南昌大学实验报告7

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：**2018/05/28**

课程名称：**Linux程序设计实验**

实验项目名称

Socket It Out(More)

实验目的

1.理解socket机制

2.熟悉多进程/线程编程

3.理解网络编程的过程

实验基础

C语言、多进/线程、Socket

实验步骤

T 1: Socket it in more processes

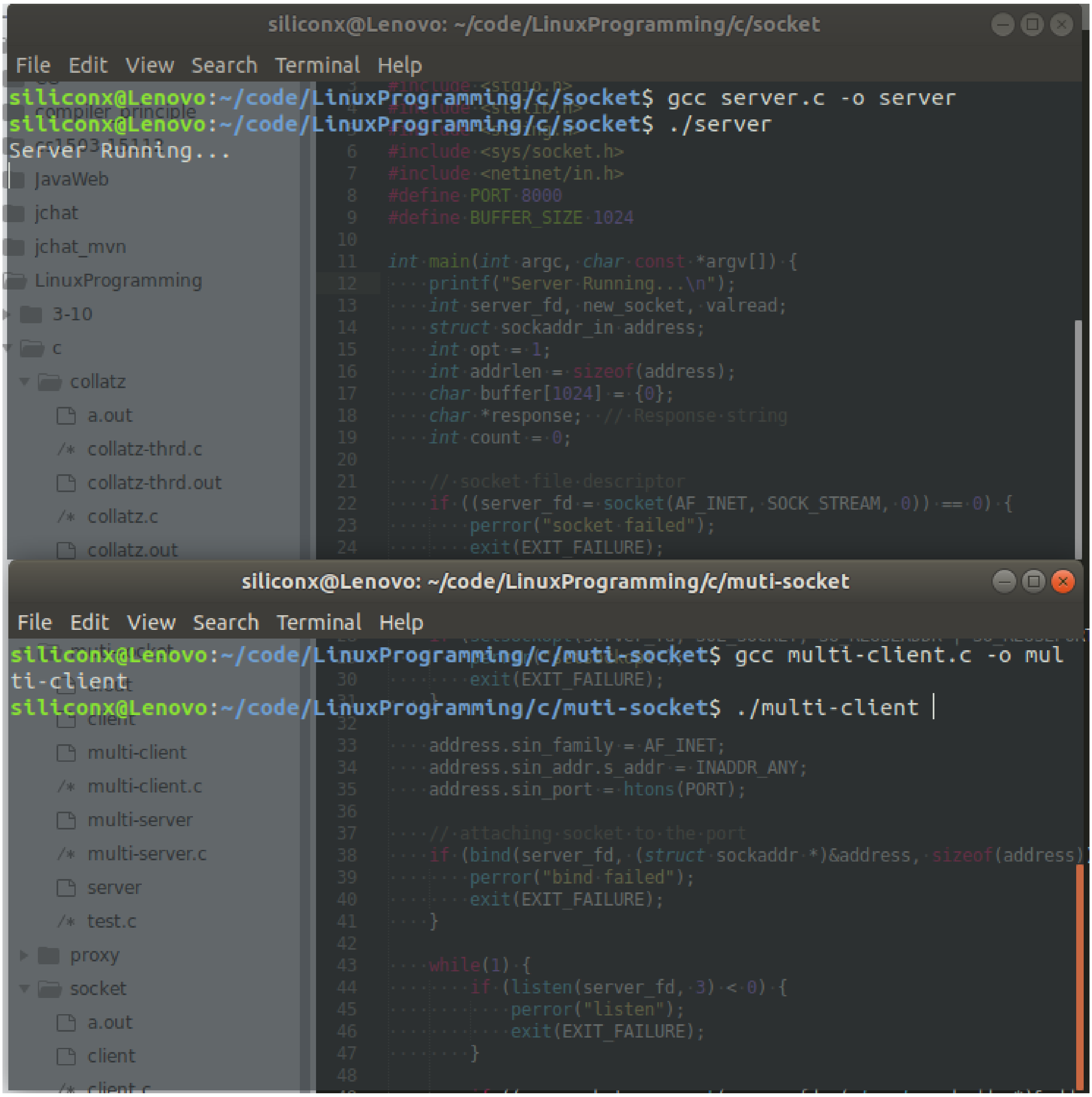
> 由于要求客户端多于100个进程，所以可以用fork()来创建进程.
>
> 顺序调用N次fork()将会产生`2^N - 1`个子进程(共有2^N个进程)，故 N >= 7

