Corso di Laboratorio di Sistemi Operativi

Alessandro Dal Palù

email: alessandro.dalpalu@unipr.it

web: www.unipr.it/~dalpalu

Qualche link utile

Informazioni sull'utilizzo della *shell* Unix/Linux

`www.pluto.it/files/ildp/HOWTO/Bash-Prog-Intro-HOWTO/Bash-Prog-Intro-HOWTO.html`

`www.pluto.it/files/ildp/guide/abs/index.html`

Google.it Google.com Google...!

La Shell

- La shell è la parte del sistema operativo dedita alla gestione dell'interazione con l'utente, ovvero, un'interfaccia a carattere:
  - l'utente impartisce i comandi al sistema digitandoli ad un apposito prompt (es: `> comando`);
  - il sistema stampa sullo schermo del terminale eventuali messaggi all'utente in seguito all'esecuzione dei comandi, facendo poi riapparire il prompt, in modo da continuare l'interazione.

- Versioni moderne forniscono X-Windows, un'interfaccia grafica (a finestre), che consente di inviare comandi tramite menu, utilizzando un mouse.

- X-Term è un emulatore di terminale che gira sotto X-Windows, fornendo localmente un'interfaccia a carattere.

Tipi di Shell

sh	Bourne shell
bash	Bourne again shell
csh	C shell
tcsh	Teach C shell
ksh	Korn shell

Quando viene invocata una shell, automaticamente al login o esplicitamente:

1. viene letto un file speciale nella home directory dello user, contenente informazioni per l'inizializzazione;
2. viene visualizzato un prompt, in attesa che l'utente invii un comando;
3. se l'utente invia un comando, la shell lo esegue e ritorna al punto 2; ad esempio, `echo $SHELL` stampa sullo schermo del terminale il percorso della shell di login, mentre il comando `bash` invoca la shell bash.

Per terminare la shell si possono usare i seguenti metodi:

- premere Ctrl-D;
- digitare i comandi `logout` o `exit`.

Caratteristiche Shell

- Ogni shell fornisce un linguaggio di programmazione. I programmi di shell sono denominati shell script.

- Uno shell script è un file di testo che contiene comandi che sono eseguiti come se l'utente li immettesse riga per riga.

- I comandi che la shell riconosce possono essere interni alla shell stessa (built-in), file eseguibili esterni o shell script.

- A parità di nome i comandi interni hanno la precedenza.

- A lezione facciamo riferimento alla *BASH*.

I comandi

- La sintassi tipica di un comando è:

  `> comando ⟨opzioni⟩ ⟨argomenti⟩`

- *Opzioni*: facoltative ed influenzano il comportamento del comando
  Generalmente consistono di un - seguiti da una lettera (es: `ls -a`)
  Per alcune opzioni esiste anche la forma estesa (es: `ls --all`)
  Alcune opzioni possono avere un argomento (es: `ls -T 2`)

- *Argomenti*: Si possono avere nessuno, uno o più argomenti.
  Alcuni argomenti sono facoltativi (es: `ls [directory]`).
  Se non specificati assumono un valore di default. (es: `ls`, `ls /etc`)

Esempi di comandi

```
> date                                  # visualizza la data corrente
> who          # visualizza gli utenti interattivi connessi al sistema
> uname -a                              # mosta informazioni sul sistema
> ps                                    # visualizza i processi attivi dell'utente
> ps ef                                 # visualizza tutti i processi
```

Diventare indipendenti: man

Consultazione del manuale on-line:

Section 0 - Everything
Section 1 - Commands
Section 2 - System Calls
Section 3 - Library Calls
Section 4 - Special Files
Section 5 - File Formats and Conversions
Section 6 - Games for Linux
Section 7 - Macro Packages and Conventions
Section 8 - System Management Commands
Section 9 - Kernel Routines

```
> man passwd
> man -a passwd
> man -s2 mkdir
> man man
```

Importante: per *uscire* da `man` digitare il tasto *q*

Micro esercizio (6 minuti)

- Aprire e chiudere una shell
- Provare a lanciare qualche comando suggerito
  (consiglio di scaricare il file pdf della lezione)
- Provare a consultare il manuale per vedere le opzioni di `ls`

Bash: edit riga di comando

- Ctrl-a : va ad inizio riga
- Ctrl-e : va a fine riga
- Ctrl-k : cancella il resto della riga
- Ctrl-y : reinserisce la stringa cancellata

La shell registra i comandi inseriti dall'utente.

