

UNIX

Storia di UNIX

Nei '60, dominano *mainframe* da milioni di \$. Hanno SO:

- enormi (milioni di righe assembler)
- quindi poco efficienti e
- pieni di errori (correggendoli se ne introducevano altri!)

Alla fine degli anni 60, fallisce il progetto del mega SO **MULTICS**. Intanto compaiono i *minicomputer* (50% prestazioni, 5% costo di un *mainframe*).

Nel 1969 un ex progettista di MULTICS, **Ken Thompson** trova alla **Bell Labs** un minicomputer per cui scrive in assembler un SO: è una versione ridotta di MULTICS: **UNIX**.

Più avanti, Thompson e **Dennis Ritchie** riscrivono UNIX in **C**, linguaggio ad alto livello, molto conciso e potente.

Essendo ad alto livello, il C è indipendente dall'hardware ciò ha permesso di *portare* UNIX su pressoché ogni hardware, oggi PC, workstation, mini e super computer. UNIX, in versioni lievemente diverse o standard (POSIX), è disponibile su tutti e domina dalle workstation in su.

La prima sessione

Come prima cosa vi verrà richiesta la *Login*.

Accanto a

Login:

scrivete lo *user id* che vi è stato assegnato e con cui sarete noti al sistema

· il sistema risponde chiedendo

Password:

- i caratteri battuti come password sono invisibili;
 - se password corretta, il sistema mostra un *prompt*
- Errori su user id o password vengono notificati:

```
..
Login: utenet1
Password:
Login incorrect
```

La shell

- *Prompt*: risposta con cui il sistema si mostra pronto per altro input dopo aver elaborato input precedente (qui la password)
- Il prompt non viene da UNIX direttamente, ma da un programma detto *shell* che, come quelli che potete scrivere voi, gira **su** UNIX
- I caratteri battuti da voi sulla tastiera sono riprodotti sullo schermo sulla **riga di comando**, accanto al prompt.
- La riga di comando viene presa in considerazione dalla shell solo quando premete il tasto etichettato *Enter* o *Return*.
- Fino ad allora, il suo contenuto si trova in un'area provvisoria detta **buffer di input** e può essere corretto con il *tasto di cancellazione*.

Combinazioni di tasti

Alcuni tasti hanno effetti di *controllo*, p.es. le combinazioni ottenute

- premendo il tasto `Ctrl e`,
- *mentre lo si tiene premuto*, un'opportuna lettera, poi lasciandoli entrambi

Alcune combinazioni importanti sono:

- `Ctrl C`: interrompe l'esecuzione di un programma
- `Ctrl S`: blocca l'output che scorre sullo schermo
- `Ctrl Q`: sblocca l'output bloccato con `Ctrl S`

Opzioni dei comandi

La shell interpreta e fa eseguire i vostri comandi.

P.es. il comando `who` serve a vedere gli utenti del sistema

L'effetto di alcuni comandi si può modificare facendoli seguire da **opzioni**

Le opzioni sono costituite da un `-` seguito da uno o più caratteri.

Es: `ls -l`

Cambio password

```
Per cambiare password:
$ passwd
New password:ciao
Re-enter new password:ciao
Password is too easily broken. Try again.
New password:ciaocaro
Re-enter new password:ciaocaro
$ passwd
Old password:ciaocaro
New password:ciaocari
Re-enter new password:ciaocari
$ passwd
Old password:ciaocara
Sorry.
$
```

Il file system

Ogni file ha un nome (*nome semplice* nel seguito) di max *n* caratteri; sui primi Unix, *n=14*, caratteri leciti: `A..Z a..z 0..9 _ . ,`, UNIX distingue tra lettere maiuscole e minuscole: `nome?noMe.`
Esempi: `lezione.doc lezione.old file_mio 19.nov.92`

Un modo semplice per mettere nel file agenda i byte che rappresentano i caratteri domani vacanza è:

```
$ echo domani vacanza > agenda
```

Per vedere sullo schermo il contenuto di agenda:

```
$ cat agenda
domani vacanza
$
cat interpreta i byte come caratteri
```

Le directory

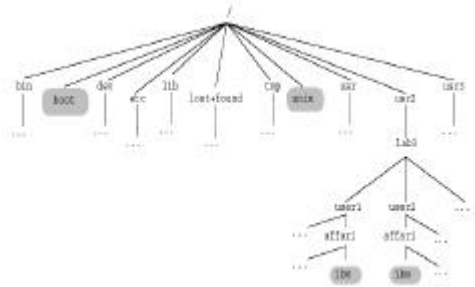
Ogni directory ha un solo *genitore*, in cui è contenuta e di cui si dice *figlia*.

