Relazione Progetto Gestione Rubrica

Relatori:

Riccardo Giannuzzi

Biribò Francesco

Ilyas Timour

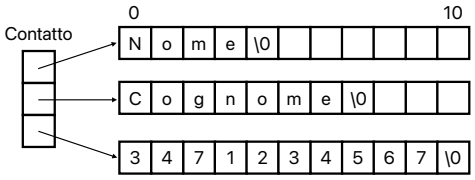
Data: 12 settembre 2024

Indice

1 Progettazione e Implementazione

1.1 Struttura del contatto

Ogni contatto della rubrica è un record con tre campi: nome, cognome e numero di telefono. Per semplicità ogni campo è una Stringa di 10 caratteri più delimitatore. Se fosse necessario la dimensione di nome e cognome potrebbe essere modificata cambiando delle costanti.



Imponiamo delle restrizioni per considerare valido un contatto:

- Nome e Cognome: non devono essere vuoti, non devono superare la lunghezza massima della stringa (10 caratteri) e devono essere composti soltanto da caratteri ASCII che siano lettere, maiuscole o minuscole, o cifre.
- Numero di telefono: deve essere lungo esattamente 10 caratteri e deve essere composto soltanto da caratteri ASCII che siano cifre.

Ci aspettiamo e vogliamo garantire che tutti i contatti memorizzati nella rubrica siano validi.

1.2 Ricerca di un Contatto nella rubrica

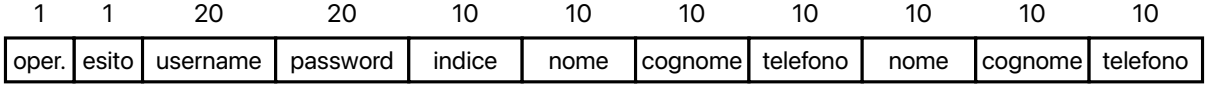
Che sia un'operazione di lettura o di modifica, la ricerca di un specifico contatto dalla rubrica è essenziale. Vogliamo che la ricerca possa essere fatta specificando i valori di un numero arbitrario di campi del contatto, perfino di nessun campo. Questo è ideale dato che, un utente che ha intenzione di cercare un contatto, probabilmente non conosce i valori di tutti i campi. Così, anche nel caso non conosca nemmeno il valore di un campo potrà comunque accedere al contenuto della rubrica .
Non è detto che, con queste modalità, una ricerca abbia un'unica corrispondenza nella rubrica, per esempio potrebbero esserci più contatti con lo stesso nome.
Quindi oltre ai valori dei campi del contatto, dovrà essere specificato l'indice della corrispondenza che si vuole cercare. Tale numero indicherà l'n-esima corrispondenza rispetto all'ordine con cui i contatti sono memorizzati dal server. Tuttavia per evitare ambiguità sulle operazioni di cancellazione e modifica, vogliamo che la combinazione dei tre campi garantisca un'unica corrispondenza in modo che tali operazioni non dipendano dall'ordine di memorizzazione dei contatti sul server.

1.3 Struttura e Memorizzazione Rubrica e Credenziali

Abbiamo scelto di memorizzare la rubrica in un file di testo sul server chiamato *rubrica.txt*. Salviamo un solo contatto per riga, scrivendo i valori dei suoi campi separati da virgola senza spazi bianchi seguiti da un carattere newline: *"nome,cognome,numero di telefono"*. Utilizziamo un approccio simile per le credenziali degli utenti. Ogni utente è una coppia (*username*, *password*), in cui l'username è univoco. Le credenziali vengono salvate in un file di testo chaimato *credenziali.txt*, sempre sul server. Per garantire una maggiore sicurezza agli utenti, viene applicata alle password una funzione hash prima che vengano memorizzate. Inoltre entrambi i file sono protetti specificandone livelli di permessi: per la rubrica *Owner* e *Group* hanno permessi sia in lettura che scrittura, mentre *Others* ha solo il permesso di read. Lo stesso vale per il file delle credenziali, per il quale però viene negata anche la lettura al gruppo *Others*. Tenendo in considerazione questo, è fondamentale che il processo del server sia avviato da un utente che abbia i permessi necessari, altrimenti non riuscirà ad accedere ai file di cui ha bisogno.

