

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)

```
1  #include <stdio.h>
2  #include <netdb.h>
3  #include <netinet/in.h>
4  #include <stdlib.h>
5  #include <string.h>
6  #include <sys/socket.h>
7  #include <sys/types.h>
8  #define BUFF_SIZE 80
9  #define PORT 5000
10 #define SA struct sockaddr
11
12 void funzionel(int sockfd);
13 int main()
14 {
15     int sockfd, connfd, len;
16     struct sockaddr_in servaddr, cli;
17     sockfd = socket(AF_INET, SOCK_STREAM, 0);
18     if (sockfd == -1) {
19         printf("Creazione socket fallita...\n");
20         exit(0);
21     }
22     else
23         printf("Socket creata correttamente...\n");
24     bzero(&servaddr, sizeof(servaddr));
25     // Ip e porta
26     servaddr.sin_family = AF_INET;
27     servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
28     servaddr.sin_port = htons(PORT);
29     if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
30         printf("socket bind failed...\n");
31         exit(0);
32     }
33     if ((listen(sockfd, 5)) != 0) {
34         printf("Errore...\n");
35         exit(0);
36     }
37     else
38         printf("In attesa...\n");
39     len = sizeof(cli);
40     connfd = accept(sockfd, (SA*)&cli, &len);
41     if (connfd < 0) {
42         printf("Errore....\n");
43         exit(0);
44     }
45     else
46         printf("Client connesso...\n");
47     funzionel(connfd);
48     close(sockfd);
49 }
```

Soluzione Ex 1 – Server (2/2)

```
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)

```
1  #include <netdb.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <sys/socket.h>
6  #define MAX 80
7  #define PORT 5000
8  #define SA struct sockaddr
9
10 void func(int sockfd);
11 int main()
12 {
13     int sockfd, connfd;
14     struct sockaddr_in servaddr, cli;
15     sockfd = socket(AF_INET, SOCK_STREAM, 0);
16     if (sockfd == -1) {
17         printf("socket creation failed...\n");
18         exit(0);
19     }
20     else
21         printf("Socket creata correttamente..\n");
22     bzero(&servaddr, sizeof(servaddr));
23     servaddr.sin_family = AF_INET;
24     servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
25     servaddr.sin_port = htons(PORT);
26     if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
27         printf("Connessione con il server fallita...\n");
28         exit(0);
29     }
30     else
31         printf("Connesso con il server..\n");
32
33     func(sockfd);
34     close(sockfd);
35 }
```

Soluzione Ex 1 – Client (2/2)

```
37 void func(int sockfd)
38 {
39     char buff[MAX];
40     int n;
41     for (;;) {
42         bzero(buff, sizeof(buff));
43         printf("Input : ");
44         n = 0;
45         while ((buff[n++] = getchar()) != '\n');
46         write(sockfd, buff, sizeof(buff));
47
48         if ((strncmp(buff, "exit", 4)) == 0) {
49             printf("Chiusura del programma...\n");
50             return;
51         }
52         bzero(buff, sizeof(buff));
53         read(sockfd, buff, sizeof(buff));
54         printf("Dal server : %s\n", buff);
55     }
56 }
```

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)

```
1  #include<stdio.h>
2  #include<netinet/in.h>
3  #include<sys/types.h>
4  #include<sys/socket.h>
5  #include<netdb.h>
6  #include<string.h>
7  #include<stdlib.h>
8  #define MAX 80
9  #define PORT 4000
10 #define SA struct sockaddr
11
12 void func(int sockfd);
13
14 int main(){
15     int sockfd;
16     struct sockaddr_in servaddr;
17     sockfd=socket(AF_INET,SOCK_DGRAM,0);
18     if(sockfd==-1){
19         printf("Creazione socket fallita...\n");
20         exit(0);
21     }else
22         printf("Socket creata correttamente..\n");
23     bzero(&servaddr,sizeof(servaddr));
24     servaddr.sin_family=AF_INET;
25     servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
26     servaddr.sin_port=htons(PORT);
27     if((bind(sockfd,(SA *)&servaddr,sizeof(servaddr)))!=0){
28         printf("socket bind failed...\n");
29         exit(0);
30     }else
31         printf("Socket successfully binded..\n");
32
33     func(sockfd);
34     close(sockfd);
35 }
```

Soluzione Ex 2 – Server (2/2)

