ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
http://www.cslab.ece.ntua.gr

Εργαστήριο Λειτουργικών Συστημάτων

7ο εξάμηνο, Ακαδημαϊκή περίοδος 2020–2021

Οδηγός προγραμματισμού BSD sockets σε περιβάλλον Linux

Εργαστήριο Υπολογιστικών Συστημάτων Ε.Μ.Π.
os-lab@lists.cslab.ece.ntua.gr

Δεκέμβριος 2020

Περιεχόμενα

Επιμέλεια: Ευ. Αγγέλου, Ευ. Κούκης

1 Εισαγωγή – Τρόποι διαδιεργασιακής επικοινωνίας και sockets

Όπως είδαμε στο μάθημα των Λειτουργικών Συστημάτων, υπάρχουν διάφοροι τρόποι διαδιεργασιακής επικοινωνίας. Ανάμεσά τους:

Μοιραζόμενη μνήμη - Shared memory
: Δύο οι περισσότερες διεργασίες επικοινωνούν, γράφοντας και διαβάζοντας σε κοινή περιοχή μνήμης.

Σωληνώσεις - Pipes
: Μονόδρομη (unidirectional) επικοινωνία ανάμεσα σε συγγενικές (με κοινό πρόγονο) διεργασίες.

Αυτοί οι μηχανισμοί επιτρέπουν την επικοινωνία ανάμεσα σε διεργασίες που εκτελούνται στο ίδιο μηχάνημα, κάτω από το ίδιο ΛΣ. Είναι εφικτό να επικοινωνήσουν διεργασίες που εκτελούνται σε διαφορετικά μηχανήματα; Για το σκοπό αυτό χρειαζόμαστε ένα *δικτυακό* πρωτόκολλο επικοινωνίας, όπως το TCP/IP, και μια προγραμματιστική διεπαφή προς τις διεργασίες για την αλληλεπίδραση με αυτό.

Τα *sockets* αποτελούν έναν τέτοιο μηχανισμό. Αναπτύχθηκαν αρχικά ως το BSD socket API, μέρος του 4.2BSD Unix ("Berkeley Sockets"), και πλέον αποτελούν το de facto API για τη χρήση δικτυακών πρωτοκόλλων τόσο σε συστήματα Unix/Linux, όσο και σε συστήματα Windows (υλοποίηση Winsock).

Παρόλο που στο παρόν κείμενο επικεντρωνόμαστε στη χρήση των sockets για επικοινωνία με χρήση των πρωτοκόλλων TCP/IP και UDP/IP, τα sockets είναι μια *γενική* διεπαφή και υποστηρίζουν πλήθος διαφορετικών πρωτοκόλλων. Ενδεικτική λίστα μπορείτε να βρείτε στο αρχείο `/usr/include/bits/socket.h` σε συστήματα Linux.

2 Μοντέλο Client-Server – Το πρωτόκολλο TCP/IP

Έστω ένας περιηγητής του παγκόσμιου ιστού (Web Browser, πρόγραμμα *πελάτης*) ο οποίος ζητά μια ιστοσελίδα από έναν εξυπηρετητή (Web Server). Ο εξυπηρετητής ανακτά το περιεχόμενο της σελίδας (συνήθως HTML) από το δίσκο, και το επιστρέφει στον περιηγητή.

Η επικοινωνία μεταξύ του browser και του server γίνεται εγκαθιστώντας μια σύνδεση του πρωτοκόλλου TCP, πάνω από το πρωτόκολλο IP (TCP/IP). Ο πελάτης

κι ο εξυπηρετητής χρησιμοποιούν τη σύνδεση μέσω *sockets*, ένα για κάθε άκρο της σύνδεσης.

Το socket είναι το σημείο ("endpoint") στο οποίο οι διεργασίες εκτελούν κλήσεις συστήματος (όπως οι `read()` και `write()` για να επικοινωνήσουν μέσω της σύνδεσης TCP/IP.

Η περιγραφή της λειτουργίας του TCP/IP ξεφεύγει από τους σκοπούς του παρόντος οδηγού. Εξαιρετική περιγραφή των πρωτοκόλλων IP μπορείτε να βρείτε στην προτεινόμενη βιβλιογραφία.

