

# ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ Εργαστήριο Άσκηση #1

Νικόλα Σιάκας 171011

## Socket Client

Ο socket client βρίσκεται στο αρχείο `client.c` και χρειάζεται ένα command line argument για να τρέξει, το hostname του socket server που θέλουμε να συνδεθούμε. Το πρώτο πράγμα που κάνει ο client είναι να συνδεθεί στον socket server καλώντας την συνάρτηση `connectToServer`. Η συνάρτηση αυτή επιστρέφει το socket descriptor που θα χρησιμοποιηθεί για την επικοινωνία του socket client με τον socket server. Ο client μετά καλεί την συνάρτηση `menu` η οποία εμφανίζει τις επιλογές για να διαλέξει ο χρήστης και επιστρέφει την επιλογή του. Στην συνέχεια αν η επιλογή του χρήστη δεν είναι η επιλογή της εξόδου ο client διαβάζει τον μήκος του πίνακα, τον πίνακα και το `a` μόνο αν η επιλογή ήταν η επιλογή του γινομένου πίνακα με έναν πραγματικό αριθμό. Σε αυτό το σημείο ο client έχει τα data που χρειάζεται αλλά πρέπει πρώτα να τα μετατρέψει σε network byte order για να τα στείλει μέσω του socket, οπότε μετατρέπει τους ακεραίους που έχει διαβάσει σε network byte order και τον πραγματικό αριθμό (στην περίπτωση γινομένου) σε string για να μπορέσει να τα στείλει. Αφού σταλούν τα δεδομένα ο client περιμένει να πάρει τα αποτελέσματα από τον server, στην περίπτωση του μέσου όρου ο server στέλνει ένα string που περιέχει το πραγματικό αριθμό. Στην περίπτωση του μικρότερου-μεγαλύτερου ο server στέλνει έναν πίνακα 2 ακεραίων, μικροτερο και μεγαλύτερο αντίστοιχα, αυτοί οι αριθμοί είναι σε network byte order οπότε ο client θα πρέπει να τα μετατρέψει στο δικό του byte order. Στην τελευταία περίπτωση ο server στέλνει ένα string το οποίο περιέχει πραγματικούς αριθμούς μέσα του, μήκους  $n * 20$  όπου  $n$  το μήκος του πίνακα  $Y$  που έδωσε ο χρήστης και 20 ένας αριθμός αρκετά μεγάλος για να χωρέσουν οι μεγαλύτεροι σε μήκος χαρακτήρων πραγματικοί αριθμοί μαζί με το τερματικό χαρακτήρα για τα strings. Αυτή η διαδικασία επαναλαμβάνεται έως ότου ο χρήστης διαλέξει να κάνει έξοδο από το πρόγραμμα, τότε στέλνετε ένα μήνυμα στον server για την επιλογή της εξόδου και κλείνει το socket.

## Socket Server / RPC Client

Το αρχείο `rpc_server_client.c` έχει δύο ρόλους, τον ρόλο του socket server και τον ρόλο του RPC client. Το πρόγραμμα αυτό δέχεται τα data από τον client και καλώντας την κατάλληλη συνάρτηση του RPC server παίρνει τα αποτελέσματα των υπολογισμών και τα στέλνει στον socket client. Το πρόγραμμα χρειάζεται ένα command line argument για να τρέξει, το hostname του RPC server. Όταν ξεκινάει η εκτέλεση του προγράμματος δημιουργείται ένα listening socket για να “ακούει” για νέες συνδέσεις και δεσμεύει την πόρτα. Όταν έρθει μια σύνδεση τότε ο server φτιάχνει ένα child process για να εξυπηρετήσει τον client που συνδέθηκε και περιμένει νέες συνδέσεις. Το child process καλεί την συνάρτηση `serve` και λαμβάνει τα data που στέλνει ο socket client, όταν τελειώσει με την επικοινωνία με καλεί την συνάρτηση που είναι υπεύθυνη για την RPC επικοινωνία. Η συνάρτηση αυτή αρχικοποιεί τα πεδία του `arguments struct` και ανάλογα με την επιλογή του χρήστη καλεί την αντίστοιχη συνάρτηση του RPC server. Για την επιλογή του μέσου όρου ο socket server παίρνει το αποτέλεσμα που είναι ένας πραγματικός αριθμός και τον μετατρέπει σε string για να μπορέσει να το στείλει σωστά μέσω του socket. Στην άλλη περίπτωση, την επιλογή του μικρότερου-μεγαλύτερου, το αποτέλεσμα βρίσκεται σε ένα struct, τα μεταφέρουμε σε ένα πίνακα 2 ακεραίων μετατρέποντας τα πρώτα σε network byte order. Στην τελευταία περίπτωση όπου το αποτέλεσμα που πρέπει να στείλουμε πίσω στον socket client είναι ένας πίνακας πραγματικών αριθμών, οπότε για να το στείλουμε πρέπει να μετατρέψουμε τους αριθμούς σε string όμως αν μετατρέψουμε έναν-έναν τους αριθμούς σε strings τότε θα πρέπει να κάνουμε  $n$  (όπου  $n$  το πλήθος των αριθμών στο διάνυσμα  $Y$ ) αποστολές μέσω του socket το οποίο θα καθυστερήσει την επικοινωνία. Μια λύση σε αυτή την καθυστέρηση είναι να στείλουμε ένα μεγάλο string με όλα τα data μαζί<sup>1</sup>, χρησιμοποιώντας την `sprintf` γράφουμε έναν-έναν τους πραγματικούς αριθμούς σε ένα buffer, δίνουμε σε κάθε αριθμό 20 χαρακτήρες για να κωδικοποιηθούν σε αυτό το διάστημα χωράνε και οι μεγαλύτεροι σε μήκος χαρακτήρων αριθμοί μαζί με τον τερματικό χαρακτήρα που τοποθετείται από τον `sprintf` για κάθε αριθμό που μετατρέπουμε. Για να σταματήσουμε το πρόγραμμα αυτό πατάμε `ctrl+c` και ο handler θα κλείσει το listening socket<sup>2</sup> πριν κάνει έξοδο.

