

UNIVERSITÀ DEGLI STUDI DI NAPOLI “PARTHENOPE”
FACOLTÀ DI SCIENZE E TECNOLOGIE
CORSO DI LAUREA IN INFORMATICA



DOCUMENTAZIONE RETI DI CALCOLATORI
Terminale Remoto

DOCENTE
Emanuel Di Nardo

CANDIDATO
Mario Mattia Senneca
matricola 0124002478
Giovanni Picardi
matricola 0124002641

Anno Accademico 2023-2024

Indice

1	Descrizione e schema dell'architettura	2
1.1	Schema	2
1.2	Architettura	2
1.2.1	Componenti dell'architettura	3
1.2.2	Flusso di Comunicazione	3
1.2.3	Flusso di Comunicazione	3
2	Dettagli implementativi dei client/server	4
2.1	Client	4
2.1.1	Connessione al server	4
2.1.2	Gestione dell'Input Utente	4
2.1.3	Comunicazione con il server	4
2.1.4	Autenticazione	5
2.1.5	Thread per la Comunicazione	5
2.2	Server	5
2.2.1	Ascolto delle Connessioni	5
2.2.2	Gestione delle Connessioni Multiple	5
2.2.3	Comunicazione con il Client	5
2.2.4	Autenticazione degli Utenti	6
2.2.5	Implementazione dei Comandi	6
3	Parti rilevanti del codice	7
3.1	Client	7
3.1.1	Connessione al server	7
3.1.2	Autenticazione e scelta utente	7
3.1.3	thread per la comunicazione	8
3.2	Server	9

3.2.1	Ascolto delle Conessioni	9
3.2.2	Autenticazione degli utenti	9
3.2.3	Implementazione dei comandi	10
3.2.4	Thread per le comunicazioni	10
4	Manuale utente con le istruzioni su compilazione ed esecuzione	12
4.1	Accesso	12
4.1.1	Registrazione	12
4.1.2	Login	13
4.2	Comandi	14
4.2.1	passwd	14
4.2.2	ls	15
4.2.3	cd, touch, mkdir	15
4.2.4	cp	16
4.2.5	mv	16
4.2.6	rm	17

Introduzione

Simulare un terminale remoto simile a Secure Shell (SSH). Utilizzare il modello client/server, con un server in grado di gestire connessioni multiple. La connessione è possibile solo per gli utenti registrati (utilizzare un file per memorizzare gli utenti).

Connessione dell'utente L'utente deve essere in grado di:

- Connettersi alla macchina remota tramite username (es. NomeApplicativo bob@127.0.0.1): Il server deve richiedere la password per procedere.
- Cambiare la propria password con il comando: Con il Comando: `passwd`.

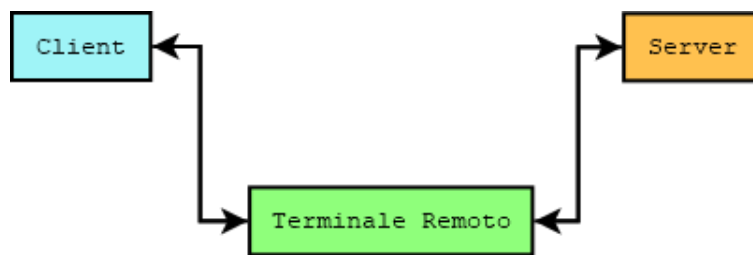
Funzionalità del terminale Il terminale deve permettere:

- La navigazione nel file system remoto: Con i Comandi: `ls`, `cd`.
- Creare file e directory: Con i comandi `mkdir` e `touch`.
- Copiare, rinominare, spostare e cancellare file: Con i Comandi `cp`, `mv`, `rm`.
- Uscire dalla sessione: Con il comando `exit`.

Capitolo 1

Descrizione e schema dell'architettura

1.1 Schema



- **Client:** Si collega al terminale remoto, scegliendo se registrarsi o fare il login. Inoltra la richiesta al server e aspetta la risposta.
- **Server:** Aspetta se un utente si registra o fa l'accesso. Verifica le autenticazioni e permette l'utilizzo dei comandi se esse sono corrette.

1.2 Architettura

Il sistema è progettato seguendo un'architettura client-server che simula un terminale remoto simile a Secure Shell (SSH).

L'architettura è basata su tre componenti principali: il client, il server e il file di memorizzazione degli utenti.

