南昌大学实验报告6

姓名：*谢志彬* 学号：*6103115112*

邮箱地址：*siliconx@163.com*

专业班级：*计算机科学与技术153*

实验日期：**2018/04/22**

课程名称：**Linux程序设计实验**

实验项目名称

Socket It Out

实验目的

1.理解**socket**机制

2.学习更多关于**C**语言的知识

3.理解网络编程的过程

实验基础

C语言、**Go**语言、**Socket**

实验步骤

Task 1: Socket it (in C)

I.编写server.c

```
// 服务端
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>

#include <netinet/in.h>
```

```c
#define PORT 8000
#define BUFFER_SIZE 1024

int main(int argc, char const *argv[]) {
    printf("Server Running\n");
    int server_fd, new_socket, valread;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    char *response;  // 响应字符串
    int count = 0;

    // socket文件描述器
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // 设置socket端口
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt))) {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    // 绑定端口
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    while(1) {
        if (listen(server_fd, 3) < 0) {
            perror("listen");
            exit(EXIT_FAILURE);
        }

        if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
(socklen_t*)&addrlen)) < 0) {
            perror("accept");
            exit(EXIT_FAILURE);
        }

        valread = read(new_socket, buffer, 1024);
        printf("Message from client(No.%d): %s\n", count, buffer);

        response = buffer;  // 返回原消息

        send(new_socket, response, strlen(response), 0);
```

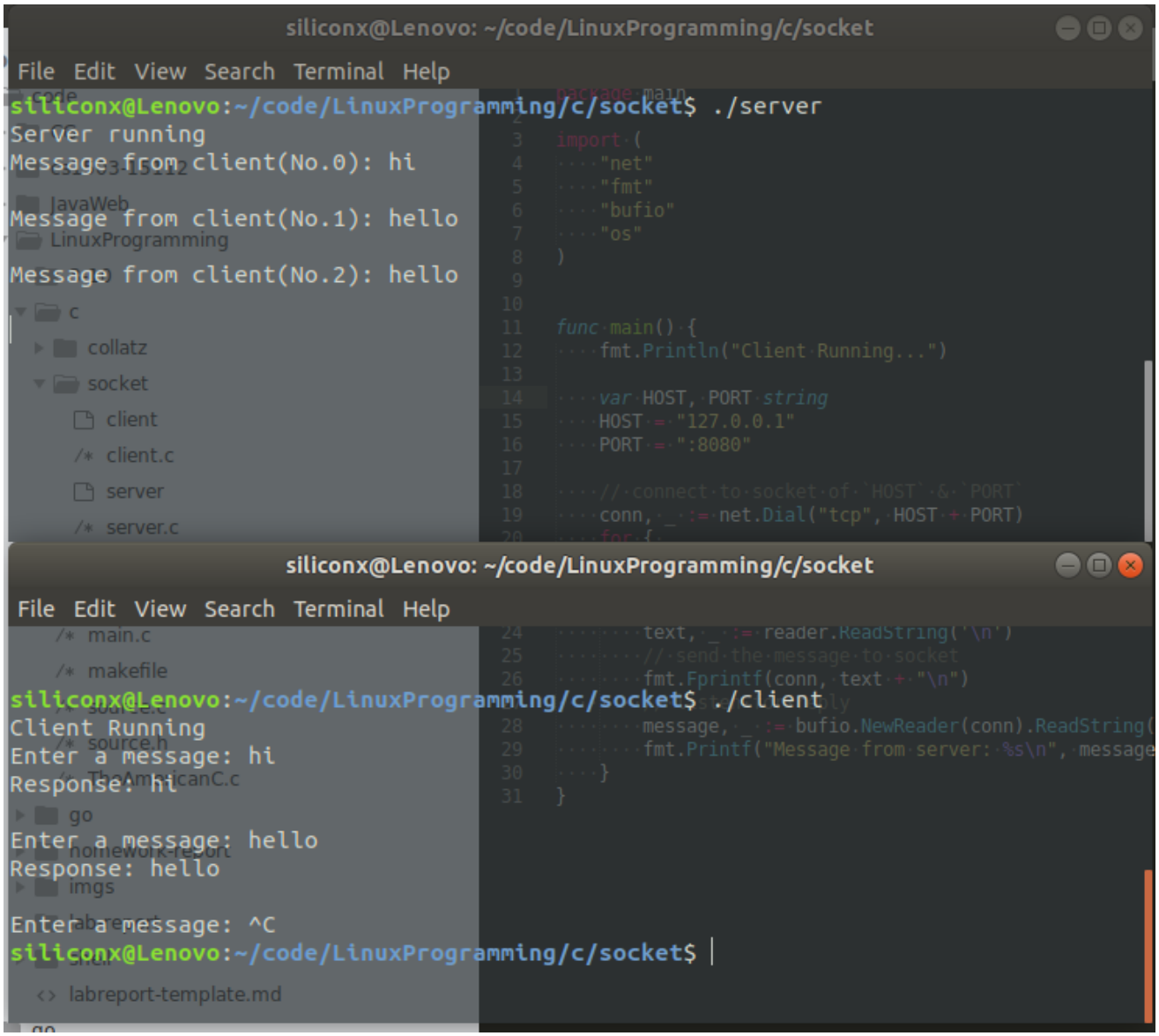
```
        count++;
    }
    return 0;
}
```

