

# 南昌大学实验报告7

---

姓名：谢志彬 学号：6103115112

邮箱地址：[siliconx@163.com](mailto: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  $N \geq 7$

#### I.编写multi-client.c

```
// Client side
#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"
#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
File Edit View Search Terminal Help
siliconx@Lenovo:~/code/LinuxProgramming/c/socket$ gcc server.c -o server
siliconx@Lenovo:~/code/LinuxProgramming/c/socket$ ./server
Server Running...
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <unistd.h>
6  #include <sys/socket.h>
7  #include <netinet/in.h>
8  #define PORT 8000
9  #define BUFFER_SIZE 1024
10
11 int main(int argc, char const *argv[]) {
12     printf("Server Running...\n");
13     int server_fd, new_socket, valread;
14     struct sockaddr_in address;
15     int opt = 1;
16     int addrlen = sizeof(address);
17     char buffer[1024] = {0};
18     char *response; // Response string
19     int count = 0;
20
21     // socket file descriptor
22     if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
23         perror("socket failed");
24         exit(EXIT_FAILURE);
25     }
26
27     // attaching socket to the port
28     if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
29         perror("bind failed");
30         exit(EXIT_FAILURE);
31     }
32
33     while(1) {
34         if (listen(server_fd, 3) < 0) {
35             perror("listen");
36             exit(EXIT_FAILURE);
37         }
38         int client_fd = accept(server_fd, (struct sockaddr *)&address, &addrlen);
39         if (client_fd < 0) {
40             perror("accept");
41             exit(EXIT_FAILURE);
42         }
43         read(client_fd, buffer, BUFFER_SIZE);
44         printf("Client: %s\n", buffer);
45         response = "I am a server";
46         write(client_fd, response, strlen(response));
47         close(client_fd);
48         count++;
49     }
50     printf("Total clients: %d\n", count);
51     return 0;
52 }
```

```
siliconx@Lenovo: ~/code/LinuxProgramming/c/multi-socket
File Edit View Search Terminal Help
siliconx@Lenovo:~/code/LinuxProgramming/c/multi-socket$ gcc multi-client.c -o multi-client
siliconx@Lenovo:~/code/LinuxProgramming/c/multi-socket$ ./multi-client |
30     exit(EXIT_FAILURE);
31 }
32
33 address.sin_family = AF_INET;
34 address.sin_addr.s_addr = INADDR_ANY;
35 address.sin_port = htons(PORT);
36
37 // attaching socket to the port
38 if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
39     perror("bind failed");
40     exit(EXIT_FAILURE);
41 }
42
43 while(1) {
44     if (listen(server_fd, 3) < 0) {
45         perror("listen");
46         exit(EXIT_FAILURE);
47     }
48     int client_fd = accept(server_fd, (struct sockaddr *)&address, &addrlen);
49     if (client_fd < 0) {
50         perror("accept");
51         exit(EXIT_FAILURE);
52     }
53     read(client_fd, buffer, BUFFER_SIZE);
54     printf("Client: %s\n", buffer);
55     response = "I am a server";
56     write(client_fd, response, strlen(response));
57     close(client_fd);
58     count++;
59 }
60 printf("Total clients: %d\n", count);
61 return 0;
62 }
```

[illegible]

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

```

// 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
File Edit View Search Terminal Help
siliconx@Lenovo:~/code/LinuxProgramming/c/muti-socket$ gcc multi-server.c -o multi-server -lpthread
siliconx@Lenovo:~/code/LinuxProgramming/c/muti-socket$ ./multi-server
Multi-Server Running...
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <string.h>
6  #include <unistd.h>
7  #include <netinet/in.h>
8  #include <pthread.h> // for threading, link with -lpthread
9
10 #define PORT 8000
11 #define BUFFER_SIZE 1024
12
13 int count = 0; // counting total requests
14 char *response; // Response string
15 char buffer[BUFFER_SIZE] = {0}; // message buffer
16
17 void *msg_handler(void*);
18
19 int main(int argc, char const *argv[]) {
20     printf("Multi-Server Running...\n");
21     int server_fd, new_socket;
22     struct sockaddr_in address;
23     int opt = 1;
24     int addrlen = sizeof(address);
25
30     if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
31         perror("socket failed");
32         exit(EXIT_FAILURE);
33     }
34
35     // Forcefully attaching socket to the port
36     if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, 1, 0) < 0) {
37         perror("setsockopt");
38         exit(EXIT_FAILURE);
39     }
40
41     address.sin_family = AF_INET;
42     address.sin_addr.s_addr = INADDR_ANY;
43     address.sin_port = htons(PORT);
44
45     // attaching socket to the port
46     if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
47         perror("bind failed");
48         exit(EXIT_FAILURE);
49     }
50     listen(server_fd, 5);
51     while (1) {
52         struct sockaddr_in client_addr;
53         socklen_t client_addr_len = sizeof(client_addr);
54         int new_socket = accept(server_fd, (struct sockaddr *)&client_addr, &client_addr_len);
55         if (new_socket < 0) {
56             perror("accept failed");
57             continue;
58         }
59         pthread_t thread;
60         pthread_create(&thread, NULL, msg_handler, (void *)new_socket);
61     }
62     return 0;
63 }
```



