《Linux应用程序开发实验报告》

题目: 嵌入式Linux网络编程

班级:\_\_ \_\_ \_\_17计科3班 \_

学号: 10417307 \_\_\_

姓名: \_\_薛冬冬\_ \_\_\_

东南大学成贤学院计算机系

2020年 月 日

**嵌入式Linux网络编程**

1. **实验目的**

1.掌握TCP/IP协议的基础知识；

2.掌握嵌入式Linux基础网络编程；

3.能够独立编写客户端、服务器端的通信程序。

1. **实验内容**

**1、客户端给服务端发消息，服务端可以回复消息。**

**答：**

（1）首先使用gcc进行编译，生成client和server。

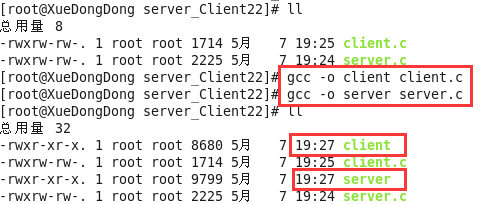


图1.编译

（2）运行可执行文件server和client。

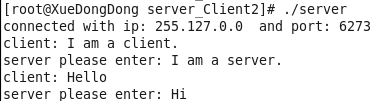


图2.服务端

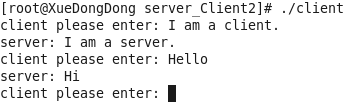


图3.客户端

（3）代码修改如下：

server.c

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/un.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define PORT 4321

#define BUFFER\_SIZE 1024

#define MAX\_QUE\_CONN\_NM 5

const char\* ip = "127.0.0.1";

int main()

{

struct sockaddr\_in server\_sockaddr,client\_sockaddr;

int sin\_size;

int sockfd, client\_fd;

char buf[BUFFER\_SIZE];

//创建客户端的socket，同服务端的含义

if((sockfd = socket(AF\_INET, SOCK\_STREAM, 0))==-1)

{

//创建失败

perror("socket");

return 1;

}

//设置sockaddr\_in 结构体中相关参数

server\_sockaddr.sin\_family = AF\_INET; /\*使用IPV4协议\*/

server\_sockaddr.sin\_port = htons(PORT);

server\_sockaddr.sin\_addr.s\_addr = inet\_addr(ip);

if(bind(sockfd, (struct sockaddr\*)&server\_sockaddr, sizeof(server\_sockaddr))== -1)

{

//命名失败

perror("bind");

return 2;

}

//调用listen函数，监听最多5个socket连接

if((listen(sockfd, MAX\_QUE\_CONN\_NM)) == -1)

{

//监听失败

perror("listen");

return 3;

}

if((client\_fd = accept(sockfd, (struct sockaddr \*)&client\_sockaddr, &sin\_size)) == -1)

{

perror("accept");

return 4;

}

else

{

printf("connected with ip: %s and port: %d\n", inet\_ntop(AF\_INET,&client\_sockaddr.sin\_addr, buf, 1024), ntohs(client\_sockaddr.sin\_port));

}

while(1)

{

memset(buf, '\0', sizeof(buf));

ssize\_t size = read(client\_fd, buf, sizeof(buf) - 1);

if(size > 0)

{

printf("client: %s\n", buf);

}

else if(size == 0)

{

printf("read is done...\n");

break;

}

else

{

perror("read");

break;

}

printf("server please enter: ");

fflush(stdout);

size = read(0, buf, sizeof(buf) - 1);

if(size > 0)

{

buf[size - 1] = '\0';

}

else if(size == 0)

{

printf("read is done...\n");

break;

}

else

{

perror("read");

break;

}

write(client\_fd, buf, strlen(buf));

}

close(sockfd);

return 0;

}

client.c

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/un.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define PORT 4321

#define BUFFER\_SIZE 1024

#define MAX\_QUE\_CONN\_NM 5

const char\* ip = "127.0.0.1";

int main(int argc, char \*argv[])

{

int sockfd;

char buf[BUFFER\_SIZE];

//创建客户端的socket，同服务端的含义

if((sockfd = socket(AF\_INET,SOCK\_STREAM,0)) == -1)

{

//创建失败

perror("socket");

return 1;

}

//设置客户端sockaddr\_in 结构体中相关参数

struct sockaddr\_in serv\_addr;

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

serv\_addr.sin\_addr.s\_addr = inet\_addr(ip);

socklen\_t addr\_len = sizeof(serv\_addr);

if((connect(sockfd,(struct sockaddr \*)&serv\_addr, sizeof(struct sockaddr)))== -1)

{

perror("connect");

return 2;

}

while(1)

{

memset(buf, '\0', sizeof(buf));

printf("client please enter: ");

fflush(stdout);

ssize\_t size = read(0, buf, sizeof(buf) - 1);

if(size > 0)

{

buf[size - 1] = '\0';

}

else if(size == 0)

{

printf("read is done...\n");

break;

}

else

{

perror("read");

return 4;

}

write(sockfd, buf, strlen(buf));

size = read(sockfd, buf, sizeof(buf));

if(size > 0)

{

buf[size] = '\0';

}

else if(size == 0)

{

printf("read is done...\n");

break;

}

else

{

perror("read");

return 5;

}

printf("server: %s\n", buf);

}

close(sockfd);

return 0;

