

## Assignment 2

### TCP:

#### Server

```

1 // Server side C/C++ program to demonstrate socket programming
2 #include <unistd.h>
3 #include <stdio.h>
4 #include <sys/socket.h>
5 #include <stdlib.h>
6 #include <netinet/in.h>
7 #include <string.h>
8 #include <arpa/inet.h>
9 #include <string.h>
10
11 int main(int argc, char *argv[]){
12
13     int socket_desc, c_read_size;
14     long unsigned int client_socket ;
15     struct sockaddr_in server, client;
16     //char client_message[1000];
17
18
19
20
21 // get a socket
22 socket_desc = socket(AF_INET, SOCK_STREAM, 0);
23 if (socket_desc == -1){
24     printf("could not create socket.");
25 }
26 puts("Socket created");
27
28 //fill the fields
29 server.sin_addr.s_addr = inet_addr("127.0.0.1");
30 server.sin_family = AF_INET;
31 server.sin_port = htons( 8888 );
32
33
34 //bind the socket to the port
35 if(bind(socket_desc, (struct sockaddr *)&server, sizeof(server))<0){
36     perror("bind failed. error");
37     return 1;
38 }
39
40 //start listening for incoming connections
41 puts("bind done");
42 listen(socket_desc, 3);
43
44 puts("waiting for incoming connections...");
45 c = accept (socket_desc, (struct sockaddr *)&client, (socklen_t*)&c);
46
47
48 if (client_socket == -1) {
49     perror("accept failed");
50     return 1;
51 }
52 puts("Connection accepted");
53
54 //while(read_size=recv(client_socket, client_message, 2000, 0)>0) {
55 while(1){
56     //variable for client message
57     char client_message[1000]="";
58     //read the message and assign it to client message
59     recv(client_socket, client_message, 3000, 0);
60     //show in console
61     puts(client_message);
62
63     //place holder for first digit
64     char str[10]="";
65     //place holder for second digit
66     char dup[10]="";
67     //variable to hold answer to the equation
68     char rstr[10]="";
69     //variable to assign an index to delete
70     int idxToDel=0;
71     //check if the equation was successful or not, default is 1 meaning success
72     int check=1;
73
74     //variable for success message
75     char success[]="Success: 1, Answer: ";
76     //variable for fail message
77     char fail[]="Success: 0, Error: Tried to divide by 0";
78
79     //add
80     //check if client message has a which is unique to add
81     char *quotPtr = strchr(client_message, 'a');
82     //if a exists
83     if(quotPtr != NULL){
84         //remove add portion from string
85         for(int i=0; i<=2; i++){
86             idxToDel = 0;
87             memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
88         }
89     }
90 }

```

```

78 //add
79 //check if client message has a which is unique to add
80 char *quotPtr = strchr(client_message, 'a');
81 //if a exists
82 if (quotPtr != NULL){
83     //remove 1st portion from string
84     for (int i=0; i<=2; i++){
85         idxToDel = 0;
86         memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
87     }
88     //copy the remain numbers to str and dup
89     strcpy(str, client_message);
90     strcpy(dup, client_message);
91     //remove second digit in str variable
92     idxToDel = 1;
93     memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
94     //remove 1st digit in dup
95     idxToDel = 0;
96     memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
97     //parse int the strings and add them
98     int x = atoi(str) + atoi(dup);
99     //convert the int to string and assign to rstr
100     sprintf(rstr, "%d", x);
101 }
102 //sub
103 //check if client message has s which is unique to sub
104 quotPtr = strchr(client_message, 's');
105 if (quotPtr != NULL){
106     //remove sub portion from string
107     for (int i=0; i<=2; i++){
108         idxToDel = 0;
109         memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
110     }
111     //copy the remain numbers to str and dup
112     strcpy(str, client_message);
113     strcpy(dup, client_message);
114     //remove second digit in str variable
115     idxToDel = 1;
116     memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
117 }
118 //mul
119 //check if client message has m which is unique to mul
120 quotPtr = strchr(client_message, 'm');
121 if (quotPtr != NULL){
122     //remove mul portion from string
123     for (int i=0; i<=2; i++){
124         idxToDel = 0;
125         memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
126     }
127     //copy the remain numbers to str and dup
128     strcpy(str, client_message);
129     strcpy(dup, client_message);
130     //remove second digit in str variable
131     idxToDel = 1;
132     memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
133     //remove 1st digit in dup
134     idxToDel = 0;
135     memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
136     //parse int the strings and mul them
137     int x = atoi(str) * atoi(dup);
138     //convert the int to string and assign to rstr
139     sprintf(rstr, "%d", x);
140 }
141 //div
142 //check if client message has v which is unique to div
143 quotPtr = strchr(client_message, 'v');
144 if (quotPtr != NULL){
145     //remove div portion from string
146     for (int i=0; i<=2; i++){
147         idxToDel = 0;
148         memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
149     }
150     //copy the remain numbers to str and dup
151     strcpy(str, client_message);
152     strcpy(dup, client_message);
153     //remove second digit in str variable
154     idxToDel = 1;
155     memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
156     //remove 1st digit in dup
157     idxToDel = 0;
158     memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
159     //parse int the strings and div them
160     int x = atoi(str) / atoi(dup);
161     //convert the int to string and assign to rstr
162     sprintf(rstr, "%d", x);
163 }

