ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
Distributed Systems

ΕΡΓΑΣΙΑ 1

Όνομα Μαθητή : Άγγελος Κωνσταντίνος

Επίθετο Μαθητή : Μεντζέλος

Αριθμός Μητρώου : 21390132

Ημερομηνία ολοκλήρωσης :

19/05/2024

Περιεχόμενα

1. Σχολιασμός Κώδικα

1.1 Το αρχείο stats.x

Το αρχείο stats.x περιέχετε από 4 δομές (2 δομές εισόδου και 2 δομές εξόδου) και από το βασικό πρόγραμμα. Αρχικά η πρώτη δομή εισόδου input1 περιέχει σαν χαρακτηριστικά έναν μεταβαλλόμενο πίνακα ακεραίων y που θα είναι το διάνυσμα y που ζητείται. Αντίστοιχα η δεύτερη δομή εισόδου input2 περιέχει σαν χαρακτηριστικά έναν μεταβαλλόμενο πίνακα ακεραίων y που θα είναι το διάνυσμα y που ζητείται και έναν double αριθμό που θα είναι η τιμή a που ζητείται. Η δομή εξόδου output1 περιέχει σαν χαρακτηριστικά έναν πίνακα ακεραίων δύο θέσεων για την ελάχιστη και μέγιστη τιμή που ζητούνται. Η δομή εξόδου output2 περιέχει έναν μεταβαλλόμενο πίνακα double που θα έχει τις τιμές μετά από το γινόμενο που ζητείτε.

Αμέσως μετά από τις δομές υπάρχει το πρόγραμμα το οποίο έχει μια version ADD_VERS που έχει 3 συναρτήσεις με βάσεις τις δομές που αναλύσαμε παραπάνω. Η πρώτη συνάρτηση είναι η double average (input1) = 1 η οποία έχει σαν έξοδο έναν αριθμό double και σαν είσοδο την δομή input1 που ανέφερα παραπάνω με σκοπό να υπολογίζω τον μέσο όρο του πίνακα y. Η συνάρτηση αυτή ισούται με 1 που αυτό μας δείχνει τον μοναδικό αριθμό ταυτότητας που έχει . Η δεύτερη συνάρτηση είναι η output1 min_max1(input1) = 2 η οποία έχει σαν έξοδο μια δομή output1 που ανέφερα και παραπάνω και μια είσοδο δομής input1 με σκοπό να βρίσκω την ελάχιστη και την μέγιστη τιμή του πίνακα y. Αντίστοιχα με την προηγούμενη συνάρτηση έτσι και εδώ μας δείχνει μοναδικό αριθμό ταυτότητας. Η τρίτη συνάρτηση είναι η output2 prod_aY(input2) = 3 η οποία έχει σαν έξοδο μια δομή output2 που ανέφερα και παραπάνω και μια είσοδο δομής input2 με σκοπό να βρίσκω το γινόμενο του πίνακα με την τιμή a. Αντίστοιχα και εδώ σε σχέση με τις δυο προηγούμενες συναρτήσεις έτσι και εδώ μας δείχνει μοναδικό αριθμό ταυτότητας.

Ο δεκαεξαδικός αριθμός που έχει στο τέλος του προγράμματος μας δείχνει μοναδικά το πρόγραμμα στο RPC σύστημα. Το αρχείο stats.x είναι αυτό με το οποίο δημιουργώ τον RPC client και RPC server.

