Tcp/ip

ΜΟΥΓΑΚΟΣ ΔΗΜΗΤΡΙΟΣ mougakosd@gmail.com

ΤΣΑΊ΄ΡΙΔΗΣ ΣΤΑΥΡΟΣ tsairidis@gmail.com

**Το μοντέλο πελάτη-server**

Ο μεγαλύτερος όγκος επικοινωνίας στην ενδιάμεση επικοινωνία υλοποιείται με την χρήση του μοντέλου server-πελάτη. Αυτοί οι όροι αναφέρονται στις δύο διαδικασίες που πραγματοποιούνται ώστε να επικοινωνούν πλάτης και serverς μεταξύ τους. Στη μία από τις δύο διαδικασίες, ο client, συνδέεται στην άλλη διαδικασία, που ο serverς, κάνει ένα αίτημα για πληροφορίες. Μια καλή αναλογία είναι ένα φυσικό πρόσωπο που τηλεφωνεί σε ένα άλλο άτομο.

Σημειώνεται πως ο client πρέπει να γνωρίζει την ύπαρξη και τη διεύθυνση του server, αλλά ο serverς δεν είναι αναγκαίο να γνωρίζει τη διεύθυνση (ή ακόμη και την ύπαρξη) του πελάτη πριν από τη δημιουργία της σύνδεσης.

Επιπρόσθετα, μόλις δημιουργηθεί μια σύνδεση, οι δύο πλευρές μπορούν να στείλουν και να λάβουν πληροφορίες

Η κλήση του συστήματος για την υλοποίηση μία σύνδεσης είναι κάπως διαφορετική για τον πελάτη σε σχέση με τον εξυπηρετητή, αλλά και στις δύο περιπτώσεις περιλαμβάνουν τη βασική κατασκευή μιας υποδοχής- socket. Ένα socket είναι το ένα άκρο ενός διαύλου επικοινωνίας μεταξύ διαδικασιών. Και στις δύο διαδικασίες, ο καθένας εκ των πελάτη-server, δημιουργεί το δικό του socket.

Τα βήματα που εμπλέκονται στη δημιουργία ενός socket από την πλευρά του πελάτη είναι τα παρακάτω:

1. Δημιουργία ενός socket με το socket () system call.

2. Σύνδεση του socket στη διεύθυνση του server χρησιμοποιώντας την connect () system call.

3. Αποστολή και λήψη δεδομένων. Υπάρχουν διαφορετικοί τρόποι να πραγματοποιηθεί αυτό, αλλά το πιο απλό είναι η χρήση read () και write ().

Τα βήματα που χρησιμοποιούνται στη δημιουργία ενός socket στην πλευρά του server είναι τα εξής:

1. Δημιουργία ενός socket με το socket () system call.

2. Σύνδεση του socket σε μια διεύθυνση χρησιμοποιώντας την bind (). Για ένα socket του server στο Διαδίκτυο, μια διεύθυνση αποτελείται από έναν αριθμό ports στh μηχανή host.

3. Ακρόαση της συνδέσης με την [listen()](http://www.linuxhowtos.org/manpages/2/listen.htm).

4. Αποδοχή σύνδεσης με accept (). Αυτή η κλήση υλοποιέιται συνήθως μέχρι να συνδεθεί ένας client με το server.

5. Αποστολή και λήψη δεδομένων.

**Τύποι Socket**

Όταν δημιουργείται ένα Socket, το πρόγραμμα πρέπει να καθορίσει τον domain της διεύθυνσης και τον τύπο του Socket. Δύο διεργασίες μπορούν να επικοινωνούν μεταξύ τους μόνο εάν τα Sockets τους είναι του ίδιου τύπου και στο ίδιο domain.

Υπάρχουν δύο ευρέως χρησιμοποιούμενες διευθύνσεις domain , το domain unix, στον οποίο επικοινωνούν δύο διαδικασίες που μοιράζονται ένα κοινό σύστημα αρχείων και το domain του Διαδικτύου στον οποίο επικοινωνούν δύο διαδικασίες που εκτελούνται σε οποιονδήποτε δύο κεντρικούς υπολογιστές στο Διαδίκτυο. Κάθε μία από αυτές έχει τη δική της μορφή διεύθυνσης.

Η διεύθυνση ενός socket στο domain του Unix είναι μια συμβολοσειρά χαρακτήρων που είναι βασικά μια καταχώρηση στο σύστημα αρχείων.

