

MP3 Report

Michael Hagaman – report, server, RRQ, report, testing

Usage:

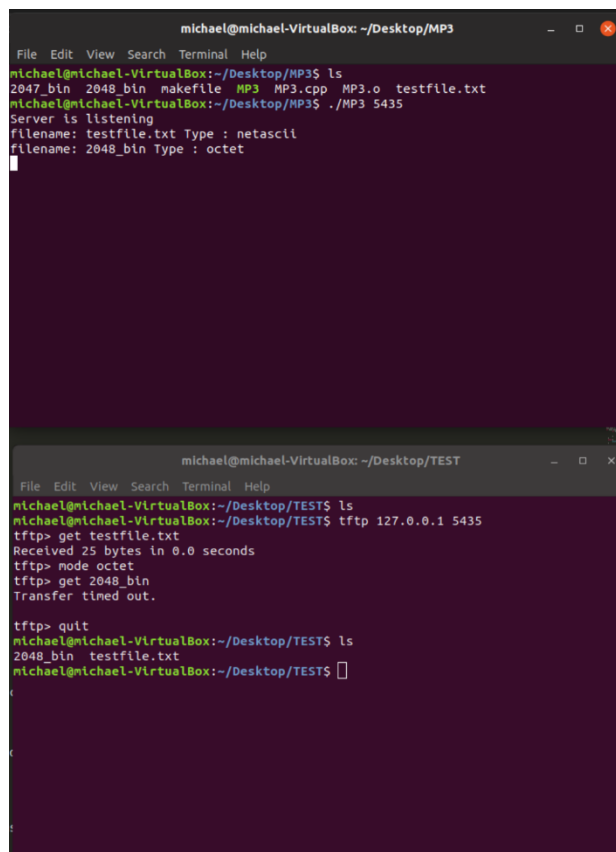
To run the code you can use the make file to compile. To run the server us `./MP3 [portnumber]`. I used the TFTP client on ubuntu. To get this to work I used `"tftp 127.0.0.1 [portnumber]"`. The server and client MUST run in different directories.

Code:

The server connects to the TFTP client and receives the initial get/put command. The server is able to parse the message and get the message type, file name, and transfer mode. The the TFTP clients sends a get request I am able to send the first 512 bytes (first packet) of data and the client saves said data as a file. I am unable to send more then one packet per transfer. This works for both octet and netascii. After the first packet is sent the client times out because the server is unable to execute the sendto function. The server is able to read the data from the file and insert it into the struct. The struct contains 3 message types for request, data transfer and ack.

Testing:

I am able to send a file that is less then 512 bytes as see below. When ever I send a file that is greater then 512 bytes the first 512 bytes is transferred and saved in the client directory.



```
michael@michael-VirtualBox: ~/Desktop/MP3
File Edit View Search Terminal Help
michael@michael-VirtualBox:~/Desktop/MP3$ ls
2047_bin 2048_bin makefile MP3 MP3.cpp MP3.o testfile.txt
michael@michael-VirtualBox:~/Desktop/MP3$ ./MP3 5435
Server is listening
filename: testfile.txt Type : netascii
filename: 2048_bin Type : octet

michael@michael-VirtualBox: ~/Desktop/TEST
File Edit View Search Terminal Help
michael@michael-VirtualBox:~/Desktop/TEST$ ls
michael@michael-VirtualBox:~/Desktop/TEST$ tftp 127.0.0.1 5435
tftp> get testfile.txt
Received 25 bytes in 0.0 seconds
tftp> mode octet
tftp> get 2048_bin
Transfer timed out.

tftp> quit
michael@michael-VirtualBox:~/Desktop/TEST$ ls
2048_bin testfile.txt
michael@michael-VirtualBox:~/Desktop/TEST$
```

Bugs:

The server is unable to send more than one packet. After the first packet is sent the server is unable to execute the sendto code. Sometimes the code hangs before it constructs the outgoing message. When this happened, you need to remake the code. Run "make clean" then "make" this should fix the hanging error. Other than that, I cannot get past the first packet.

Code:

Main:

```
int main(int argc, char *argv[])
{
    if(argc != 2){
        printf("Please enter ./server (port number)");
        exit(0);
    }
    int port_num = atoi(argv[1]);
    int sockfd;
    struct sockaddr_in serveraddr;
    int temp;
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0){
        printf("ERROR: Creation of Socket failed");
        exit(1);
    }
    memset(&serveraddr, 0, sizeof(serveraddr));
    serveraddr.sin_family = AF_INET;
    serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
    serveraddr.sin_port = htons(port_num);

    if(bind(sockfd, (const struct sockaddr *) &serveraddr, sizeof(serveraddr)) < 0){
        printf("ERROR: Socket Bind failed");
        exit(1);
    }
}
```

```

printf("Server is listening\n");

while(true){

    struct sockaddr_in clientsock;

    socklen_t clientlen = sizeof(clientsock);

    tftp_message message;

    int16_t msg_type;

    ssize_t msg;

    if((msg = recvfrom(sockfd, &message, sizeof message, 0, (struct sockaddr*)&clientsock,
&clientlen))< 0){

        printf("ERROR: RECV");

    }

    ///cout << "message value " << msg << endl;

    //printf("Received req: %d and %s\n", htons(message.request.msg_type),
message.request.DATA);

    msg_type = ntohs(message.request.msg_type);

    if (msg_type == RRQ || msg_type == WRQ) {

        if (fork() == 0) {

            handle_child(&message, msg, &clientsock, clientlen);

            exit(0);

        }

    }

    else {

        printf("ERROR: msg_type \n");

    }

}

