TCP通信实验

Server

```
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>

double send_dg(int chat_fd);

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

    int server_fd = socket(AF_INET, SOCK_STREAM, 0); // create a socket file descriptor: ipv4, TCP, ip
    if (server_fd < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in address = {
            .sin_family = AF_INET, // ipv4
            .sin_addr.s_addr = INADDR_ANY, // accept any incoming messages
            .sin_port = htons(9090), // port 9090
    }; // client addr

    int addrlen = sizeof(address);
    if (bind(server_fd, (struct sockaddr *) &address, addrlen) < 0) { // bind connection
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

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

    int stream_fd = accept(server_fd, (struct sockaddr *) &address, (socklen_t *) &addrlen); // accept connection and bind it to new file descriptor
    if (stream_fd < 0) {
        perror("accept failed");
        exit(EXIT_FAILURE);
    }

    send_dg(stream_fd);
```

```
    close(server_fd);

    return 0;
}

double send_dg(int chat_fd) {
    const int len = 1024;
    char buff[len];
    int n;
    while (1) {
        bzero(buff, len);
        read(chat_fd, buff, len);
        printf("%s\n", buff);

        bzero(buff, len);
        n = 0;
        while ((buff[n++] = getchar()) != '\n') { ;
        }

        write(chat_fd, buff, n);
        if (strncmp("exit", buff, 4) == 0) {
            printf("Server Exit\n");
            break;
        }
    }
}
```

Client

```
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
double send_dg(int chat_fd);

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

    int client_fd = socket(AF_INET, SOCK_STREAM, 0); // create a socket file
descriptor: ipv4, TCP, ip
    if (client_fd < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in address={
            .sin_family = AF_INET, // ipv4
            .sin_port = htons(9090), // port 9090
```

```
    }; // server addr

    if (inet_pton(AF_INET, "127.0.0.1", &address.sin_addr) <= 0) { // check
validity of ip addr
        perror("Invalid address");
        exit(EXIT_FAILURE);
    }

    if (connect(client_fd, (struct sockaddr *) &address, sizeof(address))) { //
connect to server
        perror("Connection failed");
        exit(EXIT_FAILURE);
    }

    send_dg(client_fd);
    close(client_fd);

    return 0;
}

double send_dg(int chat_fd) {
    const int len = 1024;
    char buff[len];
    int n;
    while (1) {
        bzero(buff, len);
        n = 0;
        while ((buff[n++] = getchar()) != '\n') { ;
        }

        write(chat_fd, buff, n);

        bzero(buff, len);
        read(chat_fd, buff, len);
        printf("%s\n", buff);
        if (strncmp("exit", buff, 4) == 0) {
            printf("Server Exit\n");
            break;
        }
    }
}
```

UDP通信实验

Server

```
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
```

```
#include <netinet/in.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>

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

    int server_fd = socket(AF_INET, SOCK_DGRAM, 0); // create a socket file
descriptor: ipv4, UDP, ip
    if (server_fd < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in address = {
            .sin_family = AF_INET, // ipv4
            .sin_addr.s_addr = INADDR_ANY, // accept any incoming messages
            .sin_port = htons(9090), // port 9090
    }; // client addr

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

    int i;
    int n = 0;
    struct sockaddr_in cliaddr; int len;
    for(;;) {
        recvfrom(server_fd, &i, sizeof(i), MSG_WAITALL, (struct sockaddr *)
&cliaddr, (socklen_t *) &len);
        if(i == 1000) {
            break;
        }
        n++;
    }
    printf("%d\n", n);

    close(server_fd);

    return 0;
}
```