```

```

117 //copy the remain numbers to str and dup
118 strcpy(str, client_message);
119 strcpy(dup, client_message);
120 //remove second digit in str variable
121 idxToDel = 1;
122 memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
123 //remove 1st digit in dup
124 idxToDel = 0;
125 memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
126 //parse int the strings and sub them
127 int x = atoi(str) - atoi(dup);
128 //convert the int to string and assign to rstr
129 sprintf(rstr, "%d", x);
130 }
131 //mul
132 //check if client message has m which is unique to mul
133 quotPtr = strchr(client_message, 'm');
134 if (quotPtr != NULL){
135     //remove mul portion from string
136     for (int i=0; i<=2; i++){
137         idxToDel = 0;
138         memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
139     }
140     //copy the remain numbers to str and dup
141     strcpy(str, client_message);
142     strcpy(dup, client_message);
143     //remove second digit in str variable
144     idxToDel = 1;
145     memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
146     //remove 1st digit in dup
147     idxToDel = 0;
148     memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
149     //parse int the strings and mul them
150     int x = atoi(str) * atoi(dup);
151     //convert the int to string and assign to rstr
152     sprintf(rstr, "%d", x);
153 }
154 //div
155 //check if client message has v which is unique to div
156 quotPtr = strchr(client_message, 'v');
157 if (quotPtr != NULL){
158     //remove div portion from string
159     for (int i=0; i<=2; i++){
160         idxToDel = 0;
161         memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
162     }
163     //copy the remain numbers to str and dup
164     strcpy(str, client_message);
165     strcpy(dup, client_message);
166     //remove second digit in str variable
167     idxToDel = 1;
168     memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
169     //remove 1st digit in dup
170     idxToDel = 0;
171     memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
172     //parse int the strings and div them
173     int x = atoi(str) / atoi(dup);
174     //convert the int to string and assign to rstr
175     sprintf(rstr, "%d", x);
176 }

```

```

158 //div
159 //check if client message has v which is unique to div
160 quotPtr = strchr(client_message, 'v');
161 //if v exists
162 if (quotPtr != NULL){
163     //remove all portion from string
164     for (int i=0; i<=2; i++){
165         idxToDel = 0;
166         memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
167     }
168
169     //copy the remain numbers to str and dup
170 strcpy(str, client_message);
171 strcpy(dup, client_message);
172 //parse each digit in str variable
173 idxToDel = 1;
174 memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
175 //first digit in dup
176 idxToDel = 0;
177 memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
178 //if one of the digits is zero change check to zero, meaning unsuccessful
179 if (atoi(str)==0 || atoi(dup)==0){
180     check=0;
181 }
182 //parse double the strings and div them
183 double x= atoi(str) / atoi(dup);
184 //convert the double to string and assign to rstr
185 sprintf(rstr, "%f", x);
186 }
187 //if there was no zero, thus successful
188 if (check==1){
189     strcat(success, rstr);
190     write(client_socket, success, strlen(success));
191 }
192 //if there was a zero, thus unsuccessful
193 if (check==0){
194     write(client_socket, fail, strlen(fail));
195 }
196 }
197
198 }
199
200
201
202 if (read_size==0){
203     puts("client disconnected");
204     fflush(stdout);
205 }
206 while ((read_size--)){
207     perror("recv failed");
208 }
209
210
211 return 0;
212 }

