Networks Lab – Client (Week 4)

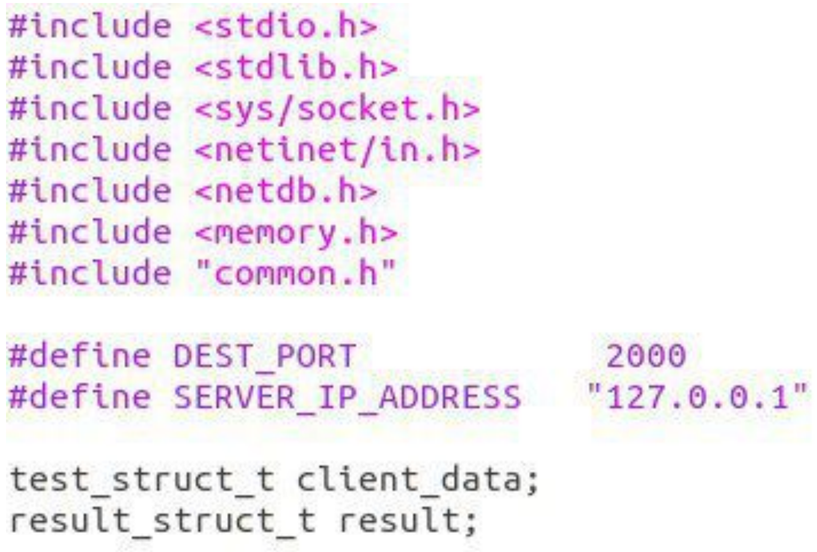
Nikita Bogomazov

Innopolis University

n.bogomazov@innopolis.ru

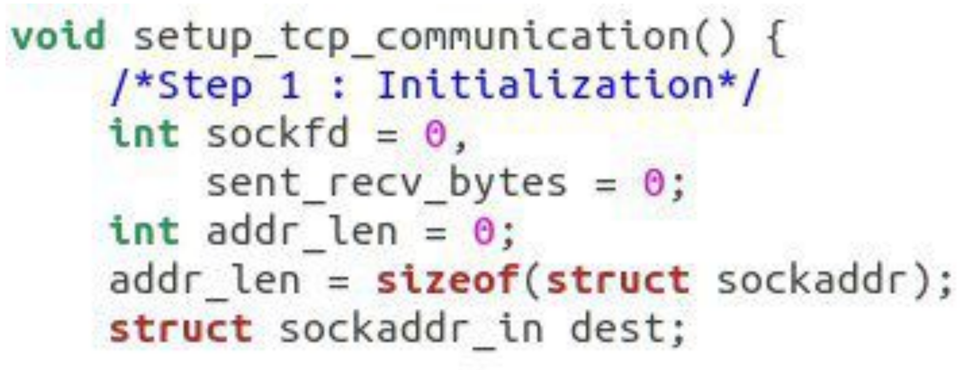
February 14, 2019

Initialize variables (1 / 2)



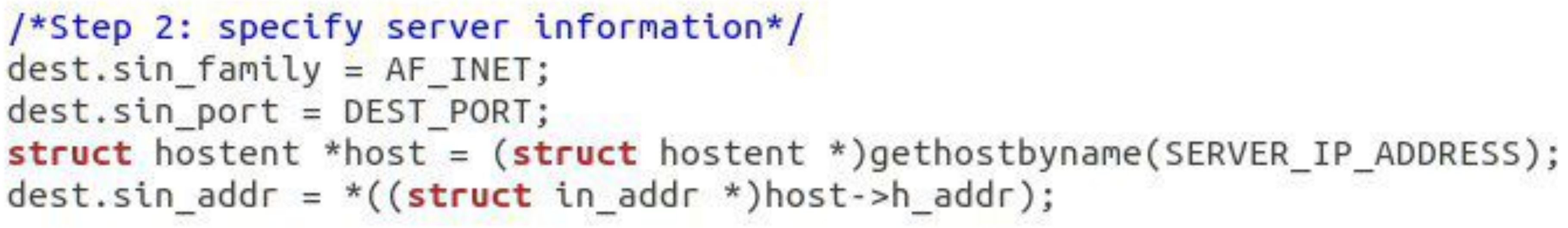
```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <memory.h>
#include "common.h"

#define DEST_PORT            2000
#define SERVER_IP_ADDRESS   "127.0.0.1"

test_struct_t client_data;
result_struct_t result;
```

