Esercitazione di Reti degli elaboratori

Prof.ssa Chiara Petrioli

SAPIENZA
UNIVERSITÀ DI ROMA

Corso di C

Christian Cardia, Gabriele Saturni

Esercitazione

Socket Programming

Esercizio 1

- Si scrivano due programmi, **client** e **server** che usano una socket TCP
- ***CLIENT->***tramite un ciclo infinito, permette all'utente di inserire un numero intero che invierà al server. A questo punto resta in attesa di una risposta dal server che stamperà sullo schermo. Se l'utente, anziché un numero, inserisce 'exit', il client dopo averlo inviato al server chiude il programma.
- ***SERVER->***tramite un ciclo infinito legge il messaggio ricevuto dal client. Se è un numero, invia al server la somma e il prodotto di tutti i numeri ricevuti. Se è 'exit', chiude il programma senza rispondere al client.
- *N.B. Si assuma che l'utente inserisca solamente un intero o la parola exit*

Consiglio…

- *Dato un array di char **buff** e un intero **i**:*
- *Per salvare il valore contenuto nella variabile intera **i,** nel buffer **buff**, si può utilizzare:* **sprintf (buff, "%d" , i);**
- *Per salvare un eventuale numero in **buff** all'interno della variabile intera **i**, si può utilizzare: sscanf (buff, "%d" , &i);*

Soluzione Ex 1 – Server (1/2)

```c
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define BUFF_SIZE 80
#define PORT 5000
#define SA struct sockaddr

void funzione1(int sockfd);
int main()
{
    int sockfd, connfd, len;
    struct sockaddr_in servaddr, cli;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("Creazione socket fallita...\n");
        exit(0);
    }
    else
        printf("Socket creata correttamente..\n");
    bzero(&servaddr, sizeof(servaddr));
    // Ip e porta
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);
    if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
        printf("socket bind failed...\n");
        exit(0);
    }
    if ((listen(sockfd, 5)) != 0) {
        printf("Errore...\n");
        exit(0);
    }
    else
        printf("In attesa...\n");
    len = sizeof(cli);
    connfd = accept(sockfd, (SA*)&cli, &len);
    if (connfd < 0) {
        printf("Errore....\n");
        exit(0);
    }
    else
        printf("Client connesso...\n");
    funzione1(connfd);
    close(sockfd);
}
```

Soluzione Ex 1 – Server (2/2)

```c
52  void funzione1(int sockfd)
53  {
54      char buff[BUFF_SIZE];
55      int somma = 0, prodotto = 1;
56      // ciclo infinito
57      for (;;) {
58          printf("\nIn attesa messaggio dal client...\n");
59          bzero(buff, BUFF_SIZE);
60          read(sockfd, buff, sizeof(buff));
61          printf("\nMessaggio dal client: %s\n", buff);
62          if (strncmp("exit", buff, 4) == 0) {
63              printf("Ricevuto Exit...\n");
64              break;
65          }
66          int i = 0;
67          //salva il valore intero in buff nella variabile intera i
68          sscanf(buff, "%d", &i);
69          //calcolo la somma e il prodotto
70          somma += i;
71          prodotto *= i;
72          //dichiaro un secondo buffer per salvare somma e prodotto
73          char buff2[BUFF_SIZE];
74          bzero(buff, BUFF_SIZE);
75          bzero(buff2, BUFF_SIZE);
76          //preparo la stringa da inviare al client...
77          strcat(buff, "La somma è ");
78          //memorizzo in buff2 l'intero somma
79          sprintf(buff2, "%d",somma);
80          strcat(buff, buff2);
81          strcat(buff, " Il prodotto è ");
82          //memorizzo in buff2 l'intero prodotto
83          sprintf(buff2, "%d",prodotto);
84          strcat(buff, buff2);
85          //invio la stringa al client
86          write(sockfd, buff, sizeof(buff));
87      }
88  }
```

