

# ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

## Εργασία 1

ΠΑΝΤΕΛΕΗΜΩΝ ΠΡΩΙΟΣ

ice18390023

6ο Εξάμηνο

ice18390023@uniwa.gr

Τμήμα ΣΤ5 Τετάρτη 14:00-16:00



ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ  
UNIVERSITY OF WEST ATTICA

## Υπεύθυνοι καθηγητές

ΜΑΜΑΛΗΣ ΒΑΣΙΛΗΣ

ΠΑΝΤΖΙΟΥ ΓΡΑΜΜΑΤΗ

ΔΟΚΑ ΑΙΚΑΤΕΡΙΝΗ

Τμήμα Μηχανικών και Πληροφορικής Υπολογιστών  
19 Απριλίου 2021

## Περιεχόμενα

1	Δημιουργία αρχείου rpc	1
2	Το αρχείο makefile	2
3	Το αρχείο header	4
4	Διαδικασίες server	7
5	Client	10
6	User	16
7	Ενδεικτικά τρεξίματα	20

## Κώδικες

1.1	Αρχείο rpcgen . . . . .	1
2.1	Αρχείο makefile . . . . .	2
3.1	Αρχείο common_func.h . . . . .	4
4.1	Αρχείο proj_client.c . . . . .	7
5.1	Αρχείο client . . . . .	10
6.1	Αρχείο user . . . . .	16

## Κατάλογος σχημάτων

7.1	Εικόνα 1 . . . . .	21
7.2	Εικόνα 2 . . . . .	22

## 1 Δημιουργία αρχείου rpc

Το αρχείο του κώδικα 1.1 κατασκευάζει ένα struct vecD όπου περιέχει έναν pointer double \*vecD\_val και έναν ακέραιο int vecD\_len. Αντίστοιχα και το vecI αλλά ο pointer είναι int. Επίσης, υπάρχει ένα struct aVec που περιέχει όλα τα δεδομένα (all vector) και ένα ακόμα struct για να επιστραφεί η μέση τιμή. Με την εντολή

```
rpcgen -cA proj.x
```

παράγουμε τα κατάλληλα αρχεία για το client-server για την επικοινωνία και ένα βασικό template για να γράψουμε τον δικό μας κώδικα υλοποίησής των ρουτινών.

```
1 typedef double vecD<>;
2
3 typedef int vecI<>;
4
5 struct aVec {
6     int X<>;
7     int Y<>;
8     int n;
9     double r;
10 };
11
12 struct avg {
13     double Ex;
14     double Ey;
15 };
16
17 program PROJ_PROG {
18     version PROJ_VERS {
19         double absolute(vecI) = 1;
20         int product(aVec) = 2;
21         avg mean(aVec) = 3;
22         vecD mulRwXY(aVec) = 4;
23     } = 1;
24 } = 0x20000000;
```

