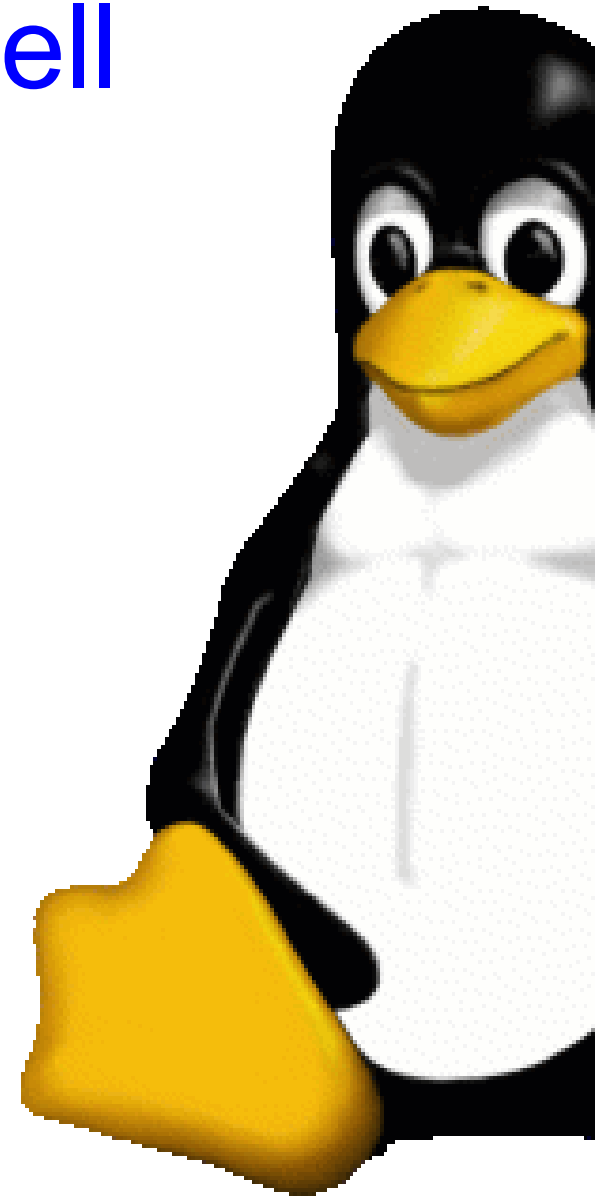
Introduzione alla Shell

LUG Ragusa

Solira
(Software Libero Ragusa)



- Mario Di Raimondo -

Cos'è una shell?

- Una interfaccia testuale attraverso cui impartire sequenze di comandi e lanciare programmi;
- l'interfaccia originale dei sistemi Unix;
- tutti i sistemi *Unix-like* (Linux, Mac OS X, BSD, ecc.) hanno una shell (più o meno nascosta);
- anche Windows ha una shell.

Perchè una shell?!

- Nonostante l'avvento di potenti e comode interfacce grafiche, l'uso della shell è ancora attuale:
  - “le abitudini sono dure a morire!”
  - velocità
  - potenza della sintassi....
  - rende l'amministrazione remota una banalità
  - “le interfacce cambiano... la shell resta sempre uguale!”
  - esistono tante shell: **bash**, csh, csh, ecc...

Gestire le directory

- **`pwd`**: ci da la directory corrente;
- **`cd`**: cambia la directory corrente;
- **`mkdir`**: crea una directory;
- **`rmdir`**: cancella una directory (già svuotata);

memoria corta? un po' di aiuto?!

- **`man`**: ti da una pagina di manuale su ogni comando.

Gestire i file

- **`ls`**: lista i file di una directory;
- **`cp`**: copia uno o più file;
- **`mv`**: sposta uno o più file (ma può anche rinominare un file);
- **`rm`**: cancella uno o più file;
- **`chmod`**: cambia i permessi di accesso ad un file.

“Keep it simple, stupid!”

- La filosofia di Unix si basa su un principio modulare:
  - ogni programma è specializzato in uno scopo ben preciso;
  - funzionalità avanzate si ottengono dalla combinazione dei programmi semplici;
- le funzionalità dei vari programmi si combinano con meccanismi tipo:
  - redirezione dell'input e dell'output;
  - *pipeline*.

Input, Output e redirezione

- Ogni programma ha i seguenti flussi di dati:
  - standard input
  - standard output
  - standard error
- si possono redirezionare:
  - redirezione dell'output:

    `nome_comando > destinazione`
  - redirezione dell'input:

    `nome_comando < sorgente`
  - redirezione dello standard error:

    `nome_comando 2> destinazione`

Pipeline e altri comandi

- lo standard output di un programma può essere messo in collegamento con lo standard input di un altro:

  `comando1 | comando 2`

- esempi di programmi specializzati:
  - **`wc`**: conteggia le parole o le linee;
  - **`sort`**: ordina le linee di un file o del flusso di dati;
  - **`gzip, gunzip`**: comprime e decomprime un file o un flusso di dati.

Bash scripting

- la bash è diventata un vero e proprio linguaggio di programmazione:
  - variabili
  - if / then / else
  - cicli for e while
  - definizioni di funzioni
- si raccolgono tutte le istruzioni in file testuali detti script che diventano dei veri e propri programmi
  - potenzialità enormi

Variabili

- esistono una serie di variabili standard che è possibile usare: HOME, PATH, SHELL, ecc.
- per ottenere il valore di una variabile si usa il prefisso **$**:

```
echo $HOME
```

- si possono assegnare proprie variabili:

```
nome_variabile= “valore della variabile”
```

Costrutti di programmazione

- if / then /else:

```
if [ condizione ]; then
   blocco di comandi
else
   altro blocco di comandi
fi
```

- ciclo for:

```
for nome_variabile in lista; do
   blocco di comandi
done
```

- ed altro: **`while`**, **`until`**, **`select`**, **`case`**, ...

Documentazione

- Il sistema è molto più complesso di quello mostrato;
- però, per fare cose interessanti non è necessario sapere tutto;
- la documentazione non manca:
  - tutorial
  - howto
  - guide