Soluzione Ex 1 – Client (1/2)

```c
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 80
#define PORT 5000
#define SA struct sockaddr

void func(int sockfd);
int main()
{
    int sockfd, connfd;
    struct sockaddr_in servaddr, cli;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket creata correttamente..\n");
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);
    if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
        printf("Connessione con il server fallita...\n");
        exit(0);
    }
    else
        printf("Connesso con il server..\n");

    func(sockfd);
    close(sockfd);
}
```

Soluzione Ex 1 – Client (2/2)

```c
void func(int sockfd)
{
    char buff[MAX];
    int n;
    for (;;) {
        bzero(buff, sizeof(buff));
        printf("Input : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n');
        write(sockfd, buff, sizeof(buff));

        if ((strncmp(buff, "exit", 4)) == 0) {
            printf("Chiusura del programma...\n");
            return;
        }
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("Dal server : %s\n", buff);
    }
}
```

Esercizio 2

Si scrivano due programmi **client** e **server** che usano una Socket UDP.

SERVER-> tramite un ciclo infinito resta in attesa di ricevere un messaggio da un client. Quando lo riceve, risponde allo stesso client con il seguente messaggio:

*Client numero: #, messaggio ricevuto: **

Dove # è un numero progressivo che viene incrementato ogni volta che si riceve un nuovo messaggio e * è il messaggio ricevuto dal client.

Se riceve la parola ‘exit’ chiude il programma senza rispondere al client.

CLIENT-> deve permettere all’utente di inserire in input da tastiera un messaggio. Se l’utente inserisce ‘exit’ il client, dopo averlo inviato al server, si chiude senza attendere. Se la parola non è ‘exit’ il client lo invia al server e resta in attesa di ricevere un messaggio che stamperà a video. Una volta ricevuta la risposta, il client deve terminare.

Eseguire più volte il client.

Soluzione Ex 2 – Server (1/2)

```c
#include<stdio.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netdb.h>
#include<string.h>
#include<stdlib.h>
#define MAX 80
#define PORT 4000
#define SA struct sockaddr

void func(int sockfd);

int main(){
    int sockfd;
    struct sockaddr_in servaddr;
    sockfd=socket(AF_INET,SOCK_DGRAM,0);
    if(sockfd==-1){
        printf("Creazione socket fallita...\n");
        exit(0);
    }else
        printf("Socket creata correttamente..\n");
    bzero(&servaddr,sizeof(servaddr));
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
    servaddr.sin_port=htons(PORT);
    if((bind(sockfd,(SA *)&servaddr,sizeof(servaddr)))!=0){
        printf("socket bind failed...\n");
        exit(0);
    }else
        printf("Socket successfully binded..\n");

    func(sockfd);
    close(sockfd);
}
```

Soluzione Ex 2 – Server (2/2)

```c
void func(int sockfd)
{
    int numeroClient = 0;
    char buff[MAX], buff2[MAX];
    int n,clen;
    struct sockaddr_in cli;
    clen=sizeof(cli);
    for(;;)
    {
        numeroClient++;
        bzero(buff,MAX);
        //in attesa di ricevere un messaggio da un client
        recvfrom(sockfd,buff,sizeof(buff),0,(SA *)&cli,&clen);
        if(strncmp("exit",buff,4)==0){
            printf("Richiesta di exit dal client...\n");
            break;
        }
        printf("\nClient numero: %d, messaggio ricevuto: %s\n",numeroClient,buff);
        bzero(buff,MAX);
        //preparo il messaggio di risposta al client
        bzero(buff2,MAX);
        sprintf(buff2, "%d",numeroClient);
        strcat(buff,"Client numero: ");
        strcat(buff,buff2);
        //invio il messaggio
        sendto(sockfd,buff,sizeof(buff),0,(SA *)&cli,clen);
    }
}
```