1.2 Socket Client

Για την υλοποίηση του socket client αρχικά δήλωσα τις βιβλιοθήκες και μία συνάρτηση για να εμφανίζω μηνύματα λάθους. Στην συνάρτηση main αρχικά δηλώνω τις μεταβλητές που θα χρειαστώ και ελέγχω αν έχουν δοθεί 3 ορίσματα δηλαδή το hostname και το port. Δηλώνω το socket ως AF_INET και μετατρέπω το port number σε ακέραιο για να το βάλω στην μεταβλητή portno. Ελέγχω αν έχει δημιουργηθεί επιτυχώς το socket και με βάση το hostname παίρνω το όνομα του για να το βάλω στην μεταβλητή server που είναι τύπου (struct hostent *). Ελέγχω επίσης αν το όνομα του server έχει την τιμή null. Γεμίζω με 0 όσο είναι το μέγεθος του serv_addr το οποίο είναι μεταβλητή τύπου (struct sockaddr_in), βάζω στο χαρακτηριστικό της δομής (sockaddr_in) serv_addr.sin_family την τιμή του AF_INET ώστε να δηλώσω σε τι family ανήκει ο server, αντιγράφω την διεύθυνση του server σε χαρακτηριστικό της δομής με όνομα serv_addr.sin_addr.s_addr και βάζω στο χαρακτηριστικό port της δομής το port που πήρα από τα ορίσματα. Γίνεται έλεγχος αν γίνει η σύνδεση με τον server και αν συμβεί με επιτυχία ξεκινάει η λειτουργία που ζητείτε από την άσκηση.

Αρχικά μέσα σε επαναληπτικό βρόγχο (μέχρι να δώσει την τιμή 4 ώστε να τερματίσει ο κώδικας) ζητάει από τον χρήστη να δώσει μια από τις τέσσερις επιλογές (1. Μέση τιμή του πίνακα y, 2. Μέγιστη και ελάχιστη τιμή του πίνακα y, 3. γινόμενο του πίνακα y με έναν double αριθμό a, 4. Έξοδος) ελέγχοντας αρχικά αν δώσει 4 να στείλει την τιμή της επιλογής στον server (send) να τερματίσεις ο βρόγχος και μετά το πρόγραμμα. Αν δεν συμβεί αυτό, στέλνει και πάλι την τιμή της επιλογής ελέγχοντας αν η επιλογή είναι 3 ώστε πρώτα να διαβάσει και να στείλει τον double αριθμό στον server. Στην συνέχεια διαβάζει το μέγεθος του διανύσματος y, δεσμεύει δυναμικά (έλεγχος για σωστή δέσμευση) το διάνυσμα y με την τιμή n που διάβασε και ξεκινάει να διαβάζει τα στοιχεία τα οποία θα βάλει στο διάνυσμα. Μόλις τελειώσει η επαναληπτική διαδικασία για να διαβάζει τα στοιχεία, στέλνει το μέγεθος και μετά τα στοιχεία του y. Όταν γίνει αυτή η διαδικασία ανάλογα με την επιλογή που έχει δώσει ο χρήστης περιμένει μέσο της εντολής recv να λάβει αυτό που επέλεξε. Αν έχει δώσει 1 τότε θα περιμένει μέχρι να λάβει τον μέσο όρο. Αντίστοιχα αν η επιλογή του είναι 2

τότε θα λάβει ένα min και ένα max. Ενώ αν δώσει 3 θα λάβει το διάνυσμα που είχε δώσει πολλαπλασιασμένο κατά τον αριθμό a.

Τέλος αποδεσμεύω ότι μνήμη χρησιμοποίησα και μόλις τερματίσει ο βρόγχος θα κλείσει το socket και θα τερματίσει το πρόγραμμα. Ο κώδικας του socket client έχει σκοπό να εξυπηρετεί τον πελάτη με τον RPC client.

1.3 Socket Server/RPC client

Για την υλοποίηση του socket server/RPC client αρχικά δήλωσα τις βιβλιοθήκες τις οποίες θα χρειαστώ , την συνάρτηση error για να εμφανίζω μηνύματα λάθους και μία δομή ώστε να μεταφέρω τις παραμέτρους που θέλω ώστε να υλοποιήσω το multithreading. Η δομή περιέχει την ακέραια τιμή του socket server και το hostname.