1.2.1 Componenti dell'architettura

- **Client:** Il client fornisce un'interfaccia utente attraverso la quale l'utente può interagire con il terminale remoto, inviando comandi al server e visualizzando i risultati restituiti.
- **Server:** Il server autentica gli utenti registrati verificando le loro credenziali memorizzate in un file di testo. Una volta autenticati, il server consente all'utente di accedere al terminale remoto e di eseguire comandi sul file system remoto.
- **File di memorizzazione degli utenti:** Gli utenti registrati sono memorizzati su un file di testo denominato `user.txt` situato sul server. Ogni riga del file contiene le informazioni dell'utente, come il nome utente e la password criptata. Quando un utente tenta di connettersi al server, il server legge il file `user.txt` per verificare le credenziali e autorizzare l'accesso.

1.2.2 Flusso di Comunicazione

Il flusso di comunicazione segue il modello client-server. Dopo aver stabilito una connessione, il client invia le credenziali dell'utente al server per l'autenticazione. Una volta autenticato, l'utente può inviare comandi al server tramite il client, che li esegue sul file system remoto e restituisce i risultati corrispondenti al client.

1.2.3 Flusso di Comunicazione

Il server è progettato per gestire più connessioni simultanee da diversi client. Questa gestione multi-threading consente al server di accettare e servire richieste concorrenti, garantendo una corretta esecuzione delle operazioni per ogni utente connesso.

Capitolo 2

Dettagli implementativi dei client/server

2.1 Client

2.1.1 Connessione al server

Il client stabilisce una connessione al server utilizzando la classe `Socket` di Java e specificando l'indirizzo IP e la porta del server.

2.1.2 Gestione dell'Input Utente

Il client accetta input dall'utente tramite la classe `BufferedReader` che legge dall'input stream del sistema (`System.in`). Questo input viene utilizzato per permettere all'utente di fare scelte come login o registrazione e per inviare comandi al server.

2.1.3 Comunicazione con il server

Il client invia comandi al server utilizzando un `PrintWriter` che scrive sull'output stream del socket. Riceve le risposte dal server tramite un `BufferedReader` che legge dall'input stream del socket.

2.1.4 Autenticazione

Se l'utente sceglie di fare il login o la registrazione, il client invia l'username e la password al server. Le risposte del server vengono visualizzate sull'output del client.

2.1.5 Thread per la Comunicazione

Il client utilizza due thread separati per gestire la comunicazione con il server. Un thread legge continuamente le risposte dal server e le visualizza sulla console del client, mentre l'altro thread legge continuamente l'input dall'utente e lo invia al server.

2.2 Server

2.2.1 Ascolto delle Connessioni

Il server utilizza un `ServerSocket` per ascoltare le connessioni in ingresso dai client sulla porta specificata.

2.2.2 Gestione delle Connessioni Multiple

Il server accetta connessioni multiple utilizzando un ciclo infinito. Per ogni nuova connessione accettata, il server crea un nuovo thread per gestire quella connessione in modo indipendente.

2.2.3 Comunicazione con il Client

Il server utilizza un `PrintWriter` per inviare messaggi al client e un `BufferedReader` per leggere i comandi inviati dal client.

2.2.4 Autenticazione degli Utenti

Se l'utente sceglie di fare il login o la registrazione, il server verifica le credenziali dell'utente e risponde di conseguenza. Se le credenziali sono valide, il server permette all'utente di accedere al terminale remoto.

2.2.5 Implementazione dei Comandi

Il server esegue i comandi inviati dal client sul file system remoto e invia i risultati al client. Questo avviene all'interno di un thread separato, che continua a leggere i comandi inviati dal client e a eseguirli.

Capitolo 3

Parti rilevanti del codice

3.1 Client

3.1.1 Connessione al server

```
1 // Creazione socket e connessione al server
2 Socket socket = new Socket("localhost", 12347);
```

3.1.2 Autenticazione e scelta utente

```
1 // Lettura del messaggio di benvenuto e visualizzazione
2 String welcomeMessage = serverReader.readLine();
3 System.out.println(welcomeMessage);
4
5 // Scelta utente e invio al server
6 String choice = userInputReader.readLine();
7 serverWriter.println(choice);
```

3.1.3 thread per la comunicazione

```
1 // Thread per la lettura delle risposte dal server
2 new Thread(() -> {
3     try {
4         String serverMessage;
5         while ((serverMessage = serverReader.readLine()) !=
6             null) {
7             System.out.println(serverMessage);
8         }
9     } catch (IOException e) {
10        e.printStackTrace();
11    }
12 }).start();
13
14 // Thread per l'invio di comandi al server
15 new Thread(() -> {
16     try {
17         String userInput;
18         while ((userInput = userInputReader.readLine()) !=
19             null) {
20             serverWriter.println(userInput);
21         }
22     } catch (IOException e) {
```

```
21         e.printStackTrace();
22     }
23 }).start();
```

