Προγραμματισμός με BSD Sockets σε περιβάλλον Linux

Εργαστήριο Λειτουργικών Συστημάτων
8ο εξάμηνο, ΣΗΜΜΥ

Εργαστήριο Υπολογιστικών Συστημάτων (CSLab)

Απρίλιος 2016

Περίγραμμα παρουσίασης

1. Εισαγωγή
2. Δικτυακά Πρωτόκολλα - TCP/IP
3. Sockets - Βασικές Έννοιες
4. Sockets API
5. Χρήσιμα εργαλεία

Διαδιεργασιακή επικοινωνία

- **Shared Memory:** Δύο οι περισσότερες διεργασίες, load/store σε κοινή μνήμη
- **Pipes:** Μονόδρομη επικοινωνία ανάμεσα σε συγγενικές διεργασίες

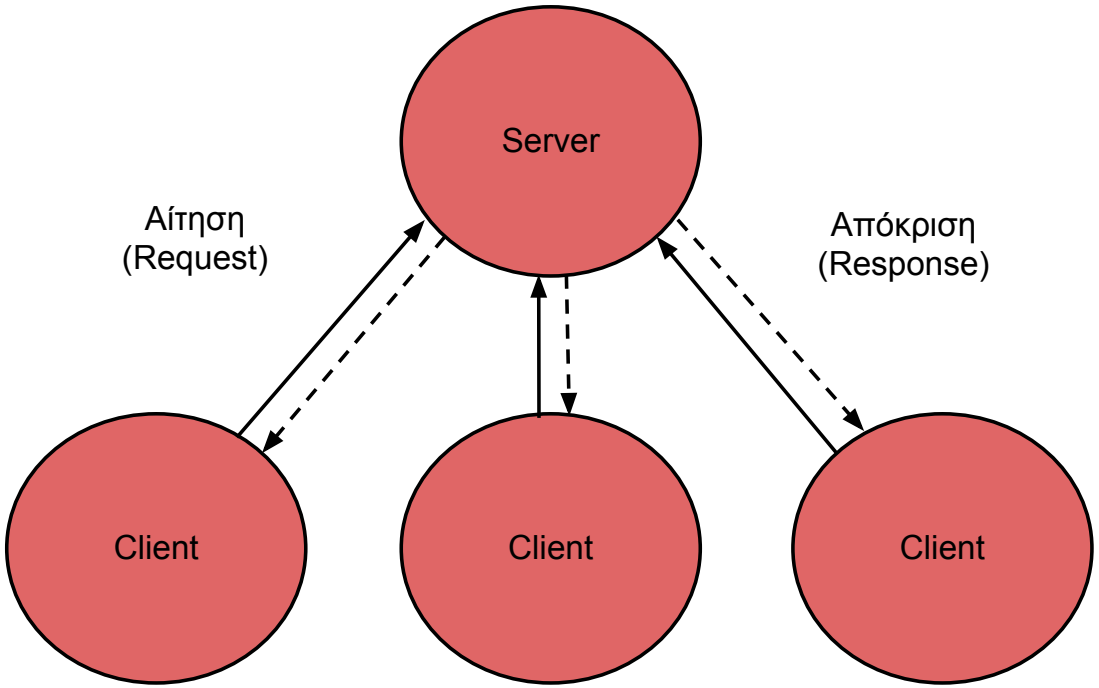
Διαδιεργασιακή επικοινωνία

- **Shared Memory:** Δύο οι περισσότερες διεργασίες, load/store σε κοινή μνήμη
- **Pipes:** Μονόδρομη επικοινωνία ανάμεσα σε συγγενικές διεργασίες

Berkeley Sockets

- Προγραμματιστική **διεπαφή** για επικοινωνία
- UNIX sockets, UDP και TCP πάνω από IPv4/IPv6
- Δίκτυο: Διεργασίες σε οποιοδήποτε μηχάνημα

Επικοινωνία με sockets



- Μοντέλο Client-Server
- Local vs. Internet sockets
- Datagram sockets (UDP)
- Connection-oriented sockets (TCP)