```

## Client

```

1 #include <stdio.h>
2 #include <sys/socket.h>
3 #include <arpa/inet.h>
4 #include <string.h>
5 #include <unistd.h>
6 #include <string.h>
7 #include <stdlib.h>
8
9 //custom function to remove spaces
10 void remove_spaces(char* s);
11
12 int main(int argc, char *argv[])
13 {
14     int sock;
15
16     struct sockaddr_in server;
17
18     //char message[1000], server_reply[2000];
19     sock = socket(AF_INET, SOCK_STREAM, 0);
20     if (sock == -1)
21         printf("ERROR opening socket");
22     puts("socket created");
23
24     server.sin_addr.s_addr=inet_addr("127.0.0.1");
25     server.sin_family = AF_INET;
26     server.sin_port = htons(8080);
27
28     if (connect(sock, (struct sockaddr *)&server, sizeof(server)) < 0){
29         printf("ERROR connecting");
30         return 1;
31     }
32     puts("connected");
33
34     while(1){
35         //create variables to store client input and server output
36         char message[10]= "", server_reply[3000]= "";
37         //message to tell user to enter the equation
38         printf("Please enter the equation: ");
39         //get the user input
40         fgets(message, 10, stdin);
41
42         //array to hold numbers entered by user
43         char *array[10];
44         //variable used to increment through the array that holds numbers
45         int i = 0;

```

```

Open  clientsocket.c  ~/Desktop  Save  -  x
serversocket.c  clientsocket.c
40 //get the user input
41 fgets(message, 10, stdin);
42
43 //array to hold numbers entered by user
44 char *array[10];
45 //index used to increment through the array that holds numbers
46 int i=0;
47 //variable to hold the equation after it is converted
48 char newstring[10]="";
49 //variable to hold first number
50 char holder[1]="";
51 //variable to hold second number
52 char holder2[1]="";
53 //remove extra spaces in the user input
54 remove_spaces(message);
55
56 //plus
57 //for when user enters addition equation
58 //check if there is a + sign
59 char *quotPtr = strchr(message, '+');
60 //if there is a + sign
61 if(quotPtr != NULL){
62     //break the message var by the + sign, leaving only the numbers
63     array[i] = strtok(message, "+");
64     //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
65     while(array[i]!=NULL){
66         array[++i] = strtok(NULL, "+");
67     }
68     //add 'add' to the newstring variable
69     strcat(newstring, "add");
70     //holder now has first digit
71     strcpy(holder, array[0]);
72     //first digit added to newstring
73     strcat(newstring, holder);
74     //holder2 now has second digit
75     strcpy(holder2, array[1]);
76     //second digit added to newstring
77     strcat(newstring, holder2);
78 }
79
80 //minus
81 //for when user enters subtraction equation
82 //check if there is a - sign
83 quotPtr = strchr(message, '-');
84 //if there is a - sign

```

```

Open  clientsocket.c  ~/Desktop  Save  -  x
serversocket.c  clientsocket.c
85 //if there is a - sign
86 if(quotPtr != NULL){
87     //break the message var by the - sign, leaving only the numbers
88     array[i] = strtok(message, "-");
89     //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
90     while(array[i]!=NULL){
91         array[++i] = strtok(NULL, "-");
92     }
93     //add 'sub' to the newstring variable
94     strcat(newstring, "sub");
95     //holder now has first digit
96     strcpy(holder, array[0]);
97     //first digit added to newstring
98     strcat(newstring, holder);
99     //holder2 now has second digit
100    strcpy(holder2, array[1]);
101    //second digit added to newstring
102    strcat(newstring, holder2);
103 }
104
105 //multiply
106 //for when user enters multiplication equation
107 //check if there is a * sign
108 quotPtr = strchr(message, '*');
109 //if there is a * sign
110 if(quotPtr != NULL){
111     //break the message var by the * sign, leaving only the numbers
112     array[i] = strtok(message, "*");
113     //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
114     while(array[i]!=NULL){
115         array[++i] = strtok(NULL, "*");
116     }
117     //add 'mul' to the newstring variable
118     strcat(newstring, "mul");
119     //holder now has first digit
120     strcpy(holder, array[0]);
121     //first digit added to newstring
122     strcat(newstring, holder);
123     //holder2 now has second digit
124     strcpy(holder2, array[1]);
125     //second digit added to newstring
126     strcat(newstring, holder2);
127 }
128
129 //divide

