厦門大學

信息学院软件工程系

《计算机网络》实验报告

题　　目 实验七 代理服务器软件

班　　级 软件工程 2019 级 1 班

姓　　名 雷鸿宇

学　　号 22920192204173

实验时间 2021 年 6 月 1 日

2021 年 6 月 1 日

填写说明

1、本文件为 Word 模板文件，建议使用 Microsoft Word 2019 打开，在可填写的区域中如实填写；

2、填表时，勿破坏排版，勿修改字体字号，打印成 PDF 文件提交；

3、文件总大小尽量控制在 1MB 以下，勿超过 5MB；

4、应将材料清单上传在代码托管平台上；

5、在学期最后一节课前按要求打包发送至 cni21@qq.com。

1 实验目的

通过完成实验，掌握基于 RFC 应用层协议规约文档传输的原理，实现符合接口且能和已有知名软件协同运作的软件。

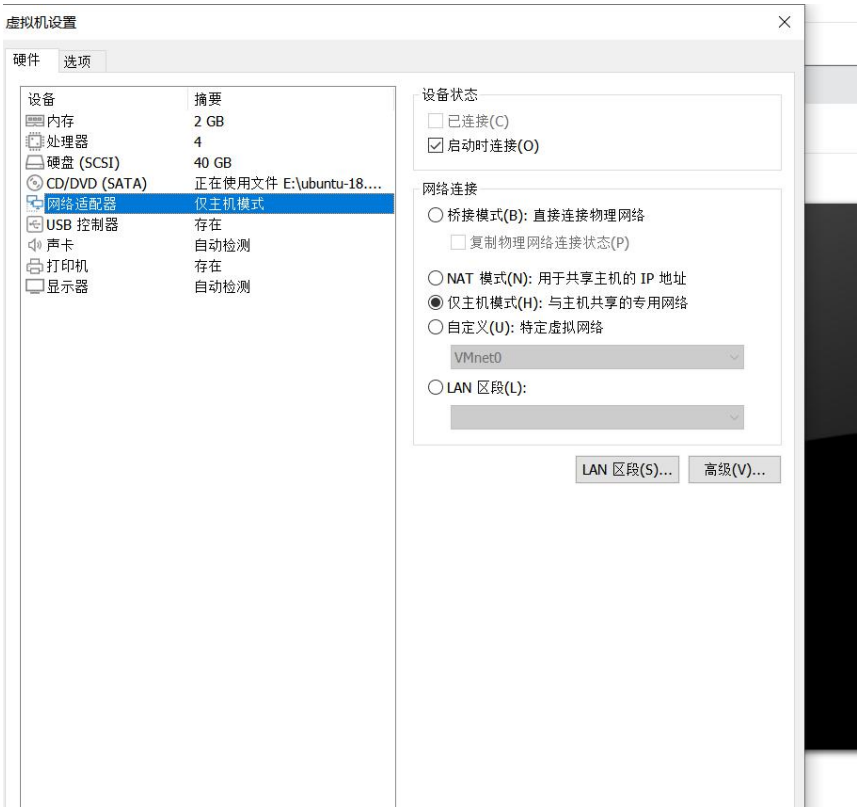
