xF1

Sending encryption to server...
Sending data successfully!


Please input messages that you want to send...
```

```
deepin@deepin-pc:~/Documents/Program/practice/test4_3/server$ ./server

Waiting to connect...
Connect successfully!

Client IP:127.0.0.1, Client Port:37718

Sending data...
File: public.pem transfer successfully!

Waiting to receive data from client...
Receiving data successfully!

Length of decrypted: 7
After decrypt: Hello!

Waiting to receive data from client...
```

五、创新说明

1. 日志记录
2. 显示登陆时间和通信时间
3. 实现消息互传
4. 实现加密传输

实验 4 基于 Libpcap 的网络嗅探程序

一、程序实践概述

1、题目名称：

基于 Libpcap 的网络嗅探程序

2、时间进度：

2017.07.03 - 2017.07.06

3、开发环境：

Linux Deepin 15.4

二、问题分析

1、功能说明：

(1)实现 icmp,udp,tcp 包的判断

(2)打印出源地址:端口 目的地址:端口

(3)实现过滤指定包，如 icmp

2、解决方案：

(1) 网络设备查找

(2) 打开网络设备

(3) 获取网络参数

(4) 编译过滤策略

(5) 设置过滤器

(6) 利用回调函数捕获数据包

(7) 关闭网络设备

三、方案设计

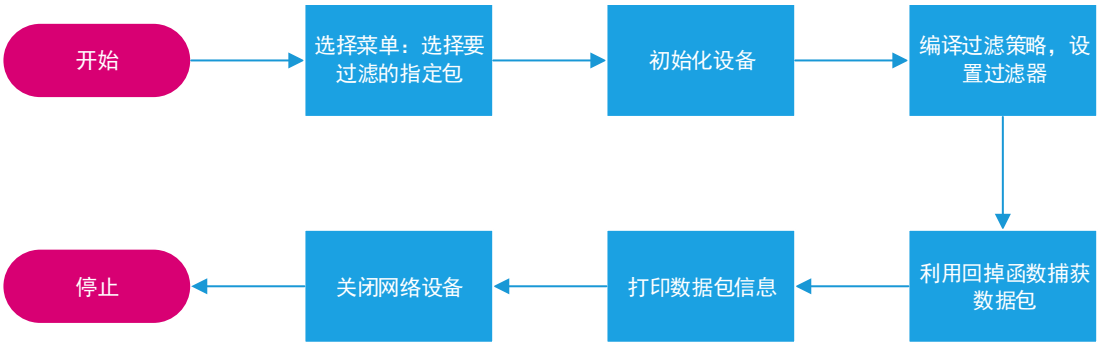
1、模块结构：

	调用函数	说明
1	void main()	主函数，选择菜单
2	void display()	打印界面及系统时间
3	pcap_t* pcap_socket(char *, const char *)	初始化
4	void capture_loop(pcap_t *, int , pcap_handler)	获取数据链路类型并设置首部长度
5	void showIPInfor(u_char *, struct pcap_pkthdr *)	打印 IP 地址信息
6	void showPacketInfor(struct ip*, u_char *, char *, char*, char *)	打印获取到的数据包信息
7	void get_packet(u_char *, struct pcap_pkthdr *, u_char *)	抓包
8	void output()	输出提示信息

2、数据结构：

```
pcap_t* p;
pcap_t* pcap_socket(char*, const char*);
```

3、总体流程：



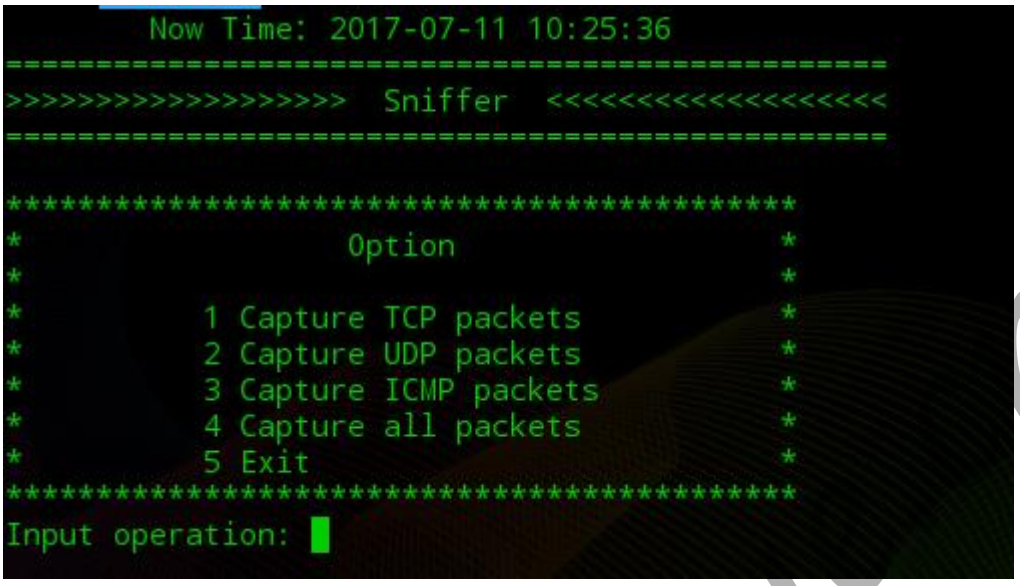
4、关键算法：

获取数据包

