ΥΠΟΔΟΧΕΙΣ (SOCKETS)

Λειτουργικά Συστήματα - Εργαστήριο

Πάνος Παπαδόπουλος

Τμήμα Πληροφορικής με Εφαρμογές στη Βιοϊατρική
Πανεπιστήμιο Θεσσαλίας

ΘΕΩΡΙΑ

ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ

- Τα **sockets** είναι ειδικές **δομές** που χρησιμοποιούνται για την **επικοινωνία μεταξύ διεργασιών/εφαρμογών**:
  - τοπικά, δηλαδή στον ίδιο υπολογιστή (**Unix sockets**)
  - μέσω internet (**Internet sockets**)
- Έχουμε δύο βασικούς **τύπους** από sockets:
  - **datagram sockets** : μετάδοση διακριτών μηνυμάτων
  - **stream sockets** : μετάδοση μιας ροής δεδομένων
- Πρέπει να ορίσουμε το **πρωτόκολλο επικοινωνίας** που θα χρησιμοποιηθεί μαζί με το socket.

UNIX SOCKETS

- #include <sys/socket.h>
  #include <sys/un.h>

```
1 struct  sockaddr_un {
2     sa_family_t sun_family;      /* AF_UNIX */
3     char        sun_path[108];   /* pathname */
4 };
```

INTERNET SOCKETS

- #include <sys/socket.h>
  #include <netinet/in.h>
  #include <arpa/inet.h>

```
1  struct sockaddr_in {
2      sa_family_t    sin_family;    /* address family: AF_INET */
3      in_port_t      sin_port;      /* port in network byte order */
4      struct in_addr sin_addr;      /* internet address */
5  };
6
7  /* Internet address. */
8  struct in_addr {
9      uint32_t       s_addr;        /* address in network byte order */
10 };
```

SOCKET() FUNCTION (1/2)

- #include <sys/types.h>
  #include <sys/socket.h>
- Prototype : int socket(int domain, int type, int protocol)
- **Δημιουργία** του socket.
- **Επιστρέφει** έναν **descriptor**.
- Η παράμετρος **domain** καθορίζει τον **τομέα επικοινωνίας**:
  - **AF_UNIX** : επικοινωνία τοπικά
  - **AF_INET** : επικοινωνία μέσω internet

SOCKET() FUNCTION (2/2)

- Η παράμετρος **type** καθορίζει τον **τύπο** του socket:
  - **SOCK_DGRAM** : datagram socket
  - **SOCK_STREAM** : stream socket
- Η παράμετρος **protocol** καθορίζει το **πρωτόκολλο επικοινωνίας** που θα χρησιμοποιηθεί μαζί με το socket.
- Συνήθως μόνο ένα πρωτόκολλο υπάρχει για να υποστηρίξει έναν συγκεκριμένο τύπο socket εντός μιας δεδομένης οικογένειας πρωτοκόλλων, οπότε το πρωτόκολλο μπορεί να οριστεί ως **0**.

BIND() FUNCTION

- #include <sys/types.h>
  #include <sys/socket.h>
- Prototype : int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen)
- **Συσχετίζει** ένα **socket** με μία **local address**.
- Αν η εκτέλεση **πετύχει** επιστρέφει **0**, αλλιώς επιστρέφει -1.
- Η παράμετρος **sockfd** είναι ο descriptor του socket που δημιουργήθηκε από τη συνάρτηση **socket()**.
- Η παράμετρος **addr** είναι ένας pointer σε ένα **struct sockaddr** (π.χ. struct sockaddr_un ή struct sockaddr_in).
- Η παράμετρος **addrlen** είναι το **μέγεθος (σε bytes)** της δομής στην οποία δείχνει το addr.

