

```

1  #include <sys/types.h>          //librerie da includere
2  #include <sys/socket.h>
3  #include <netinet/in.h>
4  #include <netdb.h>
5  #include <stdio.h>
6  #include <string.h>
7  #include <signal.h>
8  #include "mio.h"              // cerco nel percorso del file client
9
10 // definizioni di funzioni
11
12 void gestoreFineConn (int);      // gestisce segnale PIPE
13 void gestoreUscita (int);      //gestisco il segnale di uscita
14
15 int isPrimo(long);             // funzioni computazionali
16 int analizza(long,long);
17 void invia(long);
18
19 int ds_sock;                   // tutte le funzioni sono a conoscenza di
    questo parametro
20
21
22 int main (int argc, char *argv[])
23 {
24     struct sockaddr_in client;
25     char buffer [BUF_DIM];     // buffer per trasmettere messaggi
    stringa al server
26     char *ip_Server;          // ip del server con cui ci si vuole
    connettere
27     int porta_Server;         // stessa porta del server
28     long inf, sup;           // estremi dell'intervallo da analizzare
29     char risposta;           // memorizza la risposta se continuare o
    meno l'analisi
30
31     if (argc<3) // controllo che il numero dei parametri sia
    corretto
32     {
33         printf("Introdurre %s <IP_Server>
    <porta_TCP_Server>\n",argv[0]);
34         exit(-1);
35     }
36
37     ip_Server = argv[1];      // memorizzo nelle variabili i
    parametri introdotti
38     porta_Server = atoi(argv[2]);
39
40     while (3) //il ciclo finisce con la chiusura del server
41     {
42
43         if ((ds_sock = socket(AF_UNIX,SOCK_STREAM,0))<0) // creo la
    socket
44         {
45             printf ("Errore di socket");
46             printf ("Impossibile connetersi al Server \n ");
47             exit (-1);
48         }

```

```

49
50     printf("*****\n");
51     printf("Premere Control + C per uscire\n");
52     printf("\n");
53     printf ("ds server %d\n",ds_sock);
54
55     client.sin_family = AF_UNIX;           // imposta la
struttura del server a cui connettermi
56     client.sin_port = porta_Server;
57     client.sin_addr.s_addr = inet_addr(ip_Server); // converto IP
stringa in long uns.
58
59
60
61     if ((connect(ds_sock,&client, sizeof(client))) <0 )      //
provo a connettermi con il server
62     {
63         printf("Impossibile effettuare connessione\n");
64         exit (-1);
65     }
66
67     signal (SIGINT,gestoreUscita);        // armo il segnale di
uscita
68     signal (SIGPIPE,gestoreFineConn);    // armo il segnale di
chiusura PIPE
69
70     write (ds_sock,"Pronto",BUF_DIM);    // invio il mio stato di
pronto al server
71     read (ds_sock,buffer,BUF_DIM);
72     printf ("Stato connessione : %s\n",buffer); // legge lo stato
del server o fine o OK
73     if (strcmp(buffer,"fine")==0)       // il server ha finito i
pacchetti
74     {
75         printf ("Il Server ha concluso la computazione\n");
76         if ((close(ds_sock))== -1 )
77             printf ("Impossibile chiudere la socket server\n");
78         exit(-1);
79     }
80     read (ds_sock,&inf,sizeof(inf));     // leggo l'intervallo da
analizzare inviato dal server
81     read (ds_sock,&sup,sizeof(sup));
82
83     printf ("Intervallo da analizzare [%d,%d] \n",inf,sup);
84
85     analizza(inf,sup); // funzione di analisi del pacchetto
86
87     printf("\n");
88
89     printf ("Sto per chiudere la porta\n");
90
91     if ((close(ds_sock))== -1 )
92         printf ("Impossibile chiudere la socket server\n");
93
94     printf ("Ho chiuso le porte\n");
95     } // ritorno all'inizio del ciclo

```

```

96  }
97
98
99  void gestoreUscita (int sig )    // gestisco una chiusura del
client prematura
100 {
101     invia (0); // in questo modo interrompo il figlio del server
102     exit (-1);
103 }
104
105
106
107 void gestoreFineConn (int sig) // gestisco il segnale chiusura
PIPE
108 {
109     printf ("\n");
110     printf ("Il server ha chiuso la socket\n"); // avviso del
segnale ricevuto ed esco
111     exit(-1);
112 }
113
114
115 int analizza (long inf, long sup) // funzione che scandisce i
vari numeri dell'intervallo
116 {
117     long i;
118     for (i = inf; i<=sup; i++)
119         if ((isPrimo(i))==1) invia(i); // se i è primo lo invio
al figlio del server
120     invia(0); // termino la comunicazione con il figlio del
server a fine ciclo
121 }
122
123
124 int isPrimo (long numero) // funzione che controlla se il
numero è primo
125 {
126     long resto;
127     for (i=2; i<numero; i++) // non considero il numero 1 (non
inviato dal server)
128     {
129         resto = numero%i;
130         if (resto==0) return (0); // il numero non è primo
131     }
132     return(1); // se sono arrivato fino a qui il
numero è primo
133 }
134
135
136 void invia (long numero) // funzione che invia il segnale
137 {
138     write (ds_sock,&numero,sizeof(numero));
139 }
140

```