I.编写multi-client.c

```
// Client side
#include <stdio.h>
#include <stdlib.h>
```

```c
#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"
#define N 7

int main(int argc, char const *argv[]) {
    for (int i = 0; i < N; ++i) {  // create (2^N - 1) child processes
        fork();
    }

    printf("Client Running...\n");
    struct sockaddr_in address;
    int sock = 0, valread;
    struct sockaddr_in serv_addr;
    char *request = "Multi-Processes Client";  // request message
    char buffer[BUFFER_SIZE] = {0};

    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);

    // Convert IP addresses from text to binary form
    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;
    }

    send(sock, request, strlen(request), 0);
    valread = read(sock , buffer, 1024);
    printf("Message from Server: %s\n", buffer);

    return 0;
}
```

II.编译运行

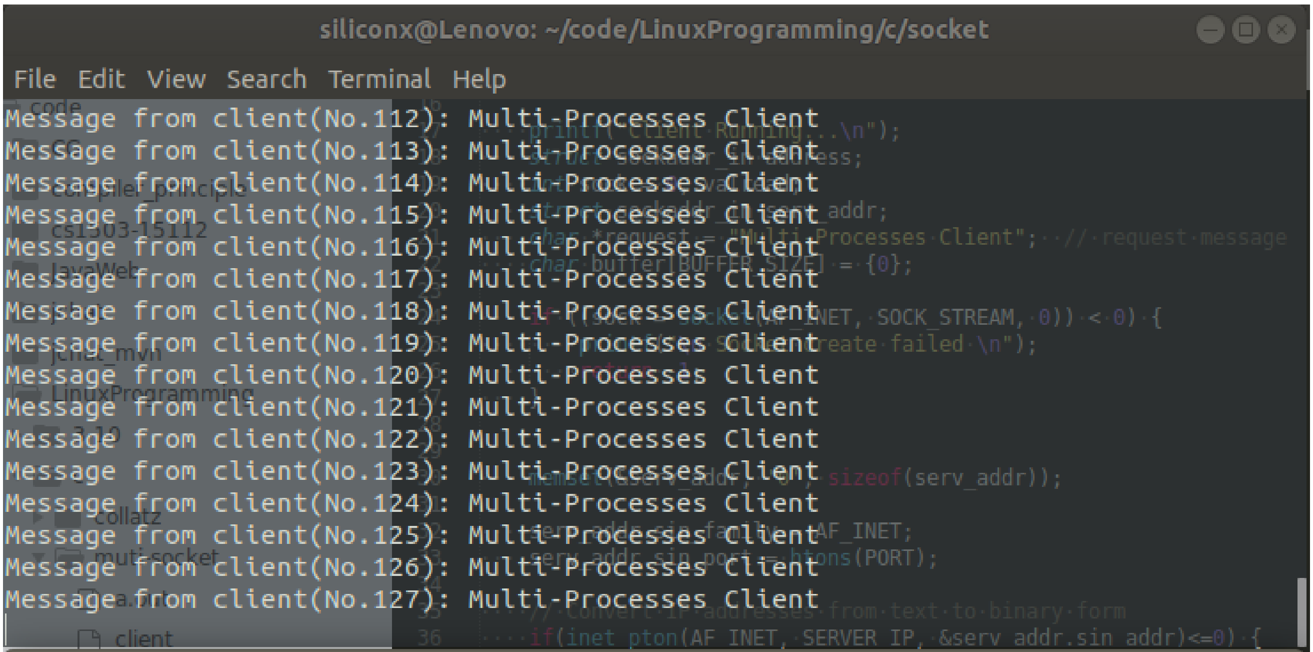


```
siliconx@Lenovo:~/code/LinuxProgramming/c/socket$ gcc server.c -o server
siliconx@Lenovo:~/code/LinuxProgramming/c/socket$ ./server
Server Running...
```

