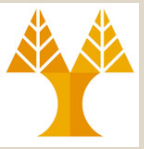
ΕΠΛ421 - Προγραμματισμός Συστημάτων



Διάλεξη 17-18:

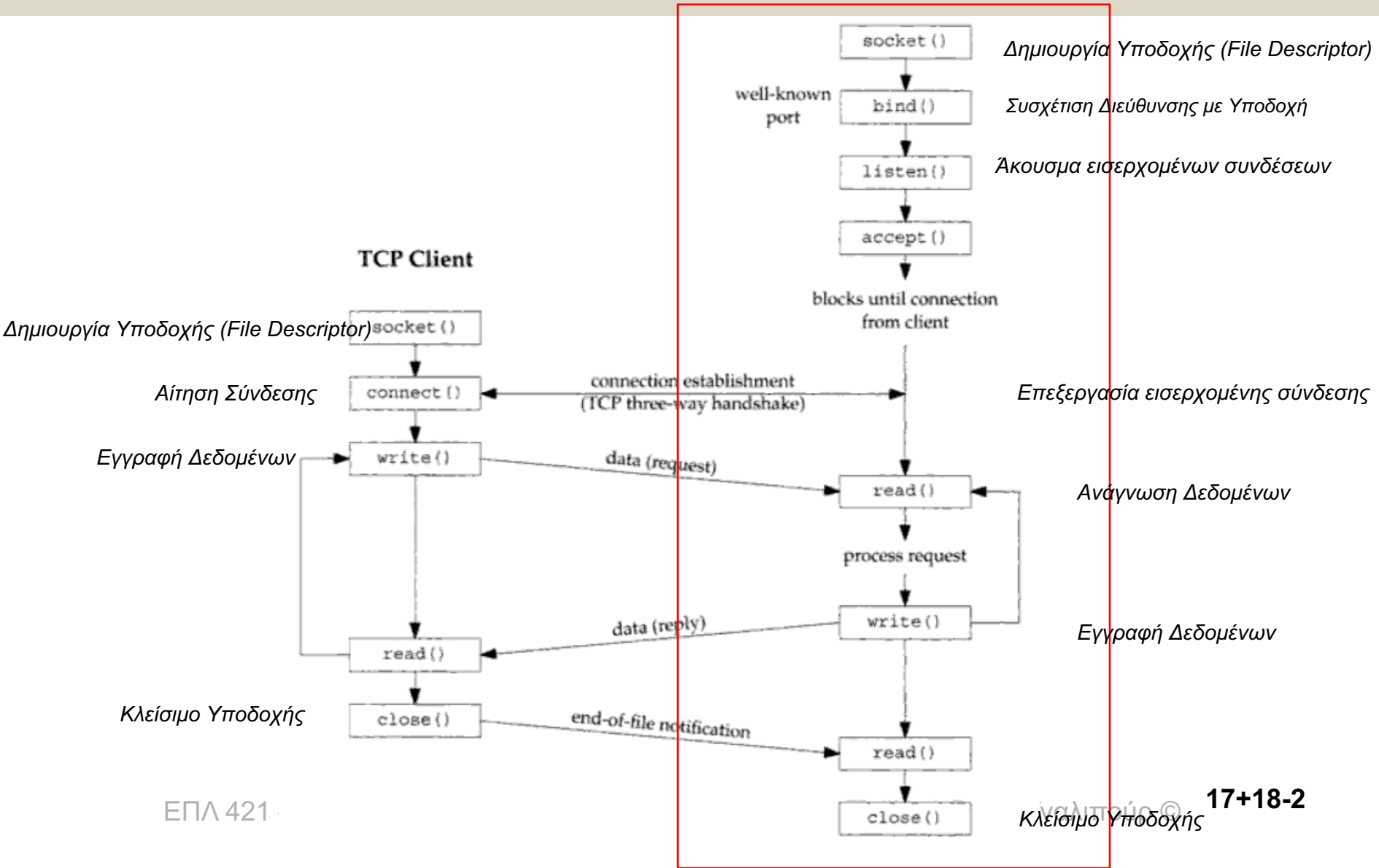
Επικοινωνία μεταξύ Διεργασιών σε Διαφορετικούς Η/Υ μέσω Δικτύου

Προγραμματισμός Υποδοχών (Socket Programming)

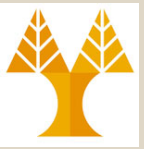
(Κεφάλαια 16/17 - Stevens & Rago)

Δημήτρης Ζεϊναλιπούρ

TCP Επικοινωνία Πελάτη / Εξυπηρετητή



Περιεχόμενο Διάλεξης



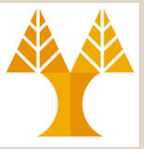
Λειτουργία του Εξυπηρετητή

- ***Δημιουργία Υποδοχής*** *(socket())*
- ***Συσχέτιση Διεύθυνσης με Υποδοχή*** *(bind()) & Δομή Διεύθυνσης Υποδοχής*
- ***Άκουσμα/ Αποδοχή εισερχομένων συνδέσεων*** *(listen(), accept()),*
- ***Μεταφορά δεδομένων*** *(read(), write(), send(), recv()),*
- ***Κλείσιμο σύνδεσης*** *(close(), shutdown()),*

Λειτουργία του Πελάτη

- ***Αίτηση σύνδεσης*** *(connect())*
- *Επιπλέον υλοποιούνται με τον ίδιο τρόπο τα read(), write(), send(), recv(), close(), shutdown(), και socket()*

Περιεχόμενο Διάλεξης



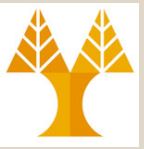
Άλλες Χρήσιμες Κλήσεις Συστήματος

- ***Μετατροπή ακολουθίας bytes*** *(little/big endian byte order),*
- ***Χειρισμός IP διεύθυνσης, Domain Names, Byte Processing, κτλ*** *(inet_addr(), inet_ntoa()), getpeername(), gethostname(), gethostbyname(), gethostbyaddr()), getsockname(), setsockopt(), bzero(), bcopy()),*

Ολοκληρωμένο Παράδειγμα Εκτέλεσης

- ***Παράδειγμα client-server.***

Δημιουργία Υποδοχής (Κλήση Συστήματος socket())



- Δημιουργία του FD ενός socket.

```
#include <sys/types.h>
#include <sys/socket.h>
int socket (int family, int type, int protocol);
```

Επιστρέφει περιγραφέα αρχείου (υποδοχής) σε επιτυχία ή -1 σε λάθος

ή PF_INET

family	Description
AF_INET	IPv4 protocols
AF_INET6	IPv6 protocols
AF_LOCAL	Unix domain protocols (Chapter 15)
AF_ROUTE	Routing sockets (Chapter 18)
AF_KEY	Key socket (Chapter 19)

ή AF_UNIX

type	Description
SOCK_STREAM	stream socket
SOCK_DGRAM	datagram socket
SOCK_SEQPACKET	sequenced packet socket
SOCK_RAW	raw socket