La struttura delle directory si dice *gerarchica* perché la relazione *genitore-figlio* determina una *gerarchia*.

Di norma una directory come user1 si trova dentro altre.

Ma esiste una directory che non è contenuta in nessun'altra:

si chiama **root** (radice dell'albero) e si indica con la barra (slash) /



Root, dir corrente e cammini

Ad ogni istante è definita una **directory corrente o di lavoro**.

File e directory, intesi come *nodi* di un albero, non possono essere individuati *univocamente* con un *nome semplice*, come p.es. *ibm*.

Per questo ogni file o directory con nome semplice *f* ha un **nome completo** o **pathname** che può avere due forme:

- **assoluta** localizza *f* sull'albero *rispetto alla root*; è data dai nodi **da / fino a f** compreso, separati da /

p.es. /usr2/lab3/user1/affari/ibm

- **relativa** localizza *f* *rispetto alla directory corrente*,

è data dai nodi **dalla directory corrente esclusa fino a f** compreso, separati da /,

p.es. affari/ibm se la dir corrente è /usr2/lab3/user1/

NB: - un n. completo è assoluto se inizia per /, relativo altrimenti - nei nomi il carattere / ha 2 usi: (1) root e (2) separatore di dir

Muoversi tra directory

Il comando `pwd` scrive sul terminale la directory corrente.

P. es.

```
$ pwd
```

```
/usr2/lab3/user1
```

Il comando `cd` senza argomento rende la home dir corrente

serve a spostarsi nella directory di lavoro, in modo da dare nomi più brevi ai file di accesso più frequente

Il comando ls mostra il contenuto della directory corrente
Il formato completo di ls è:

```
ls [opzioni] [lista di file o dir]
```

Le opzioni più importanti sono:

```
ls -a elenca anche i file (normalmente invisibili) il cui  
nome comincia per .
```

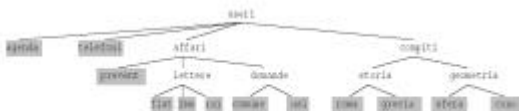
```
ls -l elenca in formato lungo
```

```
ls -t elenca a partire dal file più recente
```

```
ls -C incolonna l'elenco
```

```
ls -R elenca ricorsivamente anche le subdir
```

```
/usr2/lab3/user1 <120> ls -sC  
.login affari agenda compiti telefoni  
  
/usr2/lab3/user1 <121> ls -C affari  
domande lettere prevent  
  
/usr2/lab3/user1 <122> ls -C affari agenda  
agenda  
affari:  
domande lettere prevent  
  
/usr2/lab3/user1 <124> ls -l  
total 6  
drwxrwxrwx 1 user1 0 Nov 13 22:14 affari  
-rwxrwxrwa 1 user1 16 Nov 14 11:52 agenda  
drwxrwxrwx 1 user1 0 Nov 13 22:14 compiti  
-rwxrwxrwa 1 user1 895 Nov 14 11:52 telefoni  
  
/usr2/lab3/user1 <125> ls -lt  
total 6  
-rwxrwxrwa 1 user1 895 Nov 14 11:52 telefoni  
-rwxrwxrwa 1 user1 16 Nov 14 11:52 agenda  
drwxrwxrwx 1 user1 0 Nov 13 22:14 compiti  
drwxrwxrwx 1 user1 0 Nov 13 22:14 affari
```



si può usare il comando `ls -R dir`
`/usr2/lab3/user1 <126> ls -RC`
affari agenda compiti telefoni
affari:
domande lettere prevent
affari/domande:
comune usl
affari/lettere:
fiat ibm rai
compiti:
geometria storia
compiti/geometri:
cono sfera
compiti/storia:
grezia roma

Caratteri jolly: * e ?

- ? sta per qualsiasi carattere
- * sta per una sequenza arbitraria (di 0, 1, 2, 3...) caratteri.

```
/usr2/lab3/user1 $ echo sabato trippa > agenda1  
/usr2/lab3/user1 $ echo domenica gnocchi > agenda2  
/usr2/lab3/user1 $ ls -C agenda?  
agenda1 agenda2  
/usr2/lab3/user1 $ ls -C agenda*  
agenda agenda1 agenda2  
/usr2/lab3/user1 $ ls -C ag*  
agenda agenda1 agenda2
```

Le directory . e ..

