网络编程 –模拟远程登陆系统

121180155 张皓

2014 年 11 月 25 日

目录

1 实验目的

- 学习 Linux 操作系统下网络的编写和应用;
- 学习 tcp、udp 环境的编程控制方法。

2 实验设计思想

2.1 实验完成功能

本实验通过 socket 网络编程模拟一个远程登陆系统，用户在客户机输入一些命令，这些命令在服务器上运行，而标准输出结果重定向到客户机的文件描述符中，从而实现了在客户机模拟远程登陆的效果。

2.2 实验特点

- 使用 select 解决阻塞问题，可以有多个客户机同时运行。
- 将客户机文件描述符重定向到标准输出，从而实现远程登陆效果

3 实验内容

3.1 server.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

#define PORT 9999            //端口号
#define MAX     10                  //最大监听数
#define ADDR "127.0.0.1"    //本地ip地址

//定义字体打印时的颜色
#define LIGHT_BLUE   "\033[1;34m"
#define NONE         "\033[m"

//定义相关变量
int sockfd, newsockfd, is_connected[MAX], fd;
```

```
struct sockaddr_in addr;
int addr_len = sizeof(struct sockaddr_in);
fd_set readfd;
char msgbuffer[2048];
//定义连接成功时欢迎信息
char msg[] = LIGHT_BLUE"This is the msg from server. conneceted\nYou can input some cmd

//初始化服务器
void init_server(void)
{
    //建立socket并错误检查
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("socket err\n");
        return -1;
    }
    printf("socket created ");
    printf("socked id: %d\n", sockfd);

    //设置端口、ip地址等
    bzero(&addr, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(PORT);
    addr.sin_addr.s_addr = inet_addr(ADDR);

    //绑定端口
    if(bind(sockfd, &addr, sizeof(addr)) < 0)
    {
        printf("bind err\n");
        return -1;
    }
    printf("binded ");
    printf("local port: %d\n", PORT);
```

```
    //监听
    if(listen(sockfd, MAX) < 0)
    {
        printf("listen err\n");
        return -1;
    }
    printf("listen\n");

    //清空已连接标记
    for(fd= 0; fd < MAX; ++fd)
    {
        is_connected[fd] = 0;
    }

}

int main(void)
{
    init_server();

    //主循环
    while(1)
    {
        //清空监视集
        FD_ZERO(&readfd);
        FD_SET(sockfd, &readfd);
        //判断有无连接
        for(fd = 0; fd < MAX; ++fd)
        {
            //有连接则设置监视集
            if(is_connected[fd])
            {
                FD_SET(fd, &readfd);
            }
```

```
}

//若达到了最大连接数则直接进入下次循环
if(!select(MAX, &readfd, NULL, NULL, NULL))
{
    continue;
}

for(fd = 0; fd < MAX; ++fd)
{
    //判断监听集
    if(FD_ISSET(fd, &readfd))
    {
        //若为sockfd
        if(sockfd == fd)
        {
            //接受一个新连接
     if((newsockfd = accept(sockfd, &addr, &addr_len)) < 0)
            {
                printf("accept err\n");
            }
            //输出欢迎提示信息
            write(newsockfd, msg, sizeof(msg));
            //设置标志
            is_connected[newsockfd] = 1;
            printf("accept ");
     printf("connect from %s\n", inet_ntoa(addr.sin_addr));
        }
        else
        {
            //清空字符串
            bzero(msgbuffer, sizeof(msgbuffer));
            //读入用户输入命令
        if(read(fd, msgbuffer, sizeof(msgbuffer)) <= 0)
```

```
                    {
                        //若失败，则关闭连接
                        printf("connect closed\n");
                        is_connected[fd] = 0;