1.4 Protocollo di comunicazione Client-Server

Client e server scambiano pacchetti di lunghezza fissa definita in fase di precompilazione. Quindi ogni possibile parametro del pacchetto ha una posizione prestabilita al suo interno e una lunghezza massima in caratteri. Questo comporta che parte del pacchetto spesso non venga effettivamente utilizzata, tuttavia garantisce un livello maggiore di robustezza e sicurezza.



I parametri sono:

- Operazione: Usata dal client per indicare quale operazione richiedere al server e dal server per restituire nella risposta l'operazione che ha effettuato.
- Esito: Usata dal server per indicare con '1' il successo e con '0' il fallimento dell'operazione. In caso di fallimento possono essere specificati ulteriori esiti che corrispondono a determinati errori o casi particolari.
- Nome utente e Password: Usate dall'utente per autenticarsi.
- Indice: Indice, rispetto all'ordine di memorizzazione dei contatti, della corrispondenza da cercare in rubrica in caso di lettura.
- Dati del contatto: Usati in base all'operazione per: lettura, aggiunta, cancellazione o modifica.

Per le operazioni che modificano il contenuto dei campi di un contatto, è necessario specificare quale contatto deve essere modificato e i nuovi valori da sovrascrivere. Per questo motivo un pacchetto deve poter contenere i dati di due contatti.

1.5 Implementazione del Client

1.5.1 Connessione al server

Appena avviato il client tenta di stabilire la connessione con il server. Se sono stati specificati indirizzo ip e porta, come argomenti di avvio del programma, verranno usati tali dati (esempio "./client 54.23.132.12 54434"). Altrimenti tenterà di connettersi alla porta libera 50000 con ip localhost. Nel caso in cui la connessione fallisca applicherà una strategia di attesa esponenziale per ogni tentativo fallito. Dopo aver raggiunto il numero massimo di tentativi senza successo, il programma si chiuderà segnalando l'errore. Invece, se riesce a connettersi al server, viene mostrato il menu con il quale l'utente potrà interagire col programma.

1.5.2 Gestione dei Menu

L'utente interagisce con il programma tramite un'interfaccia utente basata su menu testuali annidati. Alcuni menu offrono una serie di opzioni selezionabili digitando il carattere corrispondente. Altri acquisiscono i dati necessari per le operazioni assicurandosi che siano validi e comunicando eventuali errori all'utente. Per rendere più chiara e interattiva l'interfaccia viene effettuata, quando si passa da un menu ad un altro, la pulizia del terminale tramite una sequenza escape ANSI.

```
============================
      GESTIONE RUBRICA
============================

[1] Leggere rubrica
[2] Autenticati
[x] Terminare sessione

Selezionare un'opzione:
```

```
====================================
      ACQUISIZIONE DATI LETTURA
====================================

Digitare i dati del contatto ( ENTER  per lasciarli vuoti):

  Nome: Mario
  Cognome: Rossi
  Numero di telefono: 3471234567
```

1.5.3 Acquisizione dati dall'utente

Abbiamo preferito evitare completamente l'uso di *scanf()* per l'acquisizione di dati dall'utente, vista la necessità di permettere all'utente di lasciare un input vuoto premendo ENTER. Abbiamo quindi definito la funzione *getOptionalInput()* che acquisisce, il numero di caratteri, considerando anche quelli vuoti, passato come parametro, dal terminale ignorando eventuali caratteri superflui e pulendo se necessario l'input buffer in modo da non sporcare eventuali letture successive.
In pratica acquisiamo i caratteri in modo sicuro tramite la funzione *fgets()* e poi in base a quanto letto gestiamo i vari casi che possono presentarsi.

1.5.4 Lettura di un contatto dalla rubrica

Vengono acquisiti dal terminale i valori inseriti dall'utente per i campi del contatto da cercare. L'utente non è obbligato a specificare tutti i campi del contatto, può addirittura lasciarli tutti vuoti. Viene quindi inviata al server la richiesta di leggere la prima corrispondenza con tali valori. Se il server restituisce una corrispondenza viene stampata e a questo punto l'utente può richiedere la corrispondenza successiva oppure inserire dati di ricerca differenti. Altrimenti viene comunicato all'utente che non vi è alcuna corrispondenza con i dati inseriti.

1.5.5 Aggiunta di un contatto alla rubrica

Per l'aggiunta, i dati del contatto devono essere validi secondo i criteri stabiliti precedentemente. Una volta acquisiti e validati i campi del contatto, si procede con la richiesta di aggiunta al server. Se il server segnala che è già presente un contatto con gli stessi dati viene comunicato l'errore. In caso contrario viene indicato che il contatto è stato aggiunto.

