计算机网络协议开发实验报告

201220064　吴浩然

一、实验目的

理解协议的逆向分析方法并掌握客户端套接字编程。

二、实验内容

对抓包文件的分析得出：客户端发送的数据包大小为33字节，第一个字节说明该包查询城市还是查询天气，第二个字节说明该包查询天气的类型，之后30个字节存储城市名，最后一字节代表查询的日期，若查询天气的类型为0x02，则代表查询未来对应天数。

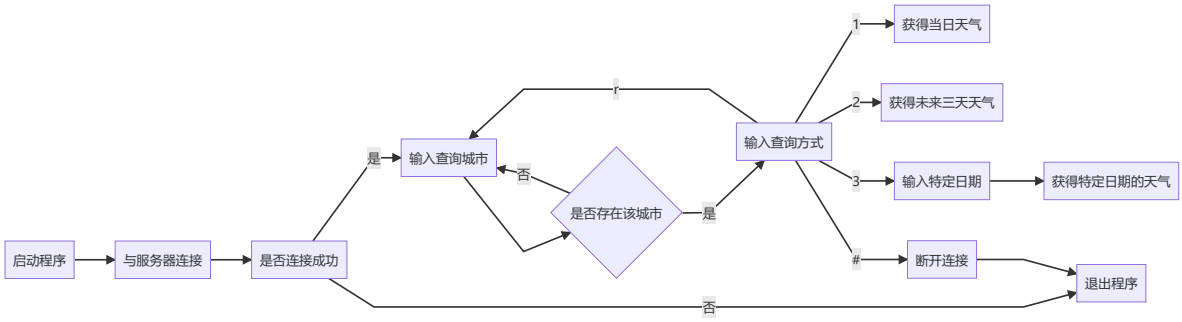
服务器端发送的数据包大小为127字节，第一个字节指示该包返回城市、天气、查询城市失败或查询天气失败，第二个字节在查询天气成功时代表返回的是一天或多天天气，之后30字节为城市名，后跟2字节的年份、1字节的月份和具体日期，之后的1字节在查询一天天气时代表具体哪一天，查询多天天气时代表查询的天数，最后的90字节，每两个字节分别代表了天气和温度。

使用结构体表示如下，其中weather使用1个字节的枚举类型来表示天气，condition结构体包含1个字节的天气和1个字节的温度，使用结构体函数来对数据包字段的读取进行简化：

```
struct sendpkt {
    uint8_t stype;
    uint8_t qtype;
    char cname[30];
    uint8_t number;
}

enum weather : uint8_t
{
    overcast = 0x00,
    sunny,
    cloudy,
    rain,
    fog,
    rainstorm,
    thunderstorm,
    breeze,
    sand_st0rm
};

struct condition
{
    weather wea;
    uint8_t temp;
};

struct recvpkt {
    uint8_t field1;
    uint8_t field2;
    char cname[30];
```

```
    uint16_t year;
    uint8_t mon;
    uint8_t day;
    uint8_t num;
    condition con[45];

    recvpkt() { memset(this, 0, sizeof(*this)); }
    bool isCitypkt() { return field1 == 0x01; }
    bool isWorngpkt() { return field1 == 0x02; }
    bool isWeapkt() { return field1 == 0x03; }
    bool isNoinfopkt() { return field1 == 0x04; }
    char getType() { return (char)field2; }
    int getYear() { return ntohs(year); }
    int getMon() { return mon; }
    int getDay() { return day; }
};
```

程序的简要设计流程图如下：



因为需要同时监视标准输入和套接字输入，所以使用select系统调用进行阻塞，使用FD_ISSET函数来对套接字的输入和标准输入进行分别处理。在设计查询流程时，因为整个流程包含在一个大循环中，所以设置整型变量menulevel来指示到达流程的哪一步骤，使用布尔变量menushow来在每个状态指示是否需要显示菜单。显示菜单的函数包含在每次循环的开始。如下所示：

```
void showMenu(int lv, bool need) {
    if(!need) return;
    if(lv == 1) showMainMenu();
    else if(lv == 2) showSubMenu();
    ...
}

int main(int argc, char** argv) {
```

```
    int menulevel = 1;
    bool menushow = true;
    ...
    while (1) {
        showMenu(menulevel, menushow);
        int rc = select(sockfd + 1, &recvmask, NULL, NULL, NULL);
        ...
    }
}
```