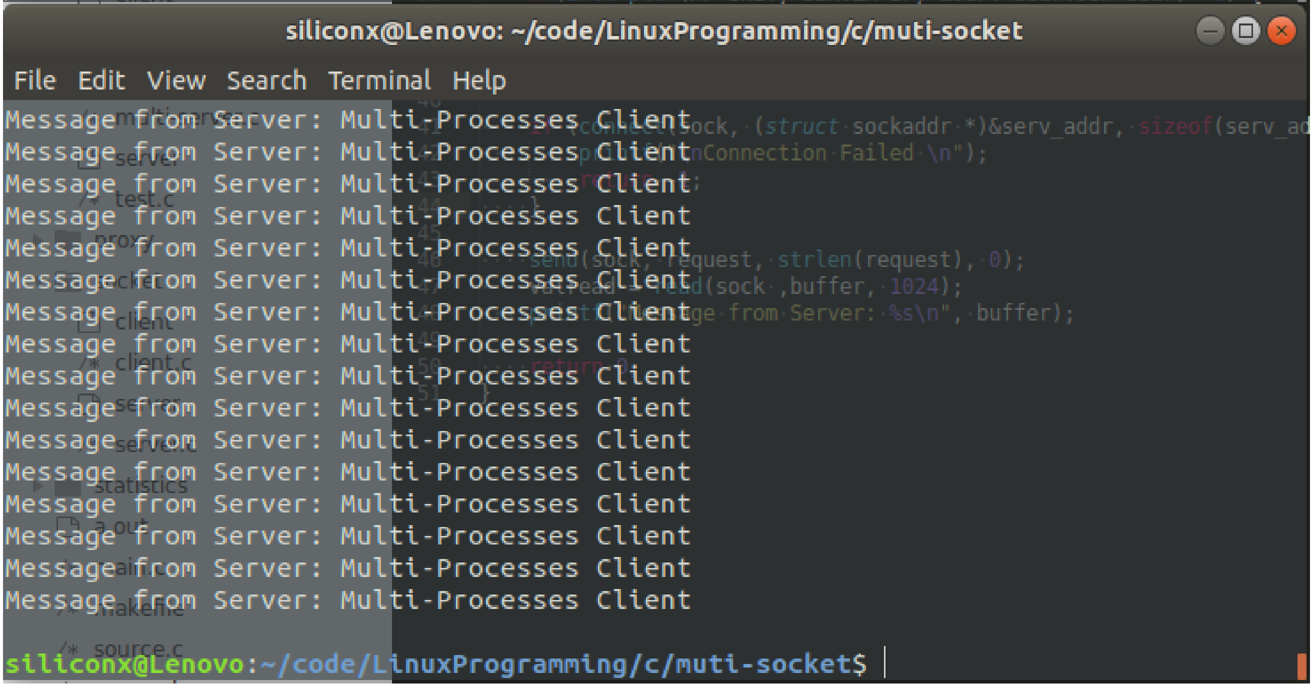
```
#include <sys/socket.h>
#include <netinet/in.h>
#define PORT 8000
#define BUFFER_SIZE 1024

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;  // Response string
    int count = 0;

    // socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
```

```
siliconx@Lenovo:~/code/LinuxProgramming/c/muti-socket$ gcc multi-client.c -o mul
ti-client
siliconx@Lenovo:~/code/LinuxProgramming/c/muti-socket$ ./multi-client
```

```
        exit(EXIT_FAILURE);

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

    // attaching socket to the port
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address))
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

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


siliconx@Lenovo: ~/code/LinuxProgramming/c/socket
File Edit View Search Terminal Help

```
Message from client(No.112): Multi-Processes Client
Message from client(No.113): Multi-Processes Client
Message from client(No.114): Multi-Processes Client
Message from client(No.115): Multi-Processes Client
Message from client(No.116): Multi-Processes Client
Message from client(No.117): Multi-Processes Client
Message from client(No.118): Multi-Processes Client
Message from client(No.119): Multi-Processes Client
Message from client(No.120): Multi-Processes Client
Message from client(No.121): Multi-Processes Client
Message from client(No.122): Multi-Processes Client
Message from client(No.123): Multi-Processes Client
Message from client(No.124): Multi-Processes Client
Message from client(No.125): Multi-Processes Client
Message from client(No.126): Multi-Processes Client
Message from client(No.127): Multi-Processes Client
```


siliconx@Lenovo: ~/code/LinuxProgramming/c/muti-socket
File Edit View Search Terminal Help