```

```

122 //holder2 now has second digit
123 strcpy(holder2, array[1]);
124 //second digit added to newstring
125 strcat(newstring, holder2);
126 }
127
128 //divide
129 //for when user enters multiplication equation
130 //for when user enters division equation
131 quotPtr = strchr(message, '/');
132 //if there is a / sign
133 if(quotPtr != NULL){
134     //remove the message from by the / sign, leaving only the numbers
135     array[1] = strtok(message, "/");
136     //split through and assign the numbers to the array first digit in index 0 and second in index 1
137     while(array[1] != NULL){
138         array[++i] = strtok(NULL, "/");
139     }
140     //add 'div' to the newstring variable
141     strcat(newstring, "div");
142     //holder now has first digit
143     strcpy(holder, array[0]);
144     //first digit added to newstring
145     strcat(newstring, holder);
146     //holder now has second digit
147     strcpy(holder2, array[1]);
148     //second digit added to newstring
149     strcat(newstring, holder2);
150 }
151
152 //display the convert string
153 puts(newstring);
154 if (send(sock, newstring, strlen(newstring), 0) < 0){
155     printf("ERROR writing to socket");
156     return 1;
157 }
158
159 if (recv(sock, server_reply, 1024, 0) < 0){
160     puts("recv failed");
161     return 1;
162 }
163 puts("server reply");
164 puts(server_reply);
165
166
167 }
168 close(sock);
169 return 0;
170 }
171
172
173 void remove_spaces(char* s) {
174     char* d = s;
175     do {
176         while (*d == ' ') {
177             ++d;
178         }
179         *s++ = *d++;
180     } while (*s++ != '\0');
181 }

```

## Console

```

yp@yp: ~/Desktop
yp@yp:~/Desktop$ gcc clientsocket.c -o client
yp@yp:~/Desktop$ ./client
Socket created
Connected
Please enter the equation: 3 + 4
add34

server reply
Success: 1, Answer: 7
Please enter the equation: 4-1
sub41

server reply
Success: 1, Answer: 3
Please enter the equation: 2*2
mul22

server reply
Success: 1, Answer: 4
Please enter the equation: 4 / 2
div42

server reply
Success: 1, Answer: 2.000000
Please enter the equation: 3/0
div30

server reply
Success: 0, Error: Tried to divide by 0
Please enter the equation:

```

```

yp@yp: ~/Desktop
yp@yp: ~/Desktop$ cd Desktop
yp@yp: ~/Desktop$ gcc serversocket.c -o server
yp@yp: ~/Desktop$ ./server
Socket created
bind done
waiting for incoming connections...
Connection accepted
add34
sub41
mul22
div42
div30

```

## UDP:

### Server

```

serversocket.c
1 // Server side C/C++ program to demonstrate socket programming
2 #include <unistd.h>
3 #include <stdio.h>
4 #include <sys/socket.h>
5 #include <stdlib.h>
6 #include <netinet/in.h>
7 #include <string.h>
8 #include <arpa/inet.h>
9 #include <string.h>
10
11 int main(int argc, char *argv[]){
12
13     int socket_desc, c_read_size;
14     long unsigned int client_socket ;
15     char server_message[2000], client_message[2000];
16     struct sockaddr_in server, client;
17     int client_struct_length = sizeof(client);
18     //char client_message[2000];
19
20
21
22
23 // get a socket in udp
24 socket_desc = socket(AF_INET, SOCK_DGRAM, 0);
25 if (socket_desc == -1){
26     printf("could not create socket.");
27 }
28 puts("Socket created");
29
30 //fill the fields
31 server.sin_addr.s_addr = inet_addr("127.0.0.1");
32 server.sin_family = AF_INET;
33 server.sin_port = htons( 8888 );
34
35
36 //bind the socket to the port
37 if (bind(socket_desc, (struct sockaddr *)&server, sizeof(server)) < 0){
38     perror("bind failed. error");
39     return 1;
40 }
41
42 //start listening for incoming connections
43 puts("bind done");
44 puts("listening for incoming message");
45
46 printf("waiting for incoming connections...");

```

```

serversocket.c
~/Desktop

45 puts("waiting for incoming connections...");
46 c = listen(sockaddr_in);
47 printf("Received message from IP: %s and port: %i\n", inet_ntoa(client.sin_addr), ntohs(client.sin_port));
48
49 if (recvfrom(socket_desc, client_message, sizeof(client_message), 0, (struct sockaddr*)&client, &client_struct_length) < 0){
50     printf("couldn't receive\n");
51     return -1;
52 }
53
54
55 /while(read_size=recv(client_socket, client_message, 2000,0)>0) {
56
57     puts(client_message);
58
59     //place holder for first digit
60     char str[10]="";
61     //place holder for second digit
62     char dup[10]="";
63     //variable to hold answer to the equation
64     char rstr[10]="";
65     //variable to assign an index to delete
66     int idxToDelete=0;
67     //check if the equation was successful or not, default is 1 meaning success
68     int check=1;
69
70     //variable for success message
71     char success[]="Success: 1, Answer: ";
72     //variable for fail message
73     char fail[]="Success: 0, Error: Tried to divide by 0";
74
75     //add
76     //check if client message has a which is unique to add
77     char *quotPtr = strchr(client_message, 'a');
78     if (quotPtr != NULL){
79         //remove add portion from string
80         for(int i=0; i<=2; i++){
81             idxToDelete = 0;
82             memmove(&client_message[idxToDelete], &client_message[idxToDelete + 1], strlen(client_message) - idxToDelete);
83         }
84
85         //copy the remain numbers to str and dup
86         strcpy(str, client_message);
87         strcpy(dup, client_message);
88     }
89

