

```

/* Client per richiedere l'invio di un file (get, versione 1) */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define LENGTH_FILE_NAME 90
#define LENGTH_PAROLA 30
#define DIM_BUFF 100

int main(int argc, char *argv[])
{
    int sd, length;
    char nome_file[LENGTH_FILE_NAME+1], parola[LENGTH_PAROLA+1], buffer[DIM_BUFF+1];
    struct hostent *host;
    struct sockaddr_in servaddr;
    int error=0;

    /* CONTROLLO ARGOMENTI ----- */
    if(argc!=3)
    {
        printf("Error:%s serverAddress serverPort\n", argv[0]);
        exit(1);
    }

    printf("Client avviato\n");

    /* PREPARAZIONE INDIRIZZO SERVER ----- */
    memset((char *)&servaddr, 0, sizeof(struct sockaddr_in));
    servaddr.sin_family = AF_INET;

    host = gethostbyname(argv[1]);
    if (host == NULL)
    {
        printf("%s not found in /etc/hosts\n", argv[1]);
        exit(2);
    }

    servaddr.sin_addr.s_addr=((struct in_addr*) (host->h_addr))->s_addr;
    // Controllo porta
    servaddr.sin_port = htons(atoi(argv[2]));

    /* CREAZIONE E CONNESSIONE SOCKET (BIND IMPLICITA) ----- */
    sd=socket(AF_INET, SOCK_STREAM, 0);
    if (sd < 0)
    {perror("apertura socket "); exit(3);}
    printf("Creata la socket sd=%d\n", sd);

    if (connect(sd,(struct sockaddr *) &servaddr, sizeof(struct sockaddr))<0)
    {perror("Errore in connect"); exit(4);}
    printf("Connect ok\n");

    /* CORPO DEL CLIENT: */
    /* ciclo di accettazione di richieste di file ----- */

```

```

printf("Richieste di trasferimento linee di file con parola, fino alla fine del file
di input\n");
printf("Inserire nome del file, EOF per terminare\n");

while (gets(nome_file))
{
    printf("Inserire parola, EOF per terminare\n");

    if(gets(parola))
    {
        // invio lunghezza nome file (con terminatore)
        length=strlen(nome_file)+1;
        length=htonl(length);
        if (write(sd, &length, (sizeof(int)))<0)
        { perror("write"); }

        // invio nome file (con terminatore)
        if (write(sd, nome_file, (strlen(nome_file)+1))<0)
        { perror("write"); }
        printf("Nome file %s inviato... \n", nome_file);

        /* Qui sotto vengono gestiti gli invii.
        * Per ogni invio, prima spedisco la lunghezza, poi
        * il dato vero e proprio.
        * In alternativa singola scrittura (write) di una
        * struttura dimensione fissa. Provare a realizzarlo.
        */

        // invio lunghezza nome file (con terminatore)
        length=strlen(parola)+1;
        length=htonl(length);
        if (write(sd, &length, (sizeof(int)))<0)
        { perror("write"); }

        // invio parola (con terminatore)
        if (write(sd, parola, (strlen(parola)+1))<0)
        { perror("write"); }
        printf("Parola %s inviata... \nDi seguito le linee ricevute:\n", parola);

        for(;;)
        {
            if(read(sd, &length, sizeof(int))<0)
            {
                error=1;
                perror("read");
                break;
            }
            length=(int)ntohl(length);
            /* Segnale di raggiunta fine file, esco dal ciclo
            * e mi preparo a ricevere una nuova richiesta
            */
            if( length == -1) break;
            // Segnale file inesistente
            else if( length == -2 )
            { printf("File richiesto inesistente\n"); break; }
            else{
                if(read(sd, &buffer, length)<0)
                {
                    error=1;
                    perror("read");
                    break;
                }
                // Scrivo a video
                printf("%s\n", buffer);
            }
        }
    }
}

```

```
    }  
  }  
  if( error == 1 ) break;  
  
  printf("Nome del file da richiedere, EOF per uscire:\n");  
  }  
  else break; // Se ricevo EOF esco dal ciclo  
} //while  
  
printf("\nClient: termino...\n");  
  
// Chiusura socket  
close(sd);  
  
exit(0);  
}
```