Soluzione Ex 2 – Client (1/1)

```c
#include<sys/socket.h>
#include<netdb.h>
#include<string.h>
#include<stdlib.h>
#include<stdio.h>
#define MAX 80
#define PORT 4000
#define SA struct sockaddr
int main()
{
    char buff[MAX];
    int sockfd,len,n;
    struct sockaddr_in servaddr;
    sockfd=socket(AF_INET,SOCK_DGRAM,0);
    if(sockfd==-1){
        printf("Creazione socket fallita...\n");
        exit(0);
    }else
        printf("Socket creata correttamente..\n");
    bzero(&servaddr,sizeof(len));
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
    servaddr.sin_port=htons(PORT);
    len=sizeof(servaddr);

    printf("\nMessaggio per il server: ");
    n=0;
    while((buff[n++]=getchar())!='\n');
    sendto(sockfd,buff,sizeof(buff),0,(SA *)&servaddr,len);
    if(strncmp("exit",buff,4)==0){
         printf("Richiesta di exit...\n");
    }else{
       bzero(buff,sizeof(buff));
       recvfrom(sockfd,buff,sizeof(buff),0,(SA *)&servaddr,&len);
       printf("Risposta dal server: %s\n",buff);
    }

    printf("\nChiusura programma...\n");
    close(sockfd);

    return 0;
}
```

Esercizio 3

Si scrivano due programmi **client** e **server** che usano una Socket TCP.

SERVER-> deve gestire una lista dove ogni elemento è composto da un Array di char di 100 elementi.

Tramite un ciclo infinito resta in attesa di ricevere un messaggio da un client. Se il messaggio contiene ‘exit’ chiude il programma. Se contiene ‘lista’, deve inviare al client tutti gli elementi della lista. Se il messaggio non è ne ‘exit’ e ne ‘lista’ allora deve aggiungere un nuovo elemento alla lista contenente il messaggio ricevuto dal client.

CLIENT-> Tramite un ciclo infinito deve permettere all’utente di inserire un messaggio da inviare al server. Se in input si ha ‘exit’, dopo averlo inviato al server chiude il programma. Se in input si ha la parola ‘lista’, dopo averla inviata al server resta in attesa di ricevere un messaggio contenente tutti gli elementi della lista. Se l’utente inserisce una cosa differente, il messaggio viene inviato al server per permettergli di aggiungerlo alla lista.

Soluzione Ex 3 – Server (1/3)

```c
#include<stdio.h>
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#define BUFF_SIZE 100
#define PORT 3000
#define SA struct sockaddr

typedef struct Elemento{
    char valore[BUFF_SIZE];
    struct Elemento *next;
}elemento;
//prototipi delle funzioni
void funzione1(int sockfd);
elemento *aggiungiElemento(elemento *ptr, char buffer[]);

int main()
{
    int sockfd, connfd, len;
    struct sockaddr_in servaddr, cli;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("Creazione socket fallita...\n");
        exit(0);
    }else
        printf("Socket creata correttamente..\n");
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);
    if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
        printf("socket bind failed...\n");
        exit(0);
    }
    if ((listen(sockfd, 5)) != 0) {
        printf("Errore...\n");
        exit(0);
    }else
        printf("In attesa...\n");
    len = sizeof(cli);
    connfd = accept(sockfd, (SA*)&cli, &len);
    if (connfd < 0) {
        printf("Errore....\n");
        exit(0);
    }else
        printf("Client connesso...\n");
    funzione1(connfd);
    close(sockfd);
}
```

Soluzione Ex 3 – Server (2/3)

```c
elemento *aggiungiElemento(elemento *ptr, char buffer[]){
    if (ptr == NULL){
      //la lista è vuota
      ptr = (elemento *)malloc(sizeof(elemento));
      if(ptr!=NULL){
        strcat(ptr->valore, buffer);
        ptr->next = NULL;
      }
      return ptr;
    }
    elemento *ptr2 = ptr;
    while(ptr2->next != NULL)
        ptr2 = ptr2->next;
    ptr2->next = (elemento *)malloc(sizeof(elemento));
    if(ptr2->next==NULL){
        printf("Non è possibile aggiungere elementi...\n");
        return ptr;
    }
    strcat(ptr2->next->valore , buffer);
    ptr2->next->next = NULL;
    return ptr;
}
```

