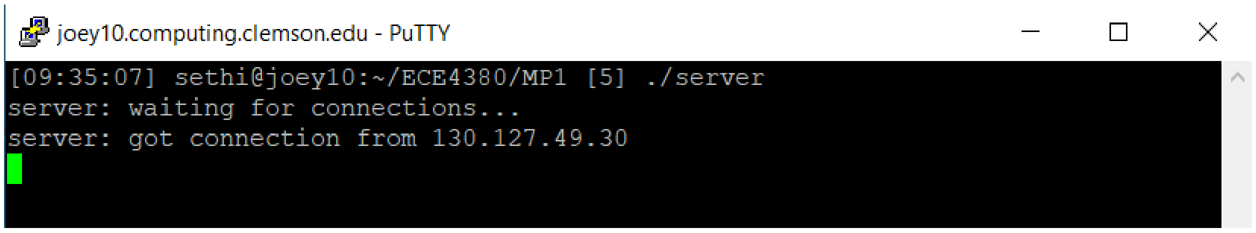
Rajat Sethi – ECE 4380 – MP1

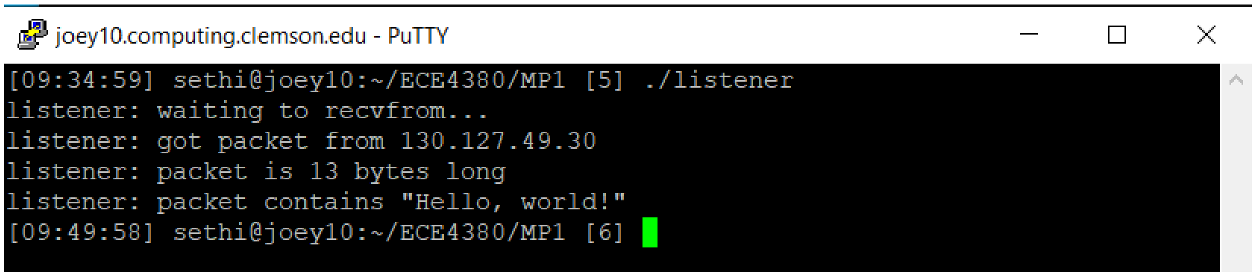
a.)

./server


joey10.computing.clemson.edu - PuTTY

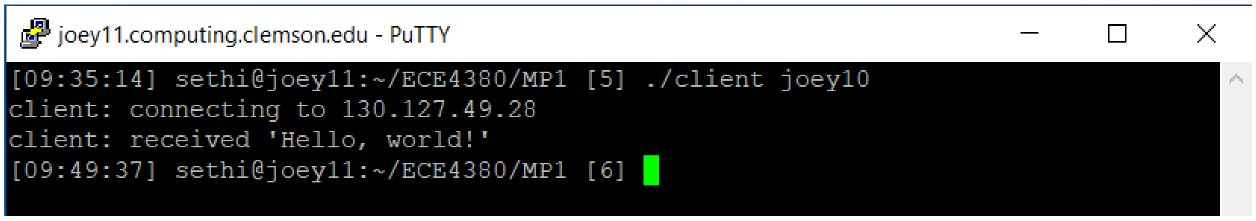
```
[09:35:07] sethi@joey10:~/ECE4380/MP1 [5] ./server
server: waiting for connections...
server: got connection from 130.127.49.30
```

./listener


joey10.computing.clemson.edu - PuTTY

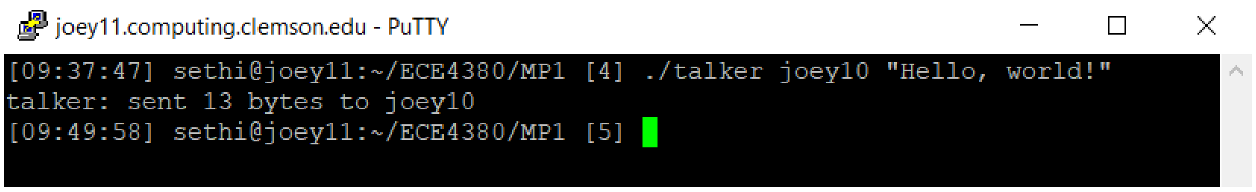
```
[09:34:59] sethi@joey10:~/ECE4380/MP1 [5] ./listener
listener: waiting to recvfrom...
listener: got packet from 130.127.49.30
listener: packet is 13 bytes long
listener: packet contains "Hello, world!"
[09:49:58] sethi@joey10:~/ECE4380/MP1 [6]
```

./client joey10


joey11.computing.clemson.edu - PuTTY

```
[09:35:14] sethi@joey11:~/ECE4380/MP1 [5] ./client joey10
client: connecting to 130.127.49.28
client: received 'Hello, world!'
[09:49:37] sethi@joey11:~/ECE4380/MP1 [6]
```