Πολύ συνοπτικά:

- Το IP επιτρέπει την αποστολή πακέτων πληροφορίας από έναν υπολογιστή σε έναν άλλο. Οι κόμβοι προέλευσης και προορισμού αναγνωρίζονται από έναν αριθμό, που ονομάζεται διεύθυνση IP, και είναι των 32 ή των 128 bits, για τα πρωτόκολλα IPv4 και IPv6 αντίστοιχα. Το IP δεν εγγυάται την σωστή παράδοση των πακέτων.
- Το πρωτόκολλο TCP χρησιμοποιεί τις υπηρεσίες του IP για να σχηματίσει *αξιόπιστες συνδέσεις*: Κάθε σύνδεση είναι ένα αμφίδρομο *ρεύμα* δεδομένων, με εγγυημένη παράδοση των bytes, με τη σειρά που στάλθηκαν. Το TCP *δεν* διατηρεί όρια μηνυμάτων: Ο παραλήπτης των bytes δεν μπορεί να διακρίνει τις ομάδες των bytes όπως παραδόθηκαν από τον αποστολέα στο TCP.
- Για τη διάκριση χωριστών ρευμάτων δεδομένων ανάμεσα σε δύο κόμβους, κάθε σύνδεση TCP ορίζεται από την τετράδα (`source IP`, `source port`, `dest IP`, `dest port`). Αν ένα από τα τέσσερα μέλη διαφέρει, πρόκειται για άλλη σύνδεση.
- το TCP αριθμεί τα πακέτα IP και φροντίζει να ανακάμπτει από το ενδεχόμενο να χαθεί ένα πακέτο, να ληφθεί διπλό, ή να ληφθεί εκτός σειράς.
- Σε προγραμματιστικό επίπεδο, κάθε σύνδεση είναι δύο *sockets*, ένα σε κάθε άκρο. Στο UNIX, το socket είναι απλώς άλλος ένας περιγραφητής αρχείου, file descriptor, ο οποίος μπορεί να χρησιμοποιηθεί για `read()`, ή `write()`.

Πολύ γνωστές υπηρεσίες του διαδικτύου βασίζονται σε συνδέσεις TCP και υλοποιούνται με sockets, όπως telnet, SSH, FTP και το Web (HTTP).

Για να δείτε το πρωτόκολλο TCP σε λειτουργία, μπορείτε να ανακτήσετε μια σελίδα του παγκόσμιου ιστού από τον server με τη χρήση του εργαλείου telnet, το οποίο είναι ένας εύκολος τρόπος να χρησιμοποιήσετε ένα socket για πραγματοποίηση σύνδεσης TCP. Αρχικά, ανοίξτε μια σύνδεση προς τον Web server, εκτελώντας `telnet www.cslab.ece.ntua.gr 80`. Η "μαγική" σταθερά `80` ορίζει τη θύρα προορισμού και υποδηλώνει ότι θέλουμε να συνδεθούμε στον WWW

server που τρέχει στον υπολογιστή `www.cslab.ece.ntua.gr` και όχι σε κάποια άλλη διεργασία. Έχοντας συνδεθεί, πληκτρολογήστε `GET /`. Τα bytes αποστέλλονται από το `telnet` μέσω του TCP socket στον WWW server, ο οποίος απαντάει στέλνοντάς μας τον κώδικα HTML της αρχικής σελίδας και έπειτα κλείνει τη σύνδεση:

```
~$ telnet www.cslab.ece.ntua.gr 80
Trying 147.102.3.1...
Connected to www.cslab.ece.ntua.gr.
Escape character is '^]'.
GET /
<html>
<head>
  <title>Computing Systems Laboratory</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-7">
  <!--css-->
  <link rel="stylesheet" TYPE="text/css" href="/css/cslab-main.css">
</head>
...
</html>
Connection closed by foreign host.
~$
```

3 Βασικές έννοιες των sockets

Όταν δημιουργούμε ένα socket, πρέπει να ορίσουμε τρεις παραμέτρους: τον χώρο ονομάτων (namespace, ή "domain"), τον τύπο της επικοινωνίας ("type") και το πρωτόκολλο επικοινωνίας το οποίο πρέπει να συμβαδίζει με τον επιλεγμένο συνδυασμό χώρου ονομάτων και τύπου επικοινωνίας.

Namespace (χώρος ονομάτων) ενός socket ορίζει το πεδίο της επικοινωνίας στο οποίο αναφέρεται το socket, το είδος των πρωτοκόλλων ("protocol family") με τα οποία πρόκειται να χρησιμοποιηθεί. Οι γνωστότεροι χώροι είναι είναι το Internet namespace (IPv4, (`PF_INET`, ή IPv6, `PF_INET6`) και το τοπικό namespace (local ή Unix domain, `PF_LOCAL` ή `PF_UNIX`). Για περισσότερες λεπτομέρειες δείτε τα manpages `ip(7)`, `unix(7)`.

