

CORSO DI LAURA TRIENNALE IN INFORMATICA – UNIFI

SISTEMI OPERATIVI A.A 23/24

Sommario

[1 - Descrizione del Progetto 3](#_Toc170649914)

[2 – Server 4](#_Toc170649915)

[1.1 - Implementazione 5](#_Toc170649916)

[3 - Client 7](#_Toc170649917)

[3 - Esempio di Esecuzione 8](#_Toc170649918)

# 1 - Descrizione del Progetto

Il presente documento illustra l’implementazione di un sistema di gestione della Rubrica, utilizzando un’architettura client-server.

Il *server* di tipo TCP/IP dettato dal SOCK\_STREAM, con dominio AF\_INET.

Il suo scopo è quello di memorizzare una rubrica di contatti che possono essere visualizzati, aggiunti, eliminati e modificati tramite comandi inviati dal client.

Il *client* invia al server, dei comandi in base all’operazione che vuole eseguire.

Possibili comandi:

1. Visualizzare i contatti della rubrica:
2. Aggiungere un nuovo contatto
3. Modificare un contatto esistente
4. Eliminare un contatto esistente
5. Uscire dal sistema

Una volta mandata la richiesta al server, riceverà indietro una risposta da parte del server.

# 2 – Server

Il *server* è il “nucleo” del programma, gestisce e accetta le connessioni client e gestisce le richieste in base ai comandi ricevuti.

Nel server vengono inizializzati la Rubrica e utente autenticato. Sono presenti 3 strutture:

**Rubrica:**

typedef struct {

    Contatto contatti[MAX\_CONTATTO];

    int totContatti;

} Rubrica;

Dove la variabile **contatti** è la lista dei contatti salvati e **totContatti** rappresenta la lunghezza dei contatti presenti.

**Contatto:**

typedef struct {

    char lastname[50];

    char firstname[50];

    char cell\_number[50];

} Contatto;

La struttura Contatto ha tre campi:

* Nome
* Cognome
* Numero di telefono

Sono le tre campi per identificare il Contatto di un persona, che vengono richiesta all’utente durante un’operazione come l’aggiunta di un nuovo utente.

**Utente:**

typedef struct {

    char username[50];

    char password[50];

} Utente;

Rappresenta l’utente autenticato. Nelle operazioni di inserimento, modifica e eliminazione di un contatto viene richiesto l’autenticazione. Nel caso positivo si potrà procedere all’operazione desiderata, altrimenti non si potrà procedere e verrà richiesto l’operazione da eseguire.

## 1.1 - Implementazione

Il server crea il socket TCP utiizzando socket()

**socket(AF\_INET, SOCK\_STREAM, 0)**

e assegna un indirizzo IP e una porta al socket creato usando bind()

**socket(AF\_INET, SOCK\_STREAM, 0).**

Mette il socket in modalità ascolto per nuove connessioni con listen()

**listen(sockfd, MAX\_N\_CLIENT)** dove MAX\_N\_CLIENT = 5 rappresenta il numero massimo di connessioni pendenti:

* Attende e accetta una connessione in arrivo con accept()

**accept(sockfd, (struct sockaddr \*)&cli\_addr, &length).**

* Dopo aver accettato una connessione, il server può leggere e scrivere dati sul nuovo socket **client\_connection.**

Dopo il setup del server viene implementato la logica per ogni richiesta del client. Quest’ultimo invia al server il comando numerico il quale leggerà tramite un **recv** e in base al comando entrerà nel switch-case.

***Case 1***: nel primo caso viene richiesto la visualizzazione della rubrica, perciò il server invierà al client la lista dei contatti tramite una send

***send(client\_connection, &rubrica, sizeof(Rubrica), 0)***

***Case 2:*** nel secondo caso avviene l’inserimento di un nuovo contatto. Avvenuto con successo l’autenticazione viene chiesto i tre campi del contatto, Nome, Cognome e il numero di telefono. Nel caso in cui non ci fosse più spazio nella rubrica allora non viene inserito il nuovo contatto e si è riportati alla richiesta del nuovo comando.

***Caso 3:*** nel terzo caso viene è richiesto la modifica di un contatto. Successivamente all’autenticazione, viene chiesto il nome e il cognome da modificare. Viene fatta una ricerca alla lista **contatti** ricercando per nome e cognome. Nel caso positivo viene chiesto al client il nuovo numero di telefono del contatto, altrimenti viene inviato al client la stringa “Contatto non trovato”.

***Caso 4:*** nel quarto caso viene richiesto l’eliminazione di un contatto se è presente. Successivamente all’autenticazione, viene chiesto il nome e cognome per la ricerca del contatto da eliminare. Se viene trovato il contatto la funzione “elimina\_contatto”, sposta tutti i contatti successivi al contatto eliminato di una posizione indietro per riempire il vuoto lasciato dal contatto eliminato, quindi decrementa il contatore totale dei contatti nella rubrica.

***Caso 9:*** in questo caso si esce dall’applicazione uccidendo il processo.

# 3 - Client

Il client si connette al server e tramite le richieste consulta la sua rubrica. Ogni client servito dal server ha una sua Rubrica personalizzata.

Una volta connesso al server con il metodo

**connect(client\_fd, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr))**

al client viene mostrato tramite una “menu”, i comandi che può eseguire.

**1 –** esegue il comando 1 richiedendo la lista della rubrica. Riceverà come risposta dal server una lista con tutti i campi del Contatto, e verrà stampato con il metodo *printRubrica*

void printRubrica(Rubrica\* rubrica){

    for(int i = 0; i < rubrica->totContatti; i++) {

        printContatto(rubrica->contatti[i]);

    }

}

**2 –** il comando 2 chiede di aggiungere un nuovo contatto dopo aver eseguito l’autenticazione. L’aggiunta del nuovo contatto avviene solo se la rubrica non è piena. Viene chiesto il nome, cognome e il numero di telefono. In quest’ultimo viene fatto un check se il numero di telefono inserito è una cifra da 0 a 9. In caso contrario viene richiesto il numero con un messaggio *“Numero non valido. Inserisci di nuovo il numero”,* altrimenti il numero viene inserito nella rubrica.

**3 –** il comando 3 chiede al server di modificare un contatto, dopo aver eseguito con successo l’autenticazione. Nella richiesta di modifica viene chiesto nome e cognome e starà al server cercare il contatto. Una volta trovato il contatto con successo viene chiesto il numero di telefono, sempre facendo il controllo sul numero, come nel comando 2.

**4 –** il comando 4 chiede di eliminare un contatto successivamente l’autenticazione. Anche in questo caso viene chiesto il nome e il cognome**.** Starà al server farà i controlli se è presente il contatto.

**9 –** chiusura della connessione del client.

# 4 - Esempio di Esecuzione