Protocol	Description
IPPROTO_TCP	TCP transport protocol
IPPROTO_UDP	UDP transport protocol
IPPROTO_SCTP	SCTP transport protocol

Εμάς μας ενδιαφέρει μόνο ο συνδυασμός AF_INET, SOCK_STREAM, IPPROTO_TCP

"AF: Address Family" "PF: Protocol Family". PF was used in older versions …

Συσχέτιση **Διεύθυνσης** με **Υποδοχή** (Κλήση Συστήματος bind())

- **Διεύθυνση** = 32-bit IP και 16-bit PORT
- **Υποδοχή** = Ο File Descriptor του Socket

```
#include <sys/types.h>
#include <sys/socket.h>
int bind (int sockfd, struct sockaddr
    *server_addr, socklen_t length);
```

Δομή που περιέχει το IP, PORT (δες επόμενη διαφάνεια)

Επιστρέφει 0 σε επιτυχία ή -1 σε λάθος

- Τώρα ο **πυρήνας** μπορεί να **καθοδηγήσει αιτήματα** από πελάτες στη **διεργασία** με το συγκεκριμένο **περιγραφέα υποδοχής!**

Δομή Διεύθυνσης Υποδοχής (struct sockaddr)

- Για να μπορούν **διαφορετικές μορφές** *διευθύνσεων* να περνούν σε συναρτήσεις υποδοχών, οι μορφές αυτές **«μετατρέπονται» (casting)** σε μια γενική δομή *διεύθυνσης* υποδοχής (***struct sockaddr*).**
  - Θυμηθείτε, ότι υπάρχουν πολλοί διαφορετικοί συνδυασμοί και θα ήταν δύσκολο να είχαμε διαφορετική δομή για κάθε συνδυασμό.
    - Σε αυτό το μάθημα μας ενδιαφέρει μόνο το κυκλωμένο

	AF_INET	AF_INET6	AF_LOCAL	AF_ROUTE	AF_KEY
SOCK_STREAM	TCP|SCTP	TCP|SCTP	Yes		
SOCK_DGRAM	UDP	UDP	Yes		
SOCK_SEQPACKET	SCTP	SCTP	Yes		
SOCK_RAW	IPv4	IPv6		Yes	Yes

Δομή Διεύθυνσης Υποδοχής (struct sockaddr_in)

- ***Δομή Ίντερνετ Διεύθυνσης για IPv4 πεδίο υποδοχών (struct sockaddr_in → <netinet/in.h>)***

```
/* Structure describing an Internet socket address.  */
struct sockaddr_in
  {
    sa_family_t sin_family;          /* Address family, e.g., AF_INET */
    in_port_t sin_port;              /* Port number, e.g., .  */
    struct in_addr sin_addr;         /* Internet IPv4 address.  */
};

/* Internet address.  */
typedef uint32_t in_addr_t; // δηλ., ένας int: π.χ.,  0x0A0C6E39 → 10.12.110.57
struct in_addr
  {
    in_addr_t s_addr;
  };
```

Ο λόγος που αναπαριστάται το IP ετσι περίπλοκα είναι για ιστορικούς λόγους (παλιά το sin_addr ήταν union από πληροφορίες, see p70 Unix Net Prog., 3ED)

Δομή Διεύθυνσης Υποδοχής (struct sockaddr_in)

Συνεπώς θα δούμε κάτι όπως το ακόλουθο στον **server**…

Θα εξηγηθούν αργότερα (διασφαλίζουν ορθότητα διάταξης των bytes)

```
int port = 30000;
struct sockaddr_in server;

server.sin_family = PF_INET; /* Internet domain */
server.sin_addr.s_addr = htonl(INADDR_ANY); /* My Internet address */
server.sin_port = htons(port); /* The given port */

/* Bind socket to address */
if (bind(sock, (struct sockaddr *) &server, sizeof (server)) < 0) {
    perror("bind");
    exit(1);
}
```

Casting
struct sockaddr_in *
σε
struct sockaddr *

Συσχέτιση Διεύθυνσης με Υποδοχή (struct sockaddr_in)

- Μπορούμε να αυτοματοποιήσουμε τη διαδικασία να πάρουμε την **IP διεύθυνση** και Port για να συσχετιστούν με συγκεκριμένη υποδοχή.
- Ρύθμιση πεδίων στο struct sockaddr_in :

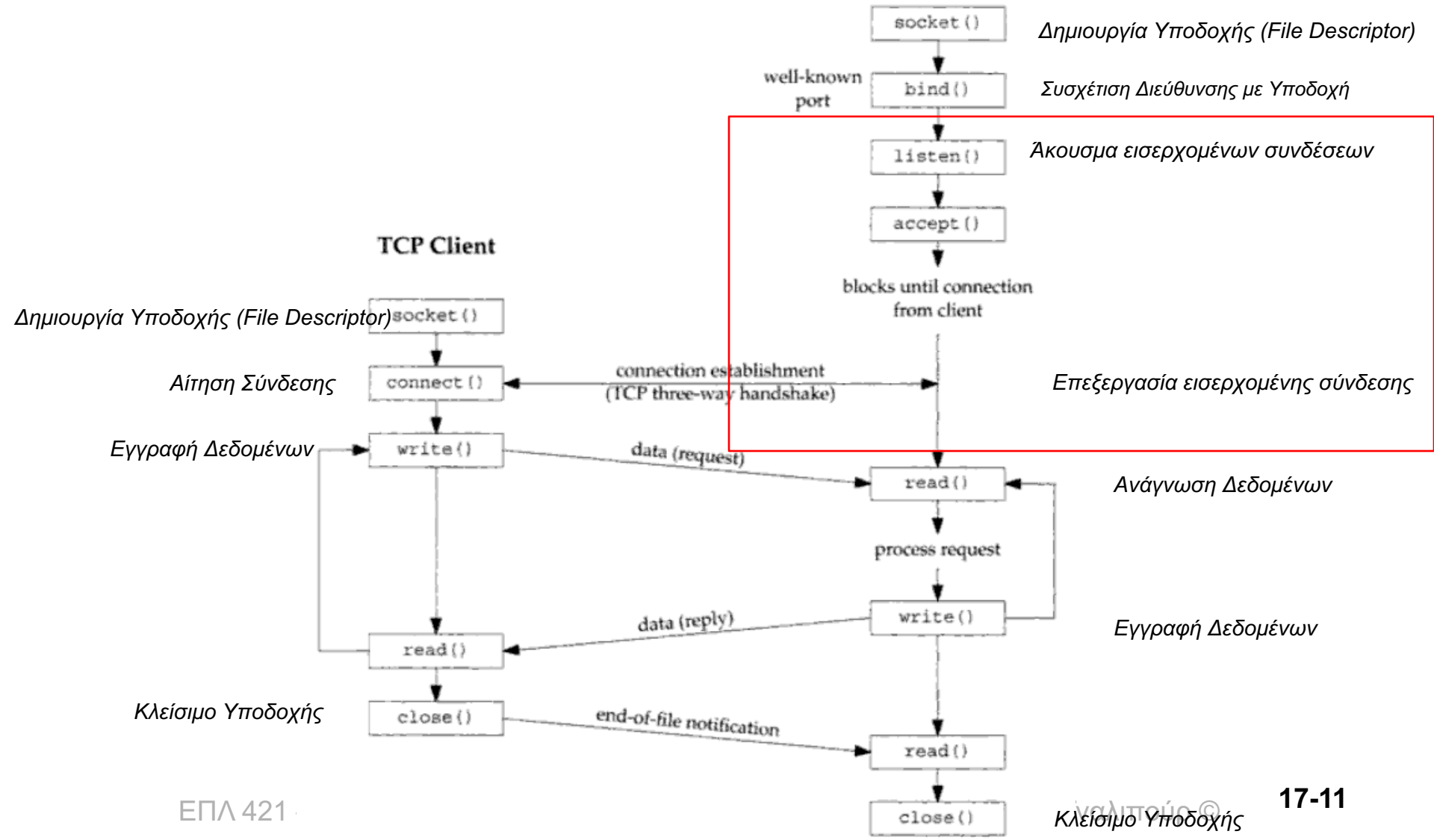
  client.sin_port = htons(0);