Soluzione Ex 3 – Server (3/3)

```c
void funzione1(int sockfd)
{
    char buff[BUFF_SIZE];
    int n;
    elemento *p = NULL; //puntatore alla lista
    // ciclo infinito
    for (;;) {
        printf("\nIn attesa messaggio dal client...\n");
        bzero(buff, BUFF_SIZE);
        read(sockfd, buff, sizeof(buff));
        printf("\nMessaggio dal client: %s\n", buff);

        if (strncmp("exit", buff, 4) == 0) {
            printf("Ricevuto Exit...\n");
            break;
        }

        if (strncmp("lista", buff, 5) == 0) {
            //inviare la lista
            elemento *p2 = p;
            bzero(buff, BUFF_SIZE);

            while(p2 != NULL){
               //controllo che non sia terminato lo spazio in buff
               if(strlen(buff) + strlen(p2->valore) >= (BUFF_SIZE - 1)){
                   break;
               }
               strcat(buff,p2->valore);
               p2 = p2->next;
            }
            //invio la stringa al client
            write(sockfd, buff, sizeof(buff));
            continue;
        }
        //se non sono entrato nei primi due if
        //significa che devo aggiungere un nuovo elemento
        //alla lista
        p = aggiungiElemento(p, buff);
        bzero(buff, BUFF_SIZE);
        strcat(buff, "Elemento aggiunto!");
        //invio il messaggio al client
        write(sockfd, buff, sizeof(buff));
    }
}
```

Soluzione Ex 3 – Client (1/2)

```c
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 100
#define PORT 3000
#define SA struct sockaddr

void func(int sockfd);

int main()
{
    int sockfd, connfd;
    struct sockaddr_in servaddr, cli;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        printf("socket creation failed...\n");
        exit(0);
    }
    else
        printf("Socket creata correttamente..\n");
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);
    if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
        printf("Connessione con il server fallita...\n");
        exit(0);
    }
    else
        printf("Connesso con il server..\n");

    func(sockfd);
    close(sockfd);
}
```

Soluzione Ex 3 – Client (2/2)

```c
void func(int sockfd)
{
    char buff[MAX];
    int n;
    for (;;) {
        bzero(buff, sizeof(buff));
        printf("Input : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n');
        write(sockfd, buff, sizeof(buff));
        if ((strncmp(buff, "exit", 4)) == 0) {
            printf("Chiusura del programma...\n");
            break;
        }
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("Dal server:\n%s\n", buff);
    }
}
```

Esercizio 4

Si scrivano due programmi **client** e **server** che usano una Socket UDP.

SERVER-> deve gestire una lista dove ogni elemento è composto da un Array di char di 100 elementi.

Tramite un ciclo infinito resta in attesa di ricevere un messaggio da un client. Se il messaggio contiene ‘exit’ chiude il programma. Se contiene ‘lista’, deve inviare allo stesso client tutti gli elementi della lista. Se il messaggio non è ne ‘exit’ e ne ‘lista’ allora deve aggiungere un nuovo elemento alla lista contenente il messaggio ricevuto dal client.

CLIENT-> deve permettere all’utente di inserire un messaggio da inviare al server. Il messaggio può contenere la parola ‘exit’ per far terminare il server, la parola ‘lista’ per ricevere e stampare tutta la lista o una parola da aggiungere alla lista. Dopodiché il client deve terminare.

Soluzione Ex 4 – Server (1/3)

```c
#include<stdio.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netdb.h>
#include<string.h>
#include<stdlib.h>
#define BUFF_SIZE 80
#define PORT 2000
#define SA struct sockaddr

typedef struct Elemento{
    char valore[BUFF_SIZE];
    struct Elemento *next;
}elemento;

//prototipi delle funzioni
void funzione1(int sockfd);
elemento *aggiungiElemento(elemento *ptr, char buffer[]);

int main()
{
    int sockfd;
    struct sockaddr_in servaddr;
    sockfd=socket(AF_INET,SOCK_DGRAM,0);
    if(sockfd==-1){
        printf("socket creation failed...\n");
        exit(0);
    }else
        printf("Socket successfully created..\n");
    bzero(&servaddr,sizeof(servaddr));
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
    servaddr.sin_port=htons(PORT);
    if((bind(sockfd,(SA *)&servaddr,sizeof(servaddr)))!=0){
        printf("socket bind failed...\n");
        exit(0);
    }else
        printf("Socket successfully binded..\n");

    funzione1(sockfd);

    close(sockfd);
}
```