1.5.6 Eliminazione di un contatto dalla rubrica

Per effettuare una eliminazione, invece di richiedere l'intera terna di dati di un contatto, viene eseguita una prima operazione di lettura per individuare il contatto da eliminare, così non si devono conoscere a priori tutti i campi del contatto. Quindi l'utente può procedere eliminando l'ultimo contatto letto oppure cercarne un altro. Dopo la richiesta di una ulteriore conferma viene inviata la richiesta al server e comunicato l'esito.

1.5.7 Modifica di un contatto sulla rubrica

La gestione è simile a quella dell'eliminazione. L'unica differenza è che dopo aver selezionato il contatto da modificare, in questo caso è necessario acquisire dall'utente i valori modificati dei campi del contatto assicurandosi che siano validi.

1.5.8 Autenticazione

Anziché acquisire le credenziali ogni volta che l'utente effettua una modifica, prima ancora di permettere dal menu di selezionare le operazioni di modifica, viene richiesto di eseguire l'autenticazione. Una volta che l'utente si è autenticato, le credenziali vengono memorizzate per quella sessione e inoltrate automaticamente per le future operazioni di modifica. Nel caso in cui durante una modifica le credenziali risultassero non più valide, a causa di una modifica nel file delle credenziali, verrà annullata l'operazione e sarà richiesta una nuova autenticazione. Per evitare un numero eccessivo di tentativi di autenticazione da parte di un utente, superato il numero di tentativi falliti ammessi, viene imposto un periodo di attesa di un minuto prima di poter ritentare. Questa attesa può essere aggirata semplicemente riavviando il programma ma rappresenta comunque un disincentivo, rendendo quindi utile la sua implementazione.

1.6 Implementazione del Server

1.6.1 Connessioni con i client

Il server apre una socket sul numero di porta specificato a riga di comando all'avvio, se non è presente, di default usa la porta 50000. Se la socket viene creata correttamente il server rimarrà in attesa di connessioni da parte dei client, tramite *accept()*. Una volta che un client si connette, il server genera un processo figlio che si occuperà di gestire le richieste di quel client, mentre il server torna in attesa di connessioni da parte di altri client. Da questo punto in poi il client comunicherà esclusivamente con il processo figlio e tramite uno scambio di pacchetti. Il processo figlio leggerà il pacchetto inviato dal client, leggerà l'operazione scritta all'interno, eseguirà l'operazione e invierà al client un pacchetto di risposta. Fatto questo tornerà alla lettura, in attesa di altre richieste dal client.

1.6.2 Gestione delle operazioni

Tra le operazioni che può eseguire il server troviamo la ricerca, l'aggiunta, la cancellazione, la modifica di un contatto e l'autorizzazione di un client. Per quanto riguarda l'autorizzazione, estrae dal pacchetto ricevuto dal client username e password, esegue l'hash della password e cerca la coppia (user,hash) tra le credenziali. Tutte le altre operazioni richiedono la lettura, ed eventualmente la scrittura sul file rubrica. L'aggiunta legge il file per controllare se il contatto è già presente o meno, in caso non ci sia lo aggiunge alla fine. La ricerca legge un contatto dalla rubrica e ogni volta che corrisponde ai parametri di ricerca indicati nel pacchetto incrementa l'indice di corrispondenza (l'i-esimo contatto che coincide con questi criteri di ricerca) fermandosi a all'indice richiesto. La cancellazione e la modifica lavorano in modo modo analogo: entrambe aprono la rubrica in modalità lettura e copiano un riga alla volta in un file temporaneo. Se il contatto letto non era quello interessato viene riscritto mentre se il contatto è quello interessato, nel caso della cancellazione viene ignorato, nel caso della modifica scriviamo il nuovo contatto al suo posto. Terminata la lettura, il file rubrica viene eliminato e il file temporaneo viene rinominato e prende il suo posto.

1.6.3 Modifiche simultanee richieste da client diversi

Se più client dovessero interagire contemporaneamente con un singolo server è necessario gestire il caso in cui entrambi i client tentino di cancellare o modificare lo stesso contatto simultaneamente. Sicuramente uno dei due eseguirà l'operazione per primo e di conseguenza il secondo client tenterà di modificare o cancellare un contatto che non è più in rubrica; dunque l'operazione non avrà successo. Segnaliamo questo caso particolare con un esito specifico, in modo che il client possa comunicarlo all'utente.
Se viene aggiunto un contatto con gli stessi dati da due client simultaneamente, sicuramente il secondo client riceverà dal server l'errore che indica che un contatto con tali dati è già presente in rubrica. Quindi, questo non rappresenta un caso particolare dovuto all'interazione di più client ma è una semplice aggiunta di un contatto già presente.