Κώδικας 1.1: Αρχείο rpcgen

## 2 Το αρχείο makefile

Στο αρχείο makefile (κώδικας 2.1) κάνουμε μερικές αλλαγές προσθέτοντας στο CC το gcc για την επιλογή compiler, στο CFLAGS το -DRPC\_SVC\_FG γιατί από default τα πάντα γίνονται στο background, στο LDLIBS το -lnsl -lm που είναι για να κάνει link την βιβλιοθήκη math και -C στο RPCGENFLAGS. Επίσης, κάτω από το target all έχει προστεθεί μία ακόμα εντολή για την μεταγλώττιση ενός επιπρόσθετου αρχείου.

```
1
2 # This is a template Makefile generated by rpcgen
3
4 # Parameters
5
6 CLIENT = proj_client
7 SERVER = proj_server
8
9 SOURCES_CLNT.c =
10 SOURCES_CLNT.h =
11 SOURCES_SVC.c =
12 SOURCES_SVC.h =
13 SOURCES.x = proj.x
14
15 TARGETS_SVC.c = proj_svc.c proj_server.c proj_xdr.c
16 TARGETS_CLNT.c = proj_clnt.c proj_client.c proj_xdr.c
17 TARGETS = proj.h proj_xdr.c proj_clnt.c proj_svc.c proj_client.c proj_server.c
18
19 OBJECTS_CLNT = $(SOURCES_CLNT.c:%.c=%.o) $(TARGETS_CLNT.c:%.c=%.o)
20 OBJECTS_SVC = $(SOURCES_SVC.c:%.c=%.o) $(TARGETS_SVC.c:%.c=%.o)
21 # Compiler flags
22
23 CC = gcc
24 CFLAGS += -g -DRPC_SVC_FG
25 LDLIBS += -lnsl -lm
26 RPCGENFLAGS = -C
27
28 # Targets
29
30 all : $(CLIENT) $(SERVER)
31     $(CC) user_client.c -o user_client
32
33 $(TARGETS) : $(SOURCES.x)
34     rpcgen $(RPCGENFLAGS) $(SOURCES.x)
35
36 $(OBJECTS_CLNT) : $(SOURCES_CLNT.c) $(SOURCES_CLNT.h) $(TARGETS_CLNT.c)
```

```
37
38 $(OBJECTS_SVC) : $(SOURCES_SVC.c) $(SOURCES_SVC.h) $(TARGETS_SVC.c)
39
40 $(CLIENT) : $(OBJECTS_CLNT)
41 $(LINK.c) -o $(CLIENT) $(OBJECTS_CLNT) $(LDLIBS)
42
43 $(SERVER) : $(OBJECTS_SVC)
44 $(LINK.c) -o $(SERVER) $(OBJECTS_SVC) $(LDLIBS)
45
46 clean:
47 $(RM) core $(TARGETS) $(OBJECTS_CLNT) $(OBJECTS_SVC) $(CLIENT) $(SERVER)
```

Κώδικας 2.1: Αρχείο makefile

### 3 Το αρχείο header

Το αρχείο `common_func.h` (κώδικας 3.1) έχει υλοποιημένες κάποιες συναρτήσεις που χρησιμοποιούνται σε παραπάνω από ένα αρχεία. Το τι κάνει η κάθε μία ξεχωριστά, εξηγείται πάνω από κάθε συνάρτηση.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define BUF_SIZE 512
5
6 #ifndef COMMON_FUNC_H
7 #define COMMON_FUNC_H
8
9 // εμφανίζει ως μήνυμα λάθους το msg στο stderr
10 // και κάνει exit με την τιμή val
11 void error(char *msg, int val)
12 {
13     perror(msg);
14     exit(val);
15 }
16
17 // αρχικοποιεί τις τιμές r και n
18 void init_values( double *r, int *n){
19
20     printf("Double number for r: ");
21     scanf("%lf", r);
22
23     printf("Length of the vectors: ");
24     scanf("%d", n);
25 }
26
27 // επιστρέφει έναν μια διεύθυνση double
28 // μεγέθους n
29 double *alloc_double(int n){
30
31     double *ptr = (double *) malloc(n*sizeof(double));
32
33     if ( !ptr ){
34         error("Allocation error",1);
35     }
36     return ptr;
37 }
38
```

```
39 // επιστρέφει έναν μια διεύθυνση int
40 // μεγέθους n
41 int *alloc_mem(int n){
42
43     int *ptr = (int *) malloc(n*sizeof(int));
44
45     if ( !ptr ){
46         error("Allocation error",1);
47     }
48     return ptr;
49 }
50
51 // αρχικοποιεί έναν πίνακα μεγέθους n από το stdin
52 // και με όνομα τον χαρακτήρα name
53 int *init_vec(int n, char name){
54
55     int i = 0;
56
57     int *ptr = alloc_mem(n);
58
59     for(; i < n; i++){
60         printf("%c[%d] = ", name, i+1);
61         scanf("%d", ptr+i);
62     }
63     return ptr;
64 }
65
66 // εμφανίζει το menu επιλογών και επιστρέφει
67 // την τιμή που διαβάστηκε από το stdin
68 int menu(){
69
70     int opt;
71
72
73     printf("\n\nChoose an option:\n");
74     printf("1) Absolute of vector X\n");
75     printf("2) Product of vector X and Y\n");
76     printf("3) Mean value of each vector\n");
77     printf("4) Product of r with the addition of X and Y\n");
78     printf("0) Stop\n");
79
80     scanf("%d", &opt);
81
82     return opt;
```