DATAGRAM SOCKETS

SENDTO() FUNCTION

- #include <sys/types.h>
  #include <sys/socket.h>
- Prototype: ssize_t sendto(int sockfd, const void *buf, size_t len, int flags, const struct sockaddr *dest_addr, socklen_t addrlen)
- **Στέλνει** το μήνυμα **buff** μεγέθους **len** στη διεύθυνση **dest_addr**.
- Αν η εκτέλεση **πετύχει** επιστρέφει **το πλήθος των χαρακτήρων που έχουν σταλεί**, αλλιώς επιστρέφει -1.
- Η παράμετρος **sockfd** είναι ο descriptor του socket που δημιουργήθηκε από τη συνάρτηση **socket()**.
- Η παράμετρος **dest_addr** είναι ένας pointer σε ένα **struct sockaddr** (π.χ. struct sockaddr_un ή struct sockaddr_in).
- Η παράμετρος **addrlen** είναι το **μέγεθος (σε bytes)** της δομής στην οποία δείχνει το dest_addr.

RECVFROM() FUNCTION

- #include <sys/types.h>
  #include <sys/socket.h>
- Prototype: ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags, struct sockaddr *src_addr, socklen_t *addrlen)
- Μπλοκάρει μέχρι να παραληφθεί το επόμενο μήνυμα.
- Το μήνυμα αποθηκεύεται στη παράμετρο **buf** και έχει μέγεθος **len**.
- Η παράμετρος **sockfd** είναι ο descriptor του socket που δημιουργήθηκε από τη συνάρτηση **socket()**.
- Στην παράμετρο **src_addr** αποθηκεύεται η διεύθυνση του αποστολέα.
- Η παράμετρος **addrlen** είναι το **μέγεθος (σε bytes)** της δομής στην οποία δείχνει το src_addr.

STREAM SOCKETS

LISTEN() FUNCTION

- #include <sys/types.h>
  #include <sys/socket.h>
- Prototype: int listen(int sockfd, int backlog)
- Σηματοδοτεί το socket που αναφέρεται από το **sockfd** ως **passive socket**, δηλαδή ως socket που θα χρησιμοποιηθεί για την **αποδοχή αιτήσεων σύνδεσης** (χρησιμοποιώντας την accept()).
- Αν η εκτέλεση **πετύχει** επιστρέφει **0**, αλλιώς επιστρέφει -1.
- Η παράμετρος **sockfd** είναι ένας **descriptor** που αναφέρεται σε ένα socket τύπου **SOCK_STREAM**.
- Η παράμετρος **backlog** ορίζει το **μέγιστο μήκος** της ουράς των εκκρεμών αιτήσεων σύνδεσης.

ACCEPT() FUNCTION (1/2)

- #include <sys/types.h>
  #include <sys/socket.h>
- Prototype: int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen)
- **Εξάγει** την πρώτη αίτηση σύνδεσης από την ουρά των εκκρεμών αιτήσεων σύνδεσης για το **passive socket** που αναφέρεται απο το **sockfd**.
- **Δημιουργεί** ένα νέο socket.
- **Επιστρέφει** τον **descriptor** του νέου socket που δημιουργήθηκε.
- Το νέο socket **δεν** ειναι σε **listening mode**.
- Το αρχικό socket **sockfd δεν αλλάζει** μετά την κλήση της accept().

ACCEPT() FUNCTION (2/2)

- Η παράμετρος **sockfd** είναι το socket που:
  - **Δημιουργήθηκε** από τη συνάρτηση **socket()**.
  - **Συσχετίστηκε** με μία **local address** χρησιμοποιώντας την συνάρτηση **bind()**.
  - **Περιμένει** για **αιτήσεις σύνδεσης** μετά τη χρήση της συνάρτησης **listen()**.
- Η παράμετρος **addr** είναι ένας pointer σε ένα **struct sockaddr** (π.χ. struct sockaddr_un ή struct sockaddr_in).
- Η παράμετρος **addrlen** είναι το **μέγεθος (σε bytes)** της δομής στην οποία δείχνει το addr.

CONNECT() FUNCTION

- #include <sys/types.h>
  #include <sys/socket.h>
- Prototype: int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen)
- **Συνδέει** το socket που αναφέρεται απο το sockfd στη διεύθυνση που καθορίζεται από το addr.
- Αν η εκτέλεση **πετύχει** επιστρέφει **0**, αλλιώς επιστρέφει -1.
- Η παράμετρος **sockfd** είναι ο descriptor του socket που δημιουργήθηκε από τη συνάρτηση **socket()**.
- Η παράμετρος **addr** είναι ένας pointer σε ένα **struct sockaddr** (π.χ. struct sockaddr_un ή struct sockaddr_in).
- Η παράμετρος **addrlen** είναι το **μέγεθος (σε bytes)** της δομής στην οποία δείχνει το addr.