2 实验环境

Vmware linux ubuntu

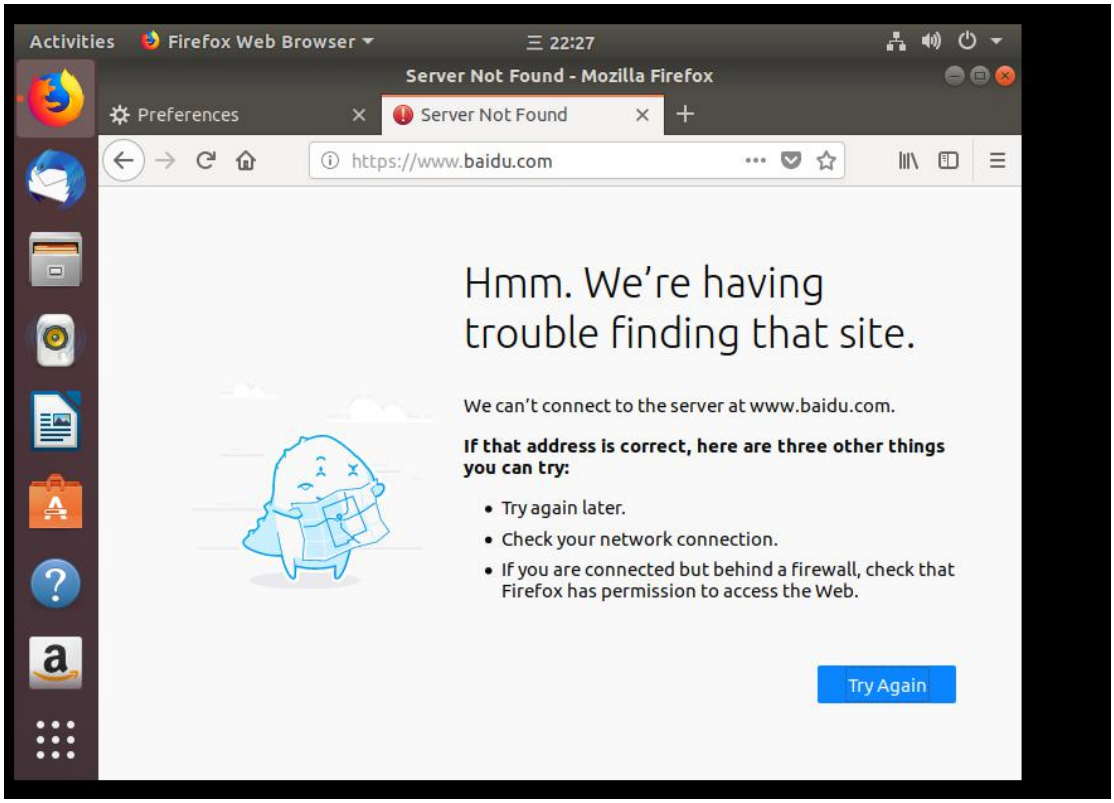
c 语言

3 实验结果

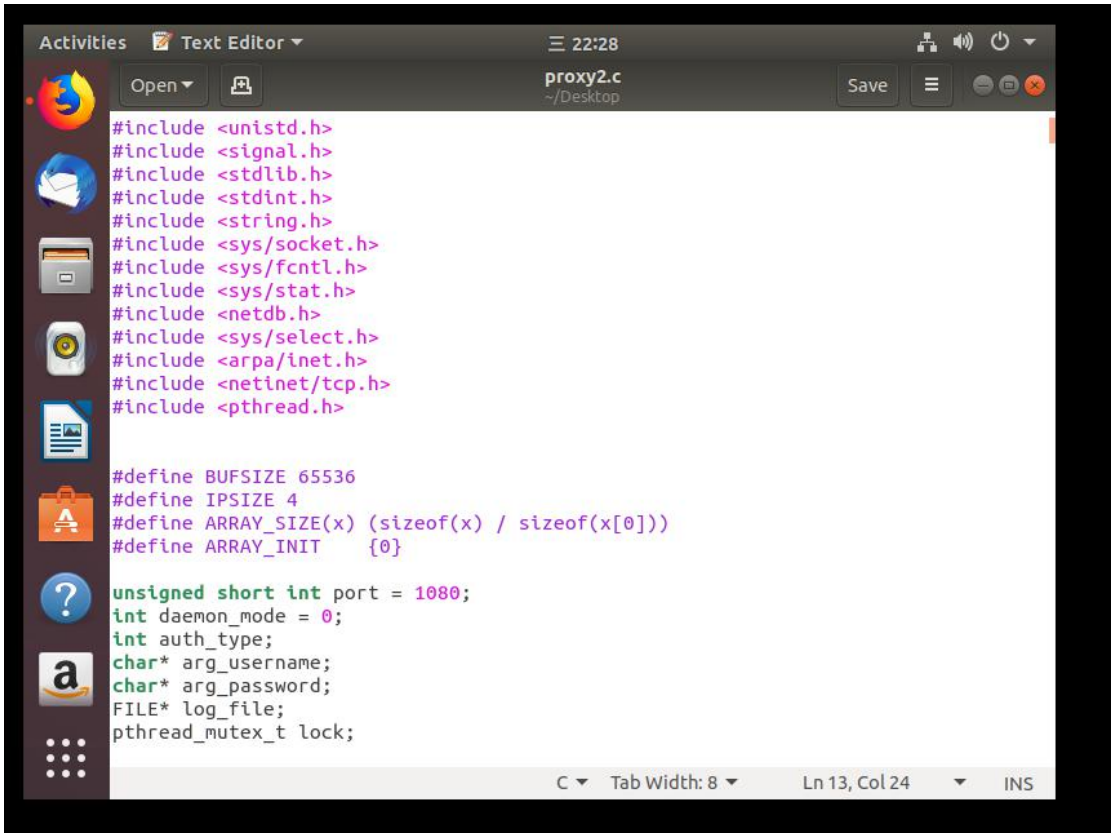
Vmware 虚拟机网络连接调整为 host-only



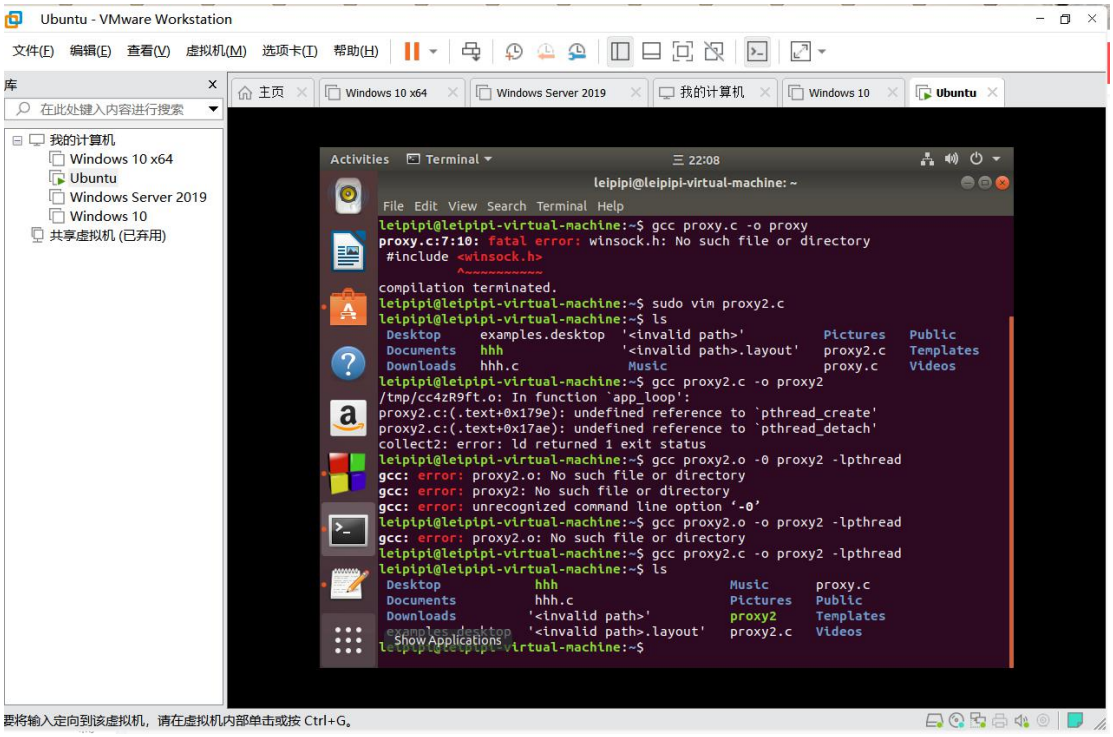
可以看到，目前无法访问



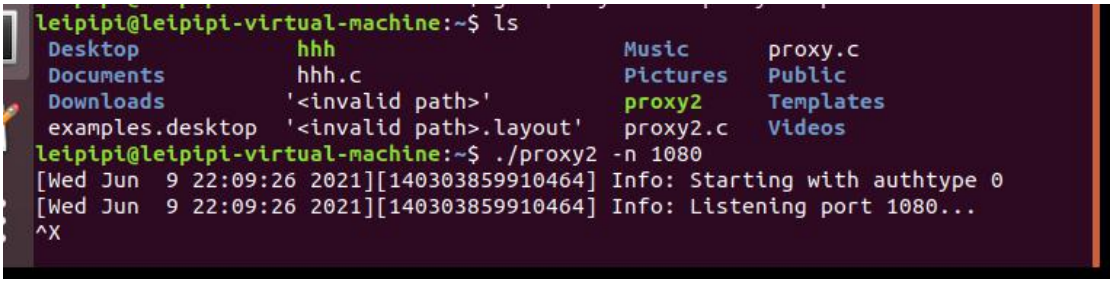
使用 proxy 示例代码(已对源代码进行阅读并进行相应注释　)


Activities
Text Editor
22:28
Open
proxy2.c
~/Desktop
Save
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/fcntl.h>
#include <sys/stat.h>
#include <netdb.h>
#include <sys/select.h>
#include <arpa/inet.h>
#include <netinet/tcp.h>
#include <pthread.h>
#define BUFSIZE 65536
#define IPSIZE 4
#define ARRAY_SIZE(x) (sizeof(x) / sizeof(x[0]))
#define ARRAY_INIT {0}
unsigned short int port = 1080;
int daemon_mode = 0;
int auth_type;
char* arg_username;
char* arg_password;
FILE* log_file;
pthread_mutex_t lock;
C
Tab Width: 8
Ln 13, Col 24
INS