3.2 Server

3.2.1 Ascolto delle Connessioni

```
1 // Creazione del ServerSocket e attesa di connessioni
2 ServerSocket serverSocket = new ServerSocket(12347);
3 while (true) {
4     Socket clientSocket = serverSocket.accept();
5     // Gestione della connessione in un nuovo thread
6 }
```

3.2.2 Autenticazione degli utenti

```
1 // Controllo delle credenziali dell'utente
2 if (!funzioni_server.isValidUser(username, password)) {
3     clientWriter.println(RED + "Errore: Utente non valido"
4         + RESET);
5     clientSocket.close();
6     continue;
7 } else {
```

```
7 // Azioni da eseguire se l'utente e' valido
8 }
```

3.2.3 Implementazione dei comandi

```
1 // Lettura e interpretazione dei comandi dal client
2 while ((clientMessage = clientReader.readLine()) != null) {
3     if (clientMessage.startsWith("ls")) clientWriter.
4         println(funzioni_server.ls());
5     else if(clientMessage.startsWith("cd ")) clientWriter.
6         println(funzioni_server.cd(clientMessage.substring
7             (3)));
8     // Altri comandi...
9 }
```

3.2.4 Thread per le comunicazioni

```
1 // Thread per la lettura delle richieste dal client e l'
2   esecuzione dei comandi
3 new Thread(() -> {
4     try {
5         // Lettura dei comandi inviati dal client e
6         // esecuzione
7     } catch (IOException e) {
8         e.printStackTrace();
9     }
10 }
```

```
7      } finally {  
8          // Eventuali operazioni di chiusura  
9      }  
10 }).start();
```

Capitolo 4

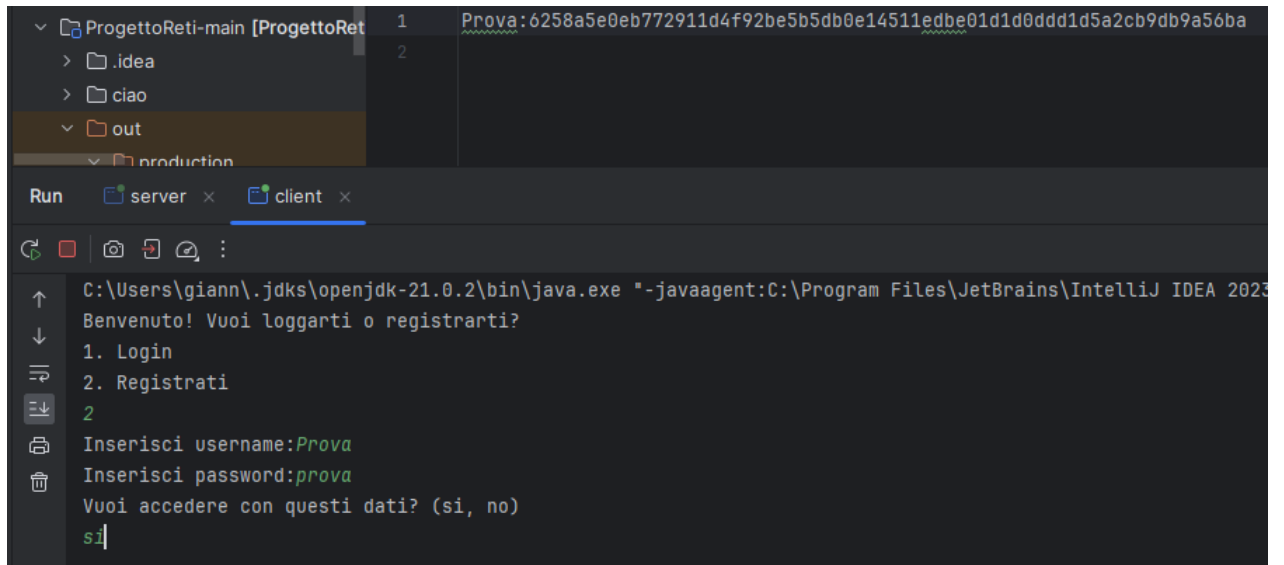
Manuale utente con le istruzioni su compilazione ed esecuzione

4.1 Accesso

4.1.1 Registrazione

Una volta avviato il terminale, l'utente si troverà davanti a due opzioni. Per la registrazione viene chiesto all'utente di inserire un username ed una password da utilizzare, dopodiché, il terminale chiederà all'utente se vuole accedere o meno con le credenziali appena create.

Le credenziali create dall'utente vengono memorizzate in un file `user.txt` e, per garantire la sicurezza dei dati, è implementato l'hashing delle password degli utenti.