TCP/IP

Internet Protocol (IP)

- Μοναδική διεύθυνση 32/128 bits ανά κόμβο (IPv4/IPv6)
- Αναξιόπιστη (best effort) μεταφορά πακέτων ανάμεσά τους

TCP/IP

Internet Protocol (IP)

- Μοναδική διεύθυνση 32/128 bits ανά κόμβο (IPv4/IPv6)
- Αναξιόπιστη (best effort) μεταφορά πακέτων ανάμεσά τους

Transmission Control Protocol (TCP)

- Αξιόπιστες συνδέσεις, **πάνω** από IP

TCP/IP

Internet Protocol (IP)

- Μοναδική διεύθυνση 32/128 bits ανά κόμβο (IPv4/IPv6)
- Αναξιόπιστη (best effort) μεταφορά πακέτων ανάμεσά τους

Transmission Control Protocol (TCP)

- Αξιόπιστες συνδέσεις, **πάνω** από IP
- Αμφίδρομο ρεύμα δεδομένων, με εγγυημένη παράδοση

TCP/IP

Internet Protocol (IP)

- Μοναδική διεύθυνση 32/128 bits ανά κόμβο (IPv4/IPv6)
- Αναξιόπιστη (best effort) μεταφορά πακέτων ανάμεσά τους

Transmission Control Protocol (TCP)

- Αξιόπιστες συνδέσεις, **πάνω** από IP
- Αμφίδρομο ρεύμα δεδομένων, με εγγυημένη παράδοση
- Αρίθμηση πακέτων, ανάκαμψη από σφάλματα

TCP/IP

Internet Protocol (IP)

- Μοναδική διεύθυνση 32/128 bits ανά κόμβο (IPv4/IPv6)
- Αναξιόπιστη (best effort) μεταφορά πακέτων ανάμεσά τους

Transmission Control Protocol (TCP)

- Αξιόπιστες συνδέσεις, **πάνω** από IP
- Αμφίδρομο ρεύμα δεδομένων, με εγγυημένη παράδοση
- Αρίθμηση πακέτων, ανάκαμψη από σφάλματα
- Δε διατηρεί πληροφορία για το όρια του μηνύματος (σε αντίθεση με UDP/SCTP)

Βασικές έννοιες των Sockets

Χώρος ονομάτων - domain

- Local namespace - ονόματα αρχείων (`PF_UNIX`)
- Internet namespace - IPv4/IPv6 addresses (32/128 bit) (`PF_INET`, `PF_INET6`)

Βασικές έννοιες των Sockets

Χώρος ονομάτων - domain

- Local namespace - ονόματα αρχείων (`PF_UNIX`)
- Internet namespace - IPv4/IPv6 addresses (32/128 bit) (`PF_INET`, `PF_INET6`)

Στυλ επικοινωνίας - type

- Connection-oriented: Με σύνδεση, ρεύμα bytes, αξιόπιστη μεταφορά (`SOCK_STREAM`)
- Datagram: Χωρίς σύνδεση, αναξιόπιστη μεταφορά διακριτών πακέτων (`SOCK_DGRAM`)

Βασικές έννοιες των Sockets

Χώρος ονομάτων - domain

- Local namespace - ονόματα αρχείων (`PF_UNIX`)
- Internet namespace - IPv4/IPv6 addresses (32/128 bit) (`PF_INET`, `PF_INET6`)