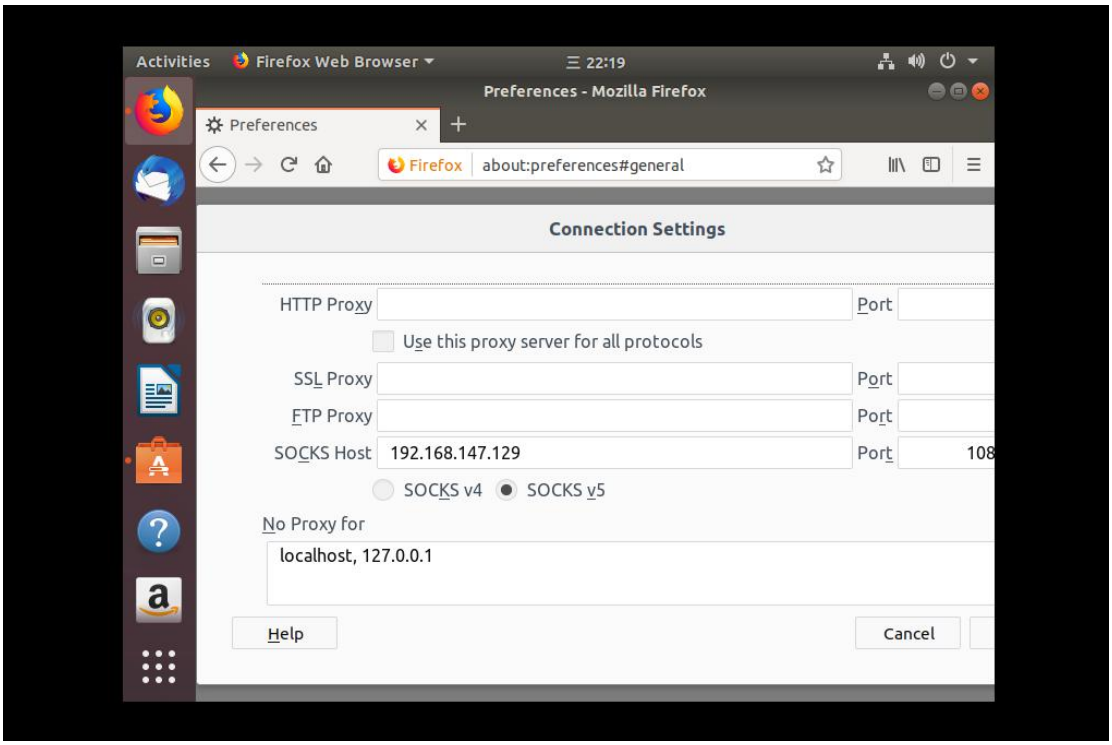
使用 vim 编译代码


Ubuntu - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
库
在此处键入内容进行搜索
我的计算机
Windows 10 x64
Ubuntu
Windows Server 2019
Windows 10
共享虚拟机 (已弃用)
主页
Windows 10 x64
Windows Server 2019
我的计算机
Windows 10
Ubuntu
Activities
Terminal
22:08
leipipi@leipipi-virtual-machine: ~
File Edit View Search Terminal Help
leipipi@leipipi-virtual-machine:~$ gcc proxy.c -o proxy
proxy.c:7:10: fatal error: winsock.h: No such file or directory
#include <winsock.h>
compilation terminated.
leipipi@leipipi-virtual-machine:~$ sudo vim proxy2.c
leipipi@leipipi-virtual-machine:~$ ls
Desktop examples.desktop '<invalid path>' Pictures Public
Documents hhh '<invalid path>.layout' proxy2.c Templates
Downloads hhh.c Music proxy.c Videos
leipipi@leipipi-virtual-machine:~$ gcc proxy2.c -o proxy2
/tmp/cc4zR9ft.o: In function `app_loop':
proxy2.c:(.text+0x179e): undefined reference to `pthread_create'
proxy2.c:(.text+0x17ae): undefined reference to `pthread_detach'
collect2: error: ld returned 1 exit status
leipipi@leipipi-virtual-machine:~$ gcc proxy2.o -0 proxy2 -lpthread
gcc: error: proxy2.o: No such file or directory
gcc: error: proxy2: No such file or directory
gcc: error: unrecognized command line option '-0'
leipipi@leipipi-virtual-machine:~$ gcc proxy2.o -o proxy2 -lpthread
gcc: error: proxy2.o: No such file or directory
leipipi@leipipi-virtual-machine:~$ gcc proxy2.c -o proxy2 -lpthread
leipipi@leipipi-virtual-machine:~$ ls
Desktop hhh Music proxy.c
Documents hhh.c Pictures Public
Downloads '<invalid path>' proxy2 Templates
examples.desktop '<invalid path>.layout' proxy2.c Videos
Show Applications
要将输入定向到该虚拟机，请在虚拟机内部单击或按 Ctrl+G。