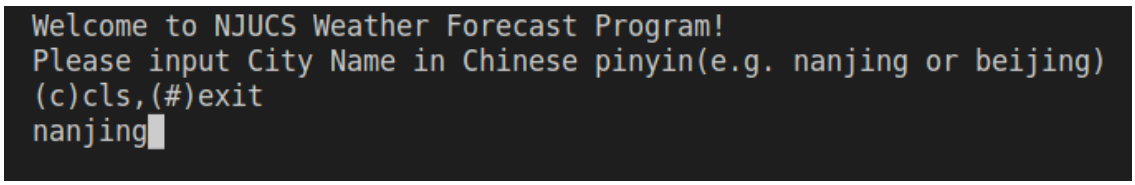
在进行标准输入后，对输入进行判断，分别处理，逻辑如下所示：

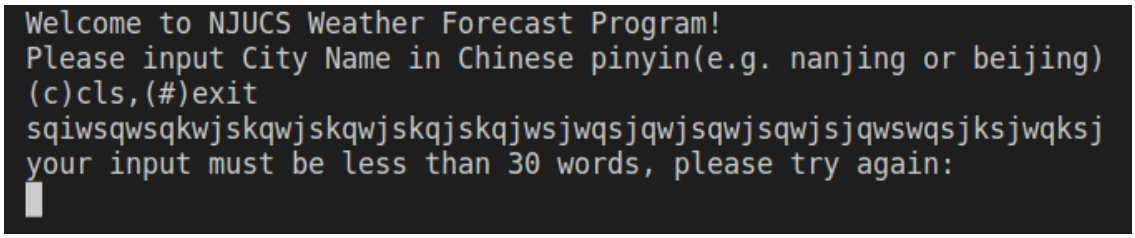
```
if(FD_ISSET(0,&recvmask) && wgets(query,MAXLINE) != NULL) {
   if(strncmp(query,"#",1) == 0) {
       ...//退出
   }
   else if(strncmp(query,"c",1) == 0) {
       ...//清屏
   }
   else if(menulevel == 1)
   {
       ...//查询城市
       sendpkt pcity = sendpkt(0x01, 0x00, query, 0x00);
       int sdsuc = send(sockfd, &pcity, sizeof(pcity), 0);
       ...
   }
   else if(menulevel == 2 && strncmp(query,"r",1) == 0)
   {
       ...//返回到menulevel为1
   }
   else if(menulevel == 2 && strncmp(query,"1",1) == 0)
   {
       ...//查询当日
       int sdsuc = send(sockfd, &ptoday, sizeof(ptoday), 0);
   }
   else if(menulevel == 2 && strncmp(query,"2",1) == 0)
   {
       ...//查询未来3天
       int sdsuc = send(sockfd, &ptoday, sizeof(ptoday), 0);
   }
   else if(menulevel == 2 && strncmp(query,"3",1) == 0)
   {
       ...//查询特定日期
   }
   ...//其他情况及错误处理
```