- `history` : elenca i comandi digitati in precedenza
- `!10` : richiama il decimo comando dell'history
- `!!` : richama il comando precedente
- `!abc` : richiama l'ultimo comando immesso che comincia per abc
- ↑↓ : naviga nella history
- *Tab* : completa il comando o il nome del file parzialmente digitato

Bash: history list (I)

L'history list è un tool fornito dalla shell bash che consente di evitare all'utente di digitare più volte gli stessi comandi:

- bash memorizza nell'history list gli ultimi 500 comandi inseriti dall'utente;
- l'history list viene memorizzata nel file `.bash_history` nell'home directory dell'utente al momento del logout (e riletta al momento del login);
- il comando `history` consente di visualizzare la lista dei comandi:

```
$ history | tail -5
    511 pwd
    512 ls -al
    513 cd /etc
    514 more passwd
    515 history | tail -5
```

- ogni riga prodotta dal comando history è detta evento ed è preceduta dal numero dell'evento.

Bash: history list (II)

Conoscendo il numero dell'evento corrispondente al comando che vogliamo ripetere, possiamo eseguirlo, usando il metacarattere `!`:

```
$ !515
 history | tail -5
    512 ls -al
    513 cd /etc
    514 more passwd
    515 history | tail -5
    516 history | tail -5
```

Se l'evento che vogliamo ripetere è l'ultimo della lista è sufficiente usare `!!`:

```
$ !!
 history | tail -5
    513 cd /etc
    514 more passwd
    515 history | tail -5
    516 history | tail -5
    517 history | tail -5
```

Bash: history list (III)

Oltre a riferirsi agli eventi tramite i loro numeri, è possibile eseguire delle `ricerche testuali` per individuare quello a cui siamo interessati:

```
$ !ls
 ls -al
 total 491
 drwxr-xr-x  16 root    root      0 Oct 15 21:35 .
 drwxr-xr-x  16 root    root      0 Oct 15 21:35 ..
 -rw-r--r--   1 root    root  87515 Jul 10 04:28 Muttrc
 drwxr-xr-x   2 root    root      0 Oct 15 21:27 WindowMaker
```

In questo modo la shell comincia a cercare a partire dall'ultimo evento, procedendo a ritroso, nell'history list un comando che inizi con `ls`.

Racchiudendo con due caratteri `?` la stringa da ricercare (e.g. `$ !?ls?`), la shell controllerà che quest'ultima appaia in un punto qualsiasi del comando (non necessariamente all'inizio).

Bash: command completion (I)

Una caratteristica molto utile della shell bash è la sua abilità di tentare di completare ciò che stiamo digitando al prompt dei comandi (nel seguito `<Tab>` indica la pressione del tasto `Tab`).

`$ pass<Tab>`

La pressione del tasto `<Tab>` fa in modo che la shell, sapendo che vogliamo impartire un comando, cerchi quelli che iniziano con la stringa `pass`. Siccome l'unica scelta possibile è data da `passwd`, questo sarà il comando che ritroveremo automaticamente nel prompt.

Se il numero di caratteri digitati è insufficiente per la shell al fine di determinare univocamente il comando, avviene quanto segue:

- viene prodotto un suono di avvertimento al momento della pressione del tasto `Tab`;
- alla seconda pressione del tasto `Tab` la shell visualizza una lista delle possibili alternative.
- digitando ulteriori caratteri, alla successiva pressione del tasto `Tab`, la lunghezza della lista diminuirà fino ad individuare un unico comando.

