ΕΡΓΑΣΤΗΡΙΟ 8

Υποδοχείς

Να πληκτρολογηθεί και να εκτελεστεί ο κώδικας των επόμενων παραδειγμάτων που επιδεικνύουν τη χρήση των σημάτων ως μέσο επικοινωνία μεταξύ διεργασιών.

Παράδειγμα 1. Σε αυτό το παράδειγμα αρχιτεκτονικής client - server με UNIX sockets, ο server λειτουργεί επαναληπτικά και επιτρέπει τη σύνδεση ενός πελάτη κάθε φορά. Ο πελάτης συνδέεται στο socket του server και αντιγράψει απλά την είσοδό του στον server ο οποίος και την εκτυπώνει.

(αρχεία un-client.c και un-server.c).

Αρχείο un-server.c

```
// Source: https://www.danlj.org/mkj/lad/src/userver.c.html
// (Johnson & Troan, pp.298-299)

#include <stdio.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#include <stdlib.h>

// Copies data from file descriptor 'from' to file descriptor 'to'
//  until nothing is left to be copied. Exits if an error occurs.

void copyData(int from, int to) {
   char buf[1024];
   int amount;
   while ((amount = read(from, buf, sizeof(buf))) > 0) {
     if (write(to, buf, amount) != amount) {
         perror("Message from write:");
         exit(1); }}
   if (amount < 0) {
      perror("Message from read:");
      exit(1); } }

// Waits for a connection on the ./sample-socket Unix domain socket. Once a connection
// has  been established, copy data  from the socket to stdout until the other end closes
// the connection, and then wait for another connection to the socket.

int main(void) {
   struct sockaddr_un address;
   int sock, conn;
   socklen_t addrLength;
   if ((sock = socket(PF_UNIX, SOCK_STREAM, 0)) < 0) {
       perror("Message from socket:");
       exit(1); }
```

```
   /* Remove any preexisting socket (or other file) */
   unlink("./sample-socket");
   address.sun_family = AF_UNIX;          /* Unix domain socket */
   strcpy(address.sun_path, "./sample-socket");

   /* The total length of the address includes the sun_family element */
   addrLength = sizeof(address.sun_family) + strlen(address.sun_path);

   if (bind(sock, (struct sockaddr *) &address, addrLength)){
      perror("Message from bind:");
      exit(1); }

  if (listen(sock, 5)){
      perror("Message from listen:");
      exit(1); }

   while ((conn = accept(sock, (struct sockaddr *) &address, &addrLength)) >= 0) {
      printf("---- getting data\n");
      copyData(conn, 1);
      printf("---- done\n");
      close(conn); }

   if (conn < 0) {
      perror("Message from accept:"); exit(1); }
   close(sock);
```

Αρχείο un-client.c

```
// Source: https://www.danlj.org/mkj/lad/src/userver.c.html
// (Johnson & Troan, pp.298-299)

#include <stdio.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#include <stdlib.h>

// Copies data from file descriptor 'from' to file descriptor 'to until nothing is left to be
// copied. Exits if an error occurs.

void copyData(int from, int to) {
   char buf[1024];
   int amount;
   while ((amount = read(from, buf, sizeof(buf))) > 0) {
     if (write(to, buf, amount) != amount) {
         perror("Message from write:"); exit(1); }}

   if (amount < 0) {
      perror("Message from read:");
      exit(1); } }
```

```
// Connect to the ./sample-socket Unix domain socket, copy stdin into the socket, and exit

int main(void) {
   struct sockaddr_un address;
   int sock;
   socklen_t addrLength;

   if ((sock = socket(PF_UNIX, SOCK_STREAM, 0)) < 0){
      perror("Message from socket:");
      exit(1); }

   address.sun_family = AF_UNIX;    /* Unix domain socket */
   strcpy(address.sun_path, "./sample-socket");

   /* The total length of the address includes the sun_family element */
   addrLength = sizeof(address.sun_family) + strlen(address.sun_path);

   if (connect(sock, (struct sockaddr *) &address, addrLength)){
      perror("Message from connect:");
      exit(1); }

   copyData(0, sock);
   close(sock);
   return 0; }
```

