Introduzione all'uso della Shell

La sintassi generale utilizzata per l'immissione dei comandi della Shell è:

```
comando [ opzioni ] [ argomenti ]
```

le parentesi quadre implicano che sia le opzioni che gli argomenti sono opzionali e possono essere omessi. I comandi più comuni eseguono una manipolazione di file e directory e accettano come argomenti percorsi dei file da creare, rimuovere, rinominare… Un percorso assoluto indica il file e tutte le directory che lo contengono, a partire dalla directory radice (/) ad esempio, sono percorsi assoluti:

- /home/utente/file.txt
- /home/utente
- /home/utente/Documenti/esercizio.c
- /usr/bin/gcc

I percorsi relativi invece assumono come punto di partenza la directory di lavoro corrente e sono, ad esempio:

- file.txt
- Documenti/esercizio.c
- Documenti/informatica/appunti.doc

si noti l'assenza del carattere '/' iniziale. Nell'uso dei percorsi relativi è spesso utile fare riferimento ad alcune directory speciali:

- `.` indica la directory corrente;
- `..` indica la directory che contiene quella corrente;
- `~` indica la directory personale dell'utente.

I comandi più comuni sono riassunti nella lista seguente: * **ls** il comando *list segments* elenca informazioni su i file che si trovano all'interno di una cartella, secondo la sintassi:

```
ls [opzioni] [file o cartella]
```

Un esempio di opzione utile è `-l`, che produce un elenco esteso con una riga per ogni file e per ognuno di essi un elenco di proprietà, come i permessi, le dimensioni e la data di ultima modifica.

- **cd** sta per *change directory* e viene utilizzato per cambiare la cartella corrente. Ad esempio, se la directory di lavoro è `/home/utente`, il comando `cd Documenti` fa in modo che la directory di lavoro diventi `/home/utente/Documenti`.

- **pwd** sta per *print working directory*, cioè *stampa la cartella di lavoro* ossia la cartella corrente. Utilizzando questo comando è possibile conoscere la cartella in cui ci si trova. Il risultato è un percorso, come ad esempio `/home/utente /Documenti`.
- **touch** viene utilizzato per creare un file vuoto, o per aggiornare la data di ultimo accesso di un file esistente. Ad esempio, il comando touch pippo . txt Crea il file pippo.txt nella directory corrente (se non esisteva prima).
- **rm** sta per *remove* e rimuove il file passato per argomento. Bisogna prestare attenzione al fatto che questo comando elimina un file in maniera definitiva, senza possibilità di recuperarlo. Ad esempio, il comando rm pippo . txt rimuove il file pippo.txt (se presente nella directory corrente).

- **mkdir** sta per *make directory* e crea una cartella nel percorso specificato come argo- mento. Ad esempio:

  ```
  mkdir prova
  ```

  crea la directory vuota dal nome prova.

- **rmdir** sta per *remove directory* e viene utilizzato per rimuovere le cartelle vuote specificate come argomento. Il comando:

  ```
  rm dir prova
  ```

  cancella la directory prova (se esistente e vuota).

- **cp** è un'abbreviazione per *copy* e, come si intuisce dal nome, è utilizzato per la copia di file e cartelle. La sintassi da utilizzare è:

  ```
  cp sorgente destinazione
  ```

- **mv** è un'abbreviazione per *move* e viene utilizzato per spostare un file da un percorso sorgente ad uno di destinazione secondo la sintassi:

  ```
  mv sorgente destinazione
  ```

  Se il percorso di sorgente e quello di destinazione differiscono solo per il nome del file, questo verrà semplicemente rinominato e rimarrà nella cartella iniziale.
- **cat** legge i file che gli sono specificati come argomento e li concatena e stampa il contenuto a video. Per questo motivo può essere utilizzato anche per ispezionare a video il contenuto di un file in maniera veloce.

- **man** permette di visualizzare a schermo il manuale di riferimento del comando che viene dato come argomento. Ad esempio, il comando:

  ```
  man ls
  ```

  mostrerà sullo schermo il manuale del comando ls.

Molti di questi comandi possono essere utilizzati per operare su file multipli quando i loro nomi seguono uno schema regolare. Ad esempio, il comando

```
rm *. c
```

cancella tutti i file nella directory corrente il cui nome termina con i caratteri `.c` (`prova.c`, `esempio.c`, `pippo.c` etc.). Il carattere asterisco rappresenta una sequenza di caratteri qualunque. Il simbolo `?`, invece, rappresenta un singolo carattere qualunque, quindi il comando:

```
cat file?.txt
```