Η διεύθυνση socket στον τομέα του Διαδικτύου αποτελείται από τη διεύθυνση Internet του κεντρικού υπολογιστή (κάθε υπολογιστής στο Διαδίκτυο έχει μια μοναδική διεύθυνση 32 bit, συχνά αναφέρεται ως η διεύθυνση IP του).

Επιπλέον, κάθε socket χρειάζεται έναν port number στον συγκεκριμένο κεντρικό υπολογιστή.

Οι αριθμοί των port είναι ακέραιοι αριθμοί 16 bit χωρίς υπογραφή.

Οι χαμηλότεροι αριθμοί παραμένουν στο Unix για τυπικές υπηρεσίες. Για παράδειγμα, ο αριθμός θύρας για το server FTP είναι to 21. Είναι σημαντικό οι τυπικές υπηρεσίες να βρίσκονται στο ίδιο port, σε όλους τους υπολογιστές ώστε οι πελάτες να γνωρίζουν τις διευθύνσεις τους.

Ωστόσο, οι αριθμοί των port άνω του 2000 είναι γενικά διαθέσιμοι.

Υπάρχουν δύο τύποι socket που χρησιμοποιούνται ευρέως, τα stream socket και τα datagram socket. Τα stream socket επεξεργάζονται τις επικοινωνίες ως συνεχή ροή χαρακτήρων, ενώ οι υποδοχές datagram πρέπει να διαβάζουν ολόκληρα τα μηνύματα ταυτόχρονα. Ο καθένας χρησιμοποιεί το δικό του πρωτόκολλο επικοινωνιών.

Οι stream sockets χρησιμοποιούν πρωτόκολλο TCP (Πρωτόκολλο Ελέγχου Μεταφοράς), το οποίο είναι ένα αξιόπιστο πρωτόκολλο προσανατολισμένο προς ροή και οι sockets datagram χρησιμοποιούν UDP (Unix Datagram Protocol), το οποίο δεν είναι τόσο αξιόπιστο.

**Κώδικας**

Ο κώδικας C, παρέχεται για έναν πολύ απλό μοντέλο πελάτη και server. Τα δύο μέρη επικοινωνούν χρησιμοποιώντας sockets ροής στον domain του Διαδικτύου. Ο κώδικας περιγράφεται λεπτομερώς παρακάτω. Ωστόσο, προτού διαβαστούν οι περιγραφές και ο κώδικας, θα πρέπει να συνταθεί και να εκτελέστουν τα δύο προγράμματα για να δούμε τι κάνουν.

Πιθανότατα δεν θα απαιτούν ειδικές «σημαίες» καταγραφής, αλλά σε ορισμένα συστήματα Solaris ίσως χρειαστεί να συνδεθούν με τη βιβλιοθήκη socket προσθέτοντας **-lsocket** στην εντολή compile.

Στην ιδανική περίπτωση, θα πρέπει να εκτελεστεί τον client και ο server σε ξεχωριστούς hosts στο Internet. Ξεκινάμε πρώτα τον server.

Ο client θα ζητήσει να εισαχθεί ένα μήνυμα.

Εάν όλα λειτουργούν σωστά, ο server θα εμφανίσει το μήνυμα σας στο stdout, θα στείλει ένα μήνυμα επιβεβαίωσης στον πελάτη και θα τερματιστεί.

Ο client θα εκτυπώσει το μήνυμα επιβεβαίωσης από το server και θα τερματιστεί.

Μπορούμε να το προσομοιώσουμεσε ένα μόνο μηχάνημα, εκτελώντας τον server σε ένα παράθυρο και τον πελάτη σε ένα άλλο. Σε αυτήν την περίπτωση, μπορούμε να χρησιμοποιήσουμε τη λέξη-κλειδί localhost ως το πρώτο όρισμα στον πελάτη.

Ο κώδικας του server χρησιμοποιεί έναν αριθμό άσχημων κατασκευών προγραμματισμού και γι 'αυτό θα το περάσουμε από γραμμή σε γραμμή.

#include <stdio.h>

Αυτό το header file περιέχει δηλώσεις που χρησιμοποιούνται στις περισσότερες εισόδους και εξόδους και συνήθως περιλαμβάνεται σε όλα τα προγράμματα C.

#include <sys/types.h>

Αυτό το header file περιέχει ορισμούς για έναν αριθμό από τύπους δεδομένων που χρησιμοποιούνται σε κλήσεις του συστήματος. Αυτοί οι τύποι χρησιμοποιούνται στα επόμενα δύο include files.