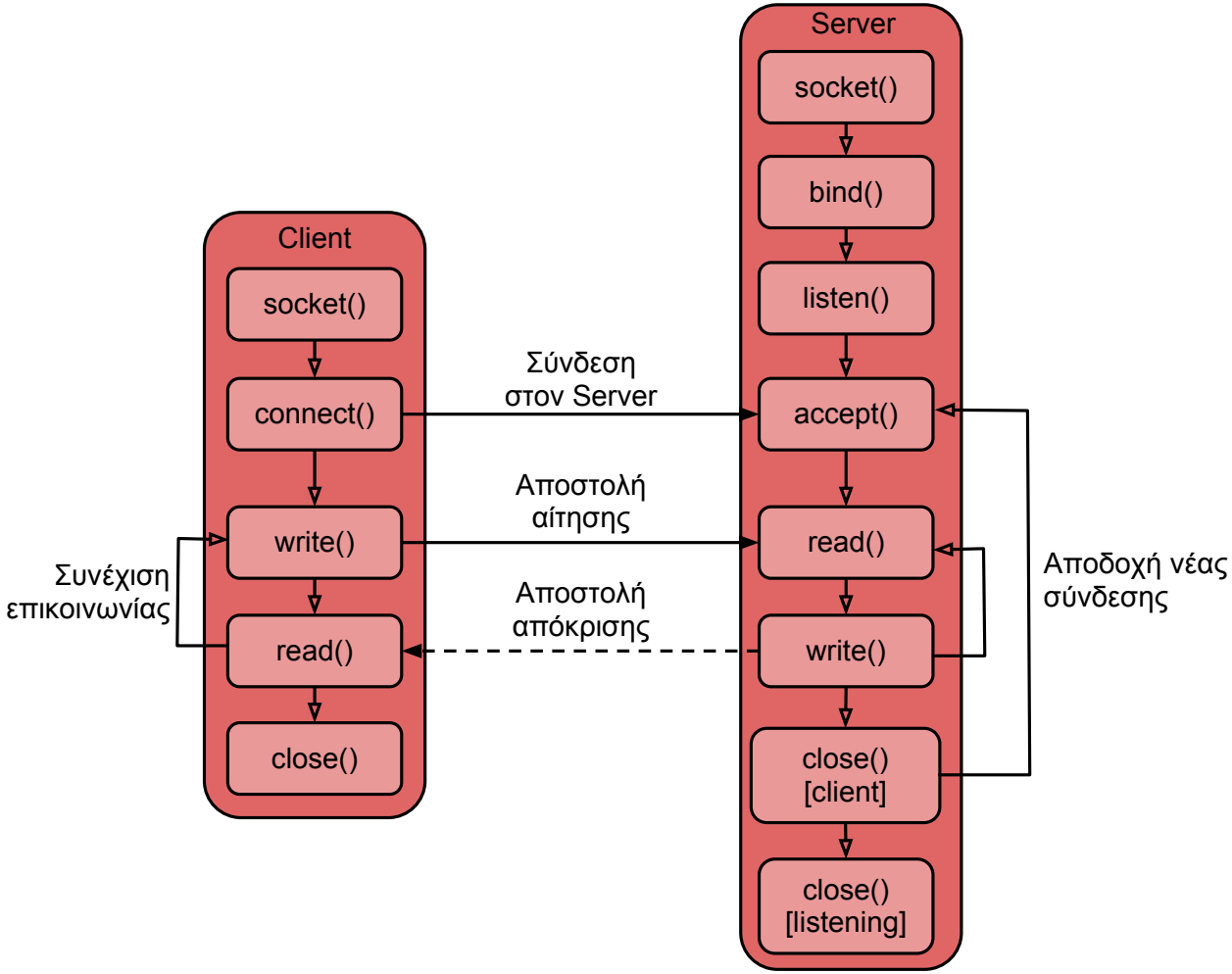
Στυλ επικοινωνίας - type

- Connection-oriented: Με σύνδεση, ρεύμα bytes, αξιόπιστη μεταφορά (`SOCK_STREAM`)
- Datagram: Χωρίς σύνδεση, αναξιόπιστη μεταφορά διακριτών πακέτων (`SOCK_DGRAM`)