```

```

serversocket.c
~/Desktop

89     strcpy(str, client_message);
90     strcpy(dup, client_message);
91
92     //remove second digit in str variable
93     idxToDelete = 1;
94     memmove(&str[idxToDelete], &str[idxToDelete + 1], strlen(str) - idxToDelete);
95     //remove first digit in dup
96     idxToDelete = 0;
97     memmove(&dup[idxToDelete], &dup[idxToDelete + 1], strlen(dup) - idxToDelete);
98     //parse int the string and add them
99     int x = atoi(str) + atoi(dup);
100     //convert the int to string and assign to rstr
101     sprintf(rstr, "%d", x);
102
103     //sub
104     //check if client message has a which is unique to sub
105     quotPtr = strchr(client_message, 's');
106     if (quotPtr != NULL){
107         //remove sub portion from string
108         for(int i=0; i<=2; i++){
109             idxToDelete = 0;
110             memmove(&client_message[idxToDelete], &client_message[idxToDelete + 1], strlen(client_message) - idxToDelete);
111         }
112
113         //copy the remain numbers to str and dup
114         strcpy(str, client_message);
115         strcpy(dup, client_message);
116         //remove second digit in str variable
117         idxToDelete = 1;
118         memmove(&str[idxToDelete], &str[idxToDelete + 1], strlen(str) - idxToDelete);
119         //remove first digit in dup
120         idxToDelete = 0;
121         memmove(&dup[idxToDelete], &dup[idxToDelete + 1], strlen(dup) - idxToDelete);
122         //parse int the string and sub them
123         int x = atoi(str) - atoi(dup);
124         //convert the int to string and assign to rstr
125         sprintf(rstr, "%d", x);
126     }
127
128     //mul
129     //check if client message has m which is unique to mul
130     quotPtr = strchr(client_message, 'm');
131     if (quotPtr != NULL){
132         //remove mul portion from string
133         for(int i=0; i<=2; i++){
134             idxToDelete = 0;
135             memmove(&client_message[idxToDelete], &client_message[idxToDelete + 1], strlen(client_message) - idxToDelete);
136         }
137
138         //copy the remain numbers to str and dup
139         strcpy(str, client_message);
140         strcpy(dup, client_message);
141         //remove second digit in str variable
142         idxToDelete = 1;
143         memmove(&str[idxToDelete], &str[idxToDelete + 1], strlen(str) - idxToDelete);
144         //remove first digit in dup
145         idxToDelete = 0;
146         memmove(&dup[idxToDelete], &dup[idxToDelete + 1], strlen(dup) - idxToDelete);
147         //parse int the string and mul them
148         int x = atoi(str) * atoi(dup);
149         //convert the int to string and assign to rstr
150         sprintf(rstr, "%d", x);
151     }
152
153     //div
154     //check if client message has d which is unique to div
155     quotPtr = strchr(client_message, 'd');
156     if (quotPtr != NULL){
157         //remove div portion from string
158         for(int i=0; i<=2; i++){
159             idxToDelete = 0;
160             memmove(&client_message[idxToDelete], &client_message[idxToDelete + 1], strlen(client_message) - idxToDelete);
161         }
162
163         //copy the remain numbers to str and dup
164         strcpy(str, client_message);
165         strcpy(dup, client_message);
166         //remove second digit in str variable
167         idxToDelete = 1;
168         memmove(&str[idxToDelete], &str[idxToDelete + 1], strlen(str) - idxToDelete);
169         //remove first digit in dup
170         idxToDelete = 0;
171         memmove(&dup[idxToDelete], &dup[idxToDelete + 1], strlen(dup) - idxToDelete);
172         //parse int the string and div them
173         int x = atoi(str) / atoi(dup);
174         //convert the int to string and assign to rstr
175         sprintf(rstr, "%d", x);
176     }
177
178     //print the result
179     printf("Result: %s\n", rstr);
180
181     //send the result to client
182     send(client_socket, rstr, strlen(rstr), 0);
183
184     //close the client socket
185     close(client_socket);
186 }
187
188 return 0;
189