Το namespace καθορίζει μοναδικά τη μορφή της διεύθυνσης που μπορεί να πάρει ένα socket. Σε κάθε protocol family (σταθερές `PF_*`) αντιστοιχεί συγκεκριμένο address family (σταθερές `AF_*`).

Διευθύνσεις socket στο τοπικό namespace είναι απλά ονόματα αρχείων (π.χ. `/var/run/mysqld/mysqld.sock` για τη γνωστή βάση δεδομένων MySQL). Στο IPv4 namespace η διεύθυνση socket περιλαμβάνει τη διεύθυνση IP και τον αριθμό πόρτας (port number).

Στυλ/τύπος επικοινωνίας ελέγχει πως αντιμετωπίζει το socket τα δεδομένα που ανταλλάσσονται. Το στυλ επικοινωνίας καθορίζει τη *σημασιολογία* για τον χρήστη του socket, απαντώντας ερωτήματα όπως: α) το socket μεταφέρει μηνύματα συγκεκριμένου μήκους, ή ένα ρεύμα ανεξάρτητων bytes; β) υπάρχει περίπτωση να χαθεί ένα μήνυμα; γ) με πόσους μπορεί κανείς να επικοινωνήσει μέσα από αυτό το socket; Θα ασχοληθούμε με δύο κύριους τύπους επικοινωνίας, τον τύπο `SOCK_STREAM`, ο οποίος παρέχει *σύνδεση* για σειριακή, αξιόπιστη, αμφίδρομη μεταφορά δεδομένων (connection-oriented) σε ρεύμα από bytes, και τον τύπο `SOCK_DGRAM`, ο οποίος προσφέρει αναξιόπιστη, μεταφορά μηνυμάτων μεταβλητού μήκους, χωρίς σύνδεση.

Πρωτόκολλο Η παράμετρος αυτή καθορίζει το πρωτόκολλο προς χρήση, για δεδομένο συνδυασμό χώρου ονομάτων και τύπου επικοινωνίας. Για παράδειγμα, ο συνδυασμός Internet namespace και `SOCK_STREAM` εξυπηρετείται από το πρωτόκολλο TCP/IP, ενώ ο συνδυασμός Internet namespace και `SOCK_DGRAM` από το UDP/IP. Ο χρήστης μπορεί να επιλέξει πρωτόκολλο αν υπάρχουν πολλά διαθέσιμα, ή να προσδιορίσει το πρωτόκολλο ως `0`, ώστε να χρησιμοποιηθεί μια προκαθορισμένη, ταιριαστή επιλογή. Για τη λίστα των διαθέσιμων πρωτοκόλλων IP δείτε το `protocols(5)` και το `/etc/protocols`.

Για περισσότερες πληροφορίες σχετικά με τα sockets, δείτε το `socket(2)` και το εγχειρίδιο της GNU C Library, στο https://www.gnu.org/software/libc/manual/html_mono/libc.html#toc_Sockets.

Δεν υποστηρίζεται κάθε συνδυασμός χώρου ονομάτων και στυλ επικοινωνίας. Για παράδειγμα, δεν είναι ακόμη αρκετά διαδεδομένη η υποστήριξη για το στυλ `SOCK_SEQPACKET` για χώρο IPv4 (`AF_INET`), η οποία βασίζεται στο πρωτόκολλο SCTP και δεν υποστηρίζεται ευρέως.