对接受到的数据包的处理为根据包的类型输出相应的内容。

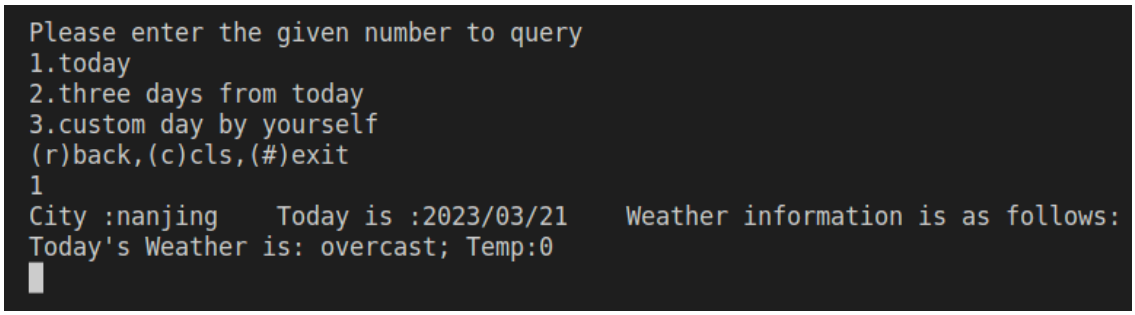
三、实验结果

1. 实现了查询是否有输入城市天气的功能，根据包的大小限制，已对输入字符串的长度进行检查。

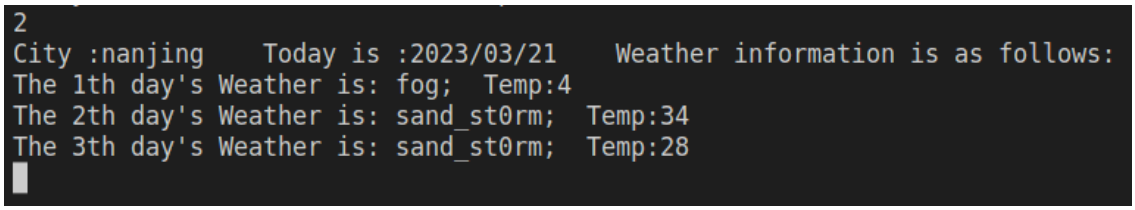




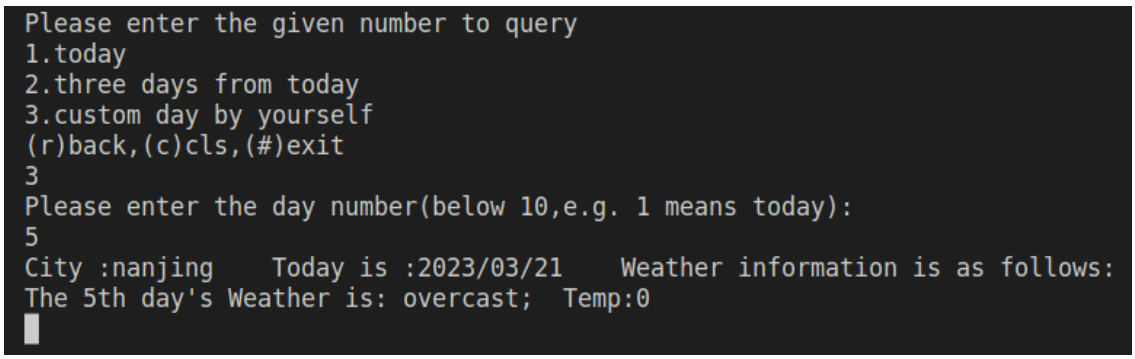
2. 实现了查询当天天气的功能。

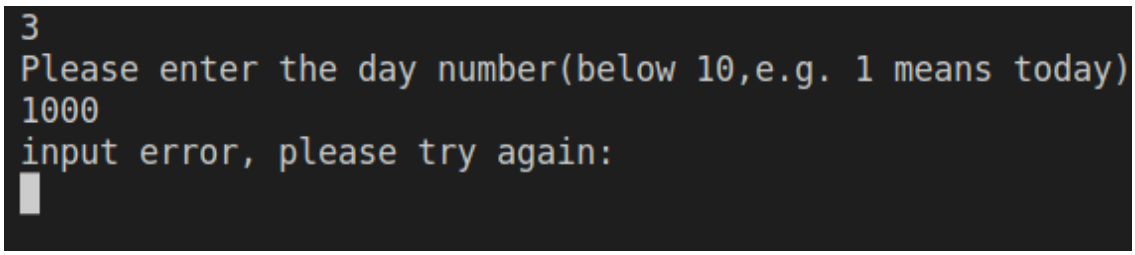


3. 实现了查询未来3天天气的功能



4. 实现了查询特定日期天气的功能，对错误的日期范围及其他输入进行了检查





5. 当查询不到城市或天气时，给出文字反馈。

6. 对错误的输入进行了检查。

```
Please enter the given number to query
1.today
2.three days from today
3.custom day by yourself
(r)back,(c)cls,(#)exit
ddqwdwqd
input error, please try again:
```

四、flag

1. 查询天气时遇到如下结果：

```
1
City: nanjing  Today is: 2023/03/21  Weather information is as follows:
Today's Weather is: Please guess: ZmxhZ3tzYW5kX3N0MHJtfQ==;  Temp:34
```

观察形式，立即想到可能是base编码，使用Base64编码解码可得：

ZmxhZ3tzYW5kX3N0MHJtfQ==

清空 加密 解密 解密为UTF-8字节流

flag{sand_st0rm}

flag为sand_st0rm，为第九种天气。

2. 观察服务器发送的数据包，可得如下形式数据：

```
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 2e 2e 2d  ........ .......-
0090  2e 20 2e 2e 20 2d 2e 20  2d 2e 2e 20 2d 2d 20 2e  . .. -. -.. -- .
```

观察形式，可能为摩尔斯电码，解码可得：

..-. .. -. -.. -- .

分割 长 -

编码 解码

FINDME

flag为FINDME

五、实验中遇到的问题及解决方案

1. 使用C语言对结构体的处理不够方便，比较麻烦，对字段的读取需要传参。

   解决方案：换用C++，使用结构体函数，在维持结构体不变的同时简化了对字段的读取和处理。

六、实验的启示和建议

系统级的套接字封装的相当优秀，大大简化了传输层连接协议的使用过程。

附：共用时间7小时