---

<sup>1</sup> Δεν μπορούμε να στείλουμε έναν πίνακα από strings όπως κάναμε με τους ακεραίους στην προηγούμενη περίπτωση γιατί δεν μπορούμε να στείλουμε έναν πίνακα από pointers (όπως είναι ένας πίνακας από strings).

<sup>2</sup> Για να μπορέσει ο handler να κλείσει το socket, αυτό πρέπει να είναι global μεταβλητή.

## RPC Server

Στο αρχείο `rpc_server_server.c` υλοποιούνται οι συναρτήσεις που καλεί ο RPC client για να εκτελέσουν τους υπολογισμούς.

## Εκτέλεση και Ενδεικτικά Τρεξίματα

Για να τρέξουμε τα προγράμματα πρώτα πρέπει να κάνουμε compile. Πηγαίνουμε στον φάκελο `server` και τρέχουμε την εντολή `make` και αυτό θα κάνει compile το RPC κομμάτι της εργασίας. Τρέχουμε τον RPC server με την εντολή `./rpc_server_server` και έπειτα τρέχουμε τον RPC client σε ένα άλλο τερματικό με την εντολή `./rpc_server_client localhost`. Ανοίγουμε ένα ακόμα τερματικό στον αρχικό φάκελο της εργασίας αυτή την φορά και κάνουμε compile τον socket client με την εντολή `gcc client.c -o client` και τον τρέχουμε με την εντολή `./client localhost`. Περιμένουμε να δούμε το παρακάτω prompt.

```
➔ ./client localhost
Connected to the server
1. Average
2. Min, Max
3. a*Y
4. Exit
> |
```

## Έλεγχος ορθότητας αποτελεσμάτων

Διαλέγουμε την επιλογή του μέσου όρου και δίνουμε τα ενδεικτικά data 3, 4, 7, -1, 3 και όπως βλέπουμε στην εικόνα 1 το αποτέλεσμα είναι σωστό ( $16/5 == 3.2$ ) και το prompt για άλλη πράξη ξανά εμφανίστηκε. Αυτή την φορά θα διαλέξουμε την επιλογή 2 και θα δώσουμε τα ίδια data και όπως βλέπουμε στην εικόνα 2 τα αποτελέσματα είναι σωστά. Δοκιμάζουμε και την επιλογή 3 με τα ίδια data και  $a = 3.14$  και τα αποτελέσματα που περιμένουμε να πάρουμε είναι τα εξής 9.42, 12.56, 21.98, -3.14, 9.42 και όπως βλέπουμε στην εικόνα 3 τα αποτελέσματα είναι σωστά. Τέλος κάνουμε έξοδο με την επιλογή 4 και όπως βλέπουμε στην εικόνα 4 το πρόγραμμα σταμάτησε. Όπως βλέπουμε στην εικόνα 5 ο server μπορεί να εξυπηρετήσει πολλούς clients ταυτόχρονα.

```

→ ./client localhost
Connected to the server
1. Average
2. Min, Max
3. a*Y
4. Exit
> 1
Length of Y: 5
Y[0] = 3
Y[1] = 4
Y[2] = 7
Y[3] = -1
Y[4] = 3
-----
Average = 3.20
-----
1. Average
2. Min, Max
3. a*Y
4. Exit
>

```

Εικόνα 1

```

1. Average
2. Min, Max
3. a*Y
4. Exit
> 2
Length of Y: 5
Y[0] = 3
Y[1] = 4
Y[2] = 7
Y[3] = -1
Y[4] = 3
-----
Min = -1
Max = 7
-----

```

Εικόνα 2

```

1. Average
2. Min, Max
3. a*Y
4. Exit
> 3
Length of Y: 5
Y[0] = 3
Y[1] = 4
Y[2] = 7
Y[3] = -1
Y[4] = 3
a = 3.14
-----
a*Y = 9.42 12.56 21.98 -3.14 9.42
-----