Παράδειγμα 2. Σε αυτό το παράδειγμα αρχιτεκτονικής client – server με Unix sockets, ο server και ο client σχετίζονται με μία σχέση πατέρα (server) – παιδιού (client) που δημιουργείται μέσω της fork και επικοινωνούν στέλνοντας ένα μήνυμα ο ένας στον άλλο. (αρχείο fork-cl-srv.c).

```
#include <stdio.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>

#define SOCKETNAME "MySocket"

int main(void) {
   struct sockaddr_un sa;
   (void)unlink(SOCKETNAME);
   strcpy(sa.sun_path, SOCKETNAME);
   sa.sun_family = AF_UNIX;
   int fd_skt, fd_client; char buf[100];
   int written; ssize_t readb;
```

```
 if (fork() == 0) { /* client */
    if ((fd_skt = socket(PF_UNIX, SOCK_STREAM, 0)) < 0) {
         perror("Message from socket [client] : ");
        exit(-1); }
    printf ("[Client] ==> Socket %d has been created\n", fd_skt);

    // Since cliend and server run concurrently, it is possible for the client
    // to call connect before the creation of the socket file by the server (ENOENT).
    // In this case the function sleeps for a second and tries again. If something
    // else goes wrong, the function fails.

    while (connect(fd_skt, (struct sockaddr *)&sa, sizeof(sa)) == -1) {
       if (errno == ENOENT) {
          sleep(1); continue; }
       else {
        perror("Message from connect [client]");
        exit(-2); }}

    // After connection the client and the server can communicate each other
    // using read and write functions

    printf ("[Client] ==> Connection has been established .. let us write to server\n");
    written = write(fd_skt, "Hello!", 7);
    if (written==-1) {
        perror("Message from write [client]");
        exit(-3); }
    else printf ("[Client] ==> %d bytes written to server\n", written);
    printf ("[Client] ==> Now let us read from server\n");
    readb = read(fd_skt, buf, sizeof(buf));
    if (readb==-1) {
        perror("Message from read [client]");
        exit(-4); }
    else printf ("[Client] ==> %zd bytes read from server\n", readb);
    printf("Client got %s\n", buf);
    close(fd_skt);
    exit(0); }

 else { /* server */
    if ((fd_skt = socket(PF_UNIX, SOCK_STREAM, 0)) < 0) {
        perror("Message from socket [server]");
        exit(-1); }
    printf ("[Server] ==> Socket %d has been created\n", fd_skt);
    if (bind(fd_skt, (struct sockaddr *) &sa, sizeof(sa))){
       perror("Message from bind [server]");
       exit(-2); }
    printf ("[Server] ==> Socket %d has been bound to address\n", fd_skt);
    if (listen(fd_skt, 5)){
       perror("Message from listen [server]");
       exit(-3); }
```

```
    printf ("[Server] ==> Listening for incoming messages\n");
    if ((fd_client = accept(fd_skt, NULL, 0))<0) {
       perror("Message from accept [server]");
       exit(-4); }
    printf ("[Server] ==> let us read from client via socket %d\n", fd_client);
    readb = read(fd_client, buf, sizeof(buf));
    if (readb == -1) {
        perror("Message from read [server] : ");
        exit(-5); }
    printf("Server got %s ==> %zd bytes read from client\n", buf, readb);
    printf ("[Server] ==> let us write to client\n");
    if (write(fd_client, "Goodbye!", 9)==-1) {
        perror("Message from write [server]");
        exit(-6); }
    close(fd_skt);
    close(fd_client);
    exit(0); }}
```

Παράδειγμα 3. Σε αυτό το παράδειγμα αρχιτεκτονικής client – server με TCP / IP sockets, ο client αποστέλλει στον server ένα μήνυμα που παραλαμβάνεται και εκτυπώνεται από αυτόν. (αρχεία server1-tcp.c και client1-tcp.c).