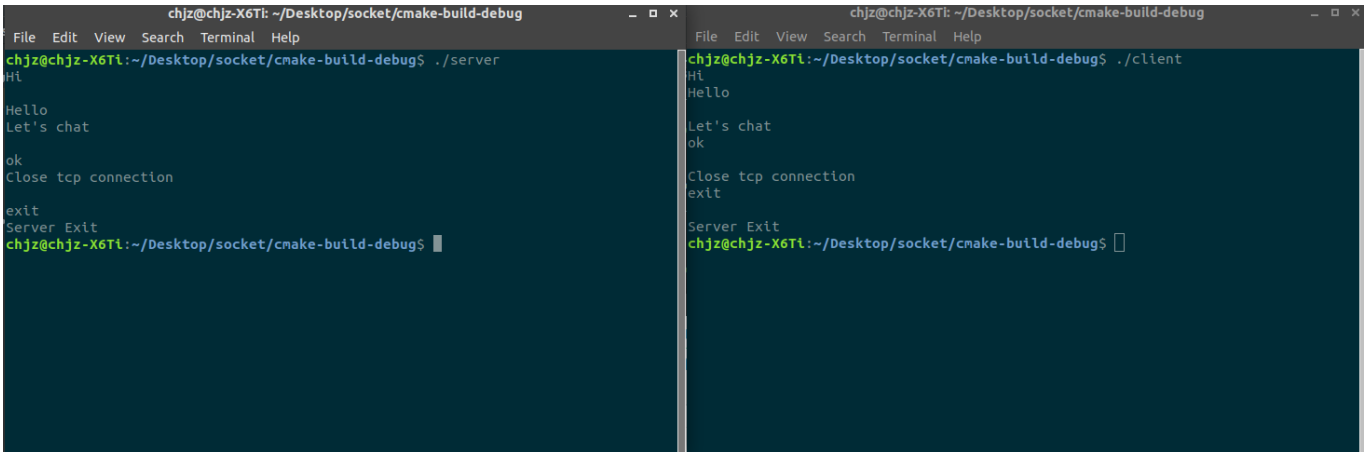
Client

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

```c
#include <string.h>
#include <arpa/inet.h>

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

    int client_fd = socket(AF_INET, SOCK_DGRAM, 0); // create a socket file
descriptor: ipv4, UDP, ip
    if (client_fd < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in address={
            .sin_family = AF_INET, // ipv4
            .sin_port = htons(9090), // port 9090
            .sin_addr.s_addr = INADDR_ANY,
    }; // server addr

    if (inet_pton(AF_INET, "127.0.0.1", &address.sin_addr) <= 0) { // check
validity of ip addr
        perror("Invalid address");
        exit(EXIT_FAILURE);
    }

    const int len = 1000;
    for (int i = 0; i < len + 1; i++) {
        sendto(client_fd, &i, sizeof(i), MSG_CONFIRM, (const struct sockaddr
*)&address, sizeof(address));
    }
    printf("1000 message sent");

    close(client_fd);

    return 0;
}
```