Πρωτόκολλο επικοινωνίας - protocol

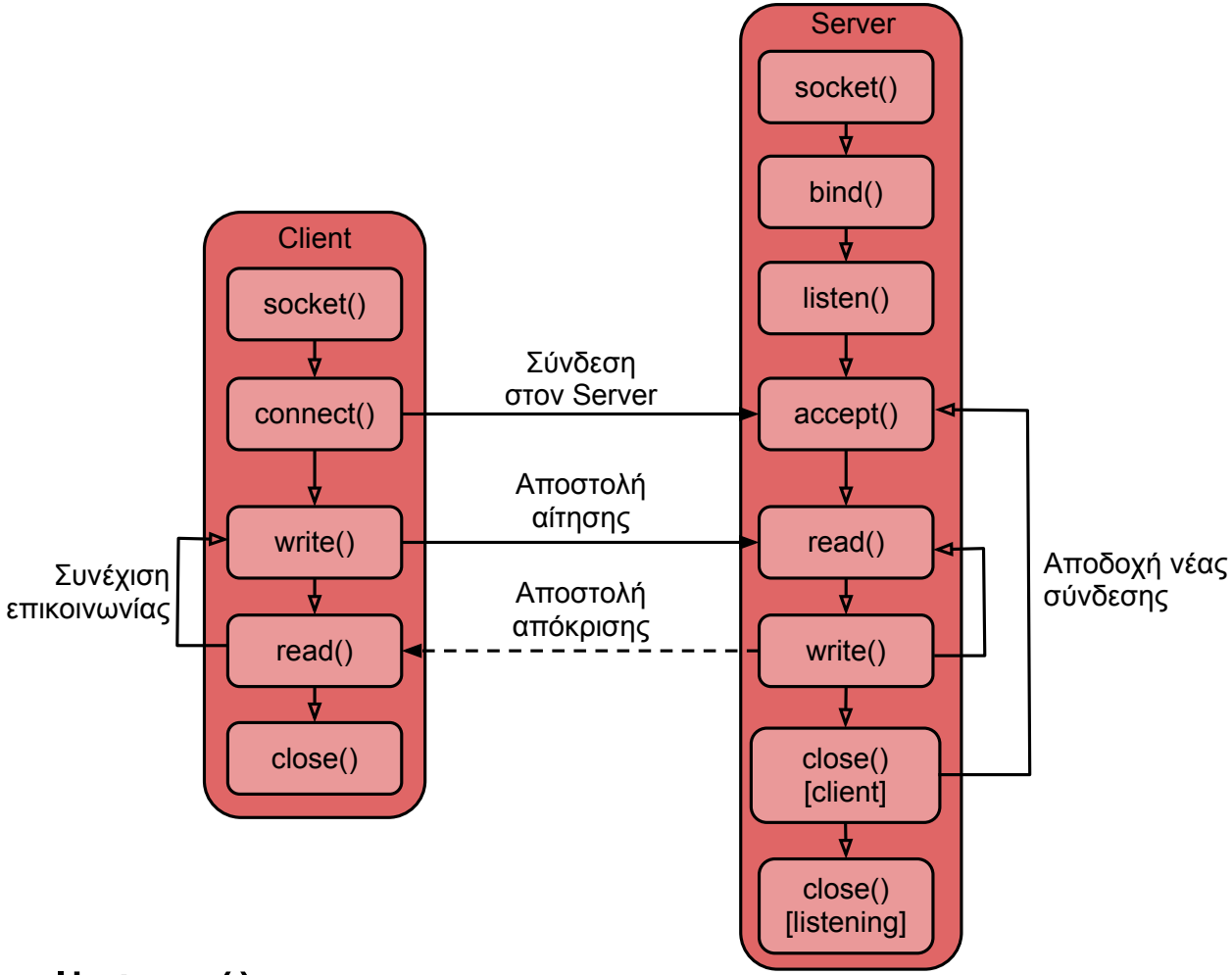
- Συγκεκριμένος τρόπος επικοινωνίας (0 $\rightarrow$ default)

Η ζωή ενός client socket



- Δημιουργία - σύνδεση - εγγραφή - ανάγνωση - κλείσιμο

Η ζωή ενός server socket



- bind() - listen()
- accept() - close() (**νέο** socket για κάθε accepted client)

Sockets API

- API **ανεξάρτητο** του υφιστάμενου πρωτοκόλλου
- Χρήση γενικού `struct sockaddr`
- Κάθε socket είναι ένα file descriptor
- Οπότε: `read()`, `write()`, `select()`, `close()`

socket()

int socket(int domain, int type, int protocol);