SEND() FUNCTION

- #include <sys/types.h>
  #include <sys/socket.h>
- Prototype: ssize_t send(int sockfd, const void *buf, size_t len, int flags)
- **Στέλνει** το μήνυμα **buff** μεγέθους **len**.
- Αν η εκτέλεση **πετύχει** επιστρέφει **το πλήθος των χαρακτήρων που έχουν σταλεί**, αλλιώς επιστρέφει -1.
- Η παράμετρος **sockfd** είναι ο descriptor του socket που δημιουργήθηκε από τη συνάρτηση **socket()**.
- Εναλλακτικά μπορούμε να καλέσουμε την συνάρτηση **sendto()** με **dest_addr=NULL** και **addrlen=0**.

RECV() FUNCTION

- #include <sys/types.h>
  #include <sys/socket.h>
- Prototype: ssize_t recv(int sockfd, void *buf, size_t len, int flags)
- Μπλοκάρει μέχρι να παραληφθεί το επόμενο μήνυμα.
- Το μήνυμα αποθηκεύεται στη παράμετρο **buf** και έχει μέγεθος **len**.
- Η παράμετρος **sockfd** είναι ο descriptor του socket που δημιουργήθηκε από τη συνάρτηση **socket()**.
- Εναλλακτικά μπορούμε να καλέσουμε την συνάρτηση **recvfrom()** με **src_addr=NULL** και **addrlen=NULL**.

UNIX DATAGRAM SOCKET - SERVER (1/2)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>

#define SIZE 1024

int main(int argc, char *argv[])
{
    int sock;
    char msg[SIZE];
    struct sockaddr_un addr; // Unix socket
    socklen_t addrlen;

    addr.sun_family = AF_UNIX; // Unix socket
    strcpy(addr.sun_path, argv[1]);
    addrlen = sizeof(addr.sun_family) + strlen(addr.sun_path) + 1;
```

UNIX DATAGRAM SOCKET - SERVER (2/2)

```
21      sock = socket(AF_UNIX, SOCK_DGRAM, 0); // Datagram socket
22      bind(sock, (struct sockaddr *)&addr, addrlen);
23
24      strcpy(msg, "");
25      while (strcmp(msg, "exit") != 0) {
26          printf("\nwaiting ...\ n");
27          recvfrom(sock, msg, SIZE, 0, (struct sockaddr *)&addr, &addrlen);
28
29          if (strcmp(msg, "exit") != 0) {
30              printf("running command : [%s]\n", msg);
31              system(msg);
32          }
33      }
34
35      close(sock);
36      unlink(argv[1]);
37
38      return 0;
39  }
```

UNIX DATAGRAM SOCKET - CLIENT (1/2)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>

#define SIZE 1024

int main(int argc, char *argv[])
{
    int sock, i;
    char msg[SIZE];
    struct sockaddr_un addr; // Unix socket
    socklen_t addrlen;

    addr.sun_family = AF_UNIX; // Unix socket
    strcpy(addr.sun_path, argv[1]);
    addrlen = sizeof(addr.sun_family) + strlen(addr.sun_path) + 1;
```

UNIX DATAGRAM SOCKET - CLIENT (2/2)

```
21      sock = socket(AF_UNIX, SOCK_DGRAM, 0); // Datagram socket
22
23      strcpy(msg, "");
24      for (i = 2; i < argc; i++) {
25          strcat(msg, argv[i]);
26          strcat(msg, " ");
27      }
28      msg[strlen(msg)-1] = '\0'; // Remove the last space
29
30      printf("sending command : %s\n", msg);
31
32      sendto(sock, msg, strlen(msg) + 1, 0,(struct sockaddr *)&addr, addrlen);
33
34      close(sock);
35
36      return 0;
37  }
```