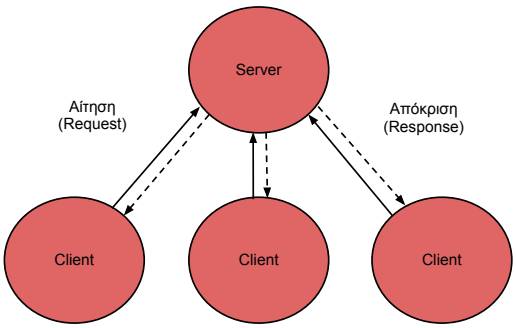
4 Βασικές κλήσεις του sockets API

Κάθε socket στο UNIX εμφανίζεται ως ένα ακόμη ανοιχτό αρχείο, προσβάσιμο από τη διεργασία μέσω ενός file descriptor. Το sockets API ορίζει ένα σύνολο κλήσεων, *ανεξάρτητων* από το είδος του πρωτοκόλλου που χρησιμοποιεί το συγκεκριμένο socket. Επειδή το sockets API ορίζεται σε C, αυτό σημαίνει ότι κάθε κλήση συστήματος δέχεται ένα γενικό όρισμα `struct sockaddr *`, το οποίο είναι δείκτης στον γενικό τύπο "διεύθυνση ενός socket". Ανάλογα με το είδος του socket, το όρισμα `struct sockaddr *` μπορεί να δείχνει σε `struct sockaddr_in` (IPv4), `struct sockaddr_in6` (IPv6), κλπ. Ομοίως, κάθε κλήση που δέχεται ως όρισμα ένα `struct sockaddr *` δέχεται κι όρισμα τύπου `socklen_t` που περιέχει το *μήκος* σε bytes της αντίστοιχης δομής. Δείτε τα `ip(7)`, `ipv6(7)` για τον ορισμό των συγκεκριμένων δομών.

Συνοπτικά τα σημαντικότερα system calls για τη χρήση των sockets είναι:

- `int socket(int domain, int type, int protocol);`
  Δημιουργεί ένα νέο socket.
- `int close(int fd);`
  Κλείνει ένα socket.
- `int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);`
  Συνδέει ένα socket με τη διεύθυνση που προσδιορίζεται από το όρισμα `addr`. Στην περίπτωση του TCP/IP, εγκαθιστά μια νέα εξερχόμενη σύνδεση από την πλευρά του client.
- `int bind(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen);`
  Αναθέτει ένα *όνομα*, δηλ. διεύθυνση, στο socket. Είναι απαραίτητο στην πλευρά του server για sockets τύπου `SOCK_STREAM`, π.χ. TCP/IP, ώστε να καθοριστεί η πόρτα στην οποία θα λαμβάνονται συνδέσεις.
- `int listen(int sockfd, int backlog);`
  Ετοιμάζει το socket να δεχτεί συνδέσεις, στην πλευρά του server.
- `int accept(int sockfd, struct sockaddr \*addr, socklen\_t \*addrlen);`
  Δέχεται μια νέα εισερχόμενη σύνδεση. Επιστρέφει ένα *νέο*, συνδεδεμένο socket το οποίο εφεξής μπορεί να χρησιμοποιηθεί για επικοινωνία με τον συγκεκριμένο πελάτη. Το αρχικό socket *δεν* επηρεάζεται, και μπορεί εκ νέου να χρησιμοποιηθεί με την `accept()`.

4.1 TCP sockets: η πλευρά του client



Σχήμα 1: Αρχιτεκτονική Client - Server.

Ας δούμε τον κύκλο ζωής ενός socket στην πλευρά του client.

Όταν δημιουργείτε ένα socket, καλώντας τη `socket()`, πρέπει να ορίσετε τις 3 παραπάνω επιλογές: τον χώρο ονομάτων, το στυλ επικοινωνίας και το πρωτόκολλο. Για την παράμετρο namespace χρησιμοποιούνται οι σταθερές που αρχίζουν με `PF_` (συντομογραφία για το "οικογένειες πρωτοκόλλων" - protocol families). Για παράδειγμα, `PF_LOCAL` ή `PF_UNIX` ορίζει τον τοπικό χώρο ονομάτων και `PF_INET` το IPv4. Για τις παραμέτρους του στυλ επικοινωνίας, χρησιμοποιούμε τις σταθερές που αρχίζουν με `SOCK_`, π.χ. `SOCK_STREAM` για connection-oriented sockets και `SOCK_DGRAM` για datagram sockets. Η τρίτη παράμετρος, το πρωτόκολλο, ορίζει τον μηχανισμό χαμηλού επιπέδου για τη μετάδοση και λήψη δεδομένων. Κάθε πρωτόκολλο είναι έγκυρο για ένα συγκεκριμένο συνδυασμό namespace-στυλ. Επειδή τις περισσότερες φορές υπάρχει κάποιο καθαρά καλύτερο πρωτόκολλο για κάθε τέτοιο συνδυασμό, ορίζοντας 0 χρησιμοποιούμε την προκαθορισμένη επιλογή.

Αν η κλήση της `socket()` επιτύχει, επιστρέφει έναν file descriptor για το socket. Μπορείτε να διαβάσετε ή να γράψετε στο socket με τη χρήση της `read()`, της `write()`, και άλλων κλήσεων συστήματος που δρουν πάνω σε file descriptors. Όταν τελειώσετε με τη χρήση του socket, μπορείτε να το κλείσετε με μια κλήση στην `close()`.