II.编写client.c

```
// 客户端
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define PORT 8000
#define BUFFER_SIZE 1024
#define SERVER_IP "127.0.0.1"

int main(int argc, char const *argv[]) {
    printf("Client Running\n");
    struct sockaddr_in address;
    int sock = 0, valread;
    struct sockaddr_in serv_addr;
    char *request;
    char buffer[BUFFER_SIZE] = {0};

    while (1) {
        if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
            printf("\n Socket create failed \n");
            return -1;
        }


        memset(&serv_addr, '0', sizeof(serv_addr));

        serv_addr.sin_family = AF_INET;
        serv_addr.sin_port = htons(PORT);

        // 将ip从字符串转成二进制形式
        if(inet_pton(AF_INET, SERVER_IP, &serv_addr.sin_addr)<=0) {
            printf("\nInvalid address\n");
            return -1;
        }

        if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
            printf("\nConnection Failed \n");
            return -1;
        }


        printf("Enter a message: ");
```

```
        fgets(request, BUFFER_SIZE, stdin);

        // 发送请求
        send(sock, request, strlen(request), 0);
        valread = read(sock ,buffer, 1024);
        printf("Response: %s\n", buffer);
    }
    return 0;
}
```

III.编译运行



Task 2: Easier Job on the Way(in Golang)

