

# Introduzione ad UNIX e la Shell

# Sommario

- 1 Introduzione a Unix
  - Storia di Unix
  - Struttura di Unix
  - Struttura del file system
  - La shell
  - I comandi Unix

# Sommario

- 1 Introduzione a Unix
  - Storia di Unix
  - Struttura di Unix
  - Struttura del file system
  - La shell
  - I comandi Unix

## Storia di Unix (1)

Il primo sistema Unix fu sviluppato nei laboratori Bell AT&T alla fine degli anni sessanta. Unix fu progettato con le seguenti caratteristiche:

- ambiente di programmazione;
- semplice interfaccia utente;
- semplici utility che possono essere combinate per realizzare potenti funzioni;
- file system gerarchico (ad albero);
- semplice interfacciamento con i dispositivi;
- sistema *multi-utente* e *multi-processo*: più utenti possono collegarsi al sistema ed eseguire processi (istanze di programmi) contemporaneamente;
- indipendente dall' architettura.

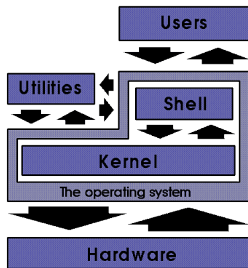
## Storia di Unix (2)

- Nel 1973 Unix è riscritto prevalentemente in **C**, un linguaggio di programmazione ad alto livello sviluppato da **Dennis Ritchie**.
- Dal 1974 Unix si diffonde prevalentemente in campo accademico grazie ad una licenza stipulata con le università per scopi educativi.
- **Come arriviamo a Linux? Richard Stallman** nel 1980 circa, iniziò a scrivere un sistema operativo chiamato GNU (GNU's Not Unix). Nel 1991 lo studente finlandese **Linus Torvalds** creò un kernel *unix-like* (conforme alla Single Unix Specification) e lo chiamò **Linux**. Il kernel Linux venne inserito dentro GNU dando vita così al sistema operativo libero GNU/Linux, più conosciuto come Linux.

# Sommario

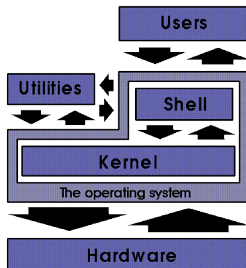
- 1 **Introduzione a Unix**
  - Storia di Unix
  - **Struttura di Unix**
  - Struttura del file system
  - La shell
  - I comandi Unix

# Unix in generale



- Le funzionalità di Unix sono organizzate logicamente a *strati*;
- Il *sistema operativo* gestisce le risorse fisiche (memoria, CPU, I/O) ed la memorizzazione dei dati (il *file system*);

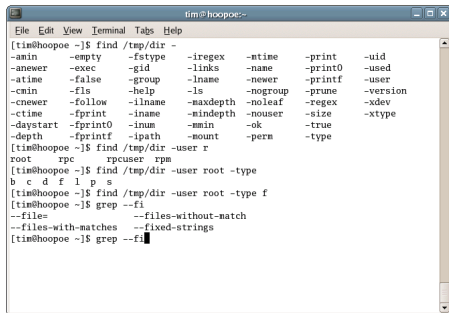
# Unix in generale



- il *kernel* realizza le operazioni su file e dispositivi e le politiche di gestione mentre
- la *shell* è un programma interprete che permette agli utenti di richiedere al kernel l'esecuzione di operazioni sul file system o sui dispositivi



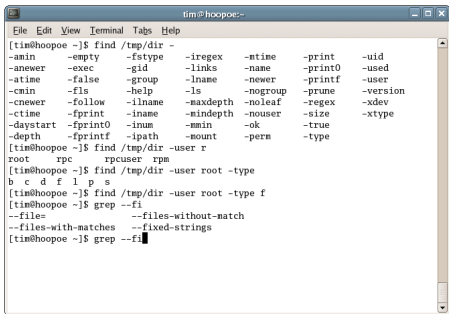
# Come è fatta una shell?



```
tim@hoopoe:~  
File Edit View Terminal Tabs Help  
[tim@hoopoe ~]$ find /tmp/dir -  
-anin      -empty      -fstype      -iregex      -mtime      -print      -uid  
-anewer    -exec        -gid         -links       -name        -print0     -used  
-atime     -false       -group       -lname       -newer       -printf     -user  
-cmin      -fls         -help        -ls          -nogroup     -prune      -version  
-cnewer    -follow      -lname       -maxdepth    -noleaf     -regex      -xdev  
-ctime     -fprint      -iname       -mindepth    -nouser      -size       -xtype  
-daystart  -fprint0     -inum        -mmin        -ok          -true  
-depth     -fprintf     -lpath       -mount       -perm        -type  
[tim@hoopoe ~]$ find /tmp/dir -user r  
root      rpc        rpcuser    rpm  
[tim@hoopoe ~]$ find /tmp/dir -user root -type  
b c d f l p s  
[tim@hoopoe ~]$ find /tmp/dir -user root -type f  
[tim@hoopoe ~]$ grep --fi  
--file=      --files-without-match  
--files-with-matches  --fixed-strings  
[tim@hoopoe ~]$ grep --f:
```