Bash: command completion (II)

Oltre a poter completare i comandi, la shell bash può anche completare i nomi dei file usati come argomento:

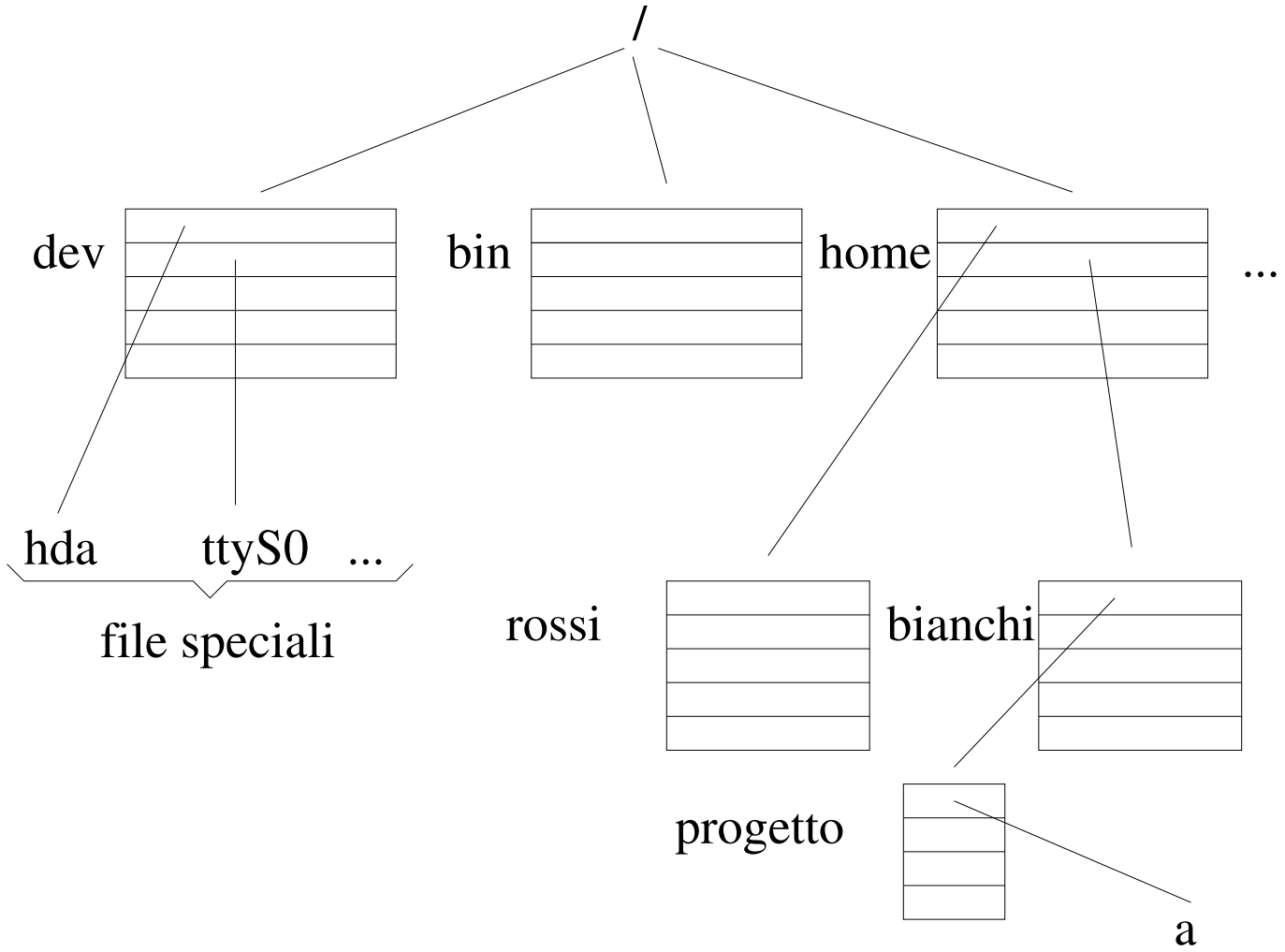
```
$ tail -2 /etc/p<Tab><Tab>
 passwd  printcap  profile

$tail -2 /etc/pa<Tab><Tab>

bianchi:fjKppCZxEvouc:500:500::/home/bianchi:/bin/bash
rossi:Yt1a4ffkGr02:501:500::/home/rossi:/bin/bash
```