```

```

127 }
128
129 //mul
130 //check if client message has m which is unique to mul
131 quotPtr = strchr(client_message, 'm');
132
133 if(quotPtr != NULL){
134     //remove mul portion from string
135     for(int i=0; i<=2; i++){
136         idxToDel = 0;
137         memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
138     }
139
140     //copy the remain numbers to str and dup
141     strcpy(str, client_message);
142     strcpy(dup, client_message);
143     //remove second digit in str variable
144     idxToDel = 1;
145     memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
146     //Remove first digit in dup
147     idxToDel = 0;
148     memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
149     //Parse the strings and mul them
150     int x = atoi(str) * atoi(dup);
151     //convert the int to string and assign to rstr
152     sprintf(rstr, "%d", x);
153 }
154
155 //div
156 //check if client message has v which is unique to div
157 quotPtr = strchr(client_message, 'v');
158 if(quotPtr != NULL){
159     //remove mul portion from string
160     for(int i=0; i<=2; i++){
161         idxToDel = 0;
162         memmove(&client_message[idxToDel], &client_message[idxToDel + 1], strlen(client_message) - idxToDel);
163     }
164
165     //copy the remain numbers to str and dup
166     strcpy(str, client_message);
167     strcpy(dup, client_message);
168     //remove second digit in str variable
169     idxToDel = 1;
170     memmove(&str[idxToDel], &str[idxToDel + 1], strlen(str) - idxToDel);
171

```

```

173     idxToDel = 0;
174     memmove(&dup[idxToDel], &dup[idxToDel + 1], strlen(dup) - idxToDel);
175     //if one of the digits is zero change check to zero, meaning unsuccessful
176     if(atoi(str)==0 || atoi(dup)==0){
177         check=0;
178     }
179     //parse double the strings and div them
180     double x= atof(str) / atof(dup);
181     //convert the double to string and assign to rstr
182     sprintf(rstr, "%f", x);
183 }
184 //if there was no zero, thus successful
185 if(check==1){
186     strcat(success, rstr);
187     if (sendto(socket_desc, success, strlen(success), 0,
188         (struct sockaddr*)&client, client_struct_length) < 0){
189         printf("can't send\n");
190         return -1;
191     }
192 }
193
194 //if there was a zero, thus unsuccessful
195 if(check==0){
196     if (sendto(socket_desc, fail, strlen(fail), 0,
197         (struct sockaddr*)&client, client_struct_length) < 0){
198         printf("can't send\n");
199         return -1;
200     }
201 }
202
203
204
205
206
207 if(read_size==0){
208     puts("client disconnected");
209     fflush(stdout);
210 }
211 else if(read_size==1){
212     perror("recv failed");
213 }
214
215
216 return 0;
217 }

```

## Client



```

1 #include <stdio.h>
2 #include <sys/socket.h>
3 #include <arpa/inet.h>
4 #include <string.h>
5 #include <unistd.h>
6 #include <string.h>
7 #include <stdlib.h>
8
9 //custom function to remove spaces
10 void remove_spaces(char* s);
11
12 int main(int argc, char *argv[])
13 {
14     int socket_desc;
15
16     struct sockaddr_in server;
17     int server_struct_length = sizeof(server);
18
19     //char message[1000], server_reply[2000];
20     socket_desc = socket(AF_INET, SOCK_DGRAM, 0);
21     if (socket_desc == -1)
22         printf("ERROR opening socket");
23     puts("socket created");
24
25     server.sin_addr.s_addr = inet_addr("127.0.0.1");
26     server.sin_family = AF_INET;
27     server.sin_port = htons(8080);
28
29 while(1){
30     //create variables to store client input and server output
31     char message[10]= "", server_reply[3000]="";
32     //message to tell user to enter the equation
33     printf("Please enter the equation: ");
34     //get the user input
35     fgets(message, 10, stdin);
36
37     //array to hold numbers entered by user
38     char *array[10];
39     //int i used to increment through the array that holds numbers
40     int i=0;
41     //variable to hold the equation after it is converted
42     char newstring[10]="";
43     //char holder[10] hold first number
44     char holder[i]="";
45     //variable to hold second number

```

```

45     //variable to hold second number
46     char holder2[i]="";
47     //remove spaces in the user input
48     remove_spaces(message);
49
50     //plus
51     //for when user enters addition equation
52     //check if there is a + sign
53     char *quotPtr = strchr(message, '+');
54     //if there is a + sign
55     if(quotPtr != NULL){
56         //break the message var by the + sign, leaving only the numbers
57         array[i] = strtok(message, "+");
58         //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
59         while(array[i]!=NULL){
60             array[++i] = strtok(NULL, "+");
61         }
62         //add 'add' to the newstring variable
63         strcat(newstring, "add");
64         //holder now has first digit
65         strcpy(holder, array[0]);
66         //first digit added to newstring
67         strcat(newstring, holder);
68         //holder now has second digit
69         strcpy(holder2, array[1]);
70         //second digit added to newstring
71         strcat(newstring, holder2);
72     }
73
74     //minus
75     //for when user enters subtraction equation
76     //check if there is a - sign
77     quotPtr = strchr(message, '-');
78     //if there is a - sign
79     if(quotPtr != NULL){
80         //break the message var by the - sign, leaving only the numbers
81         array[i] = strtok(message, "-");
82         //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
83         while(array[i]!=NULL){
84             array[++i] = strtok(NULL, "-");
85         }
86         //add 'sub' to the newstring variable
87         strcat(newstring, "sub");
88         //holder now has first digit
89         strcpy(holder, array[0]);