Soluzione Ex 4 – Server (2/3)

```c
elemento *aggiungiElemento(elemento *ptr, char buffer[]){

    if (ptr == NULL){
      //la lista è vuota
      ptr = (elemento *)malloc(sizeof(elemento));
      if(ptr!=NULL){
        strcat(ptr->valore, buffer);
        ptr->next = NULL;
      }
      return ptr;
    }
    elemento *ptr2 = ptr;
    while(ptr2->next != NULL)
        ptr2 = ptr2->next;
    ptr2->next = (elemento *)malloc(sizeof(elemento));
    if(ptr2->next==NULL){
        printf("Non è possibile aggiungere elementi...\n");
        return ptr;
    }
    strcat(ptr2->next->valore , buffer);
    ptr2->next->next = NULL;

    return ptr;
}
```

Soluzione Ex 4 – Server (3/3)

```c
47  void funzione1(int sockfd)
48  {
49      char buff[BUFF_SIZE];
50      int n,clen;
51      struct sockaddr_in cli;
52      clen=sizeof(cli);
53      elemento *p = NULL; //puntatore alla lista
54      // ciclo infinito
55      for (;;) {
56          printf("\nIn attesa di un messaggio da un client...\n");
57          bzero(buff, BUFF_SIZE);
58          recvfrom(sockfd,buff,sizeof(buff),0,(SA *)&cli,&clen);
59          printf("\nMessaggio dal client: %s\n", buff);
60          if (strncmp("exit", buff, 4) == 0) {
61              printf("Ricevuto Exit...\n");
62              break;
63          }
64          if (strncmp("lista", buff, 5) == 0) {
65              //inviare la lista
66              elemento *p2 = p;
67              bzero(buff, BUFF_SIZE);
68              while(p2 != NULL){
69                 //controllo che non siaterminato lo spazio in buff
70                 if(strlen(buff) + strlen(p2->valore) >= (BUFF_SIZE - 1)){
71                     break;
72                 }
73                 strcat(buff,p2->valore);
74                 p2 = p2->next;
75              }
76              //invio la stringa al client
77              sendto(sockfd,buff,sizeof(buff),0,(SA *)&cli,clen);
78              continue;
79          }
80          //se non sono entrato nei primi due if
81          //significa che devo aggiungere un nuovo elemento
82          //alla lista
83          p = aggiungiElemento(p, buff);
84          bzero(buff, BUFF_SIZE);
85          strcat(buff, "Elemento aggiunto!");
86          //invio il messaggio al client
87          sendto(sockfd,buff,sizeof(buff),0,(SA *)&cli,clen);
88      }
89  }
```

Soluzione Ex 4 – Client (1/1)

```c
#include<sys/socket.h>
#include<netdb.h>
#include<string.h>
#include<stdlib.h>
#include<stdio.h>
#define BUFF_SIZE 80
#define PORT 2000
#define SA struct sockaddr

int main()
{
    char buff[BUFF_SIZE];
    //dichiarazione della socket UDP
    int sockfd,len,n;
    struct sockaddr_in servaddr;
    sockfd=socket(AF_INET,SOCK_DGRAM,0);
    if(sockfd==-1){
        printf("socket creation failed...\n");
        exit(0);
    }else
        printf("Socket successfully created..\n");
    bzero(&servaddr,sizeof(len));
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
    servaddr.sin_port=htons(PORT);
    len=sizeof(servaddr);
    //socket UDP creata...

    printf("\nInput: ");
    n=0;
    while((buff[n++]=getchar())!='\n');
    sendto(sockfd,buff,sizeof(buff),0,(SA *)&servaddr,len);
    if(strncmp("exit",buff,4)==0){
      printf("Client Exit...\n");
      return 0;
    }
    bzero(buff,sizeof(buff));
    printf("\nIn attesa di risposta dal server...");
    recvfrom(sockfd,buff,sizeof(buff),0,(SA *)&servaddr,&len);
    printf("\nRisposta dal Server :\n%s\n",buff);


    close(sockfd);
}
```