```
83 }  
84  
85 #endif /* COMMON_FUNC_H */
```

Κώδικας 3.1: Αρχείο common\_func.h



## 4 Διαδικασίες server

Στο αρχείο proj\_server.c που δημιουργήθηκε με την εντολή `prcgen` υλοποιούμε τις λειτουργίες των συναρτήσεων, έτσι ώστε να έχουμε το επιθυμητό αποτέλεσμα. Η εντολή που χρειάζεται για την εκκίνηση του server είναι

```
./proj_server
```

```
1 #include "proj.h"
2 #include <math.h>
3
4 // Επιστέφεται το μέτρο του διανύσματος
5 double *
6 absolute_1_svc(vecI *argp, struct svc_req *rqstp)
7 {
8     static double result;
9
10    int i = 0;
11    int *vec = argp->vecI_val;
12    int n = argp->vecI_len;
13    result = 0.0;
14
15    for (; i < n; i++){
16        result += vec[i]*vec[i];
17    }
18
19    result = sqrt(result);
20
21    return &result;
22 }
23
24 // Επιστρέφεται το εσωτερικό γινόμενο των δύο διανυσμάτων
25 int *
26 product_1_svc(aVec *argp, struct svc_req *rqstp)
27 {
28     static int result;
29
30    int i = 0;
31    int *X = argp->X.X_val;
32    int *Y = argp->Y.Y_val;
33    int n = argp->n;
34    result = 0;
35
36    for (; i < n ; i++){
37        result += X[i] * Y[i];
```

```
38 }
39
40 return &result;
41 }
42
43 // Επιστρέφεται ένας ακέραιος πίνακας 2 στοιχείων με την
44 // μέση τιμή του X και του Y στο πρώτο στοιχείο και στο
45 // δεύτερο στοιχείο αντίστοιχα
46 avg *
47 mean_1_svc(aVec *argp, struct svc_req *rqstp)
48 {
49     static avg result;
50
51     int i = 0;
52     int *X = argp->X.X_val;
53     int *Y = argp->Y.Y_val;
54     int n = argp->n;
55     result.Ex = 0;
56     result.Ey = 0;
57
58     for(; i < n; i++){
59
60         result.Ex += X[i];
61         result.Ey += Y[i];
62     }
63
64     result.Ex /= n;
65     result.Ey /= n;
66
67     return &result;
68 }
69
70 // Η συνάρτηση multiply r with the sum of X and Y (mulRwXY)
71 // επιστρέφει ένα πίνακα double ίδιου μεγέθους με τα δύο
72 // διανύσματα
73 vecD *
74 mulrwx_1_svc(aVec *argp, struct svc_req *rqstp)
75 {
76     static vecD result;
77
78     int i = 0;
79     int *X = argp->X.X_val;
80     int *Y = argp->Y.Y_val;
81     double r = argp->r;
```

```
82 int n = argp->n;
83
84 double *ptr = (double *) malloc(n*sizeof(double));
85
86 if (!ptr){
87     printf("Allocation error\n");
88     return NULL;
89 }
90
91 for(; i < n; i++){
92
93     ptr[i] = r * (X[i] + Y[i]);
94 }
95
96
97 result.vecD_val = ptr;
98 result.vecD_len = n;
99
100 return &result;
101 }
```

Κώδικας 4.1: Αρχείο proj\_client.c

## 5 Client

Ο το αρχείο `proj_client.c` (κώδικας 5.1), επικοινωνεί με τον `server` (κώδικας 4.1) μέσω δικής του υλοποίησης. Όμως, για να επικοινωνεί με τους χρήστες (κώδικας 6.1) στην `main` επιτυχάνεται η ακρόαση (`listening`) των αιτημάτων στο `socket tcp/6969` και αφού γίνει η σύνδεση γίνεται δημιουργία νέου παιδιού (νέα διεργασία) που καλεί την συνάρτηση `proj_prog_1` για να διαχειριστεί τα αιτήματα του χρήστη και αφού επιστρέψει τελειώνει (όταν το μαζέψει η πατρική διεργασία). Η εντολή για την εκκίνηση του `client` είναι

