Esercizio sui Sockets

Realizzazione di Client/Server con Threads

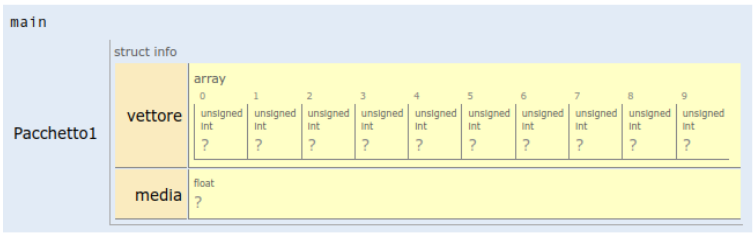
Si voglia realizzare un sistema di Client/Server con un Server e diversi Client; ogni Client invia un vettore di numeri naturali di dimensione pari a 10, e il Server risponde inviando il valore medio di tali valori. Il client stampa a video il valore medio ricevuto dal Server. Quando il Client vuole terminare, chiude la connessione. In tal caso, il Server deve automaticamente chiudere anch'esso la connessione con il client.

Si definisca un programma Server e un programma Client che realizzi quanto detto utilizzando i Sockets e i threads. Si deve supporre che per ogni client connesso, il server crei un thread apposito che gestisce la comunicazione con il client. Quando il client si disconnette, il thread finisce. Per evitare che il Server debba invocare la pthread_join, si deve cambiare lo stato di ciascun thread in modo detached. Questo è molto importante perché in questo modo le risorse del thread vengono rilasciate dal SO alla terminazione del thread.

Si supponga che il server non termini mai, e dunque e' necessario utilizzare il comando di kill per terminarlo

Client

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<sys/socket.h>

#define N 10
#define SIZE 5

typedef struct {
  unsigned int vettore[N];
  float media;
} info;

void riempi(info *p, int dim){
  int i;

  for (i=0; i<dim; i++){
     printf("Inserisci elemento di indice %d ",i);
     scanf("%u",&p->vettore[i]);
  }
  while(getchar()!='\n');
}

int main(void) {
   int sockfd, len, result, running=1;
   struct sockaddr_in address;
   info message;
   char buffer[SIZE];


   sockfd=socket(AF_INET, SOCK_STREAM, 0);
   if (sockfd==-1) {
     fprintf(stderr,"opps: creazione socket fallita ");
     exit(EXIT_FAILURE);
   }
   address.sin_family=AF_INET;
   address.sin_addr.s_addr=inet_addr("127.0.0.1");
   address.sin_port=htons(9734);
   len=sizeof(address);

   result=connect(sockfd, (struct sockaddr *)&address, len);
   if (result==-1) {
     fprintf(stderr,"opps: connessione rifiutata ");
     exit(EXIT_FAILURE);
```



```
  }

  printf("Connesso al Server \n");
  while(running) {
    printf("Vuoi Continuare ? ");
    fgets(buffer, SIZE, stdin);
    if (!strncmp(buffer, "n", 1) || !strncmp(buffer, "N", 1)) {
       running = 0;
    } else {
      riempi(&message,N);
      write(sockfd, &message, sizeof(info));
      read(sockfd, &message, sizeof(info));
      printf("La risposta che ho ricevuto e' stata: %f ", message.media);
    }
  }
  close(sockfd);
  exit(EXIT_SUCCESS);
}
```

Server

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<pthread.h>

#define N 10
#define SIZE 5

typedef struct {
  unsigned int vettore[N];
  float media;
} info;

float media(info *p, int dim){
  int i;
  float m=0;

  for (i=0; i<dim; i++) m+=p->vettore[i];
  return(m/dim);
}

/*
 * Funzione per la gestione di ciascun client
 */

void *connection_handler(void *socket_desc)
{
    int socket = *(int*)socket_desc;
    int risultato;
    int ripeti=1;
    info message;

    while(ripeti) {
        risultato=read(socket, &message, sizeof(info));
        if (risultato==0) {
             ripeti=0;
        } else {
            message.media=media(&message,N);
            write(socket, &message, sizeof(info));
        }
    }
    close(socket);
    free(socket_desc);
    pthread_exit(NULL);
```

```
}

int main(void){
   int server_sockfd, client_sockfd;
   int server_len, client_len;
   struct sockaddr_in server_address, client_address;
   int *new_sock;
   int res;
   pthread_t tid;

   server_sockfd = socket(AF_INET , SOCK_STREAM , 0);
   if (server_sockfd == -1) {
      fprintf(stderr,"Could not create socket");
      exit(EXIT_FAILURE);
   }
   printf("Socket creato\n");

   server_address.sin_family=AF_INET;
   server_address.sin_addr.s_addr=inet_addr("127.0.0.1");
   server_address.sin_port=htons(9734);
   server_len=sizeof(server_address);

   if( bind(server_sockfd,(struct sockaddr *)&server_address, server_len) ==-1){
      fprintf(stderr,"Bind failed");
      exit(EXIT_FAILURE);
   }
   printf("Bind eseguita\n");

   listen(server_sockfd, 5);
   printf("Attendo connesioni ...\n");

   while(1) {
    client_len=sizeof(client_address);
    client_sockfd=accept(server_sockfd,(struct sockaddr *)&client_address,&client_len);
    if (client_sockfd < 0){
      fprintf(stderr,"Accept failed");
      exit(EXIT_FAILURE);
    }
    printf("Connessione accettata\n");

    new_sock = (int *)malloc(sizeof(int));
    *new_sock = client_sockfd;

    res = pthread_create(&tid, NULL, connection_handler, (void *)new_sock);
    if (res != 0) {
          fprintf(stderr, "Creazione del Thread fallita");
          exit(EXIT_FAILURE);
    }
    pthread_detach(tid);
   }
}
```