The screenshot shows the IntelliJ IDEA interface. On the left, the Project Structure view displays a project named 'ProgettoReti-main' with subdirectories: '.idea', 'ciao', 'out', and 'production'. The 'Run' tab is active, showing two running processes: 'server' and 'client'. The 'client' process is selected, and its console output is visible in the bottom pane. The terminal shows the following text:

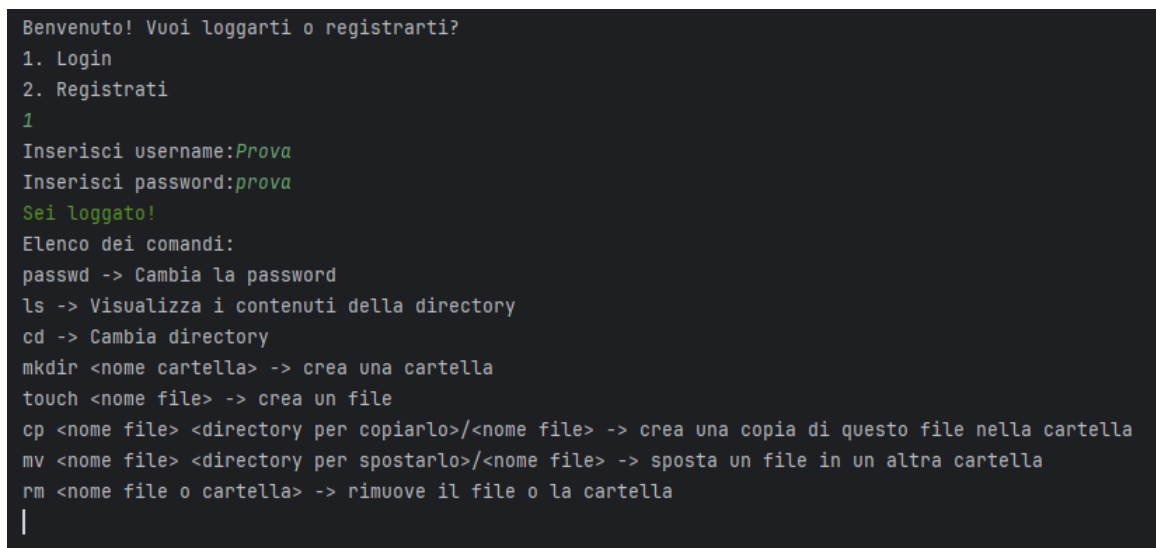
```

C:\Users\giann\.jdk\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023
Benvenuto! Vuoi loggarti o registrarti?
1. Login
2. Registrati
2
Inserisci username:Prova
Inserisci password:prova
Vuoi accedere con questi dati? (si, no)
si

```

4.1.2 Login

Una volta avviato il terminale, l'utente si troverà davanti a due opzioni. Per il login, l'utente dovrà inserire le proprie credenziali di accesso, ovvero il proprio username e la propria password. Successivamente si troverà davanti la schermata home, in cui gli saranno elencati tutti i possibili comandi utilizzabili.



The screenshot shows a terminal window with the following text:

```

Benvenuto! Vuoi loggarti o registrarti?
1. Login
2. Registrati
1
Inserisci username:Prova
Inserisci password:prova
Sei loggato!
Elenco dei comandi:
passwd -> Cambia la password
ls -> Visualizza i contenuti della directory
cd -> Cambia directory
mkdir <nome cartella> -> crea una cartella
touch <nome file> -> crea un file
cp <nome file> <directory per copiarlo>/<nome file> -> crea una copia di questo file nella cartella
mv <nome file> <directory per spostarlo>/<nome file> -> sposta un file in un'altra cartella
rm <nome file o cartella> -> rimuove il file o la cartella
|

```


4.2 Comandi

4.2.1 passwd

Il comando `passwd` consente ad un utente di modificare la propria password. Nel seguente esempio, l'utente ha effettuato l'accesso con la propria password "prova", dopodiché, tramite l'utilizzo del comando `passwd`, ha potuto aggiornare la propria password in "pippo".

```
Benvenuto! Vuoi loggarti o registrarti?
1. Login
2. Registrati
1
Inserisci username:Prova
Inserisci password:prova
Sei loggato!
Elenco dei comandi:
passwd -> Cambia la password
ls -> Visualizza i contenuti della directory
cd -> Cambia directory
mkdir <nome cartella> -> crea una cartella
touch <nome file> -> crea un file
cp <nome file> <directory per copiarlo>/<nome file> -> crea una copia di questo file nella cartella
mv <nome file> <directory per spostarlo>/<nome file> -> sposta un file in un'altra cartella
rm <nome file o cartella> -> rimuove il file o la cartella
passwd
Inserisci nuova password:
pippo
Password cambiata con successo
|
```