```
./proj_client localhost
```

```
1 #include "proj.h"
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7 #include <sys/socket.h>
8 #include <netinet/in.h>
9 #include <unistd.h>
10 #include "common_func.h"
11
12 void proj_prog_1(char *, int);
13
14 void
15 proj_prog_1(char *host, int newssockfd)
16 {
17     CLIENT *clnt;
18     double *result_1;
19     vecI absolute_1_arg;
20     int *result_2;
21     aVec aVec_1_arg;
22     avg *result_3;
23     vecD *result_4;
24
25     #ifndef DEBUG
26     clnt = clnt_create (host, PROJ_PROG, PROJ_VERS, "udp");
27     if (clnt == NULL) {
28         clnt_pcreateerror (host);
29         exit (1);
30     }
31     #endif /* DEBUG */
32
33     // get r and n
34     // Returns bytes recv on success or -1 on error
```

```
35 double r;
36 int n;
37
38 recv( newsockfd, &r, sizeof(r), 0);
39 recv( newsockfd, &n, sizeof(n), 0);
40
41 // δεσμεύουμε δυναμικά μνήμη
42 int *X = alloc_mem(n);
43 int *Y = alloc_mem(n);
44
45 // παίρνουμε τα δεδομένα για τα διανύσματα
46 recv( newsockfd, X, n*sizeof(int), 0);
47 recv( newsockfd, Y, n*sizeof(int), 0);
48
49 int i;
50
51 int opt, flag=1;
52 double ans3[2];
53
54 // κάνουμε αρχικοποίηση
55 aVec_1_arg.X.X_val = absolute_1_arg.vecI_val = X;
56 aVec_1_arg.Y.Y_val = Y;
57 aVec_1_arg.n = absolute_1_arg.vecI_len = aVec_1_arg.X.X_len = aVec_1_arg.Y.Y_len = n;
58 aVec_1_arg.r = r;
59
60 // όσο το flag δεν είναι 0 γίνεται επανάληψη
61 while(flag){
62
63     // λαμβάνουμε την επιλογή που έκανε ο χρήστης
64     recv( newsockfd, &opt, sizeof(int), 0);
65
66     // οι εντολές που θα γίνουν για κάθε αριθμό είναι
67     // είναι οι ίδιες με της επιλογές που εμφανίζονται
68     // στην συνάρτηση menu
69     switch(opt){
70
71     case 0:
72         flag = 0;
73         break;
74     case 1:
75         result_1 = absolute_1(&absolute_1_arg, clnt);
76         if (result_1 == (double *) NULL) {
77             clnt_perror (clnt, "call failed");
78         }
```

```

79     else{
80         send( newsockfd, result_1, sizeof(double), 0);
81     }
82     break;
83 case 2:
84     result_2 = product_1(&aVec_1_arg, clnt);
85     if (result_2 == (int *) NULL) {
86         clnt_perror (clnt, "call failed");
87     }
88     else{
89         send( newsockfd, result_2, sizeof(int), 0);
90     }
91     break;
92 case 3:
93     result_3 = mean_1(&aVec_1_arg, clnt);
94     if (result_3 == (avg *) NULL) {
95         clnt_perror (clnt, "call failed");
96     }
97     else{
98         ans3[0] = result_3->Ex;
99         ans3[1] = result_3->Ey;
100        send( newsockfd, ans3, sizeof(ans3), 0);
101    }
102    break;
103 case 4:
104     result_4 = mulrwxxy_1(&aVec_1_arg, clnt);
105     if (result_4 == (vecD *) NULL) {
106         clnt_perror (clnt, "call failed");
107     }
108     else{
109         send( newsockfd, result_4->vecD_val, n*sizeof(double), 0);
110     }
111     break;
112 }
113 }
114
115 #ifndef DEBUG
116     clnt_destroy (clnt);
117 #endif /* DEBUG */
118 }
119
120
121 int
122 main (int argc, char *argv[])

```