- domain: PF_{UNIX,LOCAL}, PF_INET
- type: SOCK_STREAM, SOCK_DGRAM, SOCK_SEQPACKET, ...
- protocol: IPPROTO_TCP, IPPROTO_UDP, ή 0 για το default
- Δημιουργεί το socket, επιστρέφει το file descriptor

Παράδειγμα χρήσης

```
#include <sys/types.h>
#include <sys/socket.h>
...
int sd = socket(PF_INET, SOCK_STREAM, 0);
if (sd < 0) {
    perror(”socket”);
    exit(1);
}
```

connect()

int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);

- Συνδέει το sockfd σε μία (απομακρυσμένη;) διεύθυνση
- addr: η διεύθυνση στην οποία συνδεόμαστε

Παράδειγμα: struct sockaddr_in, από **ip(7)**

```
struct sockaddr_in {
    sa_family_t    sin_family; /* address family: AF_INET */
    in_port_t      sin_port;   /* port in network byte order */
    struct in_addr sin_addr;   /* internet address */
};
```

Παράδειγμα χρήσης

```
#include <sys/types.h>
#include <sys/socket.h>
...
if (connect(sd, (struct sockaddr *)&sa, sizeof(sa)) < 0) {
    perror("connect");
    exit(1);
}
```

bind()

int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);

- Δένει το sockfd σε συγκεκριμένη διεύθυνση
- addr: η (τοπική) διεύθυνση στην οποία θα δεθεί το socket

Παράδειγμα χρήσης

```
#include <sys/types.h>
#include <sys/socket.h>
...
if (bind(sockfd, (struct sockaddr *)&sa, sizeof(sa)) < 0) {
    perror (”bind”);
    exit(1);
}
```

listen()

int listen(int sockfd, int backlog);

- Κάνει το sockfd κατάλληλο να δεχτεί συνδέσεις
- backlog: μήκος ουράς για εισερχόμενες συνδέσεις

Παράδειγμα χρήσης

```
#include <sys/types.h>
#include <sys/socket.h>
...
if (listen(sd, 5)) < 0) {
    perror(”listen”);
    exit(1);
}
```

accept()

int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);

- Αποδέχεται μια νέα εισερχόμενη σύνδεση στο sockfd
- Η διεύθυνση του συνδεδεμένου πελάτη γράφεται στον δείκτη addr
- Επιστρέφει **νέο** συνδεδεμένο socket, για επικοινωνία με τον πελάτη

Παράδειγμα χρήσης

```
#include <sys/types.h>
#include <sys/socket.h>
...
clientfd = accept(sd, (struct sockaddr *)&client_addr, &len);
if (clientfd < 0) {
    perror(”accept”);
    exit(1);
}
```

Χρήσιμα εργαλεία

netstat

```
~$ netstat -nutap
Proto Recv-Q Send-Q Local Address           Foreign Address         State       P
tcp        0      0 0.0.0.0:17500           0.0.0.0:*               LISTEN      2
tcp        1      0 192.168.2.4:41655       199.127.180.2:80        CLOSE_WAIT  1
tcp        1      0 192.168.2.4:44547       199.127.180.2:80        CLOSE_WAIT  1
tcp        0      0 192.168.2.4:44710       147.102.3.1:143         ESTABLISHED 2
```

telnet

```
~$ telnet www.cslab.ece.ntua.gr 80
Trying 147.102.3.233...
Connected to geronimo.cslab.ece.ntua.gr.
Escape character is '^]'.
GET /
<html>
...
```

netcat (nc)

```
~$ nc -v -z www.cslab.ece.ntua.gr 80
Connection to www.cslab.ece.ntua.gr 80 port [tcp/http] succeeded!
```

Επόμενα βήματα

- Σας δίνονται απλά παραδείγματα TCP/IP server και client
- Μελέτη των manpages, `man 2 select`
- Προσοχή στους δείκτες :)

Βιβλιογραφία

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

Ερωτήσεις;

Λίστα μαθήματος:
os-lab@lists.cslab.ece.ntua.gr