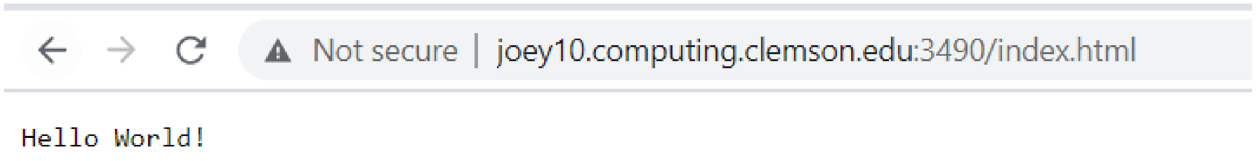
./talker joey10 “Hello, World!”


joey11.computing.clemson.edu - PuTTY

```
[09:37:47] sethi@joey11:~/ECE4380/MP1 [4] ./talker joey10 "Hello, world!"
talker: sent 13 bytes to joey10
[09:49:58] sethi@joey11:~/ECE4380/MP1 [5]
```

1.) The pairs of programs do NOT interfere with each other. This is likely because both pairs use a different port connection.

b.)

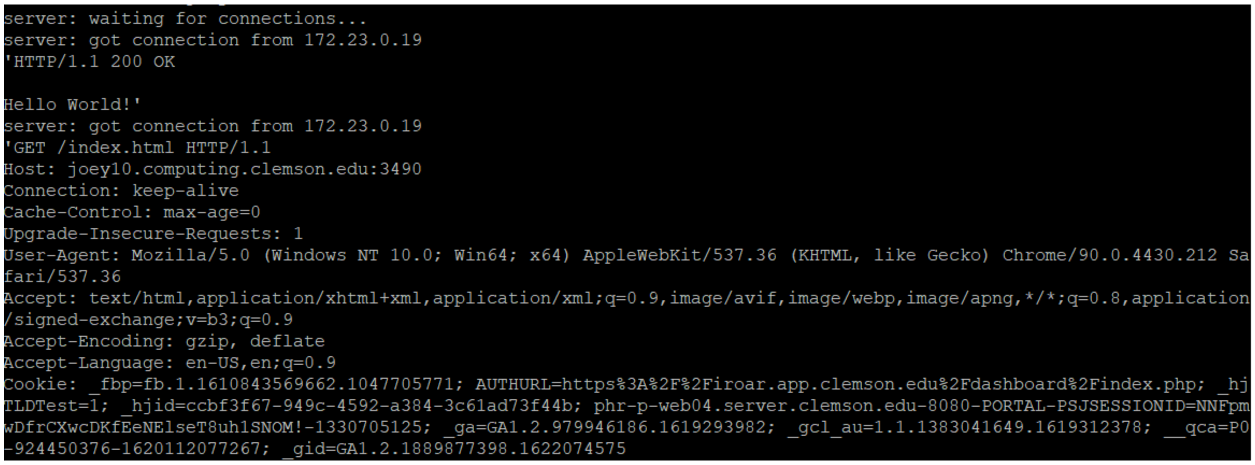


```
Hello World!
```

Changes made to server.c

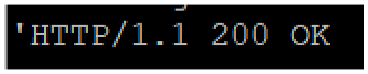
```
if (!fork()) { // this is the child process
        close(sockfd); // child doesn't need the listener
        char response[] = "HTTP/1.1 200 OK\n\nHello World!";
        if (send(new_fd, response, strlen(response), 0) == -1)
                perror("send");

        printf("'%s'\n",response);

        if ((numbytes = recv(new_fd, buf, MAXDATASIZE-1, 0)) == -1) {
                perror("recv");
                exit(1);
        }

        buf[numbytes] = '\0';

        printf("'%s'\n",buf);

        close(new_fd);
        exit(0);
}
close(new_fd);  // parent doesn't need this
```

New Server Print-Out



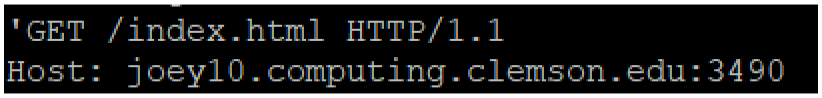
```
server: waiting for connections...
server: got connection from 172.23.0.19
'HTTP/1.1 200 OK

Hello World!'
server: got connection from 172.23.0.19
'GET /index.html HTTP/1.1
Host: joey10.computing.clemson.edu:3490
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Sa
fari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application
/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: _fbp=fb.1.1610843569662.1047705771; AUTHURL=https%3A%2F%2Firoar.app.clemson.edu%2Fdashboard%2Findex.php; _hj
TLDTest=1; _hjid=ccbf3f67-949c-4592-a384-3c61ad73f44b; phr-p-web04.server.clemson.edu-8080-PORTAL-PSJSESSIONID=NNFpm
wDfrCXwcDKfEeNElseT8uh1SNOM!-1330705125; _ga=GA1.2.979946186.1619293982; _gcl_au=1.1.1383041649.1619312378; __qca=P0
-924450376-1620112077267; _gid=GA1.2.1889877398.1622074575
```