```
123 {
124     char *host;
125
126     if (argc != 2) {
127         printf ("usage: %s <server_host> \n", argv[0]);
128         exit (1);
129     }
130
131     // βάζουμε την διεύθυνση του argv[1] στο host
132     host = argv[1];
133
134
135     int sockfd, newsockfd, portno, clilen;
136     pid_t proclد;
137     struct sockaddr_in serv_addr, cli_addr;
138     unsigned int children = 0;
139
140     // βάζουμε τον αριθμό port 6969
141     portno = 6969;
142
143     // Δημιουργεί ένα tcp socket για να ακούει όλα τα αιτήματα
144     // αν υπάρχει error επιστρέφει -1
145     sockfd = socket(AF_INET, SOCK_STREAM, 0);
146     if (sockfd < 0) {
147         error("Error opening socket", 1);
148     }
149
150     // μηδενίζει τα δεδομένα για όσα bytes είναι το δεύτερο όρισμα
151     // ξεκινώντας από την διεύθυνση της πρώτης παραμέτρου
152     bzero((char *) &serv_addr, sizeof(serv_addr));
153
154     // set up
155     serv_addr.sin_family = AF_INET;
156     serv_addr.sin_port = htons(portno);
157
158     // INADDR_ANY binds socket to all available interfaces
159     serv_addr.sin_addr.s_addr = INADDR_ANY;
160
161     // "δένουμε" το socket και το port για να μπορούμε να ακούμε τα request
162     // που γίνονται σε αυτό το socket
163     if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0){
164         error("Error on binding", 1);
165     }
166 }
```

```
167 // int listen(int sockfd, int backlog);
168 // fd, max len of queue
169 // Returns 0 on success and -1 on error
170 // ξεκινάμε να ακούμε τα μηνύματα που στέλνονται στον sockfd fd
171 if ( listen(sockfd,5) < 0){
172     error("Error, listening to port", 1);
173 }
174
175
176 while(1) {
177     printf("Waiting for a connection...\n");
178     clilen = sizeof(cli_addr); // size of struct sockaddr_in
179
180     // int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
181     // δεχόμαστε το request που έγινε στο port που είμαστε και το πρωτόκολλο
182     // tcp που έχουμε ορίσει
183     // Οπότε, δημιουργείται ένα νέο fd για την επικοινωνία με τον συγκεκριμένο
184     // client
185     newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
186
187     if (newsockfd < 0){
188         error("Error on accept", 2);
189     }
190
191     // Δημιουργούμε 2 ίδια process, όπου στον πατέρα επιστρέφεται το
192     // pid του παιδιού, ενώ στο παιδί επιστρέφεται 0
193     if (fork() == 0) { //child process
194         close (sockfd); // doesn't need to listen requests
195         printf("Connected.\n");
196         // καλούμε την διαχείριση της επικοινωνίας που γίνεται από την
197         // συνάρτηση proj_prog_1
198         proj_prog_1 (host, newsockfd);
199         exit (0) ;
200     } // child process
201
202     // κλείνουμε την επικοινωνία πελάτη για την παρούσα διεργασία
203     close(newsockfd); // closes the socket with client connection
204     children++;
205
206     // γίνεται επανάληψη μέχρι κάποιο παιδί να μην έχει αλλάξει στάδιο
207     while (children){
208         // η waitpid επιστρέφει 0 αν υπάρχουν 1 ή παραπάνω παιδιά
209         // τα οποία δεν έχουν αλλάξει κατάσταση
210         procId=waitpid((pid_t) -1, NULL, WNOHANG);
```



```
211     if (procId<0) error("waitpid error", 3);
212     else if (procId==0) break;
213     else children--;
214 }
215 printf("There are %d children\n",children);
216 }
217
218 exit (0);
219 }
```

Κώδικας 5.1: Αρχείο client

## 6 User

Ο χρήστης για να επικοινωνήσει με το "client" όπου θα καλέσει την επιθυμητή διαδικασία από τον server συνδέεται μέσω tcp στο port 6969 δημιουργώντας το socket

```
./user_client localhost 6969
```

```
1 #include <stdio.h> // i/o
2 #include <stdlib.h> // malloc
3 #include <sys/types.h> // socket connect send recv
4 #include <sys/socket.h> // families socket connect send recv
5 #include <netdb.h> // gethostbyname
6 #include <string.h> // bzero bcopy gets
7 #include <arpa/inet.h> // htons
8 #include <unistd.h> // close
9 #include "common_func.h"
10
11
12 int main (int argc, char *argv[])
13 {
14     int opt;
15     double r;
16     int *X, *Y, n;
17     int sockfd, portno;
18     struct sockaddr_in serv_addr;
19     struct hostent *server;
20
21     // check arguments
22
23     if (argc < 3) {
24         fprintf (stderr, "usage: %s <server_host> <port_number>\n", argv[0]);
25         exit (1);
26     }
27
28     // make the connection
29
30     // Μετατρέπει το δεύτερο argument σε int από αριθμητικό
31     portno = atoi(argv[2]);
32
33     // int socket( int domain, int type, int protocol)
34     sockfd = socket(AF_INET, SOCK_STREAM, 0); //file descriptor
35
36     if ( sockfd < 0) {
37         error("Error opening socket",1);
38     }
```