  – Επιλέγει ο πυρήνας ένα ελεύθερο port **τυχαία**
    - Κατάλληλο για client, Ακατάλληλο για server γιατί πρέπει να έχει ένα Well-known port

  server.sin_addr.s_addr = htonl(INADDR_ANY);

  – Χρησιμοποιείται αυτόματα η IP διεύθυνση της μηχανής (default network interface) στην οποία η συγκεκριμένη διεργασία τρέχει.
    - Κατάλληλο για client & κατάλληλο για server.
    - Προσοχή εάν έχουμε πολλαπλά network interface

TCP Επικοινωνία Πελάτη / Εξυπηρετητή



Άκουσμα Εισερχομένων Συνδέσεων (listen())

- Κλήση συστήματος *listen()*

Συνήθως 5 (αυτό δεν σχετίζεται με τον αριθμό των παράλληλων συνδέσεων με τον διαθέτη)

```
#include <sys/socket.h>
int listen (int sockfd, int num_queued_requests);
```

Επιστρέφει 0 σε επιτυχία ή -1 σε λάθος

- **Ο εξυπηρετητής ανακοινώνει ότι είναι πρόθυμος να δεκτεί αιτήσεις σύνδεσης από πελάτες.**
- Ορίζεται μια ουρά μήκους ***num_queued_requests*** *στον πυρήνα του* εξυπηρετητή στην οποία μπορούν να συσσωρεύονται αιτήσεις από πελάτες για σύνδεση στην υποδοχή που αντιστοιχεί στον περιγραφέα *sockfd*
  - *Αιτήσεις πέρα του num_queued_requests γίνονται drop, επομένως ο πελάτης πρέπει να ξανα-επιχειρήσει το 3-way TCPhandshake στην περιπτωση TCP/IP.*

Αποδοχή Εισερχομένων Αιτήσεων (Accept())

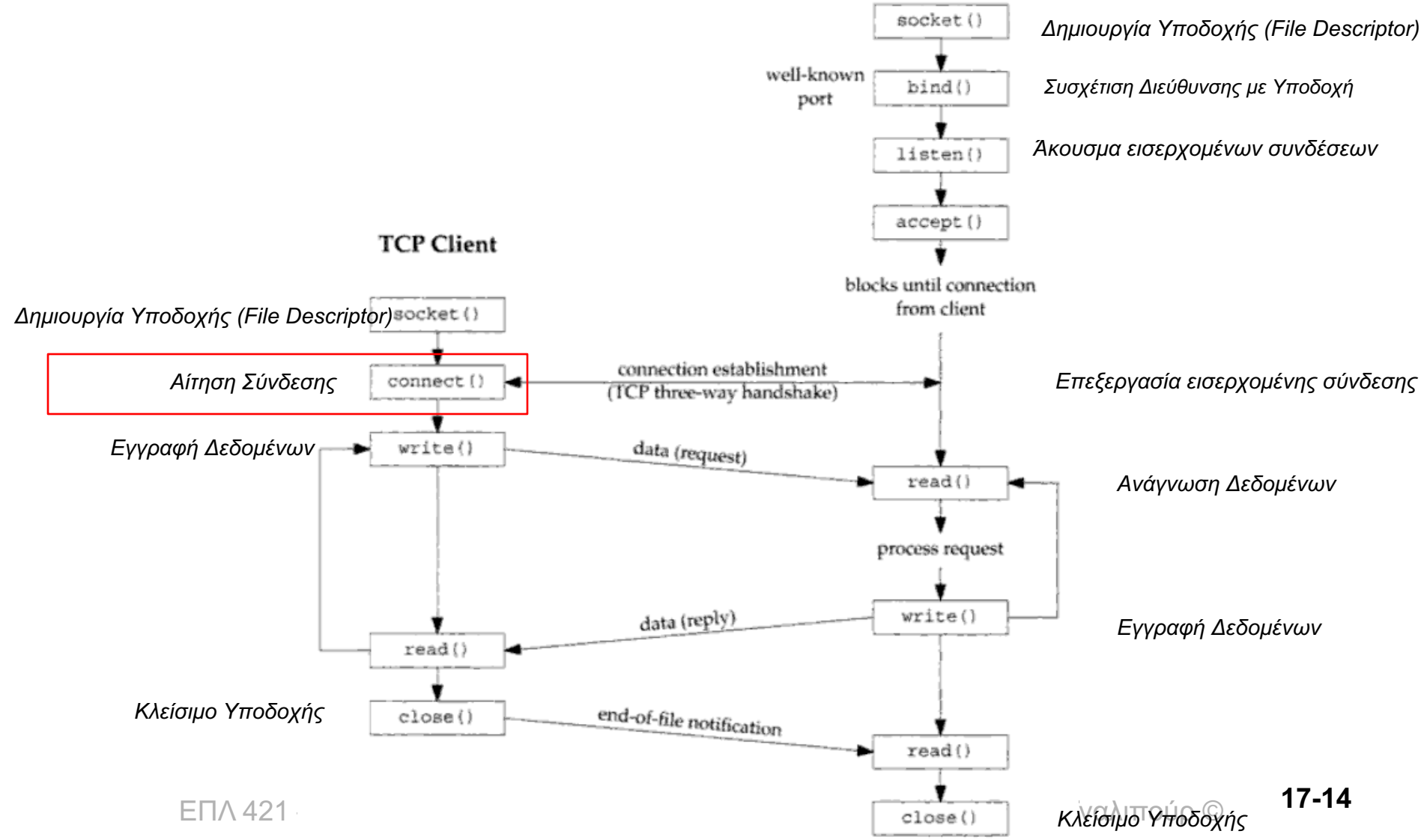
- Κλήση συστήματος *accept()*

```
#include <sys/socket.h>
int accept (int sockfd, struct sockaddr
    *incoming_address, socklen_t *length);
```

Επιστρέφει περιγραφέα αρχείου (υποδοχής) σε επιτυχία ή -1 σε λάθος

- **Αποδοχή εισερχόμενη αίτησης** σύνδεσης που έχει υποβληθεί σε ένα εξυπηρετητή στην **υποδοχή με περιγραφέα υποδοχής *sockfd*.**
- Πληροφορίες για τη διεύθυνση του **πελάτη** που συνδέθηκε επιστρέφονται μέσω της δομής ***incoming_address**, το **μέγεθος** της οποίας επιστρέφεται στο ***length**.
- Επιστρέφει ένα **νέο** περιγραφέα υποδοχής, ο οποίος πρέπει να χρησιμοποιηθεί από τον εξυπηρετητή για τη συγκεκριμένη επικοινωνία με τον πελάτη.