Η συνάρτηση για την κλήση των rpc συναρτήσεων (void add_prog_1) δέχεται ως ορίσματα την επιλογή του χρήστη, την τιμή του socket, το μέγεθος του y, το διάνυσμα y, τον double αριθμό a, την ακέραια μεταβλητή done για το πότε θα λήξει και το hostname. Οι δηλώσεις των αρχικών μεταβλητών έγιναν αυτόματα μαζί με την συνάρτηση (χωρίς τα ορίσματα που του πρόσθεσα εγώ). Μετά ανοίγει η σύνδεση με τον rpc server μέσω udp (γίνεται επίσης αυτόματα με την δημιουργία του αρχείου). Στις μεταβλητές average_1_arg , min_max1_1_arg και prod_ay_1_arg που είναι μεταβλητές τύπου δομής input1 οι δύο πρώτες και input2 η τρίτη βάζω στο χαρακτηριστικό τους y.y_len το μέγεθος του y (n) και στο χαρακτηριστικό τους y.y_val τα δέσμευσα δυναμικά με βάση την τιμή n. Έλεγξα αν έγινε η δυναμική δέσμευση και στις 3 μεταβλητές και γέμισα και τους 3 πίνακες με τις αντίστοιχες τιμές του διανύσματος y. Αρχικοποίησα κάποιες μεταβλητές τις οποίες χρειάστηκα. Με την επιλογή την οποία έχω καλείται η αντίστοιχη συνάρτηση του RPC server. Αν η επιλογή είναι 1 καλείται η average_1 με ορίσματα την δομή average_1_arg (input1) και την μεταβλητή τύπου CLIENT. Επιστρέφει έναν double αριθμό από τον RPC server και να γίνει με επιτυχία στέλνω μέσω του server socket την μεταβλητή mean που έχει την τιμή double που δέχθηκε ο RPC client. Αντίστοιχα αν η επιλογή είναι 2 καλείται η συνάρτηση min_max1_1 με ορίσματα πάλι μια δομή min_max1_1_arg (input1) και την μεταβλητή τύπου CLIENT. Επιστρέφει μια δομή τύπου (output1) και αν έρθει το αποτέλεσμα επιτυχώς στέλνω τις δύο τιμές της δομής στον socket client. Αντίστοιχα και στην 3 επιλογή αρχικοποιώ το χαρακτηριστικό της δομής prod_ay_1_arg (a) την τιμή a και καλώ την συνάρτηση prod_ay_1 με παραμέτρους την δομή prod_ay_1_arg (input2) και αν επιστρέψει επιτυχώς την δομή τύπου output2 στέλνω το γινόμενο του διανύσματος y με την τιμή του a. Τέλος της συνάρτησης void add_prog_1 αν η επιλογή είναι 4 τότε κάνει την τιμή done ίσων 1, αποδεσμεύει τις δυναμικά δεσμεύσεις που έκανα και κάνει destroy την σύνδεση με τον RPC server.

Η συνάρτηση void *handle_client είναι μια συνάρτηση η οποία δημιουργήθηκε με σκοπό να διαχειρίζεται τους πολλαπλούς clients. Αρχικά δέχομαι τα στοιχεία τις δομής που δήλωσα στην αρχή του κώδικα και δηλώνω κάποιες μεταβλητές τις οποίες θα χρειαστώ. Επαναληπτικά λοιπόν δέχεται την επιλογή του client , αν είναι 4 τελειώνει η επανάληψη και κλείνει το socket. Αν είναι 3 δέχεται τον double αριθμό (a) και μετά δέχεται το μέγεθος του y αλλά και το διάνυσμα y. Τέλος τις συνάρτησης καλείται η συνάρτηση add_prog_1 με παραμέτρους την επιλογή του χρήστη, την τιμή του socket, το μέγεθος του y, το διάνυσμα y, τον double αριθμό a, την ακέραια μεταβλητή done για το πότε θα λήξει και το hostname. Μόλις τελειώσει η συνάρτηση αποδεσμεύω την δυναμική δέσμευση που έχω κάνει και αν τελειώσει η επανάληψη κλείνει το socket και τελειώνει ο client.

Στην συνάρτηση int main δηλώνω κάποιες μεταβλητές που θα χρειαστώ, ελέγχω αν έχουν δοθεί 3 ορίσματα (hostname και port) αν έχει γίνει αυτό βάζω στις μεταβλητές host και port τα αντίστοιχα ορίσματα. Δημιουργώ το server socket που είναι τύπου AF_INET ελέγχω αν δημιουργήθηκε σωστά το socket και αμέσως μετά γεμίζω τα πεδία της δομής serv_addr (sockaddr_in) με το port , το family (AF_INET) και το address. Μετά κάνω bind, ελέγχω αν έγινε επιτυχώς και βάζω να ακούει μέχρι 5 συνδέσεις. Μέσω επανάληψης κάνω accept clients (ελέγχοντας αν έγινε σωστά το accept) και γεμίζω τα πεδία της δομής που έχω φτιάξει ώστε με την pthread_create (void * handle_client) να δέχομαι πολλαπλούς clients. Όταν τελειώσει η επανάληψη, κλείνει το socket και τερματίζει το πρόγραμμα.