#include <sys/socket.h>

Το αρχείο header socket.h περιλαμβάνει ορισμένους ορισμούς των δομών που απαιτούνται για τα sockets.

#include <netinet/in.h>

Το αρχείο header in.h περιέχει σταθερές και δομές (structures) που απαιτούνται για διευθύνσεις του domain του Ιντερνέτ

void error(char \*msg)

{

perror(msg);

exit(1);

}

Αυτή η λειτουργία χρησιμοποιείται όταν αποτύχει μια κλήση συστήματος. Εμφανίζει ένα μήνυμα σχετικά με το σφάλμα στο stderr και στη συνέχεια ακυρώνει το πρόγραμμα.

int main(int argc, char \*argv[])

{

int sockfd, newsockfd, portno, clilen, n;

Sockfd και newsockfd είναι περιγραφείς αρχείων, δηλ. δείκτες συστοιχίας μέσα στον πίνακα περιγραφής αρχείων. Αυτές οι δύο μεταβλητές αποθηκεύουν τις τιμές που επιστρέφονται από την κλήση συστήματος socket και την κλήση συστήματος αποδοχής.

Το portno αποθηκεύει τον αριθμό θύρας στον οποίο ο Server δέχεται συνδέσεις.

Το clilen αποθηκεύει το μέγεθος της διεύθυνσης του client. Αυτό είναι απαραίτητο για την αποδοχή κλήσης συστήματος.

N είναι η τιμή επιστροφής για τις κλήσεις read () και write () Δηλ. Περιέχει τον αριθμό των χαρακτήρων που διαβάζονται ή γράφονται.

char buffer[256];

Ο server διαβάζει χαρακτήρες από τo socket της σύνδεσης μέσα σε αυτό το buffer.

struct sockaddr\_in serv\_addr, cli\_addr;

Ένα sockaddr\_in είναι μια δομή που περιέχει μια διεύθυνση διαδικτύου. Αυτή η δομή ορίζεται στο netinet / in.h.

Εδώ είναι ο ορισμός:

struct sockaddr\_in

{

short sin\_family; /\* must be AF\_INET \*/

u\_short sin\_port;

struct in\_addr sin\_addr;

char sin\_zero[8]; /\* Not used, must be zero \*/

};

Μια δομή in\_addr, που ορίζεται στο ίδιο αρχείο header, περιέχει μόνο ένα πεδίο, ένα μη υπογεγραμμένο μεγάλο όνομα που ονομάζεται s\_addr.

Η μεταβλητή serv\_addr θα περιέχει τη διεύθυνση του server και το αρχείο cli\_addr θα περιέχει τη διεύθυνση του client που συνδέεται με το server.

if (argc < 2)

{

fprintf(stderr,"ERROR, no port provided

");

exit(1);

}

Ο χρήστης πρέπει να περάσει στον αριθμό θύρας στον οποίο ο server δέχεται συνδέσεις ως όρισμα. Αυτός ο κώδικας εμφανίζει ένα μήνυμα σφάλματος αν ο χρήστης δεν το κάνει αυτό.

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd < 0)

error("ERROR opening socket");

Η κλήση του socket () δημιουργεί ένα νέο socket. Απαιτούνται τρία επιχειρήματα-βήματα. Το πρώτο είναι η διεύθυνση domain του socket.

Υπενθυμίζουμε στο σημείο αυτό ότι υπάρχουν δύο πιθανές περιοχές διευθύνσεων, το domain unix για τις δύο διαδικασίες που μοιράζονται ένα κοινό σύστημα αρχείων και το domain Internet για οποιονδήποτε δύο κεντρικούς υπολογιστές στο Διαδίκτυο. Το σταθερό σύμβολο AF\_UNIX χρησιμοποιείται για το πρώτο και το AF\_INET για το τελευταίο (υπάρχουν πολλές άλλες επιλογές που μπορούν να χρησιμοποιηθούν εδώ για εξειδικευμένους σκοπούς).

Το δεύτερο επιχείρημα-βήμα είναι ο τύπος του socket. Υπενθυμίζουμε ότι υπάρχουν δύο επιλογές εδώ, μια υποδοχή ρεύματος στην οποία οι χαρακτήρες διαβάζονται σε συνεχή ροή σαν να προέρχεται από ένα αρχείο ή σε σωλήνα και μια socket datagram, στην οποία τα μηνύματα διαβάζονται όχι ολόκληρα αλλά σε κομμάτια. Οι δύο συμβολικές σταθερές είναι SOCK\_STREAM και SOCK\_DGRAM.