TCP Επικοινωνία Πελάτη / Εξυπηρετητή



Αίτηση Σύνδεσης (Connect())

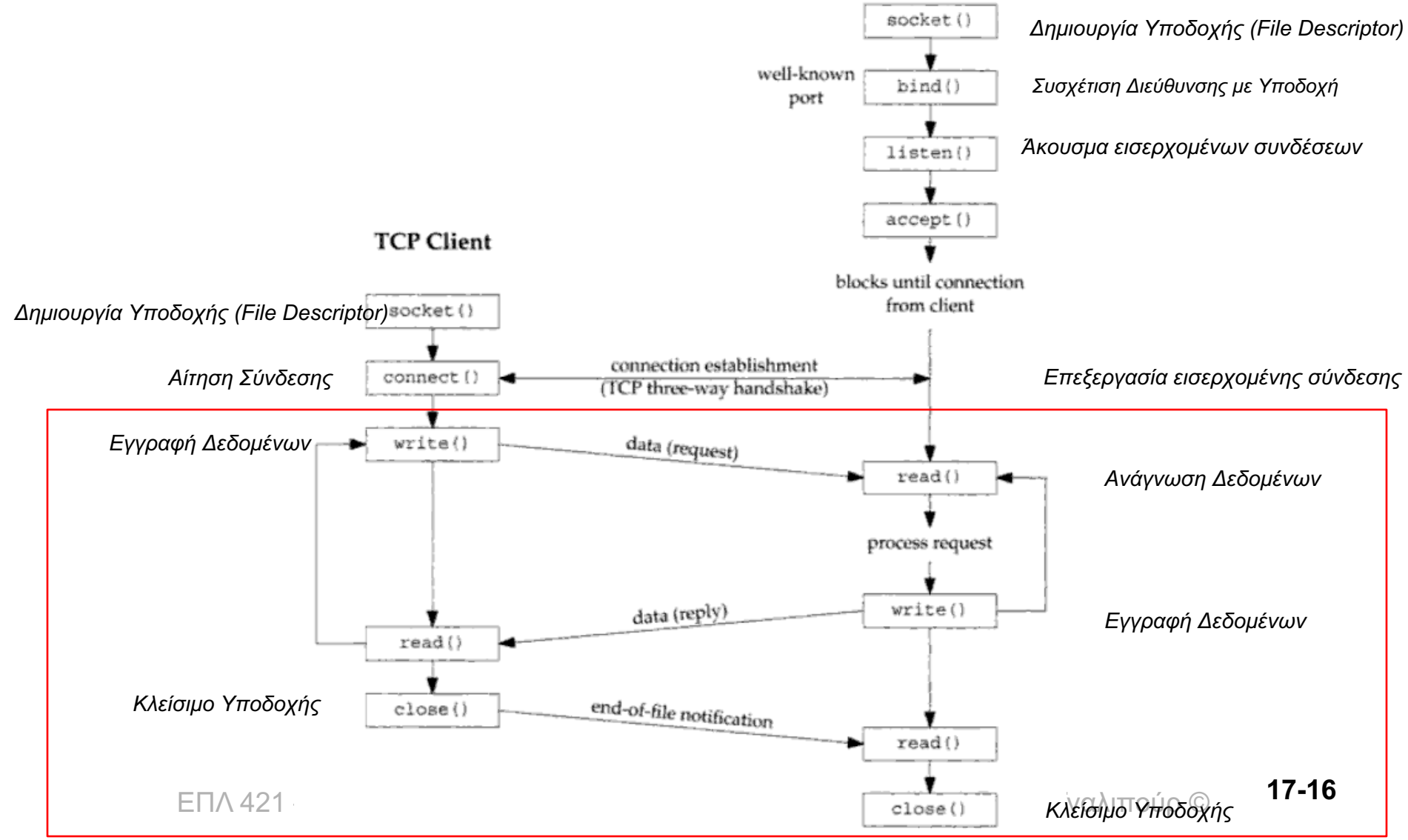
- Εκτελείται στον Client, δεδομένου του ότι έχει προηγηθεί η socket()

```
#include <sys/socket.h>
int connect(int sockfd, const struct sockaddr
    *server_address, socklen_t length);
```

Επιστρέφει 0 σε επιτυχία ή -1 σε λάθος

- **Sockfd:** FD που δημιουργήθηκε από την socket()
- **const struct sockaddr *server_address:** Διεύθυνση Υποδοχής Server (IP, Port, Family) ακριβώς όπως είδαμε και νωρίτερα.
- **socklen_t:** Μέγεθος του server_address

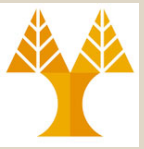
TCP Επικοινωνία Πελάτη / Εξυπηρετητή



Μεταφορά Δεδομένων & Κλείσιμο (Read()/Write()/Close())

- Εφόσον μια υποδοχή αντιπροσωπεύεται από ένα περιγραφέα αρχείου, μπορούμε να χρησιμοποιήσουμε τις κλήσεις συστήματος ***read()*** και ***write()*** για να επικοινωνήσουμε με μια υποδοχή.
- Κλήση συστήματος: ***close(sockfd);***
  - Κλείσιμο του περιγραφέα υποδοχής, αποδέσμευση του αριθμού θύρας.

```
/* Get message */
if (read(newsock, buf, sizeof(buf)) < 0) {
        perror("read"); exit(1);
}
printf("Read string: %s\n", buf);
```