Αρχείο server1-tcp.c

```
#include <stdio.h>
#include <sys/socket.h> // AF_INET and SOCK_STREAM
#include <stdlib.h>    // EXIT_SUCCESS, EXIT_FAILURE
#include <netinet/in.h> // INADDR_ANY
#include <string.h>    // bzero
#include <unistd.h>    // read and write

#define PORT 8080

int main(int argc, char const *argv[]) {
   int server_fd, new_socket, valread;
   struct sockaddr_in address;
   int opt = 1;
   int addrlen = sizeof(address);
   char buffer[1024] = {0};
   char *hello = "Hello from server";

   // Creating socket file descriptor

   if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
      perror("socket failed");
      exit(EXIT_FAILURE); }
```

```
   // Forcefully attaching socket to the port 8080


   if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
                  &opt, sizeof(opt))) {
      perror("setsockopt");
      exit(EXIT_FAILURE); }

   address.sin_family = AF_INET;
   address.sin_addr.s_addr = INADDR_ANY;
   address.sin_port = htons(PORT);

   // Forcefully attaching socket to the port 8080

   if (bind(server_fd, (struct sockaddr *)&address,
            sizeof(address)) < 0) {
      perror("bind failed");
      exit(EXIT_FAILURE); }

   if (listen(server_fd, 3) < 0) {
      perror("listen");
      exit(EXIT_FAILURE); }

   if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                                       (socklen_t*)&addrlen)) < 0) {
      perror("accept");
      exit(EXIT_FAILURE); }
   valread = read (new_socket, buffer, 1024);
   printf("%s\n",buffer );
   return 0; }
```

Αρχείο client1-tcp.c

```
#include <stdio.h>
#include <sys/socket.h> // AF_INET and SOCK_STREAM
#include <arpa/inet.h>  // storage size of serv_addr
#include <unistd.h>     // getpid, getppid, fork
#include <string.h>     // bzero

#define PORT 8080

int main(int argc, char const *argv[]) {
   int sock = 0, valread;
   struct sockaddr_in serv_addr;
   char *hello = "Hello from client";
   char buffer[1024] = {0};

   if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
      printf("\n Socket creation error \n");
      return -1; }

   serv_addr.sin_family = AF_INET;
```

```
   serv_addr.sin_port = htons(PORT);

   // Convert IPv4 and IPv6 addresses from text to binary form

   if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)<=0) {
      printf("\nInvalid address/ Address not supported \n");
      return -1; }

   if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
      printf("\nConnection Failed \n");
      return -1; }

   send(sock , hello , strlen(hello) , 0 );
   printf("Hello message sent\n");
   return 0; }
```

Παράδειγμα 4. Η εφαρμογή πελάτης συνδέεται σε έναν απομακρυσμένο Web server (port 80) η IP διεύθυνση του οποίου δίδεται από το χρήστη και διαβάζει τις 1000 πρώτες γραμμές του κώδικα HTML της κεντρικής σελίδας τις οποίες και εκτυπώνει. (αρχείο inetExample.c).

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

#define REQUEST "GET / HTTP/1.0\r\n\r\n"

int main(int argc, char * argv[]) {
   struct sockaddr_in sa;
   int fd_skt;
   char buf[1000];
   ssize_t nread;
   if (argc!=2) {
     printf ("Usage: inetExample xxx.xxx.xxx.xxx\n");
     exit (-1); }

   sa.sin_family = AF_INET;
   sa.sin_port = htons(80);
   sa.sin_addr.s_addr =  inet_addr(argv[1]);

   if ((fd_skt = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
       perror("Message from socket [client] : ");
       exit(-1); }

   if (connect(fd_skt, (struct sockaddr *)&sa, sizeof(sa)) < 0) {
      printf("\nConnection Failed \n");
      return -1; }
```