```

```

Open  clientsocket.c  ~/Desktop  Save  -  σ  x
serversocket.c  clientsocket.c

88 //holder now has first digit
89 strcpy(holder, array[0]);
90 //first digit added to newstring
91 strcat(newstring, holder);
92 //holder now has second digit
93 strcpy(holder2, array[1]);
94 //second digit added to newstring
95 strcat(newstring, holder2);
96
97 }
98
99 //multiply
100 //for when user enters multiplication equation
101 //check if there is a * sign
102 quotPtr = strchr(message, '*');
103 //if there is a * sign
104 if(quotPtr != NULL){
105     //break the message var by the * sign, leaving only the numbers
106     array[i] = strtok(message, "*");
107     //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
108     while(array[i]==NULL){
109         array[++i] = strtok(NULL, "*");
110     }
111     //add 'mul' to the newstring variable
112     strcat(newstring, "mul");
113     //holder now has first digit
114     strcpy(holder, array[0]);
115     //first digit added to newstring
116     strcat(newstring, holder);
117     //holder now has second digit
118     strcpy(holder2, array[1]);
119     //second digit added to newstring
120     strcat(newstring, holder2);
121 }
122
123 //divide
124 //for when user enters multiplication equation
125 //check if there is a / sign
126 quotPtr = strchr(message, '/');
127 //if there is a / sign
128 if(quotPtr != NULL){
129     //break the message var by the / sign, leaving only the numbers
130     array[i] = strtok(message, "/");
131     //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
132     while(array[i]==NULL){
133         array[++i] = strtok(NULL, "/");
134     }
135 }

```

```

Open  clientsocket.c  ~/Desktop  Save  -  σ  x
serversocket.c  clientsocket.c

127 if(quotPtr != NULL){
128     //break the message var by the / sign, leaving only the numbers
129     array[i] = strtok(message, "/");
130     //cycle through and assign the numbers to the array first digit in index 0 and second in index 1
131     while(array[i]==NULL){
132         array[++i] = strtok(NULL, "/");
133     }
134     //add 'div' to the newstring variable
135     strcat(newstring, "div");
136     //holder now has first digit
137     strcpy(holder, array[0]);
138     //first digit added to newstring
139     strcat(newstring, holder);
140     //holder now has second digit
141     strcpy(holder2, array[1]);
142     //second digit added to newstring
143     strcat(newstring, holder2);
144 }
145
146 //display the convert string
147 puts(newstring);
148 /*if (sendto(sock, newstring, strlen(newstring), 0) < 0){
149     printf("Error writing to socket");
150     return 1;
151 }
152
153 if (recvto(sock, server_reply, 3000, 0) < 0){
154     puts("recv failed");
155     break;
156 }*/
157
158
159 if(sendto(socket_desc, newstring, strlen(newstring), 0,
160         (struct sockaddr*)&server, server_struct_length) < 0){
161     printf("Unable to send message!\n");
162     return -1;
163 }
164
165 // Receive the server's response
166 if(recvfrom(socket_desc, server_reply, sizeof(server_reply), 0,
167         (struct sockaddr*)&server, &server_struct_length) < 0){
168     printf("Error while receiving server's msg\n");
169     return -1;
170 }
171

```

```

145
146 //display the convert string
147 puts(newstring);
148 //if (sendto(sock,newstring,strlen(newstring),0)< 0){
149     printf("ERROR writing to socket");
150     return 1;
151 }
152
153 if (recvto(sock,server_reply,1000,0)<0){
154     puts("recv failed");
155     break;
156 }
157
158 //((sendto(socket_desc, newstring, strlen(newstring), 0,
159     ((struct sockaddr*)&server, &server_struct_length) < 0){
160     printf("Unable to send message\n");
161     return -1;
162 }
163 }
164
165 // Receive the server's response:
166 //((recvfrom(socket_desc, server_reply, sizeof(server_reply), 0,
167     ((struct sockaddr*)&server, &server_struct_length) < 0){
168     printf("Error while receiving server's msg\n");
169     return -1;
170 }
171
172 puts("server_reply");
173 puts(server_reply);
174
175 }
176 }
177 close(socket_desc);
178 return 0;
179 }
180
181
182 void remove_spaces(char* s) {
183     char* d = s;
184     do {
185         while (*d == ' ') {
186             ++d;
187         }
188     } while (*s++ = *d++);
189 }