```
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client
Message from Server: Multi-Processes Client

siliconx@Lenovo:~/code/LinuxProgramming/c/muti-socket$ 
```

> 可以看到一共发送了128个请求

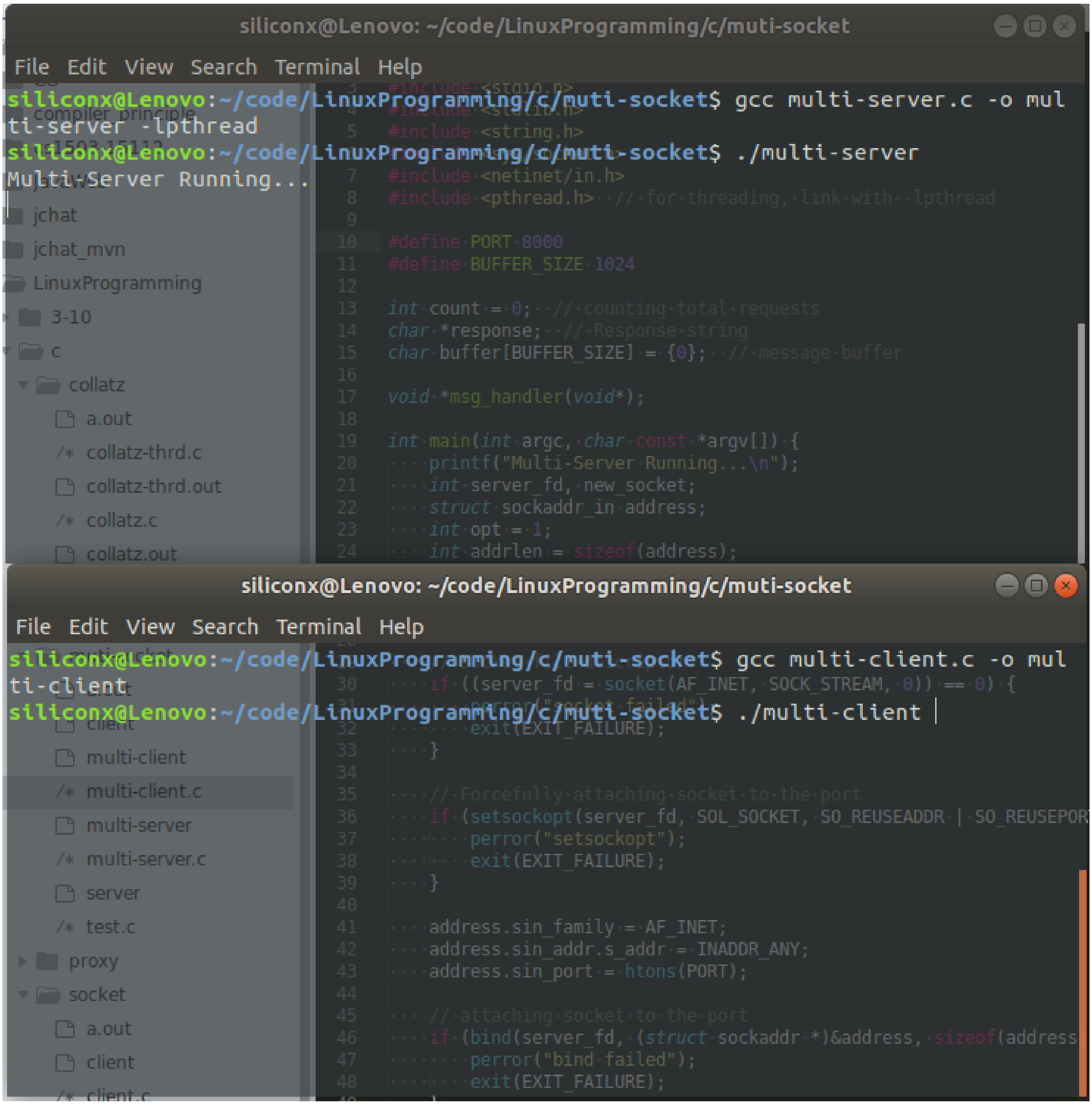
T 2: Socket it in more threads

> 使用pthread库改造服务端程序，使其支持多线程.
>
> 主线程用于接受客户端的请求，子线程用于发送服务端的响应

I.编写multi-server.c