1.4 RPC Server

Στον RPC Server υπάρχουν οι 3 συναρτήσεις οι οποίες δημιούργησα με το stats.x αρχείο με σκοπό να υπολογίζω αυτά που θέλει ο client. Υπάρχουν 3 συναρτήσεις στον κώδικα. Αρχικά η πρώτη συνάρτηση double * average_1_srv η οποία δέχεται σαν ορίσματα μια δομή τύπου input1 (αναλύθηκε στο 1.1) και μια μεταβλητή τύπου CLIENT και επιστρέφει έναν double αριθμό. Αυτή η συνάρτηση υπολογίζει το μέσο όρο του διανύσματος y. Για να γίνει αυτό αρχικοποίησα μια μεταβλητή sum ώστε να κρατάω το συνολικό άθροισμα(το result δηλώνεται αυτόματα με την δημιουργία του αρχείου). Με μια for loop παίρνω από το όρισμα της δομής το y.y_val και τα προσθέτω μεταξύ τους. Τέλος για να υπολογίσω τον μέσο όρο, διαιρώ το συνολικό άθροισμα με το μέγεθος του y που βρίσκεται στην δομή ως y.y_len και επιστέφω πίσω στον RPC client το αποτέλεσμα.

Στην δεύτερη συνάρτηση output1 * min_max1_1_srv παίρνει ακριβώς τα ίδια ορίσματα με την πρώτη ενώ επιστρέφει δομή τύπου output1 * (αναλύθηκε στο 1.1) . Αντίστοιχα είναι δηλωμένη η μεταβλητή result και αρχικοποιώ το χαρακτηριστικό min_max της δύο θέσεις του με την αρχική τιμή της δομής input1(y.y_val[0]). Επαναληπτικά λοιπόν βρίσκω την ελάχιστη και την μέγιστη τιμή του διανύσματος και όταν τις βρω και τελειώσει η επανάληψη επιστρέφω την δομή στον RPC client.

Στην τρίτη συνάρτηση output2 *prod_ay_1_svc παίρνει ως ορίσματα την δομή input2 (αναλύθηκε στο 1.1) και μια μεταβλητή τύπου CLIENT. Επιστρέφει μια δομή τύπου output2 (και αυτή αναλύθηκε στο 1.1). Όπως και στα προηγούμενα έτσι και εδώ η μεταβλητή result είναι δηλωμένη από πριν. Παίρνω το χαρακτηριστικό του output2 (pr_aY) και βάζω στο pr_aY.pr_aY_len το μέγεθος που έχει στα χαρακτηριστικά του η δομή input2. Δεσμεύω δυναμικά την τιμή του χαρακτηριστικού pr_aY , ελέγχω αν έγινε σωστά η δέσμευση και υπολογίζω επαναληπτικά το γινόμενο του χαρακτηριστικού a της δομής input2 με το διάνυσμα της δομής αυτής. Τέλος επιστρέφω την δομή αυτή πίσω στον RPC client.

2. Ενδεικτικά τρεξίματα

2.1 Μεταγλώττιση

Για να καταφέρουμε να τρέξουμε τους κώδικες θα πρέπει αρχικά με την δημιουργία του stats.x αρχείου να τρέξουμε την εντολή rpcgen -C -a stats.x ώστε να φτιάξουμε τα stats_client.c, stats_server.c μαζί με κάποια έξτρα αρχεία. Στην συνέχεια, μετονομάζουμε το Makefile.stats σε Makefile (mv Makefile.stats Makefile) ώστε να μπορέσουμε να κάνουμε compile όλα τα αρχεία με μια εντολή την make. Πριν τρέξουμε το make κάνουμε μια αλλαγή στο Makefile αρχείο βάζοντας στα παρακάτω πεδία τα συγκεκριμένα flags:

```
CFLAGS += -g -DRPC_SVC_FG -
I/usr/include/tirpc

LDLIBS += -lnsl -ltirpc -pthread

RPCGENFLAGS = -C
```