```

## Console

```

yp@yp: ~/Desktop
yp@yp: ~/Desktop$ ./client
socket created
Please enter the equation: 3 + 4
add34

server reply
Success: 1, Answer: 7
Please enter the equation: 4-1
sub41

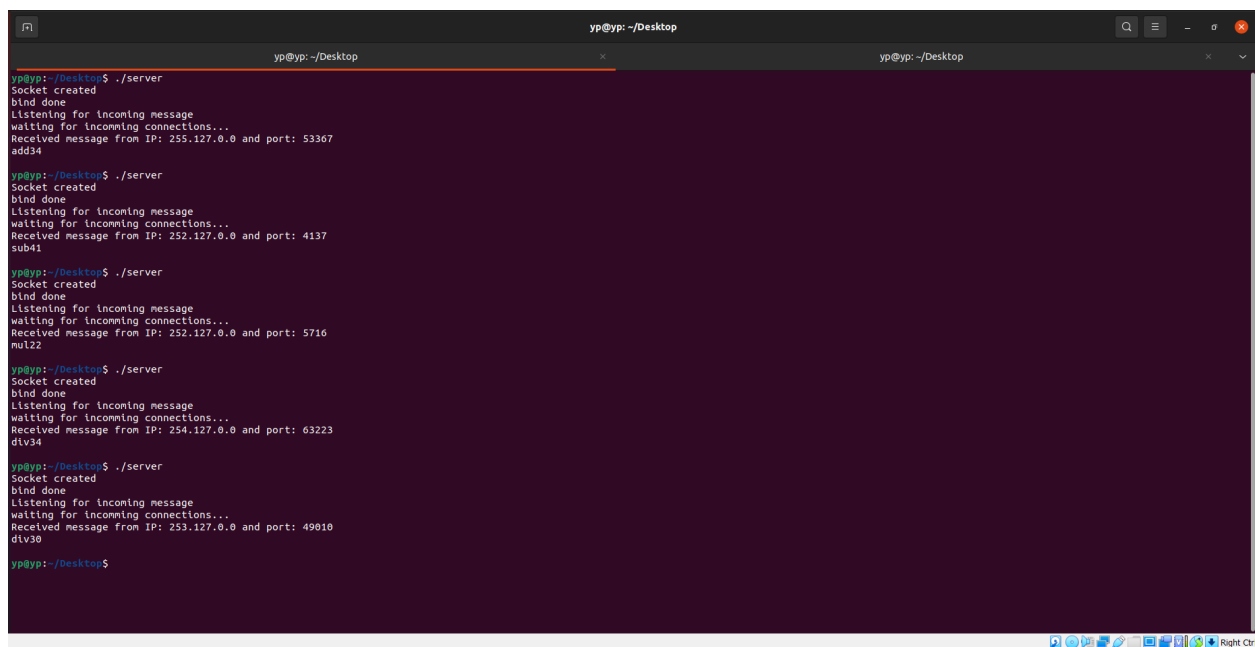
server reply
Success: 1, Answer: 3
Please enter the equation: 2 * 2
mul22

server reply
Success: 1, Answer: 4
Please enter the equation: 3/4
div34

server reply
Success: 1, Answer: 0.750000
Please enter the equation: 3/0
div30

server reply
Success: 0, Error: Tried to divide by 0
Please enter the equation:

```



```
yp@yp: ~/Desktop
yp@yp: ~/Desktop
yp@yp: ~/Desktop$ ./server
socket created
bind done
Listening for incoming message
waiting for incoming connections...
Received message from IP: 255.127.0.0 and port: 53367
add34

yp@yp: ~/Desktop$ ./server
socket created
bind done
Listening for incoming message
waiting for incoming connections...
Received message from IP: 252.127.0.0 and port: 4137
sub41

yp@yp: ~/Desktop$ ./server
socket created
bind done
Listening for incoming message
waiting for incoming connections...
Received message from IP: 252.127.0.0 and port: 5716
mU122

yp@yp: ~/Desktop$ ./server
socket created
bind done
Listening for incoming message
waiting for incoming connections...
Received message from IP: 254.127.0.0 and port: 63223
dlv34

yp@yp: ~/Desktop$ ./server
socket created
bind done
Listening for incoming message
waiting for incoming connections...
Received message from IP: 253.127.0.0 and port: 49010
dlv30

yp@yp: ~/Desktop$
```