```c
// Server Side
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <pthread.h>  // for threading, link with -lpthread

#define PORT 8000
#define BUFFER_SIZE 1024

int count = 0;  // counting total requests
char *response;  // Response string
char buffer[BUFFER_SIZE] = {0};  // message buffer

void *msg_handler(void*);

int main(int argc, char const *argv[]) {
    printf("Multi-Server Running...\n");
    int server_fd, new_socket;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);

    pthread_t tid;  // the thread identifier
    pthread_attr_t attr;  // set of attributes for the thread

    // socket file descriptor
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Forcefully attaching socket to the port
    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);

    // attaching socket to the port
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

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

    while (1) {
        if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
(socklen_t*)&addrlen)) < 0) {
            perror("accept");
            return -1;
        }

        /**
        * Now, Create thread to send message
        */

        // get the default attributes
        pthread_attr_init(&attr);

        // create the thread
        pthread_create(&tid, &attr, msg_handler, (void*) &new_socket);

        // now wait for the thread to exit
        pthread_join(tid, NULL);
    }

    return 0;
}


/**
 * This function will to handle the sending of reponse message
 */
void *msg_handler(void *new_socket) {
    int sock = *(int*) new_socket;
    read(sock, buffer, BUFFER_SIZE);

    printf("Message from Multi-Client(No.%d): %s\n", count, buffer);

    response = buffer;  // echo message

    send(sock, response, strlen(response), 0);
    count++;
}
```

II.编译运行

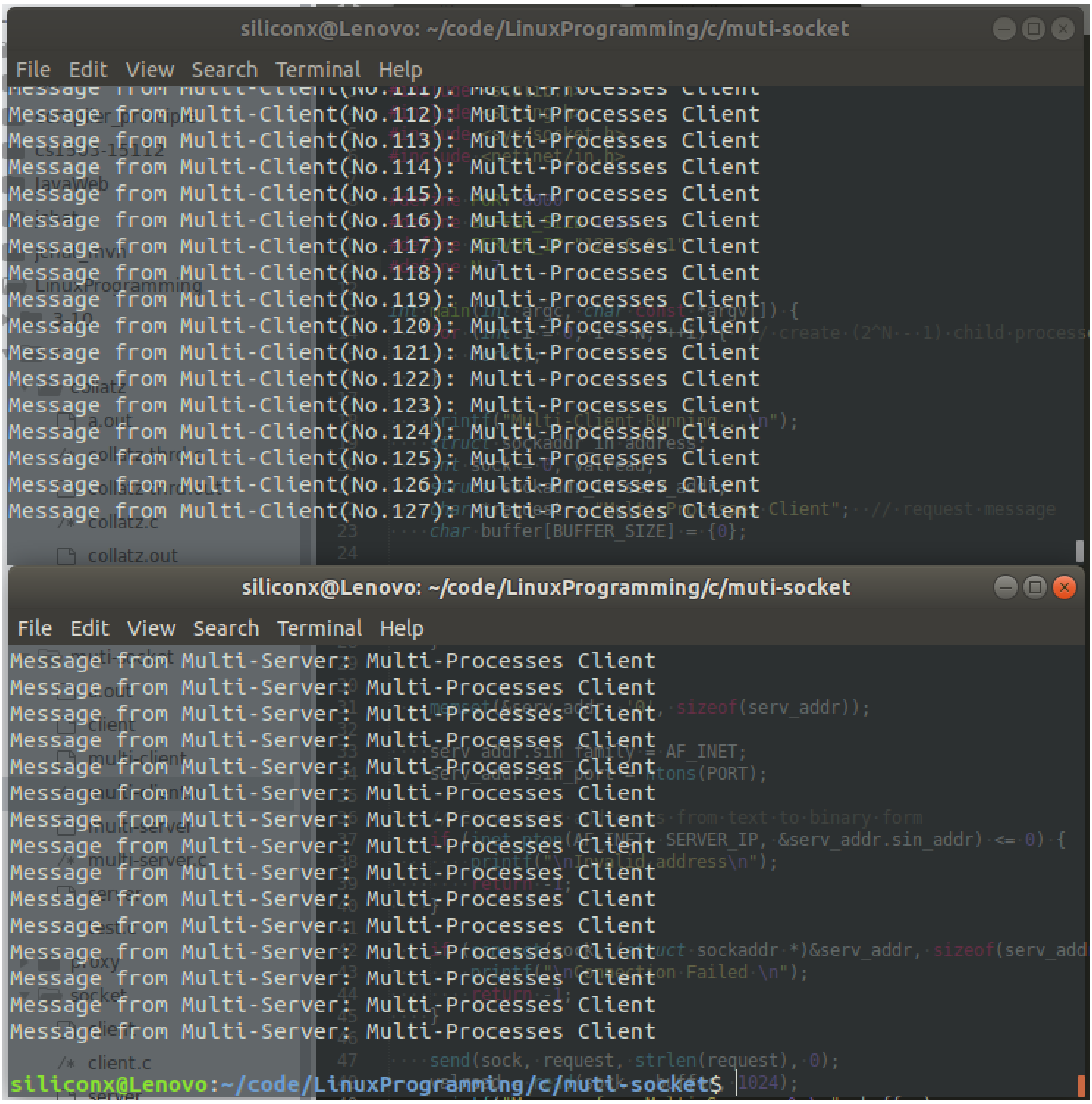