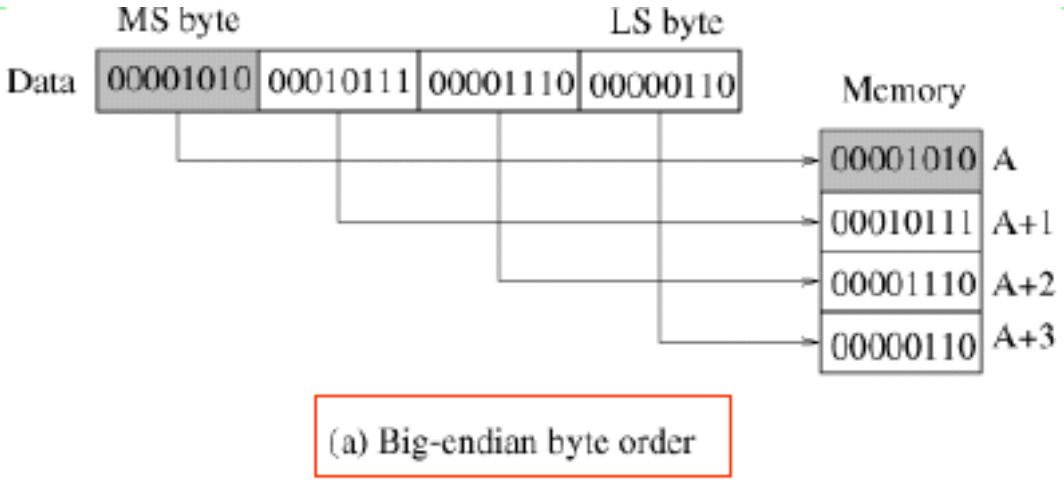
Χρήσιμες Κλήσεις Συστήματος / Συναρτήσεις Βιβλιοθήκης για Δικτυακό Προγραμματισμό

Μετατροπή Byte Ordering (Little Endian vs. Big Endian)

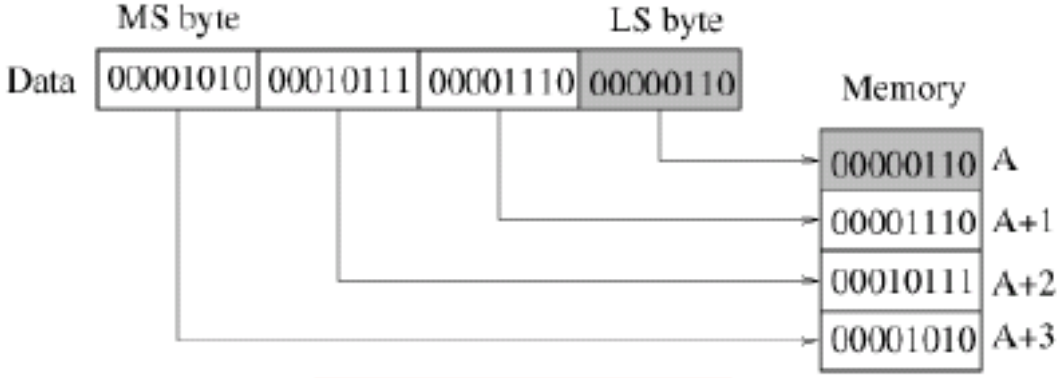
- Διαφορετικές **αρχιτεκτονικές υπολογιστών αποθηκεύουν** ακολουθίες Διφυών διαφορετικά.
- ***Little Endian* αρχιτεκτονικές** (π.χ. FreeBSD, Linux, MacOSX, Windows) αποθηκεύουν το *Least Significant Byte (LSB)* πρώτα.
  - Π.χ., int a = **0x0A0C6E39** (ακέραιος 168586809) αποθηκεύεται στη μνήμη ως **396E0C0A**
- ***Big Endian* αρχιτεκτονικές** (π.χ. Solaris on SPARC, AIX, Mac@PowerPC) αποθηκεύουν το *Most Significant Byte (MSB)* πρώτα.
  - Π.χ., int a = **0x0A0C6E39** (ακέραιος 168586809) αποθηκεύεται στη μνήμη ως **0A0C6E39**

Μετατροπή Byte Ordering (Little Endian vs. Big Endian)

Διαγραμματική Αναπαράσταση



(a) Big-endian byte order



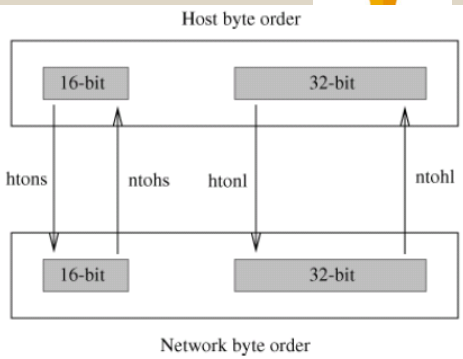
(b) Little-endian byte order

Μετατροπή Host-Network Ordering (**htons, htonl, ntohs, ntohl**)

- **Host Ordering (Διάταξη Μηχανής):** Πως μια αρχιτεκτονική αναπαριστά στη μνήμη multi-byte data (π.χ., int, float/double, structs, κτλ)
  - Little Endian ή Big Endian, ανάλογα με την αρχιτεκτονική και ΛΣ.
- **Network Ordering (Διάταξη Δικτύου)**: Πως το επίπεδο δικτύου TCP/IP αναπαριστά τα δεδομένα.
  - **Πάντα Big Endian!**

- Για να είμαστε σίγουροι ότι ένα πρόγραμμα θα λειτουργήσει ορθά σε κάθε αρχιτεκτονική (δηλ., για λόγους μεταφερσιμότητας), πρέπει να μετατρέπονται όλα τα multibyte δεδομένα σε Network Ordering
- Ακολουθούν οι σχετικές συναρτήσεις.

Μετατροπή Host-Network Ordering (**htons, htonl, ntohs, nt**



- Τέσσερις συναρτήσεις προσφέρονται για αυτή τη μετατροπή:

#include <arpa/inet.h>

Long (εννοούμε 32-bit. Παρέμεινε για ιστορικούς λόγους (Digital VAX, 1982)

Host to Network

uint32_t **h**to**nl**(uint32_t hostint32);

Επιστρέφει 32-bit (long) αριθμό σε διάταξη «δικτύου»

Short (εννοούμε 16-bit)

uint16_t **h**to**ns**(uint16_t hostint16);

Επιστρέφει 16-bit (short) αριθμό σε διάταξη «δικτύου»

Network to Host

uint32_t **n**to**hl** (uint32_t netint32); => Μετατροπή IP

Επιστρέφει 32-bit (long) αριθμό σε διάταξη «μηχανής»

uint16_t **n**toh**s**(uint16_t netint16); => Μετατροπή PORT

Επιστρέφει 16-bit (short) αριθμό σε διάταξη «μηχανής»

Μετατροπή Host-Network Ordering (32/64-bit Αρχιτεκτονικές)

- Λίγα λόγια για τύπους δεδομένων με x86 και x64 αρχιτεκτονικές
  - **I**(ntegers) L(ong) P(ointer) 32 => x86 Model
  - L(ong) P(ointer) 64 => x64 Model

Datatype	**ILP32 Model**	**LP64 Model**
char	8	8
short	16	16
int	32	32
long	**32 (4 bytes)**	**64 (8 bytes)**
pointer	**32 (4 bytes)**	**64 (8 bytes)**

Η μνήμη μπορεί να είναι μέχρι 4 GB($x10^9$) ☹!

Η μνήμη μπορεί να είναι μέχρι 16 Exabytes ($x10^{18}$) ☺!