```
void get_packet(u_char *arg, struct pcap_pkthdr *pkthdr, u_char *packet) {
    int i;
    struct ip* iphdr;
    char iphdrInfo[256];
    char srcip[256];
    char dstip[256];

    //跳过头部长度，抓取IP数据包
    packet += linklen;
    iphdr = (struct ip*)packet;
    strcpy(srcip, inet_ntoa(iphdr->ip_src));
    strcpy(dstip, inet_ntoa(iphdr->ip_dst));
    sprintf(iphdrInfo, "ID:%d  TOS:0x%x  TTL:%d  IpLen:%d  DgLen:%d",
        ntohs(iphdr->ip_id), iphdr->ip_tos, iphdr->ip_ttl,
        4 * iphdr->ip_hl, ntohs(iphdr->ip_len));
    packet += 4 * iphdr->ip_hl;

    showIPInfor(arg, pkthdr);


    for (i = 0; i < pkthdr->len; ++i) {
        printf(" %02x", packet[i]);
        if ((i + 1) % 16 == 0) {
            printf("\n");
        }
    }

    printf("\n\n");
    setbuf(stdin, NULL);
```

```
    showPacketInfor(iphdr, packet, srcip, dstip, iphdrInfo);
}
```

5、界面设计：



四、调试记录(见下页)

1. 抓取 TCP 包

```
Input operation: 1
Success! Device: wlp6s0
IP: 172.28.0.0
Sub Mask: 255.255.0.0

id: 1
Packet length: 94
Number of bytes: 94
Recieved time: Tue Jul 11 10:29:29 2017

 60 02 a2 27 00 28 06 40 20 01 0d a8 90 00 a8 11
 02 1d e0 ff fe 13 dd 3d 24 04 68 00 40 08 0c 07
 00 00 00 00 00 00 00 bc a0 fc 14 6c be af 1a f7
 00 00 00 00 a0 02 6f 90 48 ec 00 00 02 04 05 a0
 04 02 08 0a 00 0a 18 fb 7f 74 6b 15 01 03 03 07
 00 00 00 00 00 00 00 00 00 00 00 00 00 00

ICMP 144.0.168.17 -> 2.29.224.255
ID:40  TOS:0x2  TTL:32  IpLen:0  DgLen:41511
Type:96 Code:2 ID:40 Seq:1600
------------------------------------------------------------------

id: 2
Packet length: 74
Number of bytes: 74
Recieved time: Tue Jul 11 10:29:29 2017

 d8 50 01 bb d8 e7 2a 97 00 00 00 00 a0 02 72 10
 5c 64 00 00 02 04 05 b4 04 02 08 0a 00 0a 18 ff
 00 00 00 00 01 03 03 07 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00

TCP  172.28.216.128:55376 -> 64.233.189.113:443
ID:12904  TOS:0x0  TTL:64  IpLen:20  DgLen:60
****S* Seq: 0xd8e72a97 Ack: 0x0 Win: 0x7210 TcpLen: 40
------------------------------------------------------------------
```

2.抓取 UDP 包

```
Input operation: 2
Success! Device: wlp6s0
IP: 172.28.0.0
Sub Mask: 255.255.0.0

id: 1
Packet length: 75
Number of bytes: 75
Recieved time: Tue Jul 11 10:28:27 2017

 e2 b4 00 35 00 29 b9 d6 06 97 01 00 00 01 00 00
 00 00 00 00 04 61 70 70 73 03 6e 65 75 03 65 64
 75 02 63 6e 00 00 01 00 01 00 00 00 00 00 00 00
 a8 00 00 00 cb 37 64 59 43 f0 a0 0f 4b 00 00 00
 4b 00 00 00 01 00 00 00 56 00 64