Το τρίτο επιχείρημα είναι το πρωτόκολλο. Αν το όρισμα αυτό είναι μηδέν (και θα πρέπει πάντα να είναι εκτός από ασυνήθιστες περιστάσεις), το λειτουργικό σύστημα θα επιλέξει το καταλληλότερο πρωτόκολλο. Θα επιλέξει TCP για sockets ροής και UDP για sockets datagram.

Η κλήση του συστήματος socket επιστρέφει μια καταχώρηση στον πίνακα περιγραφής του αρχείου (δηλ. ένας μικρός ακέραιος αριθμός). Αυτή η τιμή χρησιμοποιείται για όλες τις επόμενες αναφορές σε αυτή το socket. Εάν η κλήση υποδοχής αποτύχει, επιστρέφει -1. Στην περίπτωση αυτή, το πρόγραμμα εμφανίζει και μήνυμα σφάλματος. Ωστόσο, αυτή η κλήση συστήματος είναι σχεδόν απίθανο να αποτύχει.

Αυτή είναι μια απλοποιημένη περιγραφή της κλήσης socket. Υπάρχουν και άλλες πολλές επιλογές για domains και τύπους, αλλά αυτές είναι οι πιο συνηθισμένες.

bzero((char \*) &serv\_addr, sizeof(serv\_addr));

Η συνάρτηση bzero () θέτει όλες τις τιμές σε ένα buffer στο μηδέν. Χρειάζονται δύο επιχειρήματα-βήματα, το πρώτο είναι ένας δείκτης στο buffer και το δεύτερο είναι το μέγεθος του buffer. Έτσι, αυτή η γραμμή αρχικοποιεί serv\_addr σε μηδενικά

portno = atoi(argv[1]);

Ο αριθμός θύρας στον οποίο ο server θα «ακούσει» τις συνδέσεις μεταβιβάζεται ως όρισμα και αυτή η εντολή χρησιμοποιεί τη λειτουργία atoi () για να το μετατρέψει από μια σειρά ψηφίων, σε έναν ακέραιο αριθμό.

serv\_addr.sin\_family = AF\_INET;

Η μεταβλητή serv\_addr είναι μια δομή τύπου struct sockaddr\_in. Αυτή η δομή έχει τέσσερα πεδία. Το πρώτο πεδίο είναι σύντομο sin\_family, το οποίο περιέχει έναν κωδικό για την οικογένεια διευθύνσεων. Θα πρέπει πάντα να οριστεί στη συμβολική σταθερά AF\_INET.

serv\_addr.sin\_port = htons(portno);

Το δεύτερο πεδίο του serv\_addr είναι το unsigned short sin\_port, το οποίο περιέχει τον αριθμό θύρας. Ωστόσο, αντί να απλώς αντιγράψει τον αριθμό θύρας σε αυτό το πεδίο, είναι απαραίτητο να το μετατρέψει σε εντολή byte δικτύου χρησιμοποιώντας τη συνάρτηση htons (), η οποία μετατρέπει έναν αριθμό θύρας σε παραγγελία byte του κεντρικού υπολογιστή σε έναν αριθμό θύρας σε σειρά byte δικτύου.

serv\_addr.sin\_addr.s\_addr = INADDR\_ANY;

Το τρίτο πεδίο του sockaddr\_in είναι μια δομή του τύπου struct in\_addr που περιέχει μόνο ένα πεδίο unsigned long s\_addr. Αυτό το πεδίο περιέχει τη διεύθυνση IP του κεντρικού υπολογιστή. Για τον κωδικό server, αυτή θα είναι πάντα η διεύθυνση IP του μηχανήματος στο οποίο εκτελείται ο server και υπάρχει μια συμβολική σταθερά INADDR\_ANY η οποία λαμβάνει αυτή τη διεύθυνση.

if (bind(sockfd, (struct sockaddr \*) &serv\_addr, sizeof(serv\_addr)) < 0)

error("ERROR on binding");

Η κλήση συστήματος bind () δεσμεύει ένα socket σε μια διεύθυνση, στην περίπτωση αυτή η διεύθυνση του τρέχοντος server και του αριθμού θύρας στον οποίο θα τρέξει ο server. Παίρνει τρία επιχειρήματα, τον περιγραφέα αρχείου socket, τη διεύθυνση στην οποία δεσμεύεται και το μέγεθος της διεύθυνσης με την οποία δεσμεύεται. Το δεύτερο επιχείρημα είναι ένας δείκτης σε μια δομή τύπου sockaddr, αλλά αυτό που περνάει είναι μια δομή τύπου sockaddr\_in και έτσι πρέπει να μεταφερθεί στον σωστό τύπο. Αυτό είναι πιθανό να αποτύχει για διάφορους λόγους, το πιο προφανές είναι ότι αυτή το socket να χρησιμοποιείται ήδη σε αυτό το μηχάνημα.