- Programma che fornisce una interfaccia testuale alle funzionalità del sistema;

# Come è fatta una shell?



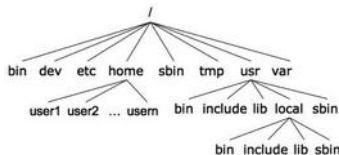
```
tim@hoopoe:~  
File Edit View Terminal Tabs Help  
[tim@hoopoe ~]$ man find  
-amin      -empty     -fstype    -iregex    -mtime     -print     -uid  
-anewer    -exec      -gid       -links     -name      -print0    -used  
-atime    -false     -group     -lname     -newer     -printf    -user  
-cmin     -fls       -help      -ls        -nogroup   -prune     -version  
-cnewer   -follow    -ilname    -maxdepth  -noleaf    -regex     -xdev  
-ctime    -fprint    -iname     -mindepth  -nouser    -size      -xtype  
-daystart -fprint0   -inum      -mmin      -ok        -true  
-depth    -fprintf   -ipath     -mout      -perm      -type  
[tim@hoopoe ~]$ man find -user r  
root      rpc      rpcuser  rpm  
[tim@hoopoe ~]$ man find /tmp/dir -user root -type  
b c d f l p s  
[tim@hoopoe ~]$ man find /tmp/dir -user root -type f  
[tim@hoopoe ~]$ man grep --fi  
--file=           --files-without-match  
--files-with-matches --fixed-strings  
[tim@hoopoe ~]$ man grep --fi
```

- Legge i comandi digitati dall'utente e li esegue (es. navigare il file system, creare file e directory, eseguire programmi).

# Sommario

- 1 **Introduzione a Unix**
  - Storia di Unix
  - Struttura di Unix
  - **Struttura del file system**
  - La shell
  - I comandi Unix

# Il file system (1)



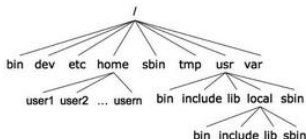
Un **file system** è il meccanismo fornito dal sistema operativo che regola l'organizzazione fisica e logica delle informazioni sui dispositivi (disco, cd-rom, dvd, ecc.).

In Unix, il file system è paragonabile alla struttura rovesciata di un *albero*

**Omogeneità:** in Unix tutto è un file (documenti, sorgenti di programmi, applicazioni, immagini...). Tre categorie di file: *ordinari*, *directory* e *dispositivi*.

## Il file system (2)

### Directory principali



Directory di sistema che si ritrovano in tutti i sistemi unix-like:

- **bin**: file eseguibili tipicamente da tutti gli utenti;
- **dev**: file speciali associati ai dispositivi (*device*);
- **etc**: file di configurazione;
- **home**: directory che contiene le home directory degli utenti;
- **sbin**: file eseguibili tipicamente dall'amministratore di sistema;
- **var**: utilizzata per il logging e lo spooling.

## Il file system (3)

### File & directory

Ogni nodo dell'albero è o un file o una directory di file, dove quest'ultima può contenere altri file e directory.

- Un **file** è una sequenza non strutturata di byte
- Una **directory** è un file che indicizza altri file

Un file, identificato da un **path name**, ed ha i seguenti attributi: tipo, permessi (diritti di accesso), nome utente proprietario, nome gruppo proprietario, dimensione, data di creazione, ultima modifica, ultimo accesso.

Il path name di un file o di una directory può essere **assoluto**, riferito alla radice della gerarchia ( / ), oppure **relativo**, riferito alla posizione dell'utente nel file system.

