

```

/* Svolgimento Compito 1 - 21/12/05 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <errno.h>
#include <fcntl.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define DIM_BUFF 100
#define LENGTH_NAME 20
#define N 10
#define K 7
#define max(a,b) ((a) > (b) ? (a) : (b))

/*****/

typedef struct
{
    /* Nome di lunghezza massima LENGTH_NAME */
    char nomeStanza[LENGTH_NAME];
    /* Tre caratteri: eventuale sospensione ('S'),
     * stato ('M' o 'P'), e terminatore '\0'
     */
    char tipo[3];
    /* Al massimo K utenti, nome di lunghezza massima LENGTH_NAME */
    char utente[K][LENGTH_NAME];
}
Stanza;

/*****/

void gestore(int signo)
{
    int stato;
    printf("esecuzione gestore di SIGCHLD\n");
    wait(&stato);
}

/*****/

int main(int argc, char **argv)
{
    int listenfd, connfd, udpfd, nready, maxfdpl;
    const int on = 1;
    /* il buffer utilizzato per la spedizione, deve contenere una stanza, piu' eventuali
     * commenti e separatori: due '\t' fra i nomi degli utenti e un '\n'
     */
    char buff[(K+1)*LENGTH_NAME+(K*2)+8];
    char nome_stanza[LENGTH_NAME];
    fd_set rset;
    int i, j, len, ris, port;
    struct sockaddr_in cliaddr, servaddr;
    Stanza stanze[N];

```

```

/* CONTROLLO ARGOMENTI ----- */
if(argc!=2)
{
    printf("Error: %s port\n", argv[0]);
    exit(1);
}
else port = atoi(argv[1]); // Controllare anche numero porta!!

printf("Server avviato\n");

/* INIZIALIZZAZIONE STRUTTURA DATI ----- */
for(i=0; i<N; i++){
    strcpy(stanze[i].nomeStanza, "L");
    strcpy(stanze[i].tipo, "L");
    for(j=0; j<K; j++) strcpy(stanze[i].utente[j], "L");
}
strcpy(stanze[0].nomeStanza, "Informatica");
strcpy(stanze[0].tipo, "P");
strcpy(stanze[0].utente[0], "Minnie");
strcpy(stanze[0].utente[2], "Paperino");
strcpy(stanze[1].nomeStanza, "Ambiente");
strcpy(stanze[1].tipo, "M");
strcpy(stanze[1].utente[0], "Topolino");
strcpy(stanze[1].utente[1], "Pluto");

/* CREAZIONE SOCKET TCP ----- */
listenfd=socket(AF_INET, SOCK_STREAM, 0);
if (listenfd <0)
{perror("apertura socket TCP "); exit(1);}
printf("Creata la socket TCP d'ascolto, fd=%d\n", listenfd);

/* INIZIALIZZAZIONE INDIRIZZO SERVER E BIND ----- */
memset ((char *)&servaddr, 0, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = INADDR_ANY;
servaddr.sin_port = htons(port);

if (setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on))<0)
{perror("set opzioni socket TCP"); exit(2);}
printf("Set opzioni socket TCP ok\n");

if (bind(listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr))<0)
{perror("bind socket TCP"); exit(3);}
printf("Bind socket TCP ok\n");

if (listen(listenfd, 5)<0)
{perror("listen"); exit(4);}
printf("Listen ok\n");

/* CREAZIONE SOCKET UDP ----- */
udpfd=socket(AF_INET, SOCK_DGRAM, 0);
if(udpfd <0)
{perror("apertura socket UDP"); exit(5);}
printf("Creata la socket UDP, fd=%d\n", udpfd);

/* INIZIALIZZAZIONE INDIRIZZO SERVER E BIND ----- */
memset ((char *)&servaddr, 0, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = INADDR_ANY;
servaddr.sin_port = htons(port);

if(setsockopt(udpfd, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on))<0)