Για την μεταγλώττιση του socket client θα πρέπει να γίνει όπως φαίνεται παρακάτω:

```
gcc -o socket_clnt socket_clnt.c
```

2.2 Εκτέλεση

Τώρα θα τρέξουμε τα τρία προγράμματα τα οποία μας ενδιαφέρουν με συγκεκριμένη σειρά. Αρχικά θα τρέξουμε τον RPC server όπως φαίνεται παρακάτω:

```
./stats_server
```

Αμέσως μετά θα τρέξουμε τον RPC client / socket server βάζοντας hostname και port σαν ορίσματα όπως φαίνεται παρακάτω:

```
./stats_client localhost 8080
```

Τέλος θα εκτελέσουμε τον socket client βάζοντας hostname και port σαν ορίσματα ίδια με του rpc client όπως φαίνεται παρακάτω:

```
./socket_clnt localhost 8080
```

2.3 Τρεξίματα

Στα ενδεικτικά τρεξίματα θα δούμε διάφορα παραδείγματα είτε με επαναληπτικότητα σε έναν χρήστη είτε εξυπηρέτηση πολλαπλών χρηστών με επαναληπτικότητα ο ένας και ο άλλος ένα τρέξιμο είτε να μην τρέχει RPC server και socket server. Για να τρέξουμε τα παραδείγματα θα πρέπει να τρέξουμε τα προγράμματα με την εξής σειρά:

1) ./stats_server 2) ./stats_client 3) ./socket_clnt

Αρχικά θα τρέξουμε το πρώτο παράδειγμα:

RPC Server

```
filegeiasou@DESKTOP-M2ECFMB:/mnt/c/Users/fileg/Documents/Code$ ./stats_server
```

RPC Client/Socket Server

```
filegeiasou@DESKTOP-
M2ECFMB:/mnt/c/Users/fileg/Documents/Code$
./stats_client localhost 8080
epil = 1
n = 4
3 5 7 9
Mean of Y: 6.000000
epil = 2
n = 7
1 2 3 4 5 6 7
Min of Y: 1
Max of Y: 7
epil = 3
n = 2
1 2
a*Y[0] = 5.000000
a*Y[1] = 10.000000
epil = 5
n = 2
1 2
```

Client Socket

```
filegeiasou@DESKTOP-
M2ECFMB:/mnt/c/Users/fileg/Documents/Code$
./socket_clnt localhost 8080
Trying to connect...
Connected.
Give selection
1)average 2)min and max 3)product a*y 4)Exit: 1
Enter the length of the vector: 4
Enter the elements of the vector: 3 5 7 9
Mean of Y: 6.000000
Give selection
1)average 2)min and max 3)product a*y 4)Exit: 2
Enter the length of the vector: 7
Enter the elements of the vector: 1 2 3 4 5 6 7
Min of Y: 1, Max of Y: 7
Give selection
1)average 2)min and max 3)product a*y 4)Exit: 3
Enter a real number: 5.0
Enter the length of the vector: 2
Enter the elements of the vector: 1 2
a*Y: 5.000000 10.000000
Give selection
1)average 2)min and max 3)product a*y 4)Exit: 5
Enter the length of the vector: 2
Enter the elements of the vector: 1 2
Give selection
1)average 2)min and max 3)product a*y 4)Exit: 4
```

Παρατηρούμε ότι με την επαναληπτική λειτουργία του menu λειτουργεί σωστά και επιστρέφονται σωστά τα αποτελέσματα.