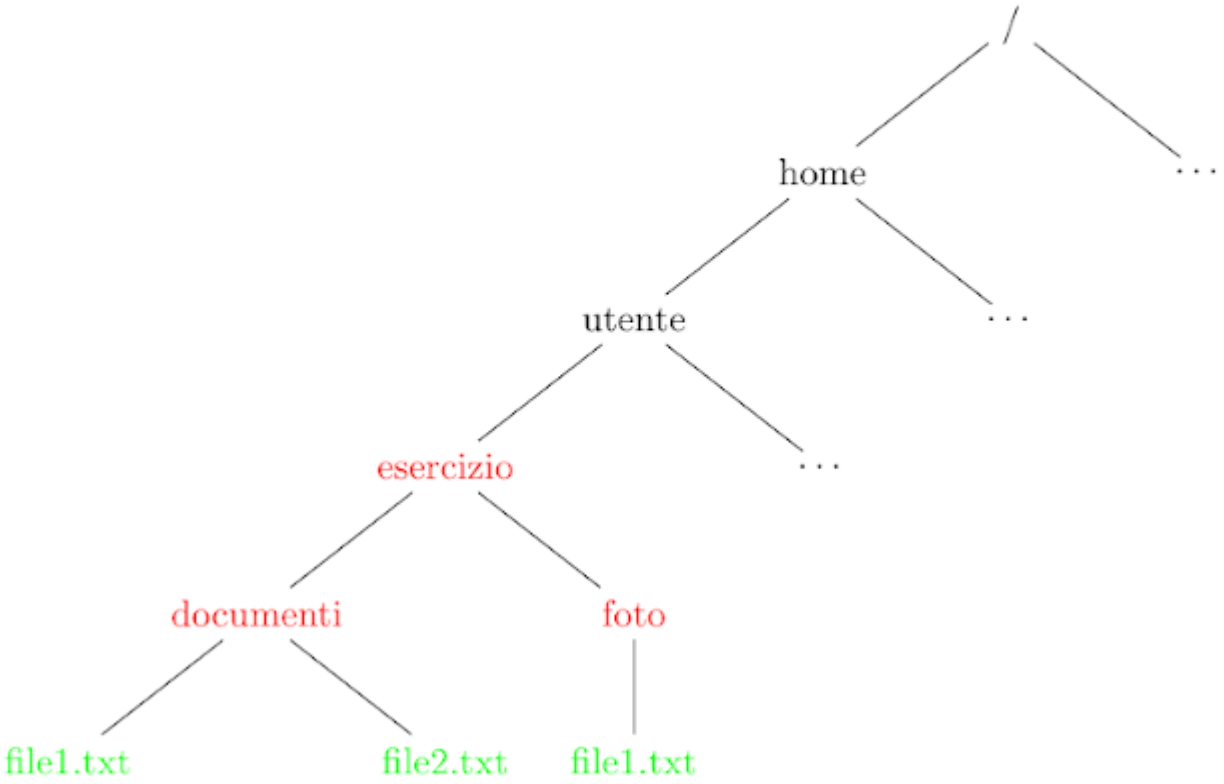
stamperebbe tutti i file nella directory corrente con nomi che seguono lo schema `file1.txt`, `file2.txt`, `fileA.txt`, `file+.txt` etc.

Esercizi

Seguono alcuni semplici esercizi di uso dei comandi riassunti nella sezione precedente. Tutti gli esercizi assumono che inizialmente la directory di lavoro sia `/home/utente`. Prima di iniziare si esegua il comando `pwd` per conoscere qual'è la directory di lavoro effettiva.

1. utilizzando i comandi `mkdir` e `touch` e come argomenti i percorsi relativi alla directory corrente, creare la directory `/home/utente/nome/utente/prova` e creare il file vuoto `/home/utente/nome/utente/prova/vuoto.txt`;
2. cancellare file e directory creati nell punto precedente con i comandi `rmdir` e `rm`;
3. ripetere i due punti precedenti utilizzando i percorsi assoluti invece di quelli relativi.

4. utilizzando i comandi `mkdir`, `touch` e aiutandosi con il comando `cd` creare la directory `esercizio` e le sue sottodirectory (in rosso) e i file (in verde) secondo lo schema:



5. cancellare directory e file creati nel punto precedente;
6. usando il comando `cp` copiare il file dal nome `stdio.h` che si strova nella directory `/usr/include`;
7. utilizzando il comando `cat` mostrare a video il contenuto del file `stdio.h`;
8. utilizzando il comando `mv` cambiare il nome del file copiato in `prova.h` e poi cancellarlo con il comando `rm`.

Comandi per lo sviluppo

Lo sviluppo di programmi in linguaggio C richiede due programmi specifici: * un editor per la scrittura dei listati, cioè dei file che contengono le istruzioni che il programma dovrà eseguire; * un compilatore che traduca le istruzioni del linguaggio di programmazione in istruzioni in linguaggio macchina che il calcolatore è in grado di eseguire. Nel seguito si assumerà l'uso dell'editor `micro` e del compilatore `gcc`.

La piattaforma SSHCode mette a disposizione anche una modalità a finestre attivabile con il comando:

```
fdi start
```

che mette a disposizione due terminali separati, ognuno in una finestra diversa. Ogni finestra è identificata da un nome che suggerisce il ruolo di quel terminale. La finestra “Help” mostra invece un riepilogo dei comandi utilizzabili nell’editor e sulla piattaforma. Si può cambiare la finestra corrente con il tasto F7 per spostarsi alla finestra successiva. Per tornare indietro si può usare la combinazione MAIUSC + F7. In alternative è possibile spostarsi in una finestra direttamente, cliccando con il mouse sul nome della finestra che compare nella barra colorato in basso.

Hello, World

Storicamente il primo programma che si realizza quando si impara un nuovo linguaggio è il programma Hello, World che stampa sul terminale la scritta “Hello, World”.

Con il comando

```
micro hello.c
```

si avvii l’editor micro nella finestra “Editor” in modo da consentire la scrittura del listato hello.c.

Si usi l’editor micro per inserire le seguenti linee nel listato

```
#include <stdio.h>
int main()
{
    puts ("Hello world!");
}
```

Dopo aver salvato il listato (CTRL + s) si usi il comando gcc nella finestra “Terminale” per compilare il programma:

gcc -Wall hello.c

Infine, sempre nella finestra “Terminale” si esegua il programma con il comando:

```
./a.out
```