## Il file system (3)

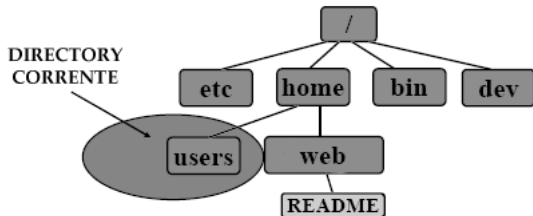
### File attributi

```
bash$ ls -l pippo.c  
-rw-r-r- 1 susanna users 1064 Feb 4 2012 pippo.c
```

- - tipo del file *regolare*
- **rw-r-r-** permessi del file (**r** lettura, **w** scrittura, **x** esecuzione)
- **susanna** owner
- **user** gruppo
- **Feb 4 2012** ultima modifica
- **1** numero di hard link
- **1064** dimensione

# Path assoluti/relativi

## Esempio



**NOME ASSOLUTO:** `/home/web/README`

**NOME RELATIVO:** `../web/README`



# Sommario

- 1 **Introduzione a Unix**
  - Storia di Unix
  - Struttura di Unix
  - Struttura del file system
  - **La shell**
  - I comandi Unix

# Tante shell

I sistemi Unix offrono diverse shell:

- **sh**: Bourne shell. La shell presente sui primi sistemi Unix.
- **bash**: shell di default per gli utenti Linux. È la shell di riferimento in questo corso.
- **csh**: La sintassi ricorda quella del linguaggio C. Richiesta, in alcuni casi, espressamente da programmatori.
- **tcsh**: “Turbo” csh. Estende la csh rendendola più “user-friendly”.
- **dash**: Debian Almquist shell. Una shell molto compatta usata in Debian e Ubuntu

Il file `/etc/shells` contiene l'elenco delle shell installate dall'amministratore e disponibili a tutti gli utenti.

## Perché usare una shell testuale?

- **Potenza e semplicità:** i comandi UNIX sono progettati per risolvere problemi specifici. Sono semplici (senza menù e opzioni nascoste) e proprio per questo potenti (es. `grep` [`<parola>`] [`<filename>`]).
- **Velocità e flessibilità:** è più veloce scrivere pochi caratteri da tastiera piuttosto che cercare un programma opportuno e usare le operazioni che fornisce sulla base delle proprie specifiche esigenze.
- **Accessibilità:** permette di accedere efficientemente ad un sistema in remoto.

# Sintassi dei comandi Unix

La sintassi tipica dei comandi UNIX è la seguente:

**comando** <opzioni> <argomenti>

- ogni comando può richiedere al kernel l'esecuzione di una particolare azione;
- i comandi esistono nel file system come file binari, generalmente eseguibili da tutti gli utenti.

<opzioni> sono facoltative e influiscono sul funzionamento del comando. Generalmente consistono nel simbolo del “-” seguito da una sola lettera.

<argomenti> si possono avere più argomenti o anche nessuno in base al comando.

## Ulteriori informazioni sulla shell

- funzione di autocompletamento (tasto TAB);
- history (freccia SU/GIU).

### Attenzione

I file system dei sistemi unix-like sono **case-sensitive**: maiuscole e minuscole sono importanti.

### Esempio

**file1**, **File1**, **FILE1**, **FiLe1** sono tutti nomi di file diversi.

# Sommario

- 1 Introduzione a Unix
  - Storia di Unix
  - Struttura di Unix
  - Struttura del file system
  - La shell
  - I comandi Unix

# Navigare nel filesystem

`cd` [`<dir>`] serve per modificare la *directory corrente*, e quindi a muoversi attraverso le directory.

Il parametro `<dir>` è opzionale — se non viene indicato, il comando porta nella home directory.

## Esempio

- Supponiamo che vogliamo accedere ai nostri documenti personali in `/home/user/documenti`
- se la directory corrente è la nostra home:  
`/home/user`
- per portarsi nella directory dei documenti basta eseguire:  
`cd documenti`
- per la navigazione risultano utili le directory: “.” (working directory), “..” (directory padre) e “~” (directory home).

## Visualizzare il contenuto di una directory

```
ls [-alsFR] [<dir1> ... <dirN>]
```

Se non viene specificata alcuna directory, si riferisce alla directory corrente.

Alcune opzioni:

- a** visualizza anche i file nascosti (il loro nome inizia per “.”);
- l** visualizza informazioni estese sui file (es. permessi, dimensione, owner, group);
- s** visualizza la dimensione in bytes;
- F** aggiunge un carattere finale al nome del file che ne denota il tipo (es. "nome/" indica una directory);
- R** visualizza ricorsivamente le sottodirectory (esegue ls ricorsivamente sulle subdir).