运行程序，使其监听 1080 端口



调整浏览器设置



部分代码（含注释）

```
unsigned short int port = 1080;//端口号
int daemon_mode = 0;
int auth_type;
char* arg_username;//用户名
char* arg_password;//密码
FILE* log_file;//日志文件，记录运行情况
pthread_mutex_t lock;//pthread 库的一个线程锁

//状态参数
enum socks {
    RESERVED = 0x00,
    VERSION4 = 0x04,
```

```
    VERSION5 = 0x05
};

enum socks_auth_methods {
    NOAUTH = 0x00,
    USERPASS = 0x02,
    NOMETHOD = 0xff
};

enum socks_auth_userpass {
    AUTH_OK = 0x00,
    AUTH_VERSION = 0x01,
    AUTH_FAIL = 0xff
};

enum socks_command {
    CONNECT = 0x01
};

enum socks_command_type {
    IP = 0x01,
    DOMAIN = 0x03
};

enum socks_status {
    OK = 0x00,
    FAILED = 0x05
};

void log_message(const char* message, ...)
{
    if (daemon_mode) {
        return;
    }

    char vbuffer[255];
    va_list args;
    va_start(args, message);
    vsnprintf(vbuffer, ARRAY_SIZE(vbuffer), message, args);
    va_end(args);

    time_t now;
    time(&now);
    char* date = ctime(&now);
    date[strlen(date) - 1] = '\0';

    pthread_t self = pthread_self();//获取自身线程

    if (errno != 0) {
        pthread_mutex_lock(&lock);//操作前将互斥锁上锁
        fprintf(log_file, "[%s][%lu] Critical: %s - %s\n", date, self,
            vbuffer, strerror(errno));
```

```
        errno = 0;
        pthread_mutex_unlock(&lock);
    }
    else {
        fprintf(log_file, "[%s][%lu] Info: %s\n", date, self, vbuffer);//错误
    }
    fflush(log_file);
}

int readn(int fd, void* buf, int n)//读取内容
{
    int nread, left = n;
    while (left > 0) {
        if ((nread = read(fd, buf, left)) == -1) {
            if (errno == EINTR || errno == EAGAIN) {
                continue;
            }
        }
        else {
            if (nread == 0) {
                return 0;
            }
            else {
                left -= nread;
                buf += nread;
            }
        }
    }
    return n;
}

int writen(int fd, void* buf, int n)//写入内容
{
    int nwrite, left = n;
    while (left > 0) {
        if ((nwrite = write(fd, buf, left)) == -1) {
            if (errno == EINTR || errno == EAGAIN) {
                continue;
            }
        }
        else {
            if (nwrite == n) {
                return 0;
            }
            else {
                left -= nwrite;
                buf += nwrite;
            }
        }
    }
    return n;
}
```

```
void app_thread_exit(int ret, int fd)//关闭线程连接和 socket
{
    close(fd);
    pthread_exit((void*)&ret);
}

int app_connect(int type, void* buf, unsigned short int portnum)//建立线程连接