Για τη δημιουργία μιας σύνδεσης για sockets τύπου `SOCK_STREAM`, ο client καλεί την `connect()` ορίζοντας τη διεύθυνση ενός server socket. Ένας client ξεκινάει τη σύνδεση, ενώ ένας server περιμένει να δεχτεί (accept) νέες συνδέσεις. Ο client καλεί την `connect()` για να ξεκινήσει τη σύνδεση από ένα τοπικό socket στη διεύθυνση προορισμού που ορίζεται στο όρισμα `addr`. Η `connect()` έχει ενιαία διεπαφή, ανεξάρτητη από τον τύπο το μήκος των διευθύνσεων που χρησιμοποιεί

το υφιστάμενο δικτυακό πρωτόκολλο, οπότε η τρίτη παράμετρός της είναι το μέγεθος σε bytes της δομής που δείχνει το όρισμα `addr`, και διαφέρει ανάλογα με το namespace του socket.

> *Ανάγνωση και Εγγραφή σε sockets*
>
> *Κάθε τεχνική για εγγραφή/ανάγνωση σε ένα file descriptor μπορεί να χρησιμοποιηθεί για την εγγραφή/ανάγνωση δεδομένων σε ένα socket. Οι κλήσεις συστήματος `send()` και `recv()` είναι ειδικές για socket file descriptors. Αν και δεν θα τις χρησιμοποιήσουμε στα παρακάτω παραδείγματα, μπορείτε να δείτε τα manpages στο σύστημα σας για τη χρήση τους.*
>
> *Πολλές φορές προκύπτει η ανάγκη* ταυτόχρονου *χειρισμού πολλών διαφορετικών sockets, π.χ. αναμονή μηνύματος από δύο file descriptors. Οι κλήσεις `select(2)` και `poll(2)` μπορούν να φανούν πολύ χρήσιμες σε αυτή την περίπτωση.*

4.2 TCP sockets: Η πλευρά του server

Ας δούμε τον κύκλο ζωής ενός socket στην πλευρά του server. Το παρόν παράδειγμα αναφέρεται σε socket της οικογένειας `AF_INET` τύπου `SOCK_STREAM`, δηλαδή TCP πάνω από IPv4.

Ο κύκλος ζωής (lifecycle) ενός server περιλαμβάνει τη δημιουργία ενός socket, το "δέσιμο" (`bind()`) του socket σε συγκεκριμένη διεύθυνση, την κλήση `listen()` που ενεργοποιεί τις συνδέσεις στο socket, διαδοχικές κλήσεις στην `accept()` για να δεχτούμε νέες συνδέσεις, και τέλος κλείσιμο του socket. Τα δεδομένα δεν διαβάζονται ή γράφονται με κλήσεις απευθείας στο socket του εξυπηρετητή. Αντίθετα, κάθε φορά που ένα πρόγραμμα δέχεται μια νέα σύνδεση, το socket API προβλέπει τη δημιουργία ενός χωριστού socket, ειδικά για το συγκεκριμένο πελάτη, το οποίο εφεξής χρησιμοποιείται για την ανταλλαγή δεδομένων μέσω της συγκεκριμένης σύνδεσης. Ας δούμε αναλυτικά τα βήματα για την αποδοχή μια σύνδεσης στην πλευρά του εξυπηρετητή.

Το server socket πρέπει να δεθεί σε συγκεκριμένη διεύθυνση, με χρήση της `bind()` ώστε να μπορεί ο client να το βρει. Πρώτη παράμετρος είναι ο περιγραφητής αρχείου του socket. Δεύτερη παράμετρος είναι ένας δείκτης σε ένα δομή που περιέχει τη διεύθυνση του socket, η μορφή της οποίας εξαρτάται από την οικογένεια διευθύνσεων στην οποία ανήκει το socket. Τρίτη παράμετρος είναι το μέγεθος της δομής που περιέχει τη διεύθυνση του socket, σε bytes.

Standard / well-known ports στο IP

Όταν χρησιμοποιούνται IPv4/IPv6 sockets, οι πελάτες πρέπει να γνωρίζουν τον αριθμό της πόρτας στην οποία ακούει ο εξυπηρετητής. Οι πόρτες από 0 έως 1023 χαρακτηρίζονται "well-known ports", και αντιστοιχίζονται από σύμβαση σε συγκεκριμένη υπηρεσία: για παράδειγμα, οι web servers (HTTP) ακούν στην πόρτα `80/tcp`*, ενώ οι mail servers (SMTP) στην πόρτα* `25/tcp`*.*

Σε ΛΣ που ακολουθούν τη φιλοσοφία του UNIX μια διεργασία πρέπει να τρέχει με δικαίωμα `root`*, για να μπορεί να δέσει ένα socket σε πόρτα μικρότερη από 1024. Μπορείτε να βρείτε μια λίστα των καθιερωμένων πορτών στο αρχείο* `/etc/services`*.*