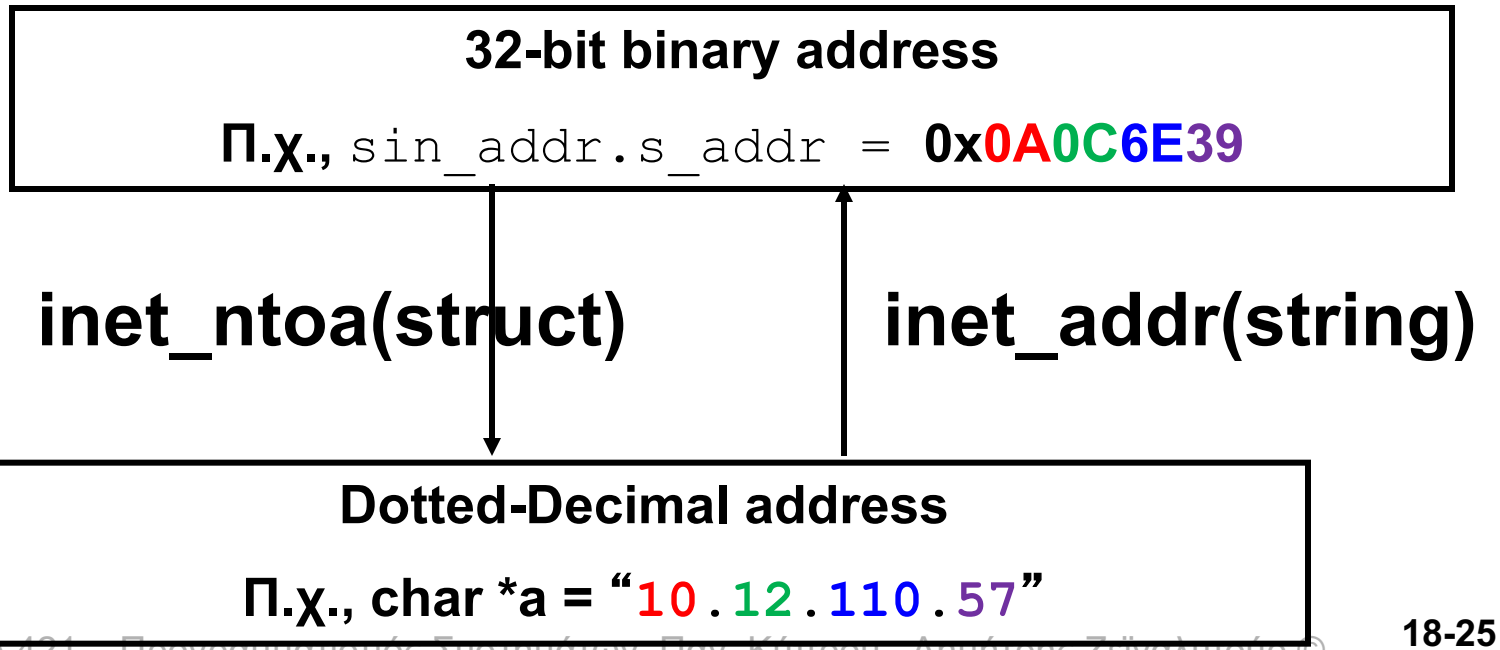
Δοκιμάστε printf("%ld", sizeof(long));στο υπολογιστή σας!

Μετατροπή Host-Network Ordering (**htons, htonl, ntohs, ntohl**)

- Οι αριθμοί **IP/PORT** είναι στα header του TCP/IP πακέτου και πρέπει ΠΑΝΤΑ να γίνεται το network ordering (εναλλακτικά δυο κόμβοι διαφορετικού endianess δεν θα καταφέρουν να μιλήσουν.)
- **Ερώτηση:** Τι γίνεται με τα υπόλοιπα δεδομένα (payload) που περιέχονται σε ένα πακέτο; Πρέπει να μετατραπούν και αυτά στο σωστό ordering;
- **Απάντηση:** Εάν μεταφέρεται ακολουθία chars (π.χ., POP3, HTTP, κτλ πρωτόκολλα) τότε δεν απαιτείται καμία μετατροπή. Εάν μεταφέρεται multi-byte data (int, short, struct, κτλ) πρέπει ο client/server να ξέρουν τι ordering χρησιμοποιείται (π.χ., μια εφαρμογή μπορεί να είναι Little Endian μια άλλη να είναι Big Endian)

Χειρισμός IP διεύθυνσης (μέσω **inet_addr / inet_ntoa**)

- Μετατροπή μεταξύ **διεύθυνσης** σε **δυαδική μορφή** και διεύθυνσης σε μορφή συμβολοσειράς (δεκαδική-με-τελείες μορφή: a.b.c.d)



Χειρισμός IP διεύθυνσης (μέσω **inet_addr**)

```
#include <arpa/inet.h>
unsigned int inet_addr(char *str);
```

Επιστρέφει την IP διεύθυνση σε δυαδική μορφή, ή -1 σε περίπτωση λάθους

- str: π.χ., «10.12.110.57»
- Επιστρέφει τη διεύθυνση σε ακολουθία bytes με *διάταξη «δικτύου»*, με βάση τη μορφή σε δεκαδικό-με-τελείες διεύθυνση / ASCII συμβολοσειρά.
- Επομένως, δε χρειάζεται να καλέσουμε τη ***hton****l().*

```
server.sin_addr.s_addr = inet_addr("10.12.110.57");
```

Χειρισμός IP διεύθυνσης (μέσω **inet_addr**)

#include <arpa/inet.h>

char* inet_**ntoa**(struct in_addr ip);

Επιστρέφει την IP διεύθυνση σε ASCII συμβολοσειρά ή -1 σε περίπτωση λάθους

– Μετατρέπει τη δυαδική 32-bit διεύθυνση, που είναι αποθηκευμένη με διάταξη «δικτύου» στη δομή *struct in_addr*, σε δεκαδικό-με-τελείες μορφή ASCII συμβολοσειρά και την επιστρέφει.

```
char *a1, *a2;
a1 = inet_ntoa(server1.sin_addr); // this is 192.168.4.14
a2 = inet_ntoa(server2.sin_addr); // this is 10.12.110.57
printf("address 1: %s\n",a1);
printf("address 2: %s\n",a2);
```

Επιπρόσθετες Συναρτήσεις (*bzero(), bcopy()*)

- Συναρτήσεις *για αρχικοποίηση ακολουθιών bytes στη μνήμη.*

void **bzero(char *buf, int count);**

- Θέτει 0 σε *count* bytes αρχίζοντας από τη διεύθυνση *buf.*
- Ίδιο με την ANSI C εντολή:
  - void * memset (void * ptr, int value, size_t num);

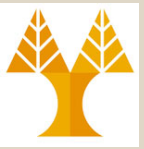
void **bcopy(char *buf1, char *buf2, int count);**

- Αντιγράφει *count* bytes αρχίζοντας από τη διεύθυνση *buf1* στη διεύθυνση *buf2.*
- Ίδιο με την ANSI C εντολή:
  - void * *memcpy* (void * dest, const void * src, size_t num);.