{
    int fd;
    struct sockaddr_in remote;
    char address[16];

    memset(address, 0, ARRAY_SIZE(address));
    //ip 模式下
    if (type == IP) {
        char* ip = (char*)buf;
        snprintf(address, ARRAY_SIZE(address), "%hhu.%hhu.%hhu.%hhu",
            ip[0], ip[1], ip[2], ip[3]);
        memset(&remote, 0, sizeof(remote));//初始化然后获取 ip 地址和端口
        remote.sin_family = AF_INET;
        remote.sin_addr.s_addr = inet_addr(address);
        remote.sin_port = htons(portnum);

        fd = socket(AF_INET, SOCK_STREAM, 0);
        if (connect(fd, (struct sockaddr*) & remote, sizeof(remote)) < 0) {
            log_message("connect() in app_connect");
            close(fd);
            return -1;
        }

        return fd;
    }
    //域模式下
    else if (type == DOMAIN) {
        char portaddr[6];
        struct addrinfo* res;
        snprintf(portaddr, ARRAY_SIZE(portaddr), "%d", portnum);
        log_message("getaddrinfo: %s %s", (char*)buf, portaddr);
        int ret = getaddrinfo((char*)buf, portaddr, NULL, &res);//服务到端口的转换，存入 res
        if (ret == EAI_NODATA) {//(No address associated with hostname
            return -1;
        }
        else if (ret == 0) {
            struct addrinfo* r;
            for (r = res; r != NULL; r = r->ai_next) {
                fd = socket(r->ai_family, r->ai_socktype,
                    r->ai_protocol);
                if (fd == -1) {
                    continue;
                }
                ret = connect(fd, r->ai_addr, r->ai_addrlen);
```

```
                if (ret == 0) {
                    freeaddrinfo(res);
                    return fd;
                }
                else {
                    close(fd);
                }
            }
        }
        freeaddrinfo(res);
        return -1;
    }

    return -1;
}

int socks_invitation(int fd, int* version)
{
    char init[2];
    int nread = readn(fd, (void*)init, ARRAY_SIZE(init));
    //若版本与5和4都不兼容
    if (nread == 2 && init[0] != VERSION5 && init[0] != VERSION4) {
        log_message("They send us %hhX %hhX", init[0], init[1]);
        log_message("Incompatible version!");
        app_thread_exit(0, fd);
    }
    log_message("Initial %hhX %hhX", init[0], init[1]);//初始化版本成功
    *version = init[0];
    return init[1];
   }
```

4 实验代码

本次实验的代码已上传于以下代码仓库：https://github.com/leipipi

5 实验总结

本实验尝试设计一个代理服务器软件，由于难度过大，时间不足，于是对示例代码进行了阅读注释，并尝试在linux上运行代码实现其功能。明白了如何基于socket进行代理服务器软件编程，实现了在linux上使浏览器通过该软件能够成功进行网页访问。同时还涉及了一些线程知识，加深了对线程应用的理解。