Come è possibile notare dall'immagine, l'utente è ora in grado di accedere al terminale con la nuova password "pippo".

```
Benvenuto! Vuoi loggarti o registrarti?
1. Login
2. Registrati
1
Inserisci username:Prova
Inserisci password:pippo
Sei loggato!
Elenco dei comandi:
passwd -> Cambia la password
ls -> Visualizza i contenuti della directory
cd -> Cambia directory
mkdir <nome cartella> -> crea una cartella
touch <nome file> -> crea un file
cp <nome file> <directory per copiarlo>/<nome file> -> crea una copia di questo file nella cartella
mv <nome file> <directory per spostarlo>/<nome file> -> sposta un file in un'altra cartella
rm <nome file o cartella> -> rimuove il file o la cartella
```

4.2.2 ls

Il comando `ls` consente di visualizzare i contenuti della directory in cui ci troviamo.

Come possiamo notare dall'immagine nella directory principale sono presenti i file e le cartelle del nostro progetto.

```
ls
Contenuto della directory:
.idea  ciao  out ProgettoReti.iml  Terminale Remoto  user.txt
```

4.2.3 cd, touch, mkdir

I comandi `cd`, `touch`, `mkdir` consentono rispettivamente di cambiare directory, creare un nuovo file e creare una nuova cartella.

Come possiamo notare nella directory principale non era presente il file `prova`. Una volta inserito il comando `"touch prova"` viene visualizzato il messaggio di creazione con successo.

Utilizzando il comando `"ls"` notiamo che nella nostra directory ora è presente il file creato precedentemente.

Inserendo il comando `"mkdir"` è possibile creare una nuova cartella.

Inoltre inserendo il comando "cd" possiamo notare che possiamo spostarci nelle diverse directory per utilizzare i prossimi comandi.

```
Contenuto della directory:
.idea  out ProgettoReti.iml  Terminale Remoto  user.txt
touch prova
File creato con successo
ls
Contenuto della directory:
.idea  out ProgettoReti.iml  prova  Terminale Remoto  user.txt
mkdir prova1
Directory creata con successo
ls
Contenuto della directory:
.idea  out ProgettoReti.iml  prova  prova1  Terminale Remoto  user.txt
```

4.2.4 cp

Il comando cp consente di copiare un file da una directory ad un'altra.

Come possiamo notare dall'immagine, inserendo il comando "cp prova prova1/prova"

È uscito il messaggio di file copiato con successo.

Il file è stato copiato dalla directory principale nella directory prova1

```
cp prova prova1/prova
File copiato con successo
ls
Contenuto della directory:
.idea  out ProgettoReti.iml  prova  prova1  Terminale Remoto  user.txt
cd prova1
Directory corrente: C:\Users\giann\OneDrive\Desktop\Reti di Calcolatori\ProgettoReti-main\.\prova1
ls
Contenuto della directory:
prova
```

4.2.5 mv

Il comando mv consente di spostare il file da una cartella all'altra.

Come possiamo notare dall'immagine nel nostro pat era presente il file prova2.

Inserendo il comando "mv prova2 prova1/prova2" il file è stato spostato dalla directory principale alla directory prova1.

```
Contenuto della directory:
.idea  out ProgettoReti.iml  prova  prova1  prova2  Terminale Remoto  user.txt
mv prova2 prova1/prova2
File spostato con successo
ls
Contenuto della directory:
.idea  out ProgettoReti.iml  prova  prova1  Terminale Remoto  user.txt
cd prova1
Directory corrente: C:\Users\giann\OneDrive\Desktop\Reti di Calcolatori\ProgettoReti-main\.\prova1
ls
Contenuto della directory:
prova2
```

4.2.6 rm

Il comando rm consente di rimuovere un file o una directory.

Scrivendo il comando "rm" possiamo rimuovere il file o la directory selezionata.

Possiamo notare dall'immagine che all'inizio era presente la directory "ciao" e dopo aver inserito il comando "rm ciao" la directory è stata rimossa.

Possiamo notarlo dal richiamo del comando "ls" che la directory è stata rimossa.

```
Contenuto della directory:
.idea  ciao  out ProgettoReti.iml  Terminale Remoto  user.txt
rm ciao
File o directory rimossi con successo
ls
Contenuto della directory:
.idea  out ProgettoReti.iml  Terminale Remoto  user.txt
```