Παράδειγμα `Client/Server`

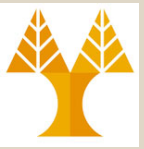
- *Γράψετε ένα πρόγραμμα πελάτη/εξυπηρετητή το οποίο θα λειτουργεί με την εξής λογική:*
  - *Ο διαθέτης δημιουργεί ένα TCP server στο PORT που προσδιορίζεται στη γραμμή εντολής.*
  - *Ο πελάτης δημιουργεί μια TCP σύνδεση με το IP, PORT που προσδιορίζεται στη γραμμή εντολής*
  - *Ο πελάτης στέλνει ένα ή περισσότερα String στον εξυπηρετητή, ο οποίος τα κάνει reverse και τα στέλνει πίσω. Αυτό επαναλαμβάνεται μέχρι να στείλει ο client το string "end" ή να διακόψει ο client.*
  - *Το πιο πάνω επαναλαμβάνεται για κάθε client που μπορεί να συνδέεται με τον server.*

Παράδειγμα **server.c**



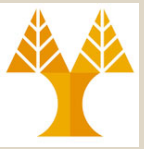
```
/* File: server.c */

#include <sys/types.h>          /* For sockets */
#include <sys/socket.h>         /* For sockets */
#include <netinet/in.h>         /* For Internet sockets */
#include <netdb.h>              /* For gethostbyaddr() */
#include <stdio.h>              /* For I/O */
#include <stdlib.h>
#include <string.h>

void reverse(char *); // function prototype for reversing func.

/* Server with Internet stream sockets */
main(int argc, char *argv[]) {
   int port, sock, newsock, serverlen, clientlen; char buf[256];
   struct sockaddr_in server, client;
   struct sockaddr *serverptr, *clientptr;
   struct hostent *rem;

   if (argc < 2) { /* Check if server's port number is given */
        printf("Please give the port number\n");
        exit(1);
   }
```

Παράδειγμα **server.c**



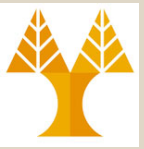
```
/* Create socket */
if ((sock = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
   perror("socket"); exit(1); }

/* Convert port number to integer */
port = atoi(argv[1]);

server.sin_family = PF_INET; /* Internet domain */
server.sin_addr.s_addr = htonl(INADDR_ANY); /* My Internet address */
server.sin_port = htons(port); /* The given port */
serverptr = (struct sockaddr *) &server;
serverlen = sizeof (server);

/* Bind socket to address */
if (bind(sock, serverptr, serverlen) < 0) {
   perror("bind"); exit(1); }

/* Listen for connections */
if (listen(sock, 5) < 0) {    /* 5 max. requests in queue */
   perror("listen"); exit(1); }
   printf("Listening for connections to port %d\n", port);
```

Παράδειγμα **server.c**



```
while(1) {
   clientptr = (struct sockaddr *) &client;
   clientlen = sizeof(client);

   /* Accept connection */
   if ((newsock = accept(sock, clientptr, &clientlen)) < 0) {
        perror("accept");
      exit(1);
   }

   /* Using IP address find DNS name (i.e., reverse DNS)*/
   if ((rem = gethostbyaddr((char *) &client.sin_addr.s_addr,
      sizeof (client.sin_addr.s_addr), client.sin_family)) == NULL) {
   herror("gethostbyaddr"); // herror(): Similar to perror but uses the h_errno
   variable (set by name resolution functions to return error values).
   exit(1);
   }

   printf("Accepted connection from %s\n", rem -> h_name);
```

Παράδειγμα `server.c`



```
/* Create child for serving the client */
switch (fork()) {
case -1:
        perror("fork"); exit(1);
case 0: /* Child process */
        do {
          bzero(buf, sizeof(buf)); /* Initialize buffer */
          if (read(newsock, buf, sizeof(buf)) < 0) { /* Get message */
            perror("read"); exit(1);
          }
          printf("Read string: %s\n", buf);
          reverse(buf); /* Reverse message */
          if (write(newsock, buf, sizeof(buf)) < 0){/* Send message */
            perror("write"); exit(1);
          }
        } while (strcmp(buf, "dne") != 0); /*Finish on "end" message*/
        close(newsock); /* Close socket */
        exit(0);
} /* end of switch */
} /* end of while(1) */
} /* end of main() */
```

Reading line-line (\n) the input

Παράδειγμα **client.c**

```
/* File: client.c */

#include <sys/types.h>          /* For sockets */
#include <sys/socket.h>         /* For sockets */
#include <netinet/in.h>         /* For Internet sockets */
#include <netdb.h>              /* For gethostbyname() */
#include <stdio.h>              /* For I/O */
#include <stdlib.h>
#include <string.h>

/* Client with Internet stream sockets */
main(int argc, char *argv[])
{
   int port, sock, serverlen; char buf[256];
   struct sockaddr_in server;
   struct sockaddr *serverptr;
   struct hostent *rem;

   /* Are server's host name and port number given? */
   if (argc < 3) {
      printf("Please give host name and port number\n"); exit(1);
   }
```

Παράδειγμα client.c

```
/* Create TCP/IP socket */
if ((sock = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
   perror("socket"); exit(1);
}

/* Find server address e.g., argv[1]="astarti.cs.ucy.ac.cy" */
if ((rem = gethostbyname(argv[1])) == NULL) {
   herror("gethostbyname"); exit(1);
}

port = atoi(argv[2]);          /* Convert port number to integer */
server.sin_family = PF_INET;   /* Internet domain */
bcopy((char *) rem -> h_addr, (char *) &server.sin_addr,
                               rem -> h_length);
server.sin_port = htons(port); /*Server's Internet address and port*/
serverptr = (struct sockaddr *) &server;
serverlen = sizeof(server);
if (connect(sock, serverptr, serverlen) < 0) { /* Request Connect */
  perror("connect"); exit(1); }
  printf("Requested connection to host %s port %d\n", argv[1], port);
```

Παράδειγμα client.c

```
do {
   bzero(buf, sizeof buf);          /* Initialize buffer */
   printf("Give input string: ");
   fgets(buf, sizeof buf, stdin); /* Read message from stdin */
   buf[strlen(buf)-1] = '\0';      /* Remove newline character */

   if (write(sock, buf, sizeof(buf)) < 0) { /* Send message */
     perror("write"); exit(1);
   }

   bzero(buf, sizeof buf); /* Initialize buffer */
   if (read(sock, buf, sizeof(buf)) < 0) { /* Receive message */
     perror("read"); exit(1);
   }
   printf("Read string: %s\n", buf);
} while (strcmp(buf, "dne") != 0); /* Finish on "end" message */
close(sock); /* Close socket */
exit(0);
}
```

Παράδειγμα Εκτέλεσης (1 server / 1 client)