## Eliminazione di file

```
rm [-rif] <file1> ... <fileN>
```

Opzioni:

- r <dir> cancella la directory con il suo contenuto;
- i prima di cancella il file chiede conferma all'utente;
- f cancella senza chiedere conferma.

## Visualizzare il contenuto di un file

```
cat [-nve] <file1> ... <fileN>
```

Opzioni:

- n** precede ogni linea con un numero;
- v** visualizza i caratteri non stampabili eccetto newline, tab e form-feed;
- e** visualizza \$ alla fine di ogni linea (quando usato insieme con l'opzione -**v**);

**cat file1 file2 file3** concatena il contenuto dei file seguendo lo stesso ordine di immissione e ne mostrerà il contenuto;

**altri comandi:** **more <file>**, **less <file>**, **pg <file>** permettono di visualizzare il contenuto di <file> poco per volta.

## Creare una directory

```
mkdir [-p] <dir1> ... <dirN>
```

I parametri **dir** indicano i nomi (path assoluti o relativi) delle directory da creare.

Opzioni:

**-p** crea eventuali directory intermedie esplicitare nei parametri **dir**.

### Esempio

- **mkdir temp** — crea directory **temp** nella directory corrente.
- **mkdir -p documenti/personali** — crea le directory **personali** dentro la directory **documenti** (se **documenti** non esiste viene creata).

## Eliminare una directory (vuota)

```
rmdir [-p] <dir1> ... <dirN>
```

I parametri **dir** indicano i nomi (pathname assoluti o relativi) delle directory da eliminare.

Opzioni:

**-p** elimina eventuali directory intermedie esplicitate nei pathname dei parametri **dir**.

### Esempio

- **rmdir temp** — elimina la directory **temp** se è vuota.
- **rmdir -p documenti/personali** — elimina le directory **personali** e **documenti**, se entrambe vuote.

# Copiare file

```
cp [-if] <file1> <file2>
```

copia **file1** in **file2** — se **file2** esiste viene sovrascritto!

```
cp [-if] <file1> ... <fileN> <dir>
```

copia i **file** nella directory **dir** — se un **file** esiste in **dir** viene sovrascritto!

Opzioni:

- i chiede conferma prima di sovrascrivere;
- f non chiede conferma prima di sovrascrivere.

## Spostare file

```
mv [-if] <file1> <file2>
```

sposta **file1** in **file2** — se **file2** esiste viene sovrascritto!

```
mv [-if] <file1> ... <fileN> <dir>
```

sposta i **file** nella directory **dir** — se un **file** esiste in **dir** viene sovrascritto!

Opzioni:

- i** chiede conferma prima di sovrascrivere;
- f** non chiede conferma prima di sovrascrivere.

## Modificare diritti

```
chmod [ugoa][[+|=][rwx...]] <file1> <file2>
```

cambia i diritti di **file1** in **file2** secondo l'espressione specificata. Esempio:

```
chmod ugo+x <file1>
```

aggiunge all'owner, gruppo e altri il permesso di lettura. Ci sono formati diversi (ottale etc ...)

Opzioni:

-**R** ricorsivo, discende le sottodirectory

I diritti corretti per la propria home sono

```
rwX-X-X
```

perche?

## I metacaratteri (wildcards)

La shell Unix riconosce alcuni caratteri speciali, chiamati **metacaratteri**, che possono comparire nei comandi.

I più comuni:

?	qualsunque carattere
*	qualsunque sequenza di caratteri

### Esempio

Supponiamo di voler copiare tutti i file `.html` di una directory nella sotto-directory `html-src`. Usando la wildcard `*` (asterisco) si può scrivere semplicemente:

```
cp *.html html-src
```



## Nomi di file e convenzioni

- Esistono precise regole che stabiliscono i nomi con cui possono venire chiamati file e directory;
- Nomi con caratteri come /, \*, & e % devono essere evitati per evitare possibili errori di sistema;
- Anche utilizzare nomi composti da parole divise da spazi non è una buona abitudine;
- Nominare file o directory usando solo caratteri alfanumerici, lettere e numeri, uniti insieme da \_ (underscore) e . (punti).

## Il comando echo

Il comando **echo** stampa sullo schermo la stringa passata come parametro al comando.