```
   write(fd_skt, REQUEST, strlen(REQUEST));
   nread = read(fd_skt, buf, sizeof(buf));
   (void) write(STDOUT_FILENO, buf, nread);
   close(fd_skt);
   exit(0); }
```

Παράδειγμα 5. Ο χρήστης καλεί την ping με μία συμβολική διεύθυνση για να ανακτήσει την πραγματική IP διεύθυνση την οποία στη συνέχεια καταχωρεί ως όρισμα στην εφαρμογή addrInfo. (αρχείο addrInfo.c).

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <netdb.h>

int main (int argc, char * argv[]) {
   int retVal; char err[50];
   struct addrinfo *infop = NULL, hint;
   memset(&hint, 0, sizeof(hint));
   hint.ai_family = AF_INET;
   hint.ai_socktype = SOCK_STREAM;
   if (argc!=2) {
     printf ("Usage: inetExample xxx.xxx.xxx.xxx\n");
     exit (-1); }
   retVal = getaddrinfo(argv[1], "80", &hint, &infop);
   if (retVal!=0) {
      strcpy (err,gai_strerror(retVal));
      printf ("Error found ==> %s\n", err);
      exit (-1); }
   for ( ; infop != NULL; infop = infop->ai_next) {
      struct sockaddr_in *sa = (struct sockaddr_in *)infop->ai_addr;
      printf("%s port: %d protocol: %d\n", inet_ntoa(sa->sin_addr),
           ntohs(sa->sin_port), infop->ai_protocol); }
   exit(0);}
```

Παράδειγμα 6. Ανταλλαγή ενός μηνύματος μεταξύ σταθμών με αυτοδύναμα πακέτα. (αρχεία server2-udp.c και client2-udp.c).

Αρχείο server2-udp.c

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

```
#include <netinet/in.h>

#define PORT    8080
#define MAXLINE 1024

int main() {
   int sockfd, len, n;
   char buffer[MAXLINE];
   char *hello = "Hello from server";
   struct sockaddr_in servaddr, cliaddr;
   if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
      perror("socket creation failed");
      exit(EXIT_FAILURE); }
   memset(&servaddr, 0, sizeof(servaddr));
   memset(&cliaddr, 0, sizeof(cliaddr));
   servaddr.sin_family    = AF_INET; // IPv4
   servaddr.sin_addr.s_addr = INADDR_ANY;
   servaddr.sin_port = htons(PORT);
   if (bind(sockfd, (const struct sockaddr *)&servaddr,
         sizeof(servaddr)) < 0) {
      perror("bind failed");
      exit(EXIT_FAILURE); }
   n = recvfrom (sockfd, (char *)buffer, MAXLINE, MSG_WAITALL,
                  (struct sockaddr *) &cliaddr, &len);
   buffer[n] = '\0';
   printf("Client : %s\n", buffer);
   sendto(sockfd, (const char *)hello, strlen(hello), MSG_CONFIRM,
            (const struct sockaddr *) &cliaddr, len);
   printf("Hello message sent.\n");
   return 0; }
```

<u>Αρχείο client2-udp.c</u>

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

#define PORT     8080
#define MAXLINE 1024

int main() {
   int sockfd, len, n;
   char buffer[MAXLINE];
   char *hello = "Hello from client";
   struct sockaddr_in     servaddr;
   if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
```

```
      perror("socket creation failed");
      exit(EXIT_FAILURE); }

   memset(&servaddr, 0, sizeof(servaddr));
   servaddr.sin_family = AF_INET;
   servaddr.sin_port = htons(PORT);
   servaddr.sin_addr.s_addr = INADDR_ANY;
   sendto(sockfd, (const char *)hello, strlen(hello), MSG_CONFIRM,
              (const struct sockaddr *) &servaddr, sizeof(servaddr));
   printf("Hello message sent.\n");
   n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL,
                   (struct sockaddr *) &servaddr, &len);
   buffer[n] = '\0';
   printf("Server : %s\n", buffer);
   close(sockfd);
   return 0; }
```

Παράδειγμα 7. Σε αυτό το παράδειγμα αρχιτεκτονικής client – server με Unix sockets, η συνάρτηση run_server μέσα από ένα infinite loop καλεί συνεχώς την accept για να συλλάβει αιτήματα σύνδεσης από τέσσερις διεργασίες – πελάτες (που υλοποιούνται από τη συνάρτηση run_client), χρησιμοποιώντας τη select. Στη συνέχεια επικοινωνεί με τον καθέναν από τους πελάτες ανταλλάσσοντας με αυτόν ένα μήνυμα. (αρχείο one-serv-n-cls-unix.c).