UNIX STREAM SOCKET - SERVER (1/3)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>

#define SIZE 1024

int main(int argc, char *argv[])
{
    int psock, sock;
    char msg[SIZE];
    struct sockaddr_un addr; // Unix socket
    socklen_t addrlen;

    addr.sun_family = AF_UNIX; // Unix socket
    strcpy(addr.sun_path, argv[1]);
    addrlen = sizeof(addr.sun_family) + strlen(addr.sun_path) + 1;
```

UNIX STREAM SOCKET - SERVER (2/3)

```
21      psock = socket(AF_UNIX, SOCK_STREAM, 0); // Stream socket
22      bind(psock, (struct sockaddr *)&addr, addrlen);
23      listen(psock, 1);
24
25      strcpy(msg, "");
26      while (strcmp(msg, "exit") != 0) {
27          printf("\nwaiting for new connection...\n");
28
29          sock = accept(psock, NULL, NULL);
30
31          printf("connection established...\n");
32
33          recv(sock, msg, SIZE, 0); // recvfrom(sock, msg, SIZE, 0, NULL, NULL);
34
35          if (strcmp(msg, "exit") != 0) {
36              printf("running command : [%s]\n", msg);
37              system(msg);
38          }
```

UNIX STREAM SOCKET - SERVER (3/3)

```
39              close(sock);
40              printf("\nconnection closed...\n");
41          }
42
43          close(psock);
44          unlink(argv[1]);
45
46          return 0;
47      }
```

UNIX STREAM SOCKET - CLIENT (1/2)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>

#define SIZE 1024

int main(int argc, char *argv[])
{
    int sock, i;
    char msg[SIZE];
    struct sockaddr_un addr; // Unix socket
    socklen_t addrlen;

    addr.sun_family = AF_UNIX; // Unix socket
    strcpy(addr.sun_path, argv[1]);
    addrlen = sizeof(addr.sun_family) + strlen(addr.sun_path) + 1;
```

UNIX STREAM SOCKET - CLIENT (2/2)

```
21      sock = socket(AF_UNIX, SOCK_STREAM, 0); // Stream socket
22      connect(sock, (struct sockaddr *)&addr, addrlen);
23
24      strcpy(msg, "");
25      for (i = 2; i < argc; i++) {
26          strcat(msg, argv[i]);
27          strcat(msg, " ");
28      }
29      msg[strlen(msg)-1] = '\0'; // Remove the last space
30
31      printf("sending command : %s\n", msg);
32
33      send(sock, msg, strlen(msg)+1, 0);
34      // sendto(sock, msg, strlen(msg) + 1, 0, NULL, 0);
35
36      close(sock);
37
38      return 0;
39  }
```

INTERNET DATAGRAM SOCKET (UDP/IP) - SERVER (1/2)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define SIZE 1024

int main(int argc, char *argv[])
{
    int sock;
    char msg[SIZE];
    struct sockaddr_in addr; // Internet socket

    addr.sin_family = AF_INET; // Internet socket
    addr.sin_addr.s_addr = htonl(INADDR_ANY); // All interfaces
    addr.sin_port = htons(atoi(argv[1]));
```

INTERNET DATAGRAM SOCKET (UDP/IP) - SERVER (2/2)

```
20      sock = socket(AF_INET, SOCK_DGRAM, 0); // Datagram socket
21      bind(sock, (struct sockaddr *)&addr, sizeof(addr));
22
23      strcpy(msg, "");
24      while (strcmp(msg, "exit") != 0) {
25          printf("\nwaiting for data (port : %s) ...\ n", argv[1]);
26          recvfrom(sock, msg, SIZE, 0, NULL, NULL);
27
28          if (strcmp(msg, "exit") != 0) {
29              printf("running command : [%s]\n", msg);
30              system(msg);
31          }
32      }
33
34      close(sock);
35      unlink(argv[1]);
36
37      return 0;
38  }
```

INTERNET DATAGRAM SOCKET (UDP/IP) - CLIENT (1/2)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define SIZE 1024

int main(int argc, char *argv[])
{
    int sock, i;
    char msg[SIZE];
    struct sockaddr_in addr; // Internet socket

    addr.sin_family = AF_INET; // Internet socket
    inet_aton(argv[1], &addr.sin_addr);
    addr.sin_port = htons(atoi(argv[2]));
```