```

```

{perror("set opzioni socket UDP"); exit(6);}
printf("Set opzioni socket UDP ok\n");

if(bind(udpfd, (struct sockaddr *) &servaddr, sizeof(servaddr))<0)
{perror("bind socket UDP"); exit(7);}
printf("Bind socket UDP ok\n");

/* AGGANCIO GESTORE PER EVITARE FIGLI ZOMBIE ----- */
signal(SIGCHLD, gestore);

/* PULIZIA E SETTAGGIO MASCHERA DEI FILE DESCRIPTOR ----- */
FD_ZERO(&rset);
maxfdpl=max(listenfd, udpfd)+1;

/* CICLO DI RICEZIONE EVENTI DALLA SELECT ----- */
for(;;)
{
    FD_SET(listenfd, &rset);
    FD_SET(udpfd, &rset);

    if ((nready=select(maxfdpl, &rset, NULL, NULL, NULL))<0)
    {
        if (errno==EINTR) continue;
        else {perror("select"); exit(8);}
    }

    /* GESTIONE RICHIESTE SPEDIZIONE STRUTTURA ----- */

    if (FD_ISSET(listenfd, &rset))
    {
        printf("Ricevuta richiesta di visualizzazione stanze\n");

        len = sizeof(struct sockaddr_in);
        if((connfd = accept(listenfd, (struct sockaddr *)&cliaddr, &len))<0)
        {
            if (errno==EINTR) continue;
            else {perror("accept"); exit(9);}
        }

        if (fork()==0)
        { /* processo figlio che serve la richiesta di operazione */
            close(listenfd);
            printf("Dentro il figlio, pid=%i\n", getpid());

            /* chiudo l'input */
            if (shutdown(connfd, SHUT_RD)<0)
            {
                if (errno==EINTR) continue;
                else {perror("shutdown(read)"); exit(8);}
            }

            for(i=0; i<N; i++){
                /* costruzione risposta */
                strcpy(buff, stanze[i].nomeStanza);
                strcat(buff, " tipo ");
                strcat(buff, stanze[i].tipo);
                strcat(buff, "\n");
                for(j=0; j<K; j++){
                    strcat(buff, stanze[i].utente[j]);
                    strcat(buff, "\t\t");
                }
                strcat(buff, "\n");
            }
        }
    }
}

```

```

        printf("Dato inviato:\n%s", buff);
        /* scrittura della riga su socket */
        if (write(connfd, buff, strlen(buff))<0)
        {perror("write"); break;}
    }

    printf("Terminato invio struttura dati\n");
    printf("Figlio %i: chiudo connessione e termino\n", getpid());
    /* libero le risorse e chiudo l'output */
    shutdown(connfd, SHUT_WR);
    exit(0);
} //figlio

/* padre chiude la socket dell'operazione */
close(connfd);

} /* fine gestione richieste visualizzazione */

/* GESTIONE RICHIESTE DI SOSPENSIONE ----- */
if (FD_ISSET(udpfd, &rset))
{
    printf("Ricevuta richiesta di sospensione stanza\n");

    len=sizeof(struct sockaddr_in);
    if (recvfrom(udpfd, nome_stanza, sizeof(nome_stanza), 0, (struct sockaddr
*)&cliaddr, &len)<0)
    {perror("recvfrom"); continue;}

    printf("Richiesta sospensione stanza %s\n", nome_stanza);
    ris=-1;
    for(i=0; i<N; i++){
        if( strcmp(stanze[i].nomeStanza, nome_stanza) == 0 ){
            if( stanze[i].tipo[0]=='S' )
                // la stanza e' gia' sospesa, esco dal ciclo e restituisco risultato
negativo
                break;
            else{
                // cambio lo stato della stanza e restituisco risultato positivo
                stanze[i].tipo[1]=stanze[i].tipo[0];
                stanze[i].tipo[0]='S';
                stanze[i].tipo[2]='\0';
                ris=0;
                break;
            }
        }
    }
}

// conversione in formato di rete
ris=htonl(ris);

if (sendto(udpfd, &ris, sizeof(ris), 0, (struct sockaddr *)&cliaddr, len)<0)
{perror("sendto"); continue;}

} /* fine gestione richieste di conteggio */

} /* ciclo for della select */

/* NEVER ARRIVES HERE */
exit(0);
}

```