```
39
40 // struct hostent *gethostbyname(const char *name);
41 // αυτό που χρειαζόμαστε από την δομή είναι το h_addr
42 // και το μέγεθος του h_length
43 server = gethostbyname( argv[1]);
44
45 if ( !server){
46     error("Error unkown host", 1);
47 }
48
49 // void bzero(void *s, size_t n);
50 // μηδενίζει κάθε byte ξεκινώντας από την διεύθυνση
51 // &serv_addr για μέγεθος sizeof(serv_addr), δηλαδή
52 // το μηδενίζει όλο
53 bzero( (char *) &serv_addr, sizeof(serv_addr) );
54
55 serv_addr.sin_family = AF_INET;
56
57 // αντιγράφει συνολικά όσα είναι τα bytes του τρίτου ορίσματος (n)
58 // από το πρώτο όρισμα (src) στο δεύτερο (dest)
59 // void bcopy(const void *src, void *dest, size_t n);
60 bcopy( (char *) server->h_addr,
61        (char *) &serv_addr.sin_addr.s_addr,
62        server->h_length);
63
64 // htons() μετατρέπει έναν unsigned short integer hostshort
65 // από host byte order σε network byte order (έχει να κάνει με
66 // τον little και big endian)
67 serv_addr.sin_port = htons(portno);
68
69 printf("Host: %s\nPort: %d\nTrying connection...\n",argv[1],portno);
70
71 // συνδέει το socket sockfd στην διεύθυνση. Το addrlen
72 // όρισμα ορίζει το μέγεθος του addr.
73 // Returns 0 on success and -1 on error.
74 // int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
75 if (connect( sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0){
76     error("Error connecting",2);
77 }
78
79 printf("Connected\n\n");
80
81 // αρχικοποιεί τις μεταβλητές με είσοδο από τον χρήστη μέσω του stdin
82 init_values(&r, &n);
```

```
83
84 // αρχικοποιεί τα διανύσματα με είσοδο από τον χρήστη μέσω του stdin
85 X = init_vec(n, 'X');
86 Y = init_vec(n, 'Y');
87
88 // send the info
89
90 // μεταδίδει το μήνυμα στο συνδεδεμένο socket
91 // Επιστρέφει τον αριθμό των bytes που στάληθηκαν στην επιτυχία και -1 σε error.
92 // ssize_t send(int sockfd, const void *buf, size_t len, int flags);
93 send( sockfd, &r, sizeof(r), 0);
94
95 send( sockfd, &n, sizeof(n), 0);
96
97 send( sockfd, X, n*sizeof(int), 0);
98 send( sockfd, Y, n*sizeof(int), 0);
99
100 // απελευθερώνουμε την δυναμικά δεσμευμένη μνήμη
101 free(X);
102 free(Y);
103
104 int flag = 1, i;
105 double ans1;
106 int ans2;
107 double ans3[2];
108 double *ans4;
109
110 while (flag){
111
112 // επιστρέφεται η επιλογή του χρήστη
113 opt = menu();
114
115 // αποστέλεται η επιλογή του χρήστη αν είναι 0 <= opt <= 4
116 if ( opt >= 0 && opt <=4)
117     send( sockfd, &opt, sizeof(opt), 0);
118
119 printf("\nAnswer:\n");
120
121 // λήψη απάντησης
122 switch(opt){
123
124 case 0:
125     // αν δωθεί 0 τότε ο χρήστης θέλει να τερματίσει
126     // και αλλάζει το flag από 1 σε 0
```