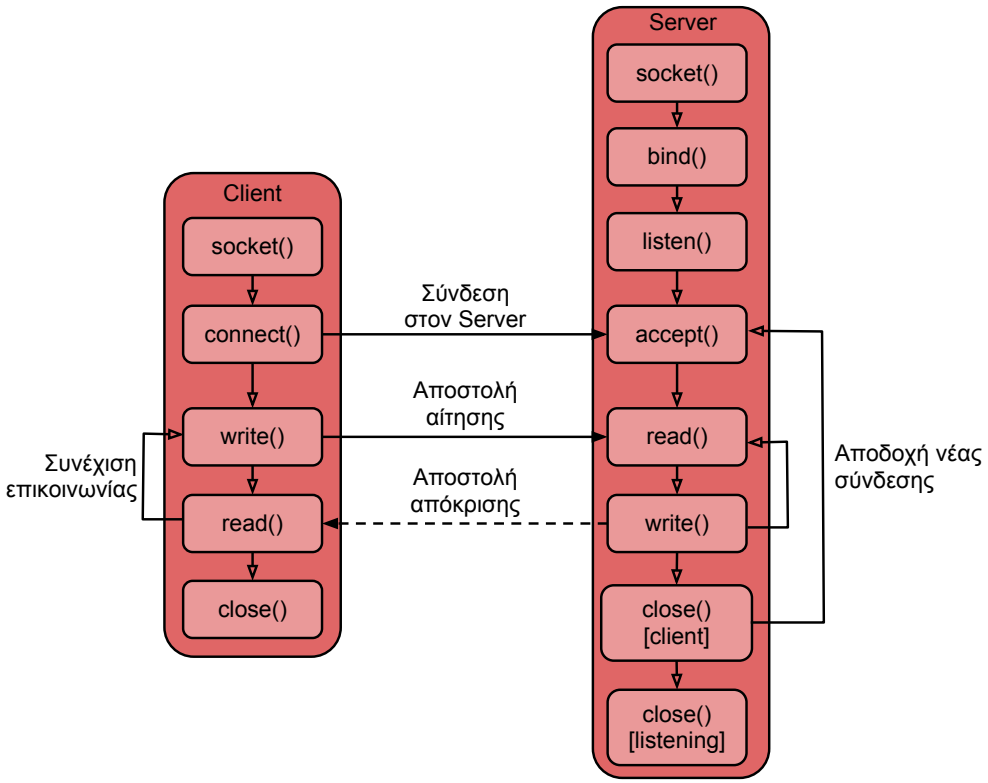
Η δυνατότητα για αποδοχή εισερχόμενων συνδέσεων σε ένα socket ενεργοποιείται με κλήση της `listen()`. Η πρώτη παράμετρος της είναι ο file descriptor του socket. Η δεύτερη ορίζει πόσες συνδέσεις μπορούν να περιμένουν στην ουρά ("backlog"). Αν η ουρά είναι γεμάτη, οι επιπλέον συνδέσεις απορρίπτονται. Αυτό δεν περιορίζει το συνολικό αριθμό των συνδέσεων που μπορεί να χειριστεί ένας server - μόνο ελέγχει τον αριθμό των πελατών που επιτρέπεται να προσπαθούν να συνδεθούν αλλά να μην έχουν γίνει ακόμη δεκτοί.

Ο server καλεί την `accept()` για να κάνει κάνει αποδεκτή μια νέα εισερχόμενη αίτηση. Αν δεν υπάρχει καμία στην ουρά, η `accept()` μπλοκάρει. Η `accept()` επιστρέφει τον περιγραφητή για ένα *νέο* συνδεδεμένο socket, το οποίο χρησιμοποιείται για επικοινωνία με το συγκεκριμένο πελάτη. Η διεύθυνση του πελάτη που μόλις έγινε αποδεκτός επιστρέφεται στη δομή που προσδιορίζεται από το όρισμα `addr`, ώστε ο server να ξέρει το όνομα (διεύθυνση IP και πόρτα TCP) του άλλου άκρου. Το αρχικό socket του server συνεχίζει να μπορεί να δεχτεί νέες εισερχόμενες συνδέσεις από clients.