In questo caso alla prima doppia pressione del tasto `Tab`, la shell presenta tre possibili alternative; digitando una `a` e premendo il tasto `Tab`, la shell ha una quantità di informazione sufficiente per determinare in modo univoco il completamento del nome di file.

Files in Unix

- Ordinari
- Directory
- Speciali

I file sono organizzati in una struttura gerarchica ad albero:



Le directory di Unix

/	E' la radice della gerarchia
/bin	Comandi per l'utente
/sbin	Comandi di amministrazione
/dev	Dispositivi di I/O
/etc	File di configurazione del sistema
/lib	Librerie
/var	File di dimensione variabile (es: logs, mailbox)
/usr	Programmi e applicazioni
/home	Directory personali degli utenti
/proc	Contiene informazioni dinamiche di sistema

Le directory . e ..

Ogni directory contiene 2 directory speciali:

- `..` E' un riferimento relativo alla directory genitore
- `.` E' un riferimento relativo alla directory stessa

Esempi:

```
cd /home/user1
ls ../user2
cd ../user2
./prog1
```

Bash: working directory

- Ogni processo ha associato una directory di lavoro (Working Directory) che può essere visualizzata (`pwd`) e modificata dinamicamente (`cd newdir`).

- La directory di lavoro serve per evitare di doversi riferire ai file e alle directory del file-system in modo assoluto, ovvero specificando l'intero percorso a partire dalla radice (/).

Bash: home directory

- Ad ogni utente viene assegnata una Home Directory che è la parte del file-system destinata a contenere i file personali dell'utente.

- Questa directory non è modificabile dall'utente ed riferita come:

  `~username/`

- Se lo username non è specificato si intende lo username dell'utente che ha generato il comando. I seguenti comandi, lanciati dall'utente user1, sono equivalenti:

- `cd ~/bin` `cd ~user1/bin` `cd /home/user1/bin`

- Il comando cd senza argomenti sposta la Working dir. sulla Home dir.

Il pathname

Ci si riferisce ai file tramite il

pathname { assoluto (rispetto a root /)
         { relativo (rispetto alla directory corrente)

Esempio:

(assoluto) `/home/bianchi/progetto/a`

(relativo) `progetto/a` (supponendo di trovarsi nella directory `/home/bianchi`)

- Present working directory:

  > `pwd`

  `/home/bianchi`

- Change directory:

  > `cd /bin`
  > `cd` (sposta l'utente nella sua home directory)

- Per spostarsi nella directory di un livello superiore:

  > `cd ..`

Il pathname (cont)

- > pwd
  /home/bianchi
  > cd ./progetto (dove . è l'alias per la directory corrente)
  > pwd
  /home/bianchi/progetto

Comandi per manipolare file e directory

- Listing dei files:
  > `ls`
  > `ls -l`
  > `ls -a` (include nomi che cominciano con .)
  > `ls -al`
  > `ls -l /bin`

- Creazione/rimozione di directory:
  > `mkdir d1`
  > `rmdir d1`

- Copia il file `f1` in `f2`: > `cp f1 f2`

- Cancellazione file `f`: > `rm f`

- Sposta/rinomina il file `f1` in `f2`:
  > `mv f1 f2`

- `cp` e `mv` come primo argomento possono prendere una lista di file; in tal caso il secondo argomento deve essere una directory:
  > `cp f1 f2 f3 d1` (copia `f1`, `f2`, `f3` nella directory `d1`)

