



2 Novembre
2021

Esercitazione 3

Socket in C

Bernardini Claudio
Corsetti Luca
Giardina Gianluca
Straccali Leonardo

Obiettivi esercitazione

Hard skills

1

Creazione e gestione socket
datagram in C

2

Gestione di file con client
filtro

1

Creazione e gestione socket
con connessione in C

2

Gestione di eliminazione di
righe da un file tramite server
parallelo

Soft skills

1

Collaborazione per raggiungere un fine comune

2

Organizzazione del proprio lavoro in team

2

Introduzione senza connessione

Client

1

Stabilisce una connessione con il servitore

2

Invia il nome del file al server

3

Attende la risposta del server contenente il numero di caratteri della parola più lunga nel file

Servitore

1

Attende richiesta di una connessione

2

Riceve il nome del file, e se esiste, identifica la parola più lunga nel file

2

Invia il numero di lettere che formano la parola (-1 se il file non esiste)

3

Introduzione con connessione

Client

1

Stabilisce una connessione
con il servitore

2

Invia il nome del file e il numero
di linea da eliminare

3

Attende la risposta del server
contenente il nuovo contenuto
del file

Servitore

1

Attende richiesta di una
connessione

2

Crea un figlio per gestire le
richieste in arrivo dai client

2

Elimina la riga dal file ricevuto e
invia il nuovo contenuto al client

4

Implementazione Client

```
while ((ok=scanf("%s", &nameFile)) != EOF )
{

    printf("nameFile: %s \n", &nameFile);

    if( ok != 1){
        /* Problema nell'implementazione della scanf. Se l'input contiene PRIMA
        * dell'intero altri caratteri la testina di lettura si blocca sul primo carattere
        * (non intero) letto. Ad esempio: ab1292\n
        *           ^      La testina si blocca qui
        * Bisogna quindi consumare tutto il buffer in modo da sbloccare la testina.
        */
        do{c=getchar(); printf("%c ", c);}
        while (c!= '\n');
        continue;
    }

    len=sizeof(servaddr);
    if(sendto(sd, &nameFile, sizeof(nameFile), 0, (struct sockaddr *)&servaddr, len)<0){
        perror("sendto");
        continue;
    }
}
```

Implementazione Server

```
while ((nread = read(in_file, &c, sizeof(c))) > 0)
{
    if (c != ' ' && c != '\n')
    {
        currentDim++;
    }
    else
    {
        if (currentDim > ris && currentDim != 0) ris = currentDim;
        currentDim=0;
    }
}

if(currentDim > ris && currentDim != 0) ris = currentDim;

close(in_file);

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

Implementazione Client

```
/* RICHIEDO e INVIO Numero linea da eliminare */
printf("Inserire numero linea da eliminare (intero): ");
char number[256];
gets(number);
write(sd,number,sizeof(number));

/*INVIO File*/
printf("Client: stampo e invio file da ordinare\n");
while((nread=read(fd_sorg, buff, DIM_BUFF))>0){
    write(1,buff,nread);    //stampa
    write(sd,buff,nread);   //invio
}
printf("Client: file inviato\n");
/* Chiusura socket in spedizione -> invio dell'EOF */
shutdown(sd,1);

/*RICEZIONE File*/
printf("Client: ricevo e stampo file senza la linea\n");
while((nread=read(sd,buff,DIM_BUFF))>0){
    write(fd_dest,buff,nread);
    write(1,buff,nread);
}
printf("Traspefimento terminato\n");
/* Chiusura socket in ricezione */
shutdown(sd, 0);
```

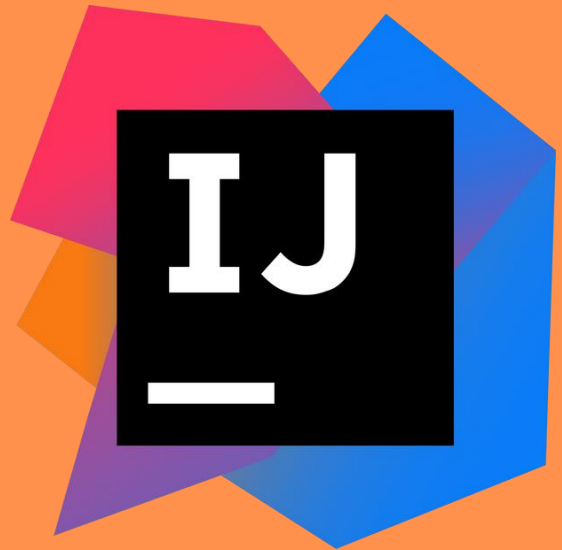

Implementazione Server Concorrente

```
//lettura linea da eliminare
read(conn_sd, &lineToDelete, sizeof(lineToDelete));
//concatenamento parametro delete della sed
strncat(lineToDelete, "d", 1);
close(listen_sd);
host = gethostbyaddr((char *)&cliaddr.sin_addr, sizeof(cliaddr.sin_addr), AF_INET);
if (host == NULL)
{
    printf("client host information not found\n");
    continue;
}
else
    printf("Server (figlio): host client e' %s \n", host->h_name);
printf("Server (figlio): eseguo l'ordinamento\n");

close(1);
close(0);
dup(conn_sd);
dup(conn_sd);
close(conn_sd);
execl("/usr/bin/sed", "sed", lineToDelete, (char *)0);
```


Gestione progetto

Coding



git



Team