4.3 Χειρισμός διευθύνσεων IP

Όπως αναφέραμε και προηγούμενα, οι διευθύνσεις των IPv4 sockets αποθηκεύονται σε δομές `struct sockaddr_in`, και αποτελούνται από δύο μέρη: μία διεύθυνση IPv4 των 32 bits, και έναν αριθμό πόρτας των 16 bits. Ομοίως, οι διευθύνσεις IPv6 αποθηκεύονται σε δομές `struct sockaddr_in6`, οι οποίες περιέχουν διευθύνσεις IPv6 των 128 bits.

Η πληροφορία αυτή αποθηκεύεται σε ανάλογα πεδία `sin_family`, `sin_addr`, `sin_port` της δομής. Για λεπτομέρειες σχετικά με τα ονόματα και τους τύπους των πεδίων, δείτε τις σελίδες `ip(7)`, `ipv6(7)`.



Σχήμα 2: Ο κύκλος ζωής των sockets, κατά την επικοινωνία client-server.

Το sockets API έχει πολλές βοηθητικές συναρτήσεις για το χειρισμό των δομών αυτών. Δείτε τουλάχιστον τις σελίδες `byteorder(3)`, `inet_pton(3)`, `inet_ntop(3)`, `gethostbyname(3)`, `getaddrinfo(3)` για τη χρήση τους.

Ονόματα στο διαδίκτυο (DNS)

Επειδή είναι ευκολότερο να θυμόμαστε ονόματα αντί για αριθμούς, η υπηρεσία ονομάτων DNS (Domain Name Service) αντιστοιχίζει ονόματα (όπως το `www.cslab.ece.ntua.gr`) σε διευθύνσεις IP, και το αντίστροφο. Οι συναρτήσεις `gethostbyname(3)`, `getaddrinfo(3)` είναι βοηθητικές συναρτήσεις που μπορούν να χρησιμοποιηθούν για να ανακτήσετε τη διεύθυνση IPv4 ή IPv6 που αντιστοιχεί σε συγκεκριμένο όνομα DNS.

4.4 UNIX domain sockets

Sockets που συνδέουν διεργασίες στον ίδιο υπολογιστή μπορούν να χρησιμοποιήσουν το τοπικό namespace που προσδιορίζεται με τα συνώνυμα `PF_LOCAL` και `PF_UNIX`. Αυτά καλούνται τοπικά sockets ή UNIX-domain sockets. Οι διευθύνσεις τους είναι ονόματα αρχείων στο τοπικό σύστημα αρχείων.

Η χρήση UNIX domain sockets επιτρέπει σε διεργασίες να εγκαταστήσουν ένα αμφίδρομο ρεύμα επικοινωνίας χρησιμοποιώντας το όνομα ενός αρχείου στο σύστημα αρχείων για να εντοπίσουν η μία την άλλη. Αυτό δεν σημαίνει ότι τα δεδομένα περνάνε από το σύστημα αρχείων: Απλώς, ένα UNIX domain socket χρησιμοποιεί το σύστημα αρχείων ως χώρο διευθύνσεών του.

Το όνομα του socket περιγράφεται σε ένα `struct sockaddr_un`, με πεδία `sun_family` (ίσο με `AF_LOCAL`), και `sun_path`, το οποίο ορίζει το όνομα αρχείου που θα χρησιμοποιηθεί.

Οποιοδήποτε όνομα αρχείου μπορεί να χρησιμοποιηθεί, αλλά η διεργασία πρέπει να έχει δικαίωμα εγγραφής στον φάκελο που το περιέχει, ώστε να μπορεί να δημιουργήσει νέα inodes. δημιουργήσει αρχεία στον φάκελο. Για τη σύνδεση σε ένα socket, μια διεργασία πρέπει να έχει δικαίωμα ανάγνωσης στο αντίστοιχο αρχείο.

Επειδή βρίσκεται μέσα σε ένα σύστημα αρχείων, ένα τοπικό socket περιγράφεται ως αρχείο. Για παράδειγμα, προσέξτε το αρχικό `s` στο παρακάτω socket, που παρέχει ο X server (το κοινό γραφικό περιβάλλον σε Linux συστήματα):

```
~$ ls -al /tmp/.X11-unix/X0
srwxrwxrwx 1 root root 0 Απρ   4 19:28 /tmp/.X11-unix/X0
```

Μπορείτε να καλέσετε την `rm` από το shell σας για να αφαιρέσετε ένα τοπικό socket, μιας και η διεύθυνσή του είναι ένα αρχείο.

Τα UNIX-domain sockets μπορούν να χρησιμοποιηθούν μόνο για επικοινωνία ανάμεσα σε δύο διεργασίες στον ίδιο υπολογιστή. Αντίθετα, τα Internet-domain sockets μπορούν να χρησιμοποιηθούν για τη σύνδεση διεργασιών σε διαφορετικά μηχανήματα που συνδέονται μέσω δικτύου.