```
37 void func(int sockfd)
38 {
39     int numeroClient = 0;
40     char buff[MAX], buff2[MAX];
41     int n, clen;
42     struct sockaddr_in cli;
43     clen=sizeof(cli);
44     for(;;)
45     {
46         numeroClient++;
47         bzero(buff,MAX);
48         //in attesa di ricevere un messaggio da un client
49         recvfrom(sockfd,buff,sizeof(buff),0,(SA *)&cli,&clen);
50         if(strncmp("exit",buff,4)==0){
51             printf("Richiesta di exit dal client...\n");
52             break;
53         }
54         printf("\nClient numero: %d, messaggio ricevuto: %s\n",numeroClient,buff);
55         bzero(buff,MAX);
56         //preparo il messaggio di risposta al client
57         bzero(buff2,MAX);
58         sprintf(buff2, "%d",numeroClient);
59         strcat(buff,"Client numero: ");
60         strcat(buff,buff2);
61         //invio il messaggio
62         sendto(sockfd,buff,sizeof(buff),0,(SA *)&cli,clen);
63     }
64 }
```

Soluzione Ex 2 – Client (1/1)

```
1 #include<sys/socket.h>
2 #include<netdb.h>
3 #include<string.h>
4 #include<stdlib.h>
5 #include<stdio.h>
6 #define MAX 80
7 #define PORT 4000
8 #define SA struct sockaddr
9 int main()
10 {
11     char buff[MAX];
12     int sockfd,len,n;
13     struct sockaddr_in servaddr;
14     sockfd=socket(AF_INET,SOCK_DGRAM,0);
15     if(sockfd==-1){
16         printf("Creazione socket fallita...\n");
17         exit(0);
18     }else
19         printf("Socket creata correttamente..\n");
20     bzero(&servaddr,sizeof(len));
21     servaddr.sin_family=AF_INET;
22     servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
23     servaddr.sin_port=htons(PORT);
24     len=sizeof(servaddr);
25
26     printf("\nMessaggio per il server: ");
27     n=0;
28     while((buff[n++]=getchar())!='\n');
29     sendto(sockfd,buff,sizeof(buff),0,(SA *)&servaddr,len);
30     if(strncmp("exit",buff,4)==0){
31         printf("Richiesta di exit...\n");
32     }else{
33         bzero(buff,sizeof(buff));
34         recvfrom(sockfd,buff,sizeof(buff),0,(SA *)&servaddr,&len);
35         printf("Risposta dal server: %s\n",buff);
36     }
37
38     printf("\nChiusura programma...\n");
39     close(sockfd);
40
41     return 0;
42 }
```

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.

```

1  #include<stdio.h>
2  #include <stdio.h>
3  #include <netdb.h>
4  #include <netinet/in.h>
5  #include <stdlib.h>
6  #include <string.h>
7  #include <sys/socket.h>
8  #include <sys/types.h>
9  #define BUFF_SIZE 100
10 #define PORT 3000
11 #define SA struct sockaddr
12
13 ▼ typedef struct Elemento{
14     char valore[BUFF_SIZE];
15     struct Elemento *next;
16 }elemento;
17 //prototipi delle funzioni
18 void funzionel(int sockfd);
19 elemento *aggiungiElemento(elemento *ptr, char buffer[]);
20
21 int main()
22 ▼ {
23     int sockfd, connfd, len;
24     struct sockaddr in servaddr, cli;
25     sockfd = socket(AF_INET, SOCK_STREAM, 0);
26     ▼ if (sockfd == -1) {
27         printf("Creazione socket fallita...\n");
28         exit(0);
29     }else
30         printf("Socket creata correttamente...\n");
31     bzero(&servaddr, sizeof(servaddr));
32     servaddr.sin_family = AF_INET;
33     servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
34     servaddr.sin_port = htons(PORT);
35     ▼ if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
36         printf("socket bind failed...\n");
37         exit(0);
38     }
39     ▼ if ((listen(sockfd, 5)) != 0) {
40         printf("Errore...\n");
41         exit(0);
42     }else
43         printf("In attesa...\n");
44     len = sizeof(cli);
45     connfd = accept(sockfd, (SA*)&cli, &len);
46     ▼ if (connfd < 0) {
47         printf("Errore...\n");
48         exit(0);
49     }else
50         printf("Client connesso...\n");
51     funzionel(connfd);
52     close(sockfd);
53 }

```

Soluzione Ex 3 – Server (1/3)

Soluzione Ex 3 – Server (2/3)