1.6.4 Gestione dei Log

Eseguiamo il logging di ogni operazione che avviene sul server su un file di log. In particolar modo per l'inizio e la terminazione della connessione, per ogni operazione richiesta da un client e anche le operazioni del manager del server. Per semplificare il logging abbiamo creato una struttura dati e una funzione che la acquisisce come parametro. La struttura dati contiene informazioni utili come la data, l'ora, l'operazione eseguita, chi l'ha richiesta, l'esito e un eventuale messaggio aggiuntivo. La funzione legge le informazioni dalla struttura e le riscrive nel file *log.txt*. Per facilitare l'inizializzazione del messaggio da scrivere, abbiamo realizzato anche una funzione alla cui passare tutti i campi, per fare una sola chiamata.

1.6.5 Chiusura del server e manager delle credenziali

Per gestire correttamente la chiusura della socket del server abbiamo ridefinito il gestore del segnale SIGINT (ctrl-c), in modo che la chiuda esplicitamente prima di terminare il processo.
Invece, nel caso sia presente nella cartella *utility* l'eseguibile *serverManager*, anziché terminare il processo, il segnale SIGINT disabilita tale segnale momentaneamente (per evitare più di 1 gestore), genera un figlio e tramite *execl()* esegue *serverManager*. In questo modo non viene bloccato il server che resta in attesa di connessioni da parte dei client. Il processo server e il processo serverManager condividono entrambi lo stesso terminale, dove verrà stampato da quest'ultimo un menu testuale per scegliere l'operazione da eseguire.

```
Server utilities - Scegli operazione da eseguire:

[+] Aggiungi utente
[-] Rimuovi utente
[x] Esci dal menu
[S] Termina server

Selezionare un'opzione:
```

L'aggiunta e la rimozione degli utenti funzionano analogamente all'aggiunta e la rimozione di un contatto dal file rubrica, con la differenza che delle password viene salvato il loro valore restituito dalla funzione di hash. Procedura che altrimenti dovrebbe essere effettuata manualmente calcolandosi l'hash della password prima di scriverlo sul file tramite un editor testo con il potenziale rischio di errori.
La terminazione del server è realizzata tramite segnale *SIGUSR2*, mentre l'uscita dal menù da *SIGUSR1*. Il server quando riceve *SIGUSR1* riabilita *SIGINT* e il manager si chiude, mentre quando riceve *SIGUSR2* esegue la chiusura della socket e il processo server termina. Anche quando termina il processo padre, cioè il processo server che accettava le connessioni, i processi figlio generati precedentemente, che soddisfano le richieste dei client che sono ancora connessi, restano in esecuzione fino a che non saranno i client a scollegarsi.

2 Istruzioni per compilazione ed esecuzione

Per la compilazione sia del client che del server è necessario avere installato un compilatore C, preferibilmente GCC. All'interno di entrambe le cartelle, client e server, è presente un makefile per automatizzare la compilazione quindi sarà sufficiente digitare *make* sulla shell all'interno della directory interessata. Si può comunque compilare file specifici direttamente tramite gcc, in tal caso consigliamo di consultare il makefile per controllare le varie dipendenze tra i file. Una volta compilati i programmi sarà possibile eseguirli digitando sulla shell il nome del file eseguibile, eventualmente specificando, come parametri da riga di comando, i dati di rete: "*./client* ⟨*IPv4*⟩ ⟨*porta*⟩" oppure "*./server* ⟨ *porta* ⟩".
Si consiglia di compilare anche il *serverManager* nella directory *utility* tramite l'apposito makefile, in modo da, in caso di segnale SIGINT sul server, avere accesso alle operazioni per gestire le credenziali degli utenti.

2.1 Esecuzione in locale

Se si intende eseguire sia client che server localmente si consiglia di aprire due terminali assicurandosi che quello su cui sarà aperto il server abbia le permissioni necessarie sui file *credenziali.txt*, *rubrica.txt* e *log.txt*. Dopodiché eseguire, in qualsiasi ordine, il client su un terminale e il server sull'altro, senza specificare parametri di rete. Per autenticarsi nel client possono essere usate le credenziali: username = "user" e password = "password", oppure se ne possono aggiungerne altre tramite il *serverManager*.