INTERNET DATAGRAM SOCKET (UDP/IP) - CLIENT (2/2)

```
21      sock = socket(AF_INET, SOCK_DGRAM, 0); // Datagram socket
22
23      strcpy(msg, "");
24      for (i = 3; i < argc; i++) {
25          strcat(msg, argv[i]);
26          strcat(msg, " ");
27      }
28      msg[strlen(msg)-1] = '\0'; // Remove the last space
29
30      printf("sending command : %s\n", msg);
31
32      sendto(sock, msg, strlen(msg) + 1, 0,(struct sockaddr *)&addr, sizeof(addr));
33
34      close(sock);
35
36      return 0;
37  }
```

INTERNET STREAM SOCKET (TCP/IP) - SERVER (1/3)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define SIZE 1024

int main(int argc, char *argv[])
{
    int psock, sock;
    char msg[SIZE];
    struct sockaddr_in addr; // Internet socket

    addr.sin_family = AF_INET; // Internet socket
    addr.sin_addr.s_addr = htonl(INADDR_ANY); // All interfaces
    addr.sin_port = htons(atoi(argv[1]));
```

INTERNET STREAM SOCKET (TCP/IP) - SERVER(2/3)

```
20      psock = socket(AF_INET, SOCK_STREAM, 0); // Stream socket
21      bind(psock, (struct sockaddr *)&addr, sizeof(addr));
22      listen(psock, 1);
23
24      strcpy(msg, "");
25      while (strcmp(msg, "exit") != 0) {
26          printf("\nwaiting for new connection (port : %s) ...\n", argv[1]);
27
28          sock = accept(psock, NULL, NULL);
29
30          printf("connection established...\n");
31
32          recv(sock, msg, SIZE, 0);
33          // recvfrom(sock, msg, SIZE, 0, NULL, NULL);
34
35          if (strcmp(msg, "exit") != 0) {
36              printf("running command : [%s]\n", msg);
37              system(msg);
38          }
```

INTERNET STREAM SOCKET (TCP/IP) - SERVER(3/3)

```
39              close(sock);
40              printf("\nconnection closed...\n");
41          }
42
43          close(psock);
44          unlink(argv[1]);
45
46          return 0;
47      }
```

INTERNET STREAM SOCKET (TCP/IP) - CLIENT (1/2)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define SIZE 1024

int main(int argc, char *argv[])
{
    int sock, i;
    char msg[SIZE];
    struct sockaddr_in addr; // Internet socket

    addr.sin_family = AF_INET; // Internet socket
    inet_aton(argv[1], &addr.sin_addr);
    addr.sin_port = htons(atoi(argv[2]));
```

INTERNET STREAM SOCKET (TCP/IP) - CLIENT (2/2)

```
21      sock = socket(AF_INET, SOCK_STREAM, 0); // Stream socket
22      connect(sock, (struct  sockaddr *)&addr, sizeof(addr));
23
24      strcpy(msg, "");
25      for  (i  =  3;  i  <  argc;  i++)  {
26          strcat(msg, argv[i]);
27          strcat(msg, "  ");
28      }
29      msg[strlen(msg)-1] = '\0';  // Remove the last space
30
31      printf("sending command : %s\n", msg);
32
33      send(sock, msg, strlen(msg) + 1,  0);
34      // sendto(sock, msg, strlen(msg) + 1,  0,  NULL, 0);
35
36      close(sock);
37
38      return  0;
39  }
```

ΑΣΚΗΣΕΙΣ

ΑΣΚΗΣΕΙΣ

1. Να γραφεί client-server εφαρμογή σε C στην οποία ο client θα στέλνει μια αριθμητική παράσταση στον server, ο server θα την υπολογίζει και θα στέλνει το αποτέλεσμα στον client.

2. Να γραφεί client-server εφαρμογή σε C η οποία θα υλοποιεί έναν απλό chat messenger.