```
127     printf("Goodbye\n");
128     flag = 0;
129     break;
130 case 1:
131     // λήψη των δεδομένων και εκτύπωση
132     recv( sockfd, &ans1, sizeof(ans1), 0);
133     printf("The value of abs is: %lf\n",ans1);
134     break;
135 case 2:
136     // λήψη των δεδομένων και εκτύπωση
137     recv( sockfd, &ans2, sizeof(ans2), 0);
138     printf("The inner product: %d\n",ans2);
139     break;
140 case 3:
141     // λήψη των δεδομένων και εκτύπωση
142     recv( sockfd, ans3, sizeof(ans3), 0);
143     printf("The mean value: Ex = %lf, Ey = %lf\n",ans3[0],ans3[1]);
144     break;
145 case 4:
146     // λήψη των δεδομένων και εκτύπωση
147     ans4 = alloc_double(n);
148     recv( sockfd, ans4, n*sizeof(double), 0);
149     printf("r(X+Y):\n");
150
151     for( i = 0; i < n; i++){
152         printf("vec[%d] = %lf\n", i+1, ans4[i]);
153     }
154     free(ans4);
155     break;
156 default:
157     printf("Unknown option\n");
158 }
159 }
160
161 close(sockfd);
162
163 return 0;
164 }
```

Κώδικας 6.1: Αρχείο user

## 7 Ενδεικτικά τρεξίματα

Τρεξίματα για διάνυσμα  $X = 1, 2, 3$ ,  $Y = 4, 5, 6$ ,  $r = 3.3$  και γίνεται κλήση με την σειρά για τις επιλογές 1,2,3 και 4.

```

padelis@padelis-linux: ~/Documents/distributed_systems/proiv5
padelis@padelis-linux:~/Documents/distributed_systems/proiv5$ ./user_client localhost 6969
Host: localhost
Port: 6969
Trying connection...
Connected
Trace:
Double number for r: 3.3      10 10 1001
Length of the vectors: 3      0 0 1 0
X[1] = 1      0 1 0100 00
X[2] = 2      1 0 0 1000 00
X[3] = 3      1 0 0 1000 00
Y[1] = 4 eps
Y[2] = 5
Y[3] = 6

Choose an option:
1) Absolute of vector X
2) Product of vector X and Y
3) Mean value of each vector
4) Product of r with the addition of X and Y
0) Stop
1

Answer:
The value of abs is: 3.741657

Choose an option:
1) Absolute of vector X
2) Product of vector X and Y
3) Mean value of each vector
4) Product of r with the addition of X and Y
0) Stop
2

Answer:
The inner product: 32

Choose an option:
1) Absolute of vector X
2) Product of vector X and Y
3) Mean value of each vector
4) Product of r with the addition of X and Y
0) Stop
3

Answer:
The mean value: Ex = 2.000000, Ey = 5.000000

Choose an option:
1) Absolute of vector X
2) Product of vector X and Y

```

```

padelis@padelis-linux:~/Documents/distributed_systems/proiv5$ ./proj_server
^C
padelis@padelis-linux:~/Documents/distributed_systems/proiv5$