}

**2、通过VMWare创建两个不同IP地址的虚拟机，使得一台虚拟机运行服务端、一台虚拟机运行客户端，通过“虚拟网络”实现真正的跨网络通信的效果。**

**答：**

（1）设置网络适配器为NAT模式

设置两个虚拟机的网络适配器为NAT模式。

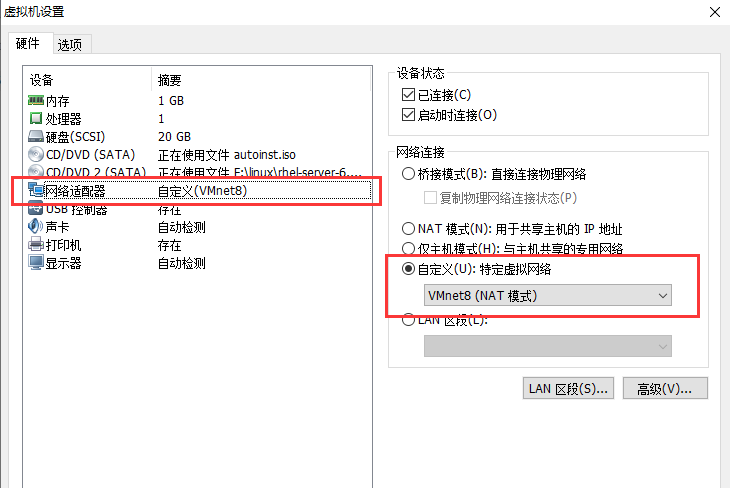


图4.虚拟机1更改网络适配器模式

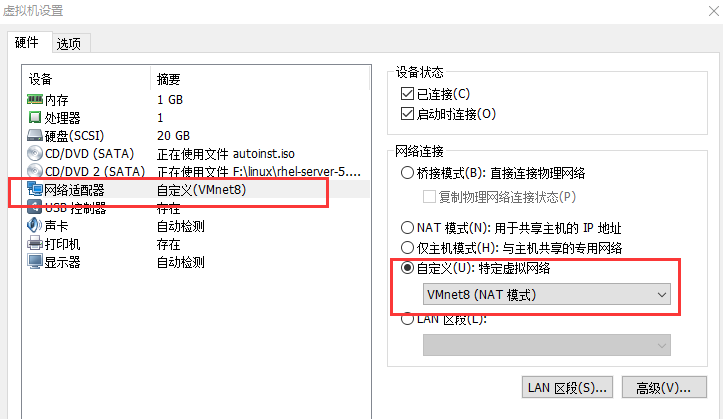


图5.虚拟机2更改网络适配器模式

（2）查看内网地址

查看两个虚拟机的内网地址

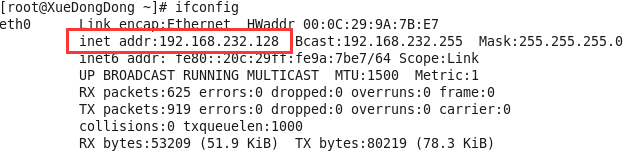


图6.虚拟机1的内网地址查看

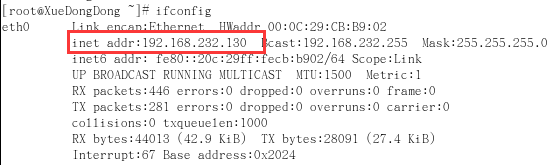


图7.虚拟机2的内网地址查看

（3）关闭防火墙

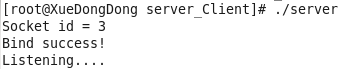


图8.虚拟机1内运行服务端



图9.虚拟机2内运行客户端发送消息报错

根据搜索资料得知是防火墙未关的原因才会报错。以下将对防火墙进行关闭：

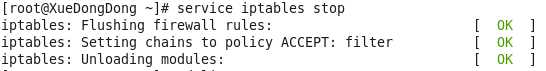


图10.虚拟机1内关闭防火墙

（4）重新运行服务端和客户端

运行服务端，等待客户端运行输入消息。

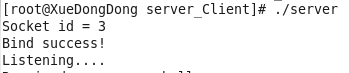


图11.虚拟机1运行服务端

运行客户端



图12.运行客户端，输入消息

服务端收到消息：

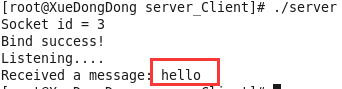


图13.虚拟机1服务端收到消息

（5）实现跨越网络通信成功。

1. **实验体会**

本次实验，与前几次实验相比，难度处于中等以上，需要自己编写代码来实现服务端可以回复客户端。我在仔细听懂老师在视频中对两个.c文件的代码讲解后，再结合本节课的知识点和自己网络上搜索的知识点后，成功完成了代码的改写，使其实现了客户端在向服务端发送信息后，服务端可以回复客户端的信息。

第二个实验我刚开始想的较为复杂，后来，根据自己搜索的资料一结合，突然感觉茅塞顿开，再看一遍突然觉得简单了很多，其中还遇到了防火墙的问题，但是将报错信息百度一下，很快就搜到了答案。

本次实验使我较好的掌握了TCP/IP协议的基础知识、Linux基础网络编程等，使我对知识点有了更深的理解。其中，socket基础编程较为重要。