I.server.go

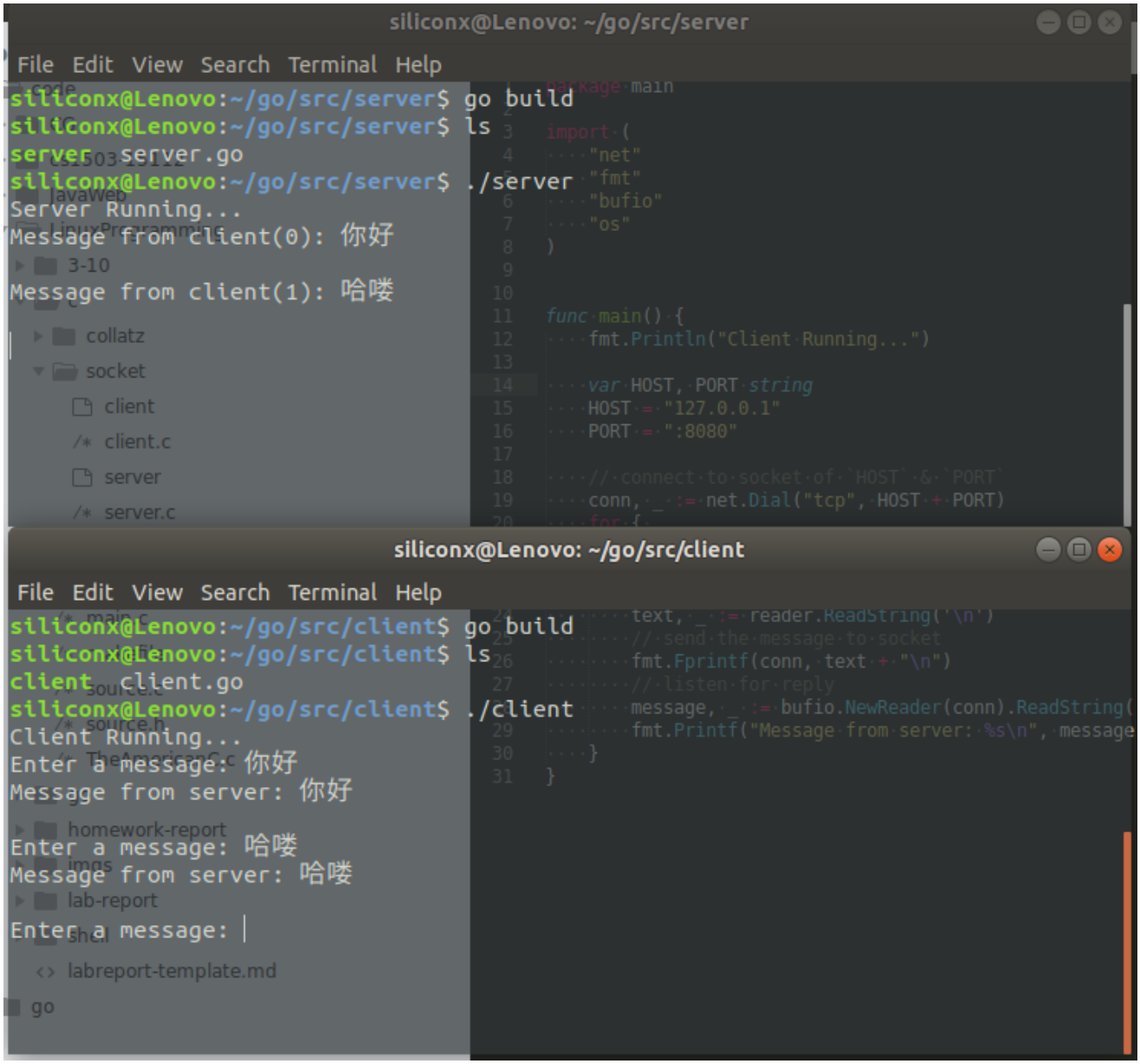
```
package main

import (
    "net"
    "fmt"
    "bufio"
)

func main() {
    fmt.Println("Server Running...")

    var PORT string
    var count int
    PORT = ":8080"
    count = 0

    // 监听TCP的 `PORT` 端口
    ln, _ := net.Listen("tcp", PORT)

    // 接受连接
    conn, _ := ln.Accept()

    // 监听中...
    for {
        message, _ := bufio.NewReader(conn).ReadString('\n')
        // 打印客户端消息
        fmt.Printf("Message from client(%d): %s\n", count, string(message))

        response := message
        // 返回原消息
        conn.Write([]byte(response + "\n"))
        count++;
    }
}
```

II.client.go

```
package main

import (
    "net"
    "fmt"
    "bufio"
    "os"
)


func main() {
```

```
    fmt.Println("Client Running...")

    var HOST, PORT string
    HOST = "127.0.0.1"
    PORT = ":8080"

    // 连接 `HOST` 和 `PORT` 对应的socket
    conn, _ := net.Dial("tcp", HOST + PORT)
    for {
        // 读取输入
        reader := bufio.NewReader(os.Stdin)
        fmt.Print("Enter a message: ")
        text, _ := reader.ReadString('\n')
        // 发送消息到socket
        fmt.Fprintf(conn, text + "\n")
        // 监听响应
        message, _ := bufio.NewReader(conn).ReadString('\n')
        fmt.Printf("Message from server: %s\n", message)
    }
}
```

II.编译运行



实验思考

- 理解使用C语言中的Socket
- 理解使用Go语言中的Socket

参考资料

- 《Golang Reference》