Initialize variables (2 / 2)



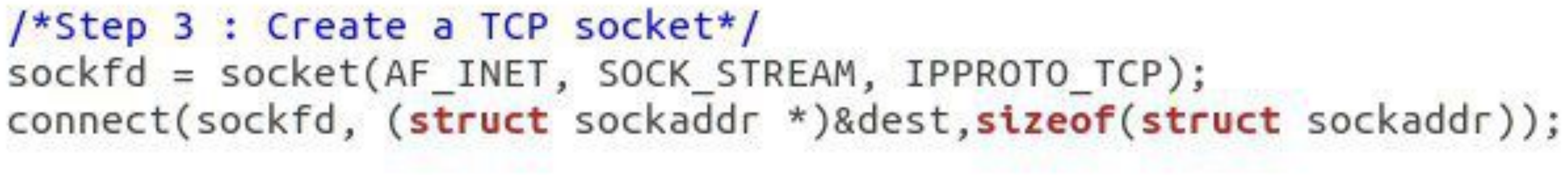
```
void setup_tcp_communication() {
    /*Step 1 : Initialization*/
    int sockfd = 0,
        sent_recv_bytes = 0;
    int addr_len = 0;
    addr_len = sizeof(struct sockaddr);
    struct sockaddr_in dest;
```

Server credentials



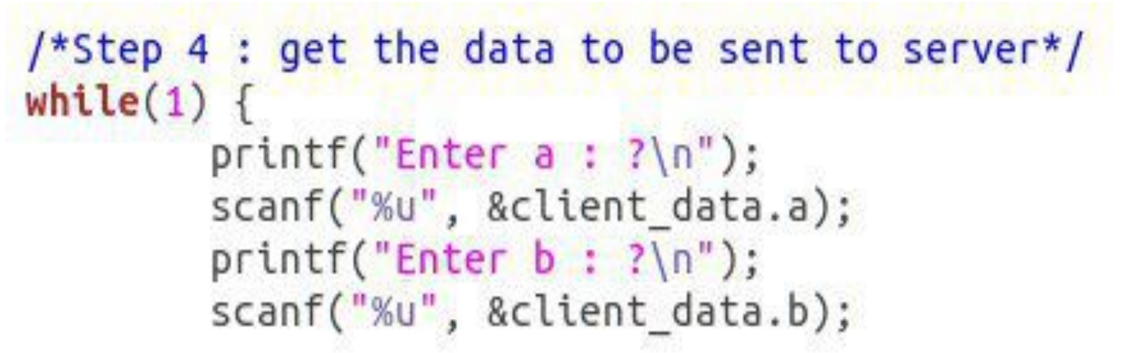
```
/*Step 2: specify server information*/
dest.sin_family = AF_INET;
dest.sin_port = DEST_PORT;
struct hostent *host = (struct hostent *)gethostbyname(SERVER_IP_ADDRESS);
dest.sin_addr = *((struct in_addr *)host->h_addr);
```

Create communication socket



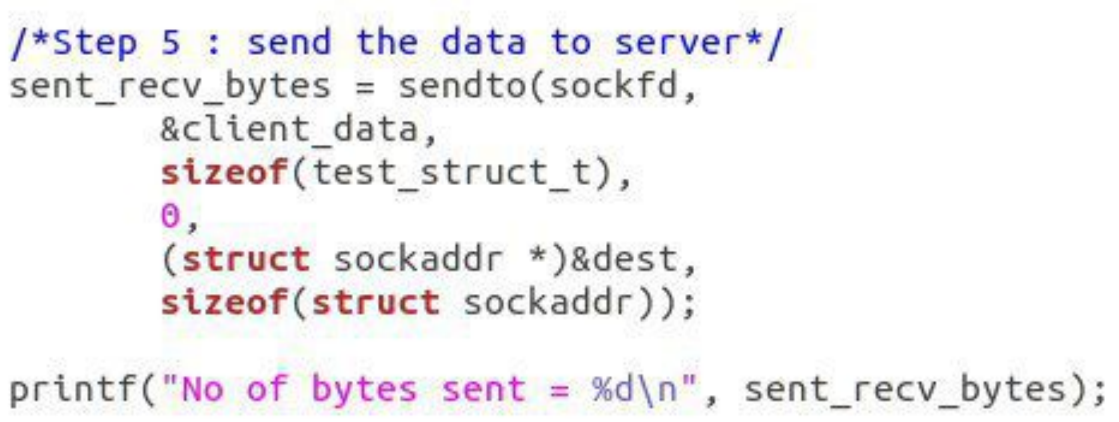
```
/*Step 3 : Create a TCP socket*/
sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
connect(sockfd, (struct sockaddr *)&dest,sizeof(struct sockaddr));
```