Ας δούμε το δεύτερο παράδειγμα με την χρήση πολλαπλών χρηστών:

RPC Server

```
filegeiasou@DESKTOP-M2ECFMB:/mnt/c/Users/fileg/Documents/Code$ ./stats_server
```

RPC Client/Socket Server

```
filegeiasou@DESKTOP-M2ECFMB:/mnt/c/Users/fileg/Documents/Code$ ./stats_client localhost 8080
epil = 1
epil = 2
n = 5
1 2 3 4 5
Mean of Y: 3.000000
n = 3
3 4 5
Min of Y: 3
Max of Y: 5
epil = 3
n = 2
1 2
a*Y[0] = 2.000000
a*Y[1] = 4.000000
```

Client Socket 1

```
filegeiasou@DESKTOPM2ECFMB:/mnt/c/Users/fileg/Document
s/Code$ ./socket_clnt localhost 8080
Trying to connect...
Connected.
Give selection
1)average 2)min and max 3)product a*y 4)Exit: 1
Enter the length of the vector: 5
Enter the elements of the vector: 1 2 3 4 5
Mean of Y: 3.000000
Give selection
1)average 2)min and max 3)product a*y 4)Exit: 3
Enter a real number: 2
Enter the length of the vector: 2
Enter the elements of the vector: 1 2
a*Y: 2.000000 4.000000
Give selection
1)average 2)min and max 3)product a*y 4)Exit: 4
```

Client Socket 2

```
filegeiasou@DESKTOPM2ECFMB:/mnt/c/Users/fileg/Documen
ts/Code$ ./socket_clnt localhost 8080
Trying to connect...
Connected.
Give selection
1)average 2)min and max 3)product a*y 4)Exit: 2
Enter the length of the vector: 3
Enter the elements of the vector: 3 4 5
Min of Y: 3, Max of Y: 5
Give selection
1)average 2)min and max 3)product a*y 4)Exit: 4
```

Παρατηρούμε ότι και με την λειτουργία πολλαπλών χρηστών το πρόγραμμα δίνει σωστά αποτελέσματα και σωστά ταυτόχρονα και στους δυο clients.

Στο τρίτο παράδειγμα θα δούμε σε περίπτωση που δεν τρέχει ένας από τους δύο server(socket/RPC) τι θα συμβεί. Πρώτα βλέπουμε όταν δεν τρέχει ο RPC Server και μετά όταν δεν τρέχει ο Socket Server.

RPC Client/Socket Server

```
filegeiasou@DESKTOPM2ECFMB:/mnt/c/Users/fileg/Document
s/Code$ ./stats_client localhost 8080
epil = 1
n = 2
1 2
call failed: RPC: Unable to receive; errno =
Connection refused
```

Client Socket

```
filegeiasou@DESKTOPM2ECFMB:/mnt/c/Users/fileg/Documen
ts/Code$ ./socket_clnt localhost 8080
Trying to connect...
Connected.
Give selection
1)average 2)min and max 3)product a*y 4)Exit: 1
Enter the length of the vector: 2
Enter the elements of the vector: 1 2
```

RPC Server

```
filegeiasou@DESKTOPM2ECFMB:/mnt/c/Users/file
g/Documents/Code$ ./stats_server
```

Client Socket

```
filegeiasou@DESKTOPM2ECFMB:/mnt/c/Users/file
g/Documents/Code$ ./socket_clnt localhost
8080
Trying to connect...
ERROR connecting: Connection refused
```

Παρατηρούμε ότι αν δεν τρέχει ο RPC server δεν μπορούμε να πάρουμε τα αποτελέσματα με σκοπό να μην εμφανίζει κάτι στον client. Αν αντίστοιχα δεν τρέχει ο RPC client/socket server έχουμε ως αποτέλεσμα να μην μπορεί να συνδεθεί ο client ώστε να ζητήσει αυτά που θέλει.

3. Επίλογος

Κλείνοντας το σχολιασμό μου στο κώδικα τον οποίο υλοποίησα, έχω δώσει σωστές απαντήσεις στα ερωτήματα της άσκησης. Ο socket server/RPC client έχει υλοποιηθεί με AF_INET σύνδεση, επαναληπτικά, με την χρήση multithreading και ο υπολογισμός γίνεται με βάση του RPC server (όπως ορίσαμε το stats.x αρχείο) και όχι μέσο τοπικής συνάρτησης. Στα ενδεικτικά τρεξίματα βλέπουμε ότι οι κώδικες οι οποίοι έχω υλοποιήσει μεταγλωττίζονται και τρέχουν με σωστά αποτελέσματα.