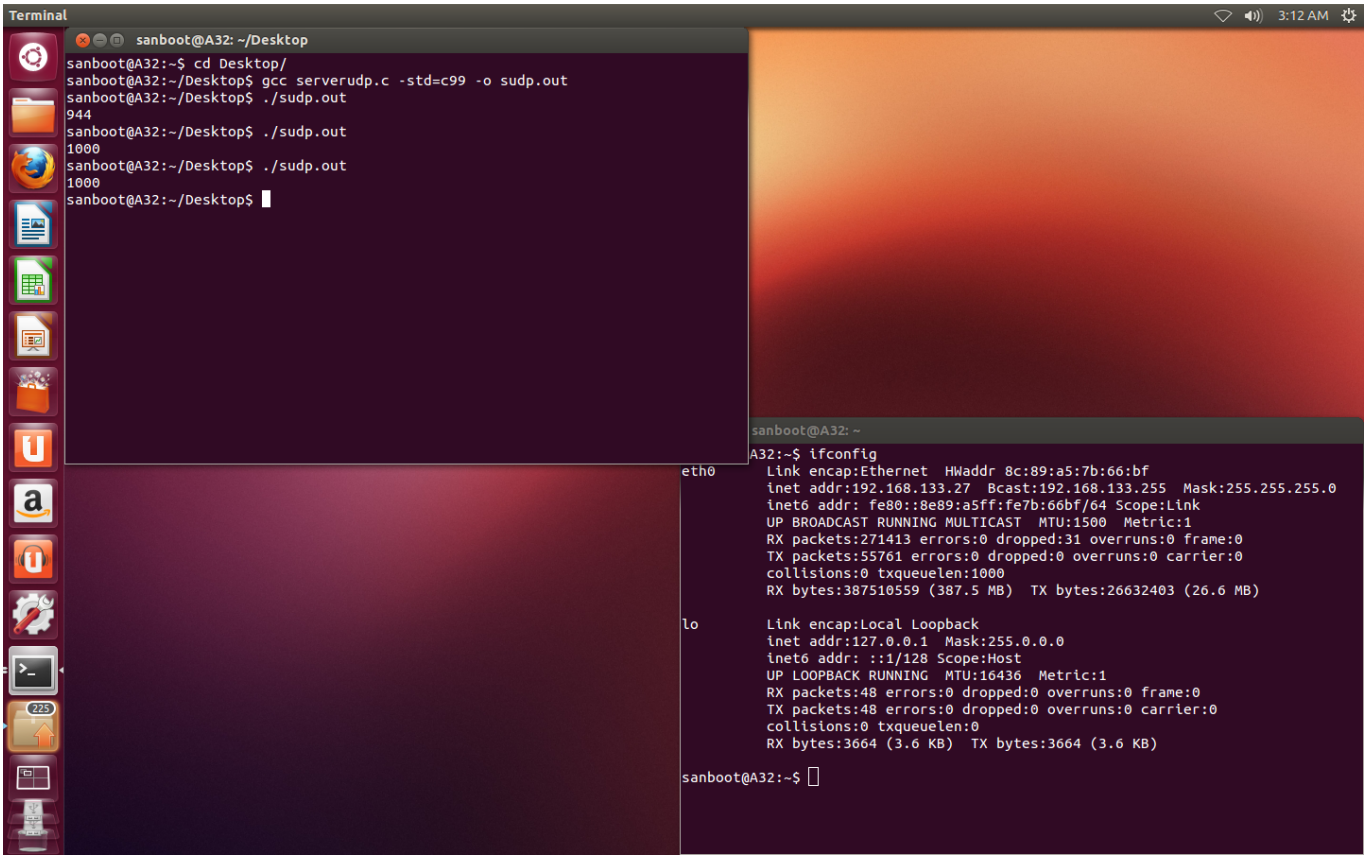
实验结果

TCP通信

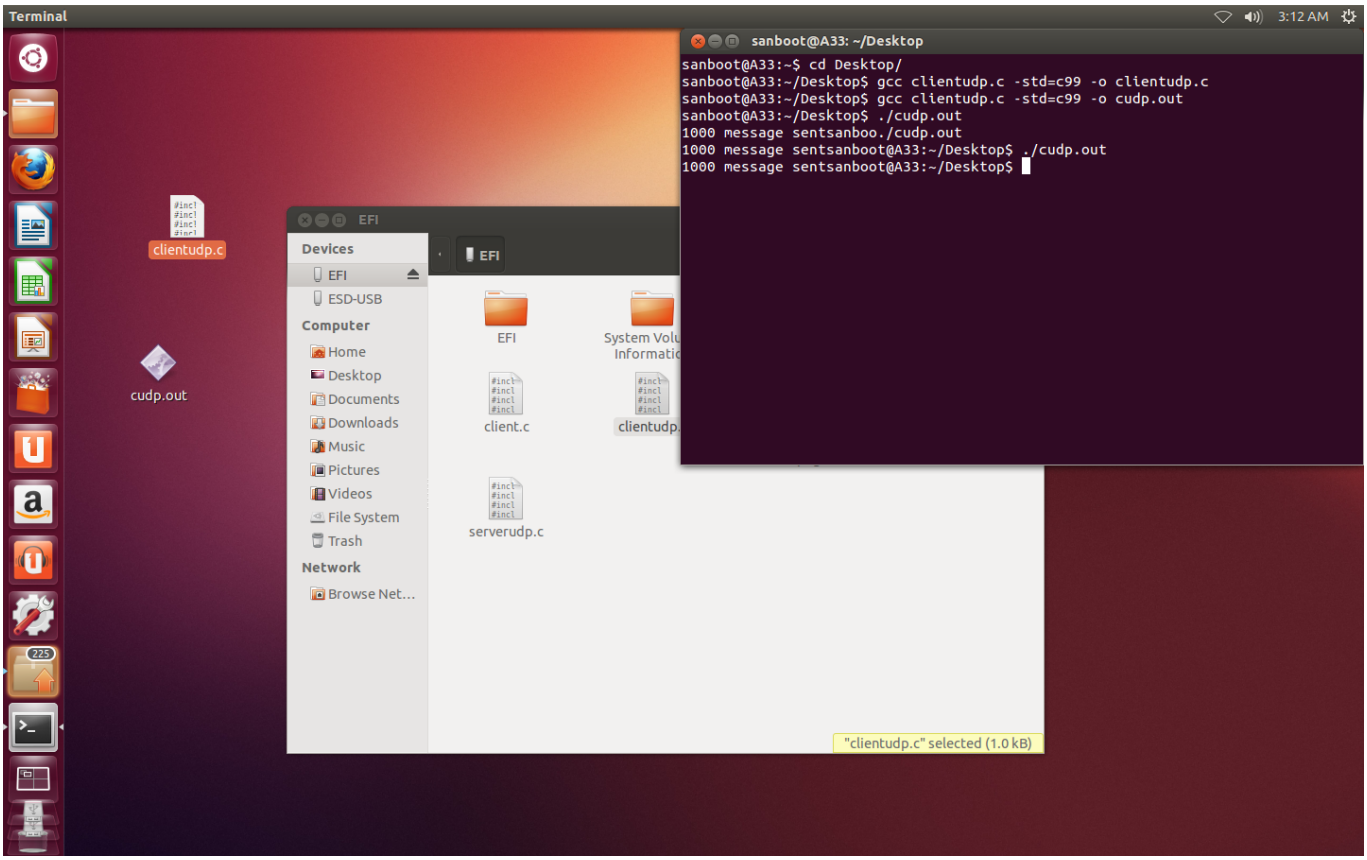


UDP通信

接收端



发送端


Terminal
3:12 AM
sanboot@A33: ~/Desktop
sanboot@A33:~$ cd Desktop/
sanboot@A33:~/Desktop$ gcc clientudp.c -std=c99 -o clientudp.c
sanboot@A33:~/Desktop$ gcc clientudp.c -std=c99 -o cudp.out
sanboot@A33:~/Desktop$ ./cudp.out
1000 message sentsanboo./cudp.out
1000 message sentsanboot@A33:~/Desktop$ ./cudp.out
1000 message sentsanboot@A33:~/Desktop$
clientudp.c
cudp.out
EFI
Devices
EFI
ESD-USB
Computer
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
File System
Trash
Network
Browse Net...
client.c
serverudp.c
"clientudp.c" selected (1.0 kB)