```

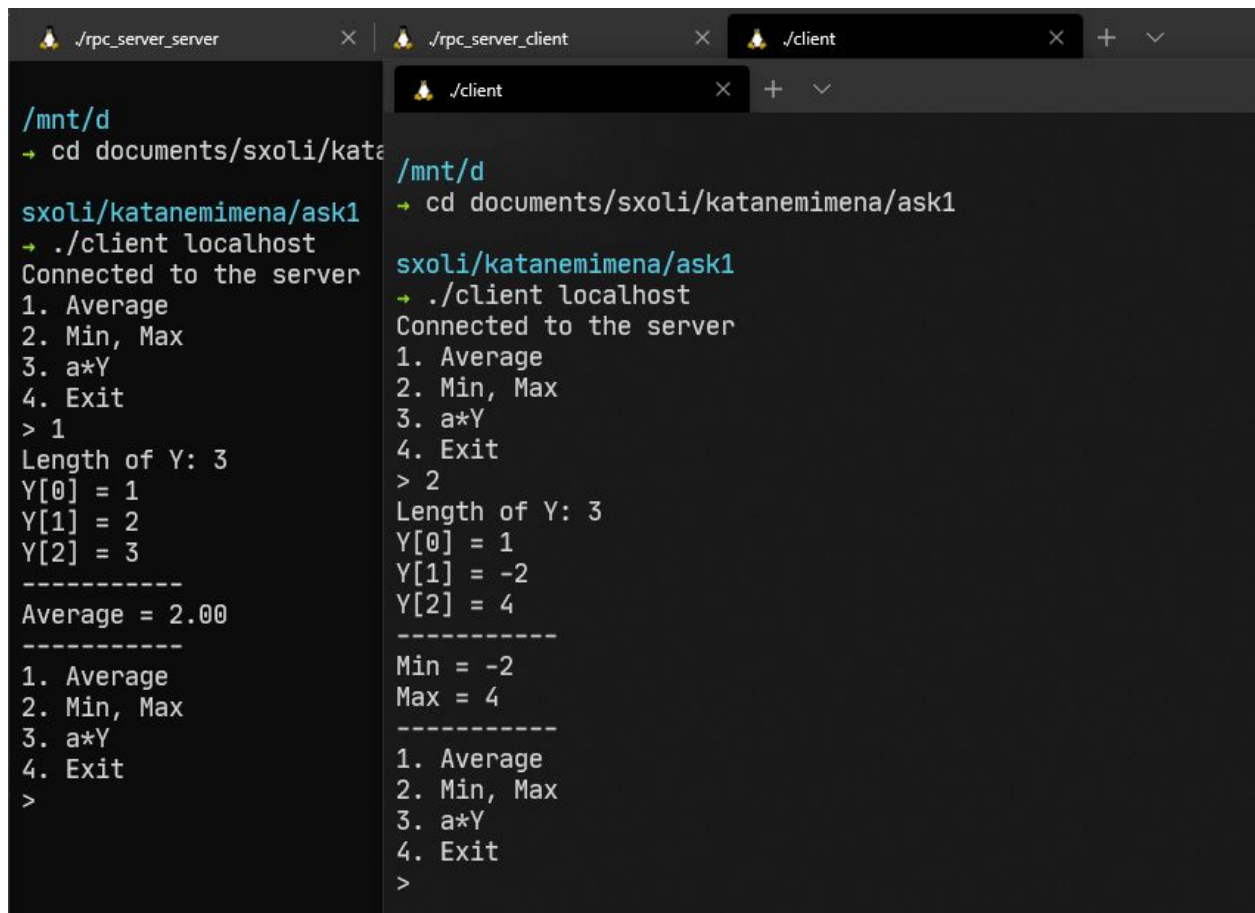
Εικόνα 3

```

1. Average
2. Min, Max
3. a*Y
4. Exit
> 4
sxoli/katanemimena/ask1
→ |

```

Εικόνα 4



```
./rpc_server_server      x  ./rpc_server_client      x  ./client      x  +  v
/mnt/d
→ cd documents/sxoli/katanemimena/ask1
sxoli/katanemimena/ask1
→ ./client localhost
Connected to the server
1. Average
2. Min, Max
3. a*Y
4. Exit
> 1
Length of Y: 3
Y[0] = 1
Y[1] = 2
Y[2] = 3
-----
Average = 2.00
-----
1. Average
2. Min, Max
3. a*Y
4. Exit
>

./client      x  +  v
/mnt/d
→ cd documents/sxoli/katanemimena/ask1
sxoli/katanemimena/ask1
→ ./client localhost
Connected to the server
1. Average
2. Min, Max
3. a*Y
4. Exit
> 2
Length of Y: 3
Y[0] = 1
Y[1] = -2
Y[2] = 4
-----
Min = -2
Max = 4
-----
1. Average
2. Min, Max
3. a*Y
4. Exit
>
```

Εικόνα 5