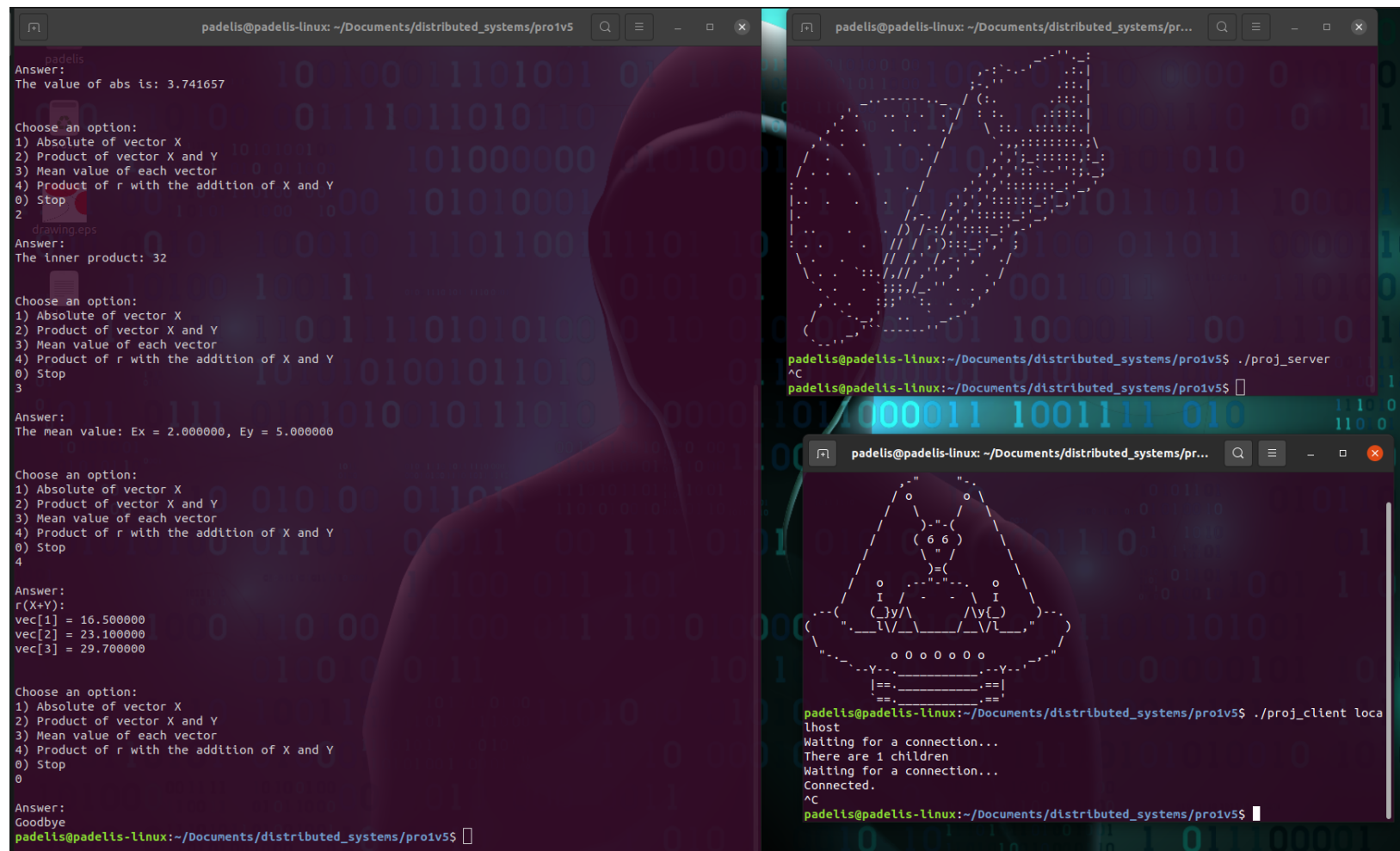
```

```

padelis@padelis-linux:~/Documents/distributed_systems/proiv5$ ./proj_client localhost
Waiting for a connection...
There are 1 children
Waiting for a connection...
Connected.
^C
padelis@padelis-linux:~/Documents/distributed_systems/proiv5$

```

Σχήμα 7.1: Εικόνα 1



```

padelis@padelis-linux: ~/Documents/distributed_systems/pro1v5
padelis
Answer:
The value of abs is: 3.741657

Choose an option:
1) Absolute of vector X
2) Product of vector X and Y
3) Mean value of each vector
4) Product of r with the addition of X and Y
0) Stop
2
drawing.eps
Answer:
The inner product: 32

Choose an option:
1) Absolute of vector X
2) Product of vector X and Y
3) Mean value of each vector
4) Product of r with the addition of X and Y
0) Stop
3
Answer:
The mean value: Ex = 2.000000, Ey = 5.000000

Choose an option:
1) Absolute of vector X
2) Product of vector X and Y
3) Mean value of each vector
4) Product of r with the addition of X and Y
0) Stop
4
Answer:
r(X+Y):
vec[1] = 16.500000
vec[2] = 23.100000
vec[3] = 29.700000

Choose an option:
1) Absolute of vector X
2) Product of vector X and Y
3) Mean value of each vector
4) Product of r with the addition of X and Y
0) Stop
0
Answer:
Goodbye
padelis@padelis-linux:~/Documents/distributed_systems/pro1v5$

padelis@padelis-linux:~/Documents/distributed_systems/pr...
padelis@padelis-linux:~/Documents/distributed_systems/pro1v5$ ./proj_server
^C
padelis@padelis-linux:~/Documents/distributed_systems/pro1v5$

padelis@padelis-linux:~/Documents/distributed_systems/pr...
padelis@padelis-linux:~/Documents/distributed_systems/pro1v5$ ./proj_client loca
lhost
Waiting for a connection...
There are 1 children
Waiting for a connection...
Connected.
^C
padelis@padelis-linux:~/Documents/distributed_systems/pro1v5$

```

Σχήμα 7.2: Εικόνα 2