UDP  172.28.216.128:58036 -> 202.118.1.29:53
ID:4838  TOS:0x0  TTL:64  IpLen:20  DgLen:61
------------------------------------------------------------------

id: 2
Packet length: 75
Number of bytes: 75
Recieved time: Tue Jul 11 10:28:27 2017

 a3 22 00 35 00 29 40 82 a4 7d 01 00 00 01 00 00
 00 00 00 00 04 61 70 70 73 03 6e 65 75 03 65 64
 75 02 63 6e 00 00 1c 00 01 00 00 00 00 00 00 00
 a8 00 00 00 cb 37 64 59 cb da a2 0f 4b 00 00 00
 4b 00 00 00 01 00 00 00 56 00 64

UDP  172.28.216.128:41762 -> 202.118.1.29:53
ID:4839  TOS:0x0  TTL:64  IpLen:20  DgLen:61
------------------------------------------------------------------
```

3.抓取 ICMP 包

(1)打印设备信息，等待抓取 ICMP 包

```
Input operation: 3
Success! Device: wlp6s0
IP: 172.28.0.0
Sub Mask: 255.255.0.0
```

(2)使用 ping 命令

```
deepin@deepin-pc:~/Desktop$ ping nextlegend.cn
PING nextlegend.cn (118.89.236.39) 56(84) bytes of data.
```

(3)打印抓取的 ICMP 包信息

```
Input operation: 3
Success! Device: wlp6s0
IP: 172.28.0.0
Sub Mask: 255.255.0.0

id: 1
Packet length: 98
Number of bytes: 98
Recieved time: Tue Jul 11 10:31:14 2017

 08 00 1f ef 1d 6c 00 01 72 38 64 59 00 00 00 00
 19 3f 0c 00 00 00 00 00 10 11 12 13 14 15 16 17
 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27
 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00

ICMP 172.28.216.128 -> 118.89.236.39
ID:20051  TOS:0x0  TTL:64  IpLen:20  DgLen:84
Type:8 Code:0 ID:7532 Seq:1
------------------------------------------------------------------------
```

4. 抓取所有类型的包

```
Success! Device: wlp6s0
IP: 172.28.0.0
Sub Mask: 255.255.0.0

id: 1
Packet length: 86
Number of bytes: 86
Recieved time: Tue Jul 11 10:26:21 2017

 60 00 00 00 00 20 3a ff fe 80 00 00 00 00 00 00
 6e c2 17 ff fe 41 f1 84 ff 02 00 00 00 00 00 00
 00 00 00 01 ff b6 96 2a 87 00 2f a0 00 00 00 00
 20 01 0d a8 90 00 a8 11 b4 0e a4 e1 73 b6 96 2a
 01 01 6c c2 17 41 f1 84 00 00 00 00 00 00 00 00
 00 00 00 00 00 00

------------------------------------------------------------------------

id: 2
Packet length: 86
Number of bytes: 86
Recieved time: Tue Jul 11 10:26:22 2017

 60 00 00 00 00 20 3a ff fe 80 00 00 00 00 00 00
 6e c2 17 ff fe 41 f1 84 ff 02 00 00 00 00 00 00
 00 00 00 01 ff f1 37 5f 87 00 d0 fa 00 00 00 00
 20 01 0d a8 90 00 a8 11 3c 26 88 90 23 f1 37 5f
 01 01 6c c2 17 41 f1 84 00 00 00 00 00 00 00 00
 00 00 00 00 00 00

------------------------------------------------------------------------
```

5.输出提示信息

```
^CReceived: 72 packet(s)
Dropped 0 packet(s)
```

五、创新说明

1.可自主选择需要抓取的指定包

2.可打印通信双方的 IP 地址信息

3.可获取通信双方的子网掩码

考核标准	得分
（1）正确理解和掌握实验所涉及的概念和原理（20%）；	
（2）按实验要求合理设计数据结构和程序结构（20%）；	
（3）运行结果正确（20%）；	
（4）认真记录实验数据，原理及实验结果分析准确（20%）；	
(5)实验过程中,具有严谨的学习态度和认真、踏实、一丝不苟的科学作风(10%);	
（7）实验报告规范（10%）。	
合计	