```
#include <stdio.h>
#include <sys/socket.h> // AF_INET and SOCK_STREAM
#include <arpa/inet.h>  // storage size of serv_addr
#include <unistd.h>     // getpid, getppid, fork
#include <string.h>     // bzero
#include <stdlib.h>
#include <errno.h>
#include <sys/un.h>

#define SOCKETNAME "MySocket"
```

// Ο κώδικας του server

```
static int run_server(struct sockaddr_un *sap) {
   int fd_skt, fd_client, fd_hwm = 0, fd;
   char buf[100];
   fd_set set, read_set;
   ssize_t nread;
   printf ("Server pid is %d\n", getpid());
   if ((fd_skt = socket(AF_UNIX, SOCK_STREAM, 0)) < 0) {
      printf("\n Socket creation error \n"); return -1; }
   if (bind(fd_skt, (struct sockaddr *)sap, sizeof(*sap)) < 0) {
      perror("Bind"); exit(EXIT_FAILURE); }
   if (listen(fd_skt, SOMAXCONN)<0) {
      perror("Listen"); exit(EXIT_FAILURE); }
   if (fd_skt > fd_hwm) fd_hwm = fd_skt;
   FD_ZERO(&set);
```

```
   FD_SET(fd_skt, &set);
   while (1) {
      read_set = set;
      if (select(fd_hwm+1, &read_set, NULL, NULL, NULL)==-1) {
         perror("Select"); exit(EXIT_FAILURE); }
      for (fd = 0; fd <= fd_hwm; fd++)
         if (FD_ISSET(fd, &read_set)) {
            if (fd == fd_skt) {
               if ((fd_client = accept(fd_skt, NULL, 0))<0) {
                  perror("Select"); exit(EXIT_FAILURE); }
               FD_SET(fd_client, &set);
               if (fd_client > fd_hwm) fd_hwm = fd_client; }
            else {
               if ((nread = read(fd, buf, sizeof(buf)))<0) {
                   perror("Read"); exit(EXIT_FAILURE); }
               if (nread == 0) {
                  FD_CLR(fd, &set);
                  if (fd == fd_hwm) fd_hwm--;
                  close(fd); }
               else {
                  printf("Server got \"%s\"\n", buf);
                  if (write(fd, "Goodbye!", 9)==-1) {
                     perror("Write"); exit(EXIT_FAILURE); }}}}}
   close(fd_skt);
   close(fd_client);
   printf ("Server terminates\n");
   return 1; }
```

// Ο κώδικας του client

```
static int run_client(struct sockaddr_un *sap) {
   if (fork() == 0) {
      int fd_skt;
      char buf[100];
      if ((fd_skt = socket(AF_UNIX, SOCK_STREAM, 0)) < 0) {
          printf("\n Socket creation error \n"); return -1; }
      while (connect(fd_skt, (struct sockaddr *)sap, sizeof(*sap)) == -1) {
         if (errno == ENOENT) {
            sleep(1);
            continue; }
         else {
          perror("Message from connect [client]");
          exit(-2); }}
      snprintf (buf, sizeof(buf), "Hello from %ld!", (long)getpid());
      if (write (fd_skt, buf, strlen(buf)+1) < 0) {
         perror("Write"); exit(EXIT_FAILURE); }
      if ((read(fd_skt, buf, sizeof(buf))) < 0) {
         perror("Read"); exit(EXIT_FAILURE); }
      printf("Client %d got %s\n", getpid(), buf);
      close(fd_skt);
     printf ("Client %d terminates\n", getpid());
```

```
    exit(EXIT_SUCCESS); }
    return 1; }

int main(void) {
   struct sockaddr_un sa;
   int nclient;
   (void)unlink(SOCKETNAME);
   strcpy(sa.sun_path, SOCKETNAME);
   sa.sun_family = AF_UNIX;
   for (nclient = 1; nclient <= 4; nclient++)
       run_client(&sa);
   run_server(&sa);
   exit(EXIT_SUCCESS); }
```