listen(sockfd,5);

Η κλήση συστήματος ακρόασης επιτρέπει στη διαδικασία να «ακούει» στην υποδοχή για τις συνδέσεις. Το πρώτο επιχείρημα είναι ο περιγραφέας αρχείου υποδοχής και το δεύτερο είναι το μέγεθος της ουράς αναμονής, δηλαδή ο αριθμός των συνδέσεων που μπορούν να περιμένουν ενώ η διαδικασία χειρίζεται μια συγκεκριμένη σύνδεση. Αυτό πρέπει να οριστεί σε 5, το μέγιστο μέγεθος που επιτρέπεται από τα περισσότερα συστήματα. Εάν το πρώτο επιχείρημα είναι έγκυρο socket, αυτή η κλήση δεν μπορεί να αποτύχει, οπότε ο κώδικας δεν ελέγχει για σφάλματα.

clilen = sizeof(cli\_addr);

newsockfd = accept(sockfd, (struct sockaddr \*) &cli\_addr, &clilen);

if (newsockfd < 0)

error("ERROR on accept");

Η κλήση συστήματος accept () προκαλεί τη διαδικασία μπλοκαρίσματος έως ότου ένας υπολογιστής- client συνδεθεί με το server. Έτσι, ξυπνάει τη διαδικασία όταν έχει συνδεθεί επιτυχώς μια σύνδεση από έναν client. Επιστρέφει ένα νέο περιγραφέα αρχείου και όλη η επικοινωνία σε αυτή τη σύνδεση θα πρέπει να γίνει χρησιμοποιώντας το νέο περιγραφέα αρχείου. Το δεύτερο επιχείρημα είναι ένας δείκτης αναφοράς στη διεύθυνση του πελάτη στο άλλο άκρο της σύνδεσης και το τρίτο επιχείρημα είναι το μέγεθος αυτής της δομής

bzero(buffer,256);

n = read(newsockfd,buffer,255);

if (n < 0) error("ERROR reading from socket");

printf("Here is the message: %s

",buffer);

Να σημειωθεί ότι θα φτάνει σε αυτό το σημείο μόνο αφού ο client έχει συνδεθεί επιτυχώς στον server μας. Αυτός ο κώδικας προετοιμάζει το buffer χρησιμοποιώντας τη λειτουργία bzero () και στη συνέχεια διαβάζει από την socket. Να σημειωθεί ότι η κλήση ανάγνωσης χρησιμοποιεί τον νέο περιγραφέα αρχείου, αυτόν που επιστρέφεται από το accept (), όχι ο αρχικός περιγραφέας αρχείου που επιστρέφεται από την υποδοχή ().Επίσης ότι το read () θα μπλοκάρει μέχρι να υπάρχει κάτι που να διαβάζει στην υποδοχή, δηλαδή αφού ο πελάτης έχει εκτελέσει ένα write ().

Θα διαβάστει είτε ο συνολικός αριθμός χαρακτήρων στο socket είτε το 255, όποιο από τα δύο είναι μικρότερο, και να επιστρέψει τον αριθμό των χαρακτήρων που διαβάζονται.

n = write(newsockfd,"I got your message",18);

if (n < 0) error("ERROR writing to socket");

Μόλις δημιουργηθεί μια σύνδεση, και οι δύο άκρες μπορούν να διαβάζουν και να γράφουν στη σύνδεση. Φυσικά, οτιδήποτε γράφεται από τον client θα διαβαστεί από το server και ότι γράφει ο server θα διαβαστεί από τον client. Αυτός ο κώδικας απλά γράφει ένα σύντομο μήνυμα στον client. Το τελευταίο επιχείρημα της εγγραφής είναι το μέγεθος του μηνύματος.

return 0;

}

Έτσι τερματίζει το πρόγραμμα. Δεδομένου ότι το κύριως κηρύχθηκε ότι είναι τύπου int όπως καθορίζεται από το πρότυπο ascii, ορισμένοι προγραμματιστές διαμαρτύρονται αν δεν επιστρέφουν τίποτα.

**Κωδικός Client**

Όπως και πριν, θα περάσουμε από το πρόγραμμα client.c ,γραμμή με γραμμή.