I caratteri `.` e `..` sono nomi speciali di directory:
`.` rappresenta la directory corrente
`..` rappresenta la directory genitore di quella corrente

L'esempio sotto va letto ricordando il solito albero delle directory:

```
/usr2/lab3/user1 $ ls -C
affari agenda compiti telefoni
/usr2/lab3/user1 $ ls -C .
affari agenda compiti telefoni
/usr2/lab3/user1 $ cd aff*/let*
/usr2/lab3/user1/affari/lettere $ ls -C ../..
affari agenda compiti telefoni
/usr2/lab3/user1/affari/lettere $ cd ../lettere
/usr2/lab3/user1/affari/lettere $ cd ../dom*/../../user1
/usr2/lab3/user1 $
```

Creare e cancellare directory

`mkdir d` crea una directory di nome `d`
`rmdir d` cancella la directory `d` purché sia vuota

```
/usr2/lab3/user1 $ mkdir tmp
/usr2/lab3/user1 $ ls -l
total 6
drwxrwxrwx 1 user1 0 Nov 13 22:14 affari
-rwxrwxrwx 1 user1 1747 Nov 14 11:52 agenda
drwxrwxrwx 1 user1 0 Nov 13 22:14 compiti
-rwxrwxrwa 1 user1 895 Nov 14 11:52 telefoni
drwxrwxrwx 1 user1 0 Nov 15 11:58 tmp
/usr2/lab3/user1 $ mkdir tmp/d1/d2
mkdir: path not found for "tmp/d1/d2"
/usr2/lab3/user1 $ mkdir tmp/d1
/usr2/lab3/user1 $ ls -l tmp/d1
total 0
/usr2/lab3/user1 $ rmdir tmp/d1
/usr2/lab3/user1 $ ls -l tmp/d1
ls: File or directory "tmp/d1" is not found
```

Copiare file: cp

`cp f1 [f2 ...] dir` crea delle copie dei file `f1...` dentro `dir`

- `dir` deve esistere come directory
- se `f1` esiste già dentro `dir` viene sovrascritto

`cp f1 f2` crea una copia del file `f1` di nome `f2`

- `f2` deve essere un file o non esistere (altrimenti vedi caso precedente)
- se `f2` esiste già viene sovrascritto

```
/usr2/lab3/user1 $ ls -C
affari agenda compiti telefoni tmp
/usr2/lab3/user1 $ cp agenda telefoni tmp
/usr2/lab3/user1 $ ls -l tmp
total 6
-rwxrwxrwa 1 user1 0 17 Nov 15 23:40 agenda
-rwxrwxrwa 1 user1 0 895 Nov 15 23:40 telefoni
/usr2/lab3/user1 $ echo domani lavoro > agenda1
/usr2/lab3/user1 $ cp agenda1 tmp/agenda
/usr2/lab3/user1 $ ls -l tmp
total 3
-rwxrwxrwa 1 user1 0 15 Nov 14 23:44 agenda
-rwxrwxrwa 1 user1 0 895 Nov 14 23:40 telefoni
```

Con `cp -r f1 [f2 ...] dir`, se `f1..` è una directory, viene copiata ricorsivamente, cioè insieme alle sue subdirectory.

```
/usr2/lab3/user1 $ cp -r affari tmp
/usr2/lab3/user1 $ ls -RC tmp/affari
tmp/affari:
domande lettere prevent
tmp/affari/domande:
comune usl
tmp/affari/lettere:
fiat ibm rai
```

Spostare i file: mv

- `mv f1 [f2 ...] dir` sposta `f1...` dentro `dir`
- `dir` deve esistere come directory
 - se `f1` esiste già dentro `dir` viene sovrascritto
 - `f1 [f2 ...]` può essere una directory (verrà copiata ricorsivamente dentro `dir`)
- `mv f1 f2` cambia il nome di `f1` in `f2`
- `f2` deve essere un file o non esistere (altrimenti vedi caso precedente)
 - se `f2` esiste già viene sovrascritto

```
/usr2/lab3/user1 $ ls -C
affari agenda compiti telefoni tmp
/usr2/lab3/user1 $ echo ciao > tmp/saluto
/usr2/lab3/user1 $ ls -C tmp/saluto
tmp/saluto
/usr2/lab3/user1 $ mv tmp/saluto .
/usr2/lab3/user1 $ ls -C
affari agenda compiti saluto telefoni tmp
/usr2/lab3/user1 $ ls -C tmp/saluto
ls: File or directory "tmp/saluto" is not found
/usr2/lab3/user1 $ mv saluto tmp/file.ciao
/usr2/lab3/user1 $ ls -C
affari agenda compiti telefoni tmp
/usr2/lab3/user1 $ ls -C tmp/file*
tmp/file.ciao
/usr2/lab3/user1 $
```

Cancellare file: rm

- `rm f1 f2 ...` cancella (rimuove) i file `f1 f2 ...`
- `rm -r dir` cancella la directory `dir` insieme con tutte le subdirectory
- **Attenzione:** *non* c'è modo di recuperare i file cancellati!
- `rm *` e `rm -r dir` sono **molto** pericolosi!