```
102 ▼ elemento *aggiungiElemento(elemento *ptr, char buffer[]){
103 ▼     if (ptr == NULL){
104         //la lista è vuota
105         ptr = (elemento *)malloc(sizeof(elemento));
106 ▼     if(ptr!=NULL){
107         »     strcat(ptr->valore, buffer);
108         »     ptr->next = NULL;
109         »     }
110         return ptr;
111     }
112     elemento *ptr2 = ptr;
113     while(ptr2->next != NULL)
114         ptr2 = ptr2->next;
115     ptr2->next = (elemento *)malloc(sizeof(elemento));
116 ▼ »     if(ptr2->next==NULL){
117         »     printf("Non è possibile aggiungere elementi...\n");
118         »     return ptr;
119         »     }
120     »     strcat(ptr2->next->valore , buffer);
121     ptr2->next->next = NULL;
122     return ptr;
123 }
```

```

56 void funzional(int sockfd)
57 {
58     char buff[BUFF_SIZE];
59     int n;
60     elemento *p = NULL; //puntatore alla lista
61     // ciclo infinito
62     for (;;) {
63         printf("\nIn attesa messaggio dal client...\n");
64         bzero(buff, BUFF_SIZE);
65         read(sockfd, buff, sizeof(buff));
66         printf("\nMessaggio dal client: %s\n", buff);
67
68         if (strncmp("exit", buff, 4) == 0) {
69             printf("Ricevuto Exit...\n");
70             break;
71         }
72
73         if (strncmp("lista", buff, 5) == 0) {
74             //inviare la lista
75             elemento *p2 = p;
76             bzero(buff, BUFF_SIZE);
77
78             while(p2 != NULL){
79                 //controllo che non sia terminato lo spazio in buff
80                 if(strlen(buff) + strlen(p2->valore) >= (BUFF_SIZE - 1)){
81                     break;
82                 }
83                 strcat(buff, p2->valore);
84                 p2 = p2->next;
85             }
86             //invio la stringa al client
87             write(sockfd, buff, sizeof(buff));
88             continue;
89         }
90         //se non sono entrato nei primi due if
91         //significa che devo aggiungere un nuovo elemento
92         //alla lista
93         p = aggiungiElemento(p, buff);
94         bzero(buff, BUFF_SIZE);
95         strcat(buff, "Elemento aggiunto!");
96         //invio il messaggio al client
97         write(sockfd, buff, sizeof(buff));
98     }
99 }

```

Soluzione Ex 3 – Server (3/3)

Soluzione Ex 3 – Client (1/2)

```
1  #include <netdb.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <sys/socket.h>
6  #define MAX 100
7  #define PORT 3000
8  #define SA struct sockaddr
9
10 void func(int sockfd);
11
12 int main()
13 {
14     int sockfd, connfd;
15     struct sockaddr_in servaddr, cli;
16     sockfd = socket(AF_INET, SOCK_STREAM, 0);
17     if (sockfd == -1) {
18         printf("socket creation failed...\n");
19         exit(0);
20     }
21     else
22         printf("Socket creata correttamente..\n");
23     bzero(&servaddr, sizeof(servaddr));
24     servaddr.sin_family = AF_INET;
25     servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
26     servaddr.sin_port = htons(PORT);
27     if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
28         printf("Connessione con il server fallita...\n");
29         exit(0);
30     }
31     else
32         printf("Connesso con il server..\n");
33
34     func(sockfd);
35     close(sockfd);
36 }
```


Soluzione Ex 3 – Client (2/2)

```
38 void func(int sockfd)
39 ▼ {
40     char buff[MAX];
41     int n;
42 ▼     for (;;) {
43         bzero(buff, sizeof(buff));
44         printf("Input : ");
45         n = 0;
46         while ((buff[n++] = getchar()) != '\n');
47         write(sockfd, buff, sizeof(buff));
48 ▼         if ((strncmp(buff, "exit", 4)) == 0) {
49             printf("Chiusura del programma...\n");
50             break;
51         }
52         bzero(buff, sizeof(buff));
53         read(sockfd, buff, sizeof(buff));
54         printf("Dal server:\n%s\n", buff);
55     }
56 }
```

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.