5 Παραδείγματα χρήσης TCP sockets

Σας δίνονται σύντομα παραδείγματα χρήσης TCP sockets από την πλευρά του πελάτη (`socket-client.c`) και την πλευρά του εξυπηρετητή (`socket-server.c`).

Το `socket-server.c` υλοποιεί έναν απλό "echo" server: ακούει σε προκαθορι-

σμένη πόρτα TCP και δέχεται μια νέα εισερχόμενη σύνδεση. Τότε, ξεκινά να διαβάζει συνεχώς δεδομένα από αυτή, τα οποία στέλνει πάλι πίσω αλλάζοντας τα μικρά γράμματα σε κεφαλαία. Μόνο μία σύνδεση είναι ενεργή κάθε φορά: όταν αυτή κλείσει, ο εξυπηρετητής προχωρά στην αποδοχή της επόμενης.

Αντίστοιχα, ο πελάτης στο `socket-client.c`, δέχεται ως ορίσματα στη γραμμή εντολών το όνομα ενός μηχανήματος στο οποίο εκτελείται ο server και την πόρτα στην οποία ακούει, επιλύει το συγκεκριμένο όνομα σε διεύθυνση IPv4, κι αν αυτό είναι εφικτό, συνδέεται εκεί, στέλνει ένα προκαθορισμένο μήνυμα, και τυπώνει στο χρήστη την απάντηση που λαμβάνει από τον server.

Ένα παράδειγμα χρήσης φαίνεται στα ακόλουθα.

Αρχικά ξεκινάμε τον εξυπηρετητή:

```
$ ./socket-server
Created TCP socket
Bound TCP socket to port 35001
Waiting for an incoming connection...
```

Οπότε από το ίδιο ή κάποιο άλλο μηχάνημα, ενεργοποιούμε τον πελάτη:

```
$ ./socket-client daedalus.cslab.ece.ntua.gr 35001
Created TCP socket
Connecting to remote host... Connected.
I said:
Hello there!
Remote says:
HELLO THERE!
Done.
```

Ταυτόχρονα, ο server αναγνωρίζει την εισερχόμενη σύνδεση:

```
Incoming connection from 147.102.3.98:40029
Peer went away
Waiting for an incoming connection...
```

Έχοντας τερματίσει τον server, ο client δεν μπορεί πλέον να συνδεθεί:

```
$ ./socket-client daedalus.cslab.ece.ntua.gr 35001
Created TCP socket
Connecting to remote host... connect: Connection refused
```

6 Χρήσιμα εργαλεία

Κατά τον πειραματισμό με TCP sockets, θα σας φανούν πολύ χρήσιμα εργαλεία όπως τα:

- `netstat`
  Εμφανίζει μια λίστα των ανοιχτών sockets και της κατάστασης στην οποία βρίσκεται κάθε ένα από αυτά. Δείτε το manpage `netstat(8)` για πολύ χρήσιμες παραμέτρους, π.χ. η εντολή `netstat -ta` δείχνει μόνο τα TCP sockets, *μαζί* με αυτά που περιμένουν νέα εισερχόμενη σύνδεση (`LISTENING`).

- `nc / telnet`
  Χρήσιμα εργαλεία για την πραγματοποίηση δικτυακών συνδέσεων απευθείας από τη γραμμή εντολών, π.χ. με `nc www.ntua.gr 80`. Δείτε τα `nc(1)`, `telnet(1)`.

- `tcpdump`
  Το tcpdump είναι ένας "packet sniffer". Χρησιμοποιήστε το σε μηχάνημα όπου έχετε δικαίωμα `root` για να καταγράψετε και να αναλύσετε όλα τα πακέτα που περνάνε από συγκεκριμένη δικτυακή διεπαφή, π.χ. την κάρτα δικτύου `eth0` του μηχανήματός σας: `tcpdump -ni eth0`. Δείτε τη σελίδα `tcpdump(8)` για όλες τις λεπτομέρειες.

7 Βιβλιογραφία

- UNIX Network Programming, Vol. 1: The Sockets Networking API (3rd Edition), by W. Richard Stevens, Bill Fenner, Andrew M. Rudoff (2003), link
- TCP/IP Illustrated, Volume 1: The Protocols (2nd Edition), by Kevin R. Fall, W. Richard Stevens (2011) link
- The Linux Programming Interface, by Michael Kerrisk (2010), link