### Esempi

```
$ echo Ciao!
```

```
Ciao!
```

```
$ ls
```

```
data-new data1 data2 inittab esempi1.txt
```

```
$ echo data*
```

```
data-new data1 data2
```

```
$ echo data?
```

```
data1 data2
```

# Redirezione

Di default i comandi Unix prendono l'input da tastiera (**standard input - stdin**) e mandano l'output ed eventuali messaggi di errore su video (**standard output - stdout**, **standard error - stderr**). L'input/output in Unix può essere rediretto da/verso file, utilizzando opportuni metacaratteri:

Metacarattere	Significato
>	ridirezione dell'output
>>	ridirezione dell'output (append)
<	ridirezione dell'input
<<	ridirezione dell'input dalla linea di comando

## Redirezione — Esempi

```
$ echo pippo Topolino > file.txt
```

```
$ cat file.txt
```

```
pippo Topolino
```

```
$ echo e anche Minnie » file.txt
```

```
$ cat file.txt
```

```
pippo Topolino e anche Minnie
```

```
$ cat list1 list2 > biglist
```

```
$ sort biglist > sortbiglist
```

# Pipe

Il metacarattere “|” (pipe) serve per comporre comandi “in cascata” in modo che l’output di ciascuno sia fornito in input al successivo. L’output dell’ultimo comando è l’output della pipeline (di default sullo standard output).

**command1 | command2** — l’output dell’esecuzione del primo comando viene passato come input del secondo comando.

## Esempio

```
ls | more
```

Effetto è quello di visualizzare l’output di **ls** una pagina per volta.

## Il comando SSH

SSH (*Secure SH*) è un comando che consente se eseguito sul computer **A** (*client*) di aprire una shell remota verso un computer **B** (*server*) su cui è in esecuzione un demone SSH.

```
ssh [opzioni] nomeutente@host [comando]
```

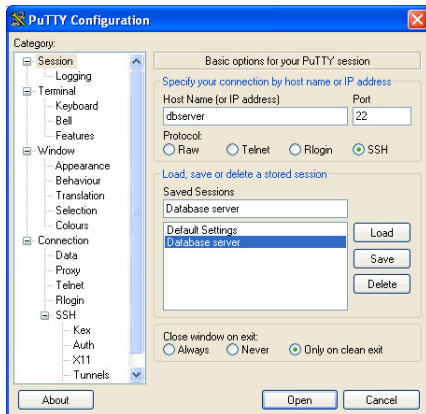
### Esempio

```
ssh mencagli@olivia.cli.di.unipi.it
```

Per uscire dalla shell remota e tornare alla shell locale digitare il comando **exit**.

# Client SSH per Windows

PUTTY è un client SSH per Windows che potete scaricare gratuitamente (cercatelo su Google)!



## Documentazione dei comandi

- `man comando`: mostra la pagina del manuale di `comando`, con istruzioni sull'uso e sulle opzioni disponibili, es. `man ls`;
- `man -k word`: ricerca le descrizioni di pagine di manuale che contengono "word", es. `man -k cat`;
- `apropos word`: cerca la stringa 'word' nelle pagine di manuale di tutti i comandi Unix. Utile per trovare il nome esatto di un comando che compie l'azione 'word';
- `whatis comando`: descrive la funzione di `comando`;
- `comando -help`.



## Altri comandi utili (1)

- **pwd (print working directory)** — visualizza il percorso assoluto della directory corrente;
- **head** — visualizza le prime linee di un file di testo  
es. `head -10 esempio.txt` visualizza le prime 10 righe di `esempio.txt`;
- **tail** — visualizza le ultime linee di un file di testo  
es. `tail -10 esempio.txt` — visualizza le ultime 10 righe di `esempio.txt`;
- **sort** — ordina le linee di un file di testo lessicograficamente  
es. `sort esempio.txt` — ordina le righe di `esempio.txt`.

## Altri comandi utili (2)

- **gzip/gunzip** — compressione/decompressione di file  
es. `gzip esempio.txt` — ottengo il file compresso `esempio.txt.gz`;
- **bzip2/bunzip2** — compressione/decompressione di file;
- **tar** — creazione/estrazione da archivi;
- **zip/unzip** e **rar/unrar** — creazione e estrazione di archivi compressi;
- **file <nome>** — visualizza il tipo del file `<nome>`, es.  
`file lezione1.pdf` — stampa  
`lezione1.pdf: PDF document, version X.X.`