```

Handler:

```
void handle_child(tftp_message *temp, ssize_t msg, struct sockaddr_in *clientsock, socklen_t clientlen){
```

```
    int newsockfd;
```

```
    uint16_t mess_type;
```

```
    FILE *fd;
```

```
    fstream fs;
```

```
    if((newsockfd = socket(AF_INET, SOCK_DGRAM, 0))==-1){
```

```
        printf("ERROR SOCKET CHILD");
```

```
        exit(1);
```

```
    }
```

```
    printf("filename: ");
```

```
    char *filename = ((char*) temp->request.DATA);
```

```
    mess_type = ntohs(temp->request.msg_type);
```

```
    char *transfertype = strchr(filename, '\0') + 1;
```

```
    string ttstring(transfertype);
```

```
    string filestring(filename);
```

```
    cout<<filestring;
```

```
    printf(" Type : ");
```

```
    cout<<ttstring<<endl;
```

```
    if(!file_exists(filestring)){
```

```
        printf("FILE NOT FOUND");
```

```
        exit(1);
```

```
    }
```

```
    if (ttstring.compare("netascii")==0){
```

```
        //fopen
```

```
}
```

```
fd = fopen(filename, "r");
```

```
if(fd == NULL){
```

```
    printf("ERROR: fopen");
```

```
    exit(1);
```

```
}
```

```
if(mess_type == RRQ){
```

```
    tftp_message inboundmsg;
```

```
    //tftp_message outboundmsg;
```

```
    ssize_t read_length;
```

```
    int tempp;
```

```
    uint8_t out_data[512];
```

```
    uint16_t NUMpacket = 1;
```

```
    bool data_remaining = true;
```

```
    while(data_remaining){
```

```
        tftp_message outboundmsg;
```

```
        //cout << "before read" << endl;
```

```
        read_length = fread(out_data,1,sizeof out_data, fd);
```

```
        //printf("Out data: %s\n", out_data);
```

```
        cout << "Read lenght" << read_length << endl;
```

```
        if (read_length < 512){
```

```
            data_remaining = false;
```

```
        }
```

```
        outboundmsg.data_transfer.msg_type = htons(DATA);
```

```
        outboundmsg.data_transfer.packetNUM = htons(NUMpacket);
```

```

        NUMpacket++;

        memcpy(outboundmsg.data_transfer.data, out_data, read_length);

        //cout << "IN loop iter:" << NUMpacket << endl;

        //cout << "data_remaining "<< data_remaining << endl;

        int attempts=0;

        //cout << "attempts" << attempts << endl;

        while(attempts<9){

            printf("HERE");

            //tempp = sendto(newsockfd ,&outboundmsg, (4+read_length) ,0,
            (struct sockaddr *)&clientsock, clientlen);

            if((tempp = sendto(newsockfd ,&outboundmsg, (4+read_length) ,0,
            (struct sockaddr *)&clientsock, clientlen)) <0){

                printf("ERROR sendto in RRQ loop\n");

                //exit(0);

                //      //continue;

            }

            if((tempp = recvfrom(newsockfd, &inboundmsg, sizeof
            inboundmsg, 0, (struct sockaddr *)&clientsock, &clientlen))< 0){

                printf("ERROR: RECV in RRQ loop\n");

            }

            attempts++;

            //printf("inboundmsg: ");

            //printf("Received req: %d and %s \n", htons(inboundmsg.acknow.msg_type),
            htons(inboundmsg.acknow.packetNUM));

        }

        //cout << "After loop" << endl;

    }

```

```
    }  
    else if(mess_type == WRQ){  
    }  
    //printf("Transfer done");  
    //fclose(fd);  
    //close(newsockfd);  
    //exit(0);  
}
```

Letter from the Author:

I worked extremely hard on this code and it frustrating that I could not complete it. I am an ELEN student with little c/c++ experience. ELEN students only have to take CSCE 121 which does not teach you how to do anything near what is required for these assignments. The only prereq for this course is stats (ECEN 303). It is my opinion that these machine problems are incredibly difficult and do not assist me learning the material in the course. The protocols are simple to understand but difficult to implement with out any help from a professor/TA. I enjoyed the course overall since network architecture is an interesting subject.