Terminal 1 cs4122.in.cs.ucy.ac.cy

```
bash-3.1$ gcc server.c -o server
bash-2.05b$ ./server 30000
Listening for connections to port 30000
Accepted connection from cs4122.in.cs.ucy.ac.cy
Read string: Test no 1
Read string: test no 2
Read string: test no 3
Read string: end
^C
bash-3.1$
```

cs4122.in.cs.ucy.ac.cy Terminal 2

```
bash-3.1$ gcc client.c -o client
bash-3.1$ ./client localhost 30000
Requested connection to host localhost port 30000
Give input string: Test no 1
Read string: 1 on tseT
Give input string: test no 2
Read string: 2 on tset
Give input string: test no 3
Read string: 3 on tset
Give input string: end
Read string: dne
bash-3.1$
```

Παράδειγμα Εκτέλεσης (1 server / 2 clients)

astarti.cs.ucy.ac.cy

```
bash-2.05b$ ./server 30000
Listening for connections to port 30000
Accepted connection from cs4122.in.cs.ucy.ac.cy
Read string: Test no 1
Read string: Test no 2
Read string: Test no 3
Read string: end
Accepted connection from cs4042.in.cs.ucy.ac.cy
Read string: message 1
Read string: message 2
Read string: end
^C
bash-2.05b$
```

cs4122.in.cs.ucy.ac.cy

```
bash-3.1$ ./client
      astarti.cs.ucy.ac.cy
      30000
Requested connection to host
      astarti.cs.ucy.ac.cy port
      30000
Give input string: Test no 1
Read string: 1 on tseT
Give input string: Test no 2
Read string: 2 on tseT
Give input string: Test no 3
Read string: 3 on tseT
Give input string: end
Read string: dne
bash-3.1$
```

cs4042.in.cs.ucy.ac.cy

```
bash-3.1$ ./client
      astarti.cs.ucy.ac.cy
      30000
Requested connection to host
      astarti.cs.ucy.ac.cy port
      30000
Give input string: message 1
Read string: 1 egassem
Give input string: message 2
Read string: 2 egassem
Give input string: end
Read string: dne
```

Επιπρόσθετες Συναρτήσεις (gethostbyname)

DNS Resolution : DNS name => IP address

```
#include <netdb.h>
```

struct hostent *gethostbyname(char *name);

Επιστρέφει ένα δείκτη σε δομή struct *hostent* σε περίπτωση επιτυχίας ή ένα *NULL* δείκτη σε περίπτωση λάθους

```
struct hostent {
    char *h_name;          /* name of the host. */
    char **h_aliases;      /* pointer to alternate host name array. */
    int h_addrtype;        /* The type of address being returned; usually AF_INET.*/
    int h_length;          /* The length of the address in bytes. */
    char **h_addr_list;    /* pointer to array of network addresses for the host.
                                  Host addresses are in Network Byte Order. */
};
#define h_addr h_addr_list[0]  /* The first address in h_addr_list. */
// Ένα host μπορεί να έχει πολλαπλά IP addresses.
```

nslookup www.google.com

Επιπρόσθετες Συναρτήσεις (gethostbyname)

Π.χ., ο client κάνει κάτι όπως το ακόλουθο:

```
struct hostent *rem;
if ((rem = gethostbyname("astarti.cs.ucy.ac.cy")) == NULL) {
    // Σε περίπτωση λάθους στο Address Resolution της gethostbyname() , μπορούμε
    να καλέσουμε την herror(),δείχνοντας το συγκεκριμένο λάθος στη σφαιρική
    μεταβλητή h_errno (#include <netdb.h>, void herror(const char *s);
    herror("gethostbyname"); exit(1);
}
// αντιγραφή πρωτου resolved ονόματος στο sin_addr struct
bcopy((char *) rem -> h_addr,  (char *) &server.sin_addr, rem -> h_length);
server.sin_family = PF_INET;  /* Internet domain */
server.sin_port = htons(port); /*Server's Internet address and port*/
serverptr = (struct sockaddr *) &server;
Connect()...
```

Επιπρόσθετες βοηθητικές συναρτήσεις (συνέχεια)

Reverse DNS Resolution : IP => DNS Name

#include <netdb.h>

struct hostent *gethostbyaddr(char *addr, int len, int type);

Επιστρέφει ένα δείκτη σε δομή struct *hostent* σε περίπτωση επιτυχίας ή ένα *NULL* δείκτη σε περίπτωση λάθους

•Επιστρέφει ένα δείκτη σε δομή struct *hostent* για ένα υπολογιστή, που δεδομένης της διεύθυνσης του *addr*, του μεγέθους της *len* και του είδους της *type* (AF_INET), στο πεδίο *h_name* της επιστρεφόμενης δομής τοποθετείται το όνομα του υπολογιστή.

Π.χ.,

```
struct hostent *rem;

/* Find client's address */
if ((rem = gethostbyaddr((char *) &client.sin_addr.s_addr,
                         sizeof (client.sin_addr.s_addr), client.sin_family)) == NULL) {
        herror("gethostbyaddr"); exit(1);
}

printf("Accepted connection from %s\n", rem -> h_name);
//π.χ., Accepted connection from cs4122.in.cs.ucy.ac.cy
```

Επιπρόσθετες βοηθητικές συναρτήσεις (συνέχεια)

- **server.sin_port = htons(0)** ή **client.sin_port = htons(0);** μας δίνει 1 αριθμό θύρας (τον επόμενο διαθέσιμο)
- Πως μπορεί τώρα η εφαρμογή να βρεί την πραγματική θύρα που διατέθηκε από το σύστημα για τη σύνδεση;
- Κλήση συστήματος *getsockname()*

```
#include <socket.h>
int getsockname(int sockfd, struct sockaddr *address,
    int *addresslen);
```

Επιστρέφει 0 σε επιτυχία, ή -1 σε λάθος

- Επιστρέφει στη δομή **address* τη διεύθυνση με την οποία έχει συνδεθεί η υποδοχή που αντιστοιχεί στον περιγραφέα υποδοχής και στο **addresslen* το μέγεθος της διεύθυνσης αυτής.

Επιπρόσθετες βοηθητικές συναρτήσεις (συνέχεια)

- Κλήση συστήματος *setsockopt()*

#include <socket.h>

int setsockopt(int sockfd, int level, int option, const void *val, socklen_t len);

Επιστρέφει 0 σε επιτυχία, ή -1 σε λάθος

- Ρύθμιση επιλογών σχετικά με συγκεκριμένη υποδοχή.
- Πολύ χρήσιμη στην περίπτωση που ενώ μια διεργασία κλείσει μια θύρα, αυτή δε μπορεί να ξαναχρησιμοποιηθεί (να ξανασυνδεθεί - rebind) για κάποιο χρονικό διάστημα, εξαιτίας ίσως κάποιου μέρους της υποδοχής που παραμένει στον πυρήνα για κάποιο λόγο.
- Αυτό το πρόβλημα μπορεί να λυθεί ως ακολούθως:

```
#include <sys/types.h>
#include <sys/socket.h>
int yes=1;
if (setsockopt(server,
        SOL_SOCKET,SO_REUSEADDR,&yes,sizeof(int)) == -1) {
   perror("setsockopt"); exit(1); }
//
   http://beej.us/guide/bgnet/output/html/multipage/setsockoptman.ht
   ml
```