LA SHELL

Il formato di immisione della maggior parte dei comandi segue questo pattern:

`comando [opzioni] argomenti`

I comandi più utili per la gestione di file sono:

- `pwd` - stampa la directory (cartella) corrente
- `ls` - elenca i file della direcotry corrente
- `cd directory` - spostati in un'altra directory
- `rm nomefile1 nomefile2 ...` - rimouvi uno o più file
- `mkdir nome_nuova_cartella` - crea nuova directory
- `rmdir` - rimuovi una directory vuota (se non vuota allora `rm -fR directory`)
- `cp file_da_copiare destinazione/` - copia un file in una nuova directory
- `mv file_da_spostare destinazione/` - sposta/taglia file ed eventualmente rinomina
- `man comando` - mostra il manuale di un comando

La struttura delle directory è gerarchica ed assomiglia molto ad un albero:
consideriamo la posizione del file `file.doc` come `/home/utente/file.doc`
la radice dell'albero è sempre `/`
un ramo successivo è `/home`
un ramo di `/home` è `/home/utente`
infine la foglia (il file) è `/home/utente/file.doc`
Questo modo di rappresentare il path (percorso) si dice assoluto perché parte sempre dalla radice.
Se invece ci trovassimo nella cartella `/home`, potremmo riferirci al file `files.doc` partendo dalla cartella corrente, ovvero `utente/file.doc`
Ad esempio, se noi ci trovassimo nella cartella `/home` e volessimo cancellare il file di prima, otterremmo lo stesso risultato con questi percorsi:

```
rm /home/utente/file.doc
rm utente/file.doc
```

Un'ultima considerazione importante riguarda le directory speciali ".", ".." e "∼":
la prima indica la directory corrente, la seconda è la directory padre, infine, "∼" è una scorciatoia per saltare alla directory home (principale) dell'utente loggato sul sistema.
In altre parole se si esegue `cd .` si rimane nella directory corrente, `cd ..` si torna indietro di una directory e `cd ∼` si salta alla home dell'utente (su Ubuntu per scrivere ∼ basta premere `F12`).

HELLO WORLD

Per scrivere del codice dobbiamo avviare un editor di testo, con Ubuntu possiamo eseguire da terminale `gedit hello.c &` dove `gedit` è l'editor e `hello.c` è il nome del file che andiamo a creare o modificare se esiste già.
Nel caso di Cygwin possiamo usare l'editor da riga di comando `nano hello.c`, ma è preferibile utilizzare un editor grafico di Windows come notepad o Notepad++.
Il codice del primo programma contiene alcuni elementi di base che verranno sempre riutilizzati in futuro (i commenti non sono indispensabili per il funzionamento del programma):

```
1 //questo è un commento
2 #include <stdio.h> //è una direttiva per includere il file header di una
     libreria che contiene alcune funzioni utili, e.g. printf()
3
4 int main() {// questo è il "punto d'inizio" del programma
5
```

```
6   printf("Hello world\n");
7     //printf() è una funzione della libreria stdio che stampa sul terminale
      ciò che è contenuto tra le virgolette del primo argomento della funzione (
      ovvero "Hello world\n")
8     //lo "\n" indica il carattere speciale new line, ovvero l'andata a capo
9
10  return 0; // Qui il programma termina e viene "restituito" uno zero al S.O.
11    //Qualsiasi codice  presente dopo il return non verrà eseguito
12
13 }
```

Una volta salvato il file (`Ctrl + S`) bisogna compilarlo in formato eseguibile dalla macchina: `gcc -Wall -o primo_prog hello.c`, dove l'opzione `-o` indica al compilatore `gcc` di chimare l'eseguibile generato "primo_prog" per comodità, mentre `-Wall` indica al acompilatore di mostrare tutti gli avvisi e non solo gli errori.
Per eseguire il programma è sufficiente digitare il percorso relativo dell'eseguibile rispetto alla directory ".", ovvero: `./primo_prog`
L'output del programma sarà stampato sul terminale:

```
$ Hello world
```

Le shortcut

Le shortcut sono delle scorciatoie da tastiera per velocizzare il lavoro e ridurre la possibilità di commettere errori. Particolarmente utili sono quelle del terminale:

- ↑ (tasto su) per ottenere l'ultimo comando eseguito in ordine cronologico
- ⇤⇥ (tab) per auto-completare il nome di file e comandi in corso di digitazione, ad esempio digito `gedit he`, premo ⇤⇥ ed il terminale completerà automaticamente `he` in `hello.c`

Ubuntu offre inoltre:

- Ctrl + Alt + T per aprire un nuovo terminale
- tenendo premuto Alt e premendo ripetutamente ⇤⇥ si passa da una finestra all'altra
- Ctrl + S per salvare il file che si sta modificando con gedit