Send data (1 / 2)



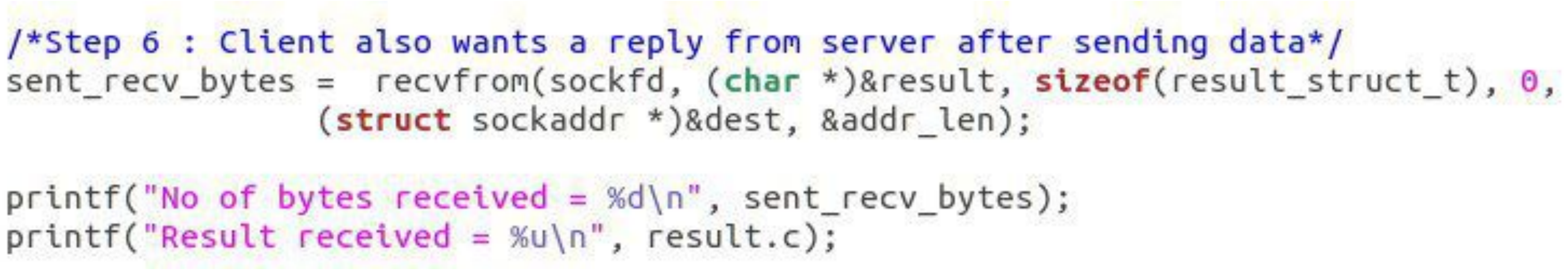
```
/*Step 4 : get the data to be sent to server*/
while(1) {
        printf("Enter a : ?\n");
        scanf("%u", &client_data.a);
        printf("Enter b : ?\n");
        scanf("%u", &client_data.b);
```

Send data (2 / 2)



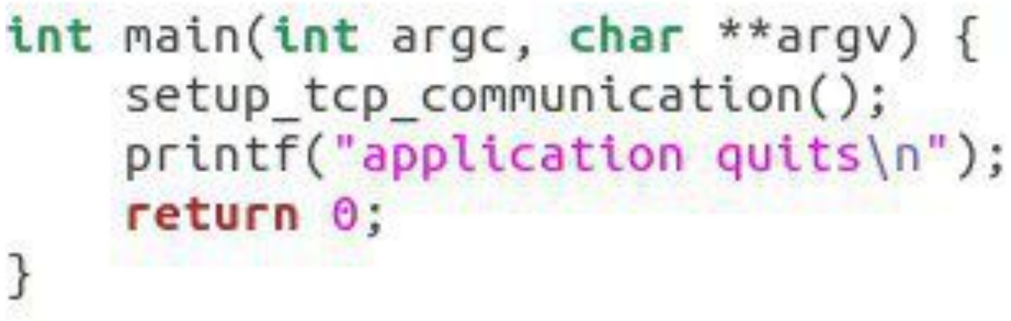
```
/*Step 5 : send the data to server*/
sent_recv_bytes = sendto(sockfd,
       &client_data,
       sizeof(test_struct_t),
       0,
       (struct sockaddr *)&dest,
       sizeof(struct sockaddr));

printf("No of bytes sent = %d\n", sent_recv_bytes);
```

Get response



```
/*Step 6 : Client also wants a reply from server after sending data*/
sent_recv_bytes =  recvfrom(sockfd, (char *)&result, sizeof(result_struct_t), 0,
                (struct sockaddr *)&dest, &addr_len);

printf("No of bytes received = %d\n", sent_recv_bytes);
printf("Result received = %u\n", result.c);
```

Main



```c
int main(int argc, char **argv) {
    setup_tcp_communication();
    printf("application quits\n");
    return 0;
}
```

Your task

1) Run server.c and then client.c (order matters) and make sure that your connection is working
2) Modify the type of your msg to represent a student info (name, age, group) and try to send it to server (Modify the server code also)
3) Run your network configuration from week 2 and set up your client and server on the distant machines. After that send the modified student info and take screenshots of the result. Submit an archive with modified code and screens as one (1!!!!!!) file.