START_LINE



```
'HTTP/1.1 200 OK
```

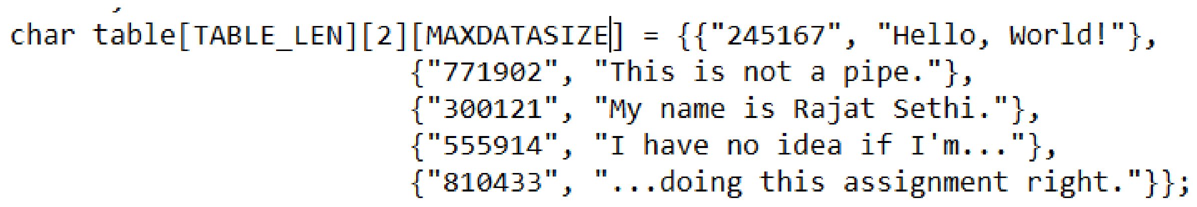
MESSAGE_HEADER



```
'GET /index.html HTTP/1.1
Host: joey10.computing.clemson.edu:3490
```

c.)

Table of "valid/arbitrary" identification numbers that the server can receive, along with their respective messages to show a successful send.



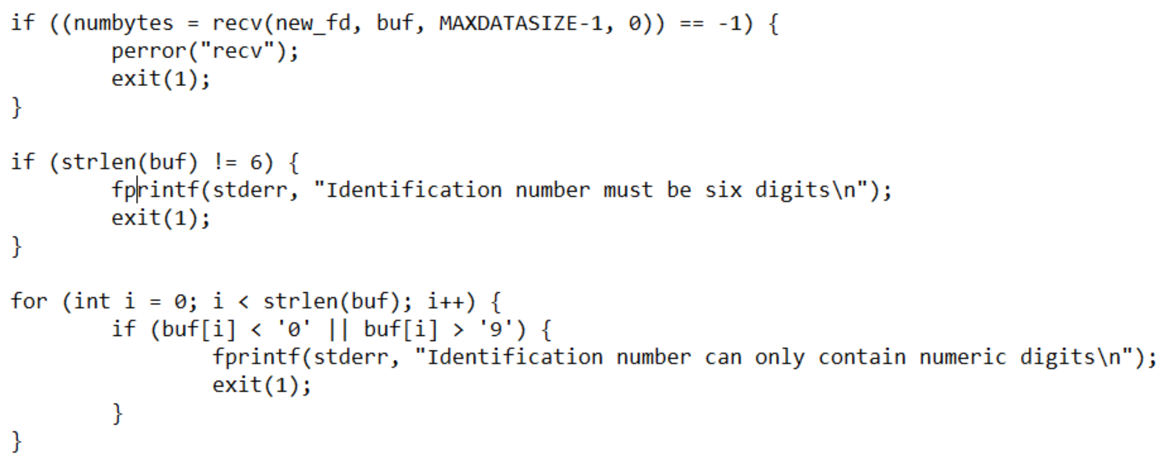
```
char table[TABLE_LEN][2][MAXDATASIZE] = {{"245167", "Hello, World!"},
                                          {"771902", "This is not a pipe."},
                                          {"300121", "My name is Rajat Sethi."},
                                          {"555914", "I have no idea if I'm..."},
                                          {"810433", "...doing this assignment right."}};
```

How the server protects itself from invalid id-numbers:

The first if-statement receives the data from the client.

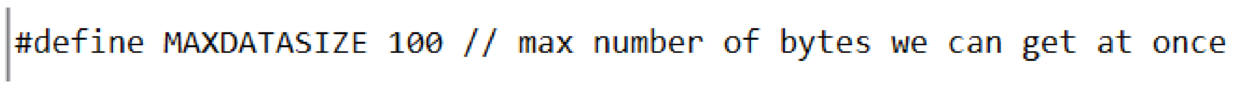
The second if-statement verifies that the id-number is exactly six digits.

The third if-statement verifies that all of the characters are numerical digits.



```
if ((numbytes = recv(new_fd, buf, MAXDATASIZE-1, 0)) == -1) {
        perror("recv");
        exit(1);
}

if (strlen(buf) != 6) {
        fprintf(stderr, "Identification number must be six digits\n");
        exit(1);
}

for (int i = 0; i < strlen(buf); i++) {
        if (buf[i] < '0' || buf[i] > '9') {
                fprintf(stderr, "Identification number can only contain numeric digits\n");
                exit(1);
        }
}
```

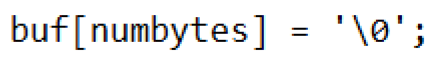
Limit to data that the server can send and receive:



```
#define MAXDATASIZE 100 // max number of bytes we can get at once
```

Since the server can only receive 100 bytes from the client, it is implied that the client can also receive 100 bytes from the server.

In the situation that the server sends too much data, the client places a delimiter on the string for however much memory it can hold.



```
buf[numbytes] = '\0';
```

Credit for original server and client programs go to Brian “Beej” Hall.