```

1  #include<stdio.h>
2  #include<netinet/in.h>
3  #include<sys/types.h>
4  #include<sys/socket.h>
5  #include<netdb.h>
6  #include<string.h>
7  #include<stdlib.h>
8  #define BUFF_SIZE 80
9  #define PORT 2000
10 #define SA struct sockaddr
11
12 ▼ typedef struct Elemento{
13     char valore[BUFF_SIZE];
14     struct Elemento *next;
15 }elemento;
16
17 //prototipi delle funzioni
18 void funzione1(int sockfd);
19 elemento *aggiungiElemento(elemento *ptr, char buffer[]);
20
21 int main()
22 ▼ {
23     int sockfd;
24     struct sockaddr_in servaddr;
25     sockfd=socket(AF_INET,SOCK_DGRAM,0);
26 ▼     if(sockfd==-1){
27         printf("socket creation failed...\n");
28         exit(0);
29     }else
30         printf("Socket successfully created...\n");
31     bzero(&servaddr,sizeof(servaddr));
32     servaddr.sin_family=AF_INET;
33     servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
34     servaddr.sin_port=htons(PORT);
35 ▼     if((bind(sockfd,(SA *)&servaddr,sizeof(servaddr)))!=0){
36         printf("socket bind failed...\n");
37         exit(0);
38     }else
39         printf("Socket successfully binded...\n");
40
41     funzione1(sockfd);
42
43     close(sockfd);
44 }

```

Soluzione Ex 4 – Server (1/3)

Soluzione Ex 4 – Server (2/3)

```
91 ▼ elemento *aggiungiElemento(elemento *ptr, char buffer[]){
92
93 ▼     if (ptr == NULL){
94         //la lista è vuota
95         ptr = (elemento *)malloc(sizeof(elemento));
96 ▼         if(ptr!=NULL){
97             » »   strcat(ptr->valore, buffer);
98             ptr->next = NULL;
99         }
100         return ptr;
101     }
102     elemento *ptr2 = ptr;
103     while(ptr2->next != NULL)
104         ptr2 = ptr2->next;
105     ptr2->next = (elemento *)malloc(sizeof(elemento));
106 ▼ »   if(ptr2->next==NULL){
107     »   »   printf("Non è possibile aggiungere elementi...\n");
108     »   »   return ptr;
109     »   }
110 »   strcat(ptr2->next->valore , buffer);
111     ptr2->next->next = NULL;
112
113     return ptr;
114 }
```

Soluzione Ex 4 – Server (3/3)

```
47 void funzone1(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                 //controlla che non si esaurisca 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)

```
1  #include<sys/socket.h>
2  #include<netdb.h>
3  #include<string.h>
4  #include<stdlib.h>
5  #include<stdio.h>
6  #define BUFF_SIZE 80
7  #define PORT 2000
8  #define SA struct sockaddr
9
10 int main()
11 {
12     char buff[BUFF_SIZE];
13     //dichiarazione della socket UDP
14     int sockfd,len,n;
15     struct sockaddr_in servaddr;
16     sockfd=socket(AF_INET,SOCK_DGRAM,0);
17     if(sockfd==-1){
18         printf("socket creation failed...\n");
19         exit(0);
20     }else
21         printf("Socket successfully created..\n");
22     bzero(&servaddr,sizeof(len));
23     servaddr.sin_family=AF_INET;
24     servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
25     servaddr.sin_port=htons(PORT);
26     len=sizeof(servaddr);
27     //socket UDP creata...
28
29     printf("\nInput: ");
30     n=0;
31     while((buff[n++]=getchar())!='\n');
32     sendto(sockfd,buff,sizeof(buff),0,(SA *)&servaddr,len);
33     if(strncmp("exit",buff,4)==0){
34         printf("Client Exit...\n");
35         return 0;
36     }
37     bzero(buff,sizeof(buff));
38     printf("\nIn attesa di risposta dal server...");
39     recvfrom(sockfd,buff,sizeof(buff),0,(SA *)&servaddr,&len);
40     printf("\nRisposta dal Server : \n%s\n",buff);
41
42
43     close(sockfd);
44 }
```