3 Esempi di esecuzione

Il client interagisce con il server attraverso menu testuali che acquisiscono i dati necessari e comunicano gli esiti delle operazioni all'utente. Di seguito alcuni esempi di esecuzione:

Lettura

```
Individuata corrispondenza con successo

Corrispondenza 1:
  Nome: Mario
  Cognome: Rossi
  Numero di telefono: 347 908 1326


============================
       MENU DI LETTURA
============================

[1] Inserire dei dati da cercare differenti
[2] Leggere corrispondenza successiva con gli stessi dati
[x] Terminare la lettura

Selezionare un'opzione: 2
```

```
Non sono state individuate ulteriori corrispondenze con i dati inseriti

============================
       MENU DI LETTURA
============================

[1] Inserire dei dati da cercare differenti
[x] Terminare la lettura

Selezionare un'opzione:
```

Aggiunta

```
========================================
      ACQUISIZIONE DATI AGGIUNTA
========================================

Digitare i dati del contatto (non possono essere lasciati vuoti):

  Nome: Giulia
  Cognome: Bianchi
  Numero di telefono: 3479804671
```

```
Contatto digitato:
  Nome: Giulia
  Cognome: Bianchi
  Numero di telefono: 347 980 4671


============================
      MENU DI AGGIUNTA
============================

[1] Procedere con l'aggiunta del contatto
[x] Annullare aggiunta del nuovo contatto

Selezionare un'opzione:
```

Autenticazione

```
============================
      AUTENTICAZIONE
============================

Tentativi rimanenti: 3

Username: user
Password:
```

```
Credenziali non valide

============================
      AUTENTICAZIONE
============================

[1] Ritenta l'autenticazione
[any other key] Torna al menu principale

Selezionare un'opzione:
```

```
Autenticato con successo

============================
      GESTIONE RUBRICA
============================

[1] Leggere rubrica
[2] Modificare rubrica
[x] Terminare sessione

Selezionare un'opzione:
```

Cancellazione

```
Individuata corrispondenza con successo

Corrispondenza 1:
  Nome: Luigi
  Cognome: Verdi
  Numero di telefono: 347 890 3216


================================
      MENU DI CANCELLAZIONE
================================

[1] Inserire dei dati di ricerca differenti
[2] Leggere contatto successivo con gli stessi dati
[3] Procedere con l'eliminazione
[x] Terminare l'eliminazione

Selezionare un'opzione: 3
```

```
Corrispondenza 1:
  Nome: Luigi
  Cognome: Verdi
  Numero di telefono: 347 890 3216


================================
      CONFERMA ELIMINAZIONE
================================

Vuoi procedere? Il contatto verrà eliminato permanentemente dalla rubrica:

[Y] si
[any other key] no

Selezionare un'opzione: y
```

Modifica dello stesso contatto da parte di due client

```
Corrispondenza 1:
  Nome: Mario
  Cognome: Rossi
  Numero di telefono: 347 908 1326


Contatto modificato:
  Nome: Simone
  Cognome: Costa
  Numero di telefono: 347 678 2431


============================
      CONFERMA MODIFICA
============================

Vuoi procedere? La modifica del contatto è irreversibile

[Y] si
[any other key] no

Selezionare un'opzione: y
```

```
Corrispondenza 1:
  Nome: Mario
  Cognome: Rossi
  Numero di telefono: 347 908 1326


Contatto modificato:
  Nome: Tommaso
  Cognome: Romano
  Numero di telefono: 347 567 4258


============================
      CONFERMA MODIFICA
============================

Vuoi procedere? La modifica del contatto è irreversibile

[Y] si
[any other key] no

Selezionare un'opzione: y
```

```
Contatto modificato con successo

============================
      MENU DI MODIFICA
============================

[1] Aggiungere nuovo contatto
[2] Cancellare contatto dalla rubrica
[3] Modificare contatto sulla rubrica
[x] Terminare la modifica

Selezionare un'opzione:

Modifica fallita, il contatto da modificare non è più in rubrica

============================
      MENU DI MODIFICA
============================

[1] Aggiungere nuovo contatto
[2] Cancellare contatto dalla rubrica
[3] Modificare contatto sulla rubrica
[x] Terminare la modifica

Selezionare un'opzione:
```