                        close(fd);
                    }
                    else
                    {
                        //将fd重定向到stdout
                        dup2(fd, 1);
                        //调用系统命令
                        system(msgbuffer);
                    }
                }
            }
        }
    }
    return 0;
}
```

3.2 client.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>

#define PORT    9999            //远程端口
#define REMOTE_IP "127.0.0.1"    //远程ip地址
```

```
//定义变量
int fd;
struct sockaddr_in addr;
char mybuffer[2048];

//初始化客户机
void init_client(void)
{
    //创建socket并错误检查
    if((fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("socket err\n");
        return -1;
    }
    printf("socket created ");
    printf("socked id: %d\n", fd);

    //设置远程端口和ip地址
    bzero(&addr, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(PORT);
    addr.sin_addr.s_addr = inet_addr(REMOTE_IP);

    //连接远程服务器
    if(connect(fd, &addr, sizeof(addr)) < 0)
    {
        printf("connect err\n");
        return -1;
    }
    printf("connected ");
    printf("remote ip: %s ", REMOTE_IP);
    printf("remote port: %d\n", PORT);

    //接受远程服务器的输出欢迎信息并显示
```

```
    recv(fd, mybuffer, sizeof(mybuffer), 0);
    printf("%s\n", mybuffer);
}

int main(void)
{
    init_client();
    while(1)
    {
        //清空字符串
        bzero(mybuffer, sizeof(mybuffer));
        //读入一个命令
        read(STDIN_FILENO, mybuffer, sizeof(mybuffer));
        //发送信息
        if(send(fd, mybuffer, sizeof(mybuffer), 0) < 0)
        {
            printf("send err\n");
            return -1;
        }
        //清空字符串
        bzero(mybuffer, sizeof(mybuffer));
        //接受并输出信息
        recv(fd, mybuffer, sizeof(mybuffer), 0);
        printf("%s\n", mybuffer);
    }
    return 0;
}
```

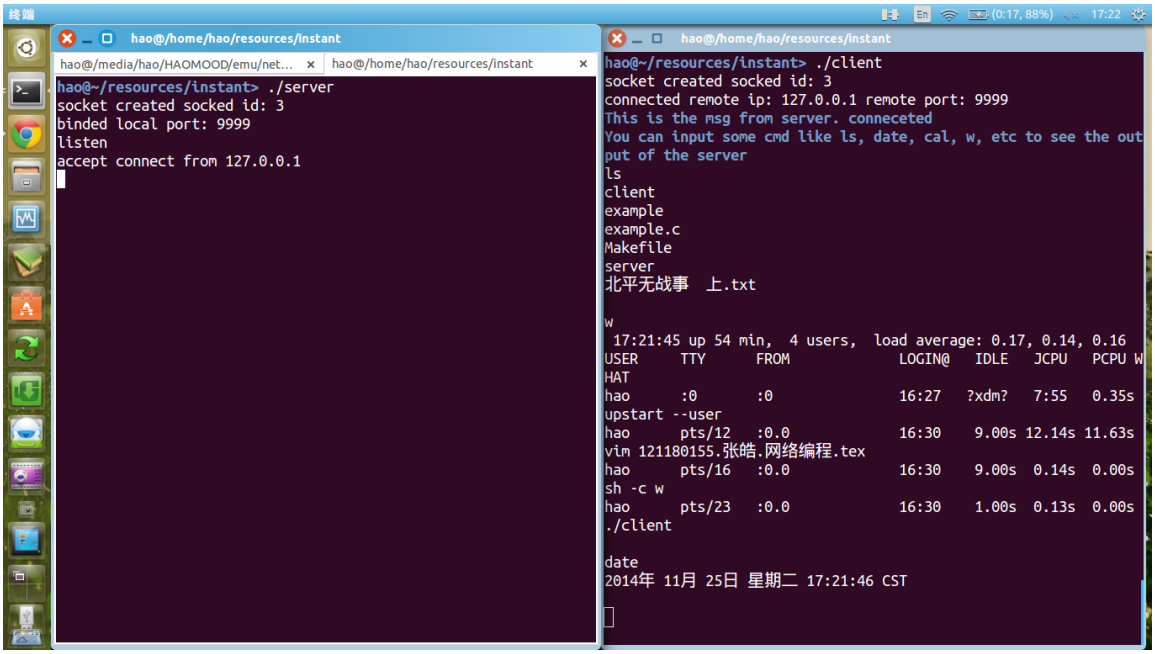
3.3 Makefile

```
all: server client
CFLAGS =
%: %.c
    gcc -o $@ $(CFLAGS) $^
```

```
clean:
    rm server client
```

3.4 实验运行效果

```
>make
#运行服务器
>./server
#新建终端，运行客户机
>./client
```



4 实验分析与小结

由图中可以看出，程序完成了模拟远程登陆的效果，将服务机的输出信息反应到客户机中。

日后可以改进的地方在于可以模拟一个 ftp 系统，从而达到客户机和服务器之间发送和接受文件的作用。

5 参考文献

- 微处理器与嵌入式系统 (试用教材). 方元, 彭成磊. 南京大学.2014