```
siliconx@Lenovo:~/code/LinuxProgramming/c/muti-socket$ gcc multi-server.c -o mul
ti-server -lpthread
siliconx@Lenovo:~/code/LinuxProgramming/c/muti-socket$ ./multi-server
Multi-Server Running...
```

```
#include <string.h>
#include <netinet/in.h>
#include <pthread.h> // for threading, link with -lpthread

#define PORT 8000
#define BUFFER_SIZE 1024

int count = 0; // counting total requests
char *response; // Response string
char buffer[BUFFER_SIZE] = {0}; // message buffer

void *msg_handler(void*);

int main(int argc, char const *argv[]) {
    printf("Multi-Server Running...\n");
    int server_fd, new_socket;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
```

```
siliconx@Lenovo:~/code/LinuxProgramming/c/muti-socket$ gcc multi-client.c -o mul
ti-client
siliconx@Lenovo:~/code/LinuxProgramming/c/muti-socket$ ./multi-client
```

```
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Forcefully attaching socket to the port
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPOR
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }

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

    // attaching socket to the port
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address
        perror("bind failed");
        exit(EXIT_FAILURE);
```


siliconx@Lenovo: ~/code/LinuxProgramming/c/muti-socket
File Edit View Search Terminal Help

```
Message from Multi-Client(No.111): Multi-Processes Client
Message from Multi-Client(No.112): Multi-Processes Client
Message from Multi-Client(No.113): Multi-Processes Client
Message from Multi-Client(No.114): Multi-Processes Client
Message from Multi-Client(No.115): Multi-Processes Client
Message from Multi-Client(No.116): Multi-Processes Client
Message from Multi-Client(No.117): Multi-Processes Client
Message from Multi-Client(No.118): Multi-Processes Client
Message from Multi-Client(No.119): Multi-Processes Client
Message from Multi-Client(No.120): Multi-Processes Client
Message from Multi-Client(No.121): Multi-Processes Client
Message from Multi-Client(No.122): Multi-Processes Client
Message from Multi-Client(No.123): Multi-Processes Client
Message from Multi-Client(No.124): Multi-Processes Client
Message from Multi-Client(No.125): Multi-Processes Client
Message from Multi-Client(No.126): Multi-Processes Client
Message from Multi-Client(No.127): Multi-Processes Client
```

siliconx@Lenovo: ~/code/LinuxProgramming/c/muti-socket
File Edit View Search Terminal Help

```
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
Message from Multi-Server: Multi-Processes Client
siliconx@Lenovo:~/code/LinuxProgramming/c/muti-socket$
```

实验思考

- 理解并使用多进程机制
- 理解并使用多线程机制
- 将多线程运用到socket上