

```

/* Client per controllare l'esistenza di linea con parola in un file remoto */

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

int main(int argc, char **argv)
{
    struct hostent *host;
    struct sockaddr_in clientaddr, servaddr;
    int sd, len, ris, i, j;
    char nome_file[LENGTH_FILE_NAME+1], parola[LENGTH_PAROLA+1],
    buffer[LENGTH_FILE_NAME+LENGTH_PAROLA+2];

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

    /* PREPARAZIONE INDIRIZZO CLIENT E SERVER ----- */
    memset((char *)&clientaddr, 0, sizeof(struct sockaddr_in));
    clientaddr.sin_family = AF_INET;
    clientaddr.sin_addr.s_addr = INADDR_ANY;
    /* Passando 0 ci leghiamo ad un qualsiasi indirizzo libero,
     * ma cio' non funziona in tutti i sistemi.
     * Se nel nostro sistema cio' non funziona come si puo' fare?
     */
    clientaddr.sin_port = 0;

    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);
    }
    else
    {
        servaddr.sin_addr.s_addr=((struct in_addr *) (host->h_addr))->s_addr;
        // Controllare anche numero porta
        servaddr.sin_port = htons(atoi(argv[2]));
    }

    printf("Client avviato\n");

    /* CREAZIONE SOCKET ----- */
    sd=socket(AF_INET, SOCK_DGRAM, 0);
    if(sd<0) {perror("apertura socket"); exit(3);}
    printf("Creata la socket sd=%d\n", sd);
    /* BIND SOCKET, a una porta scelta dal sistema ----- */

```

```

if(bind(sd, (struct sockaddr *) &clientaddr, sizeof(clientaddr))<0)
{perror("bind socket "); exit(1);}
printf("Client: bind socket ok, alla porta %i\n", clientaddr.sin_port);

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

printf("Richiesto controllo parola in file\n");
printf("Qualsiasi tasto per procedere, EOF per terminare\n");
printf("Nome del file: ");

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

    if(gets(parola))
    {
        // costruzione messaggio applicativo
        // nome_file seguito da zero binario e parola seguita da zero binario
        strcpy(buffer, nome_file);
        i=strlen(nome_file)+1;
        j=0;

        while( i<(strlen(nome_file)+1+strlen(parola)+1) )
        {
            buffer[i]=parola[j];
            i++; j++;
        }

        len=sizeof(servaddr);
        if (sendto(sd, buffer, (strlen(buffer)+strlen(parola)+2), 0, (struct sockaddr
*)&servaddr, len)<0)
        {
            perror("scrittura socket");
            printf("Nome del file: ");
            continue; // se questo invio fallisce il client torna all'inizio del ciclo
        }

        /* ricezione del risultato */
        printf("Attesa del risultato...\n");
        if (recvfrom(sd, &ris, sizeof(ris), 0, (struct sockaddr *)&servaddr, &len)<0)
        {
            perror("recvfrom");
            printf("Nome del file: \n");
            continue; // se questa ricezione fallisce il client torna all'inizio del ciclo
        }

        ris=ntohl(ris);
        if(ris==0) printf("Il file richiesto contiene la parola\n");
        else printf("Il file richiesto non contiene la parola\n");

        printf("Nome del file: \n");
    }
    else break; // se ricevo EOF esco dal ciclo
} // while

printf("\nClient: termino...\n");
close(sd);

```

```
    exit(0);  
}
```