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

Τα αρχεία κεφαλίδων είναι τα ίδια με αυτά του server με μία προσθήκη. Το αρχείο netdb.h ορίζει τον φορέα hostent, ο οποίος θα χρησιμοποιηθεί παρακάτω.

void error(char \*msg)

{

perror(msg);

exit(0);

}

int main(int argc, char \*argv[])

{

int sockfd, portno, n;

struct sockaddr\_in serv\_addr;

struct hostent \*server;

Η συνάρτηση σφάλματος () είναι ίδια με αυτή του server, όπως και οι μεταβλητές sockfd, portno και n. Η μεταβλητή serv\_addr θα περιέχει τη διεύθυνση του server με τον οποίο θέλουμε να συνδεθούμε. Είναι τύπου struct sockaddr\_in.

Όλος αυτός ο κώδικας είναι ο ίδιος με αυτόν στον server.

server = gethostbyname(argv[1]);

if (server == NULL)

{

fprintf(stderr,"ERROR, no such host

");

exit(0);

}

Η μεταβλητή argv [1] περιέχει το όνομα ενός κεντρικού υπολογιστή στο Internet, π.χ. Cs.rpi.edu. Η λειτουργία (function):

struct hostent \*gethostbyname(char \*name)

Λαμβάνει ένα τέτοιο όνομα ως επιχείρημα και επιστρέφει έναν δείκτη σε ένα hostent που περιέχει πληροφορίες σχετικά με τον συγκεκριμένο κεντρικό υπολογιστή.

Το πεδίο char \* h\_addr περιέχει τη διεύθυνση IP.

Εάν αυτή η δομή είναι NULL, το σύστημα δεν θα μπορούσε να εντοπίσει έναν κεντρικό υπολογιστή με αυτό το όνομα.

Παλαιότερα, αυτή η λειτουργία λειτούργησε με την αναζήτηση ενός αρχείου συστήματος που ονομάζεται / etc / hosts αλλά με την εκρηκτική ανάπτυξη του Internet, έγινε αδύνατο για τους διαχειριστές συστημάτων να διατηρήσουν αυτό το αρχείο τρέχοντα. Έτσι, ο μηχανισμός με τον οποίο λειτουργεί αυτή η λειτουργία είναι πολύπλοκος, συχνά περιλαμβάνει την αναζήτηση μεγάλων βάσεων δεδομένων σε ολόκληρη τη χώρα.

bzero((char \*) &serv\_addr, sizeof(serv\_addr));

serv\_addr.sin\_family = AF\_INET;

bcopy((char \*)server->h\_addr,

(char \*)&serv\_addr.sin\_addr.s\_addr,

server->h\_length);

serv\_addr.sin\_port = htons(portno);

Αυτός ο κώδικας ορίζει τα πεδία στον server serv\_addr. Πολλά από αυτά είναι τα ίδια με αυτά του server. Ωστόσο, επειδή ο server πεδίου-> h\_addr είναι μια συμβολοσειρά χαρακτήρων, χρησιμοποιούμε τη συνάρτηση:

void bcopy(char \*s1, char \*s2, int length)

Η οποία αντιγράφει τα bytes από s1 έως s2.

if (connect(sockfd,&serv\_addr,sizeof(serv\_addr)) < 0)

error("ERROR connecting");

Η συνάρτηση σύνδεσης καλείται από τον πελάτη για να δημιουργήσει μια σύνδεση με το server. Παίρνει τρία επιχειρήματα, τα αρχεια του socket , τη διεύθυνση του κεντρικού υπολογιστή με τον οποίο επιθυμεί να συνδεθεί (συμπεριλαμβανομένου του αριθμού θύρας) και το μέγεθος αυτής της διεύθυνσης. Αυτή η συνάρτηση επιστρέφει το 0 στην επιτυχία και το -1 αν αποτύχει.

Παρατηρούμε ότι ο client πρέπει να γνωρίζει τον αριθμό θύρας του server, αλλά δεν χρειάζεται να γνωρίζει τον δικό του αριθμό θύρας. Αυτό συνήθως εκχωρείται από το σύστημα όταν καλείται σύνδεση.

printf("Please enter the message: ");

bzero(buffer,256);

fgets(buffer,255,stdin);

n = write(sockfd,buffer,strlen(buffer));

if (n < 0)

error("ERROR writing to socket");

bzero(buffer,256);

n = read(sockfd,buffer,255);

if (n < 0)

error("ERROR reading from socket");

printf("%s

",buffer);

return 0;

}