Un esempio d'uso del comando `ls`

Eseguendo il comando `ls -l /bin` si ottiene il seguente output:

```
lrwxrwxrwx    1 root     root            4 Dec  5  2000 awk->gawk
-rwxr-xr-x    1 root     root         5780 Jul 13  2000 basename
-rwxr-xr-x    1 root     root       512540 Aug 22  2000 bash ...
```

da sinistra a destra abbiamo:

1. tipo di file (`-` file normale, `d` directory, `l` link, `b` block device, `c` character device),
2. permessi,
3. numero di hard link al file,
4. proprietario del file,
5. gruppo del proprietario del file,
6. grandezza del file in byte,
7. data di ultima modifica,
8. nome del file.

I permessi dei file

Linux è un sistema multiutente. Ci sono 4 categorie di utenti:
root, owner (u), group (g), world (o)
L'amministratore del sistema (root) ha tutti i permessi (lettura, scrittura, esecuzione) su tutti i file. Per le altre categorie di utenti l'accesso ai file è regolato dai permessi:

```
> ls -l /etc/passwd
-rw-r--r--    1 root     root          981 Sep 20 16:32 /etc/passwd
```

Il blocco di caratteri `rw-r--r--` rappresenta i permessi di accesso al file. I primi 3 (`rw-`) sono riferiti all'owner. Il secondo blocco di 3 caratteri (`r--`) è riferito al group e l'ultimo blocco (`r--`) è riferito alla categoria world.

La prima posizione di ogni blocco rappresenta il permesso di lettura (`r`), la seconda il permesso di scrittura (`w`) e la terza il permesso di esecuzione (`x`). Un trattino (-) in una qualsiasi posizione indica l'assenza del permesso corrispondente.

N.B.: per "attraversare" una directory, bisogna avere il permesso di esecuzione su di essa.

Il comando `chmod`

L'owner di un file può cambiarne i permessi tramite il comando `chmod`:

- `> chmod 744 f1` (imposta i permessi del file `f1` a `rwxr--r--`)
  Infatti: `rwxr--r--` $\rightsquigarrow$ `111 100 100` $\rightsquigarrow$ `7 4 4` (leggendo ogni gruppo in ottale)

- `> chmod u=rwx,go=r f1` (produce lo stesso effetto del comando precedente)
  dove `u` rappresenta l'owner, `g` il gruppo e `o` il resto degli utenti (world)
  Inoltre:
  `+` aggiunge i permessi che lo seguono,
  `-` toglie i permessi che lo seguono,
  `=` imposta esattamente i permessi che lo seguono.
  Quindi l'effetto di `chmod g+r f1` è in generale diverso da `chmod g=r f1`.

Ulteriori comandi

- `umask` è utilizzato per assegnare i permessi di default ai nuovi file. Viene inserito tipicamente nei file di inizializzazione della shell.

- La sintassi è ottale, ma diversamente da `chmod` gli 1 indicano rimozione del permesso rispetto al default di Unix (666 per i file e 777 per le directory).

- Esempio: `umask 002` sottrae i permessi w alla terza tripletta (others) i nuovi file avranno permessi 664 mentre le directory avranno 775.

Ulteriori comandi

- Visualizzazione del contenuto di un file:

  ```
  > cat f1
  > more f1
  > tail f1
  > head f1
  ```

Esercizi

- Esplorate il vostro file system. Qual è il pathname della vostra home directory?
- Visualizzate i file della vostra home directory ordinati in base alla data di ultima modifica.
- Che differenza c'è tra i comandi `cat`, `more`, `tail`?
- Trovate un modo per ottenere l'elenco delle subdirectory contenute ricorsivamente nella vostra home.
- I seguenti comandi che effetto producono? Perché?

```
> cd
> mkdir d1
> chmod 444 d1
> cd d1
```

Editare un file

- Esistono vari programmi standard per editare un file
- Il programma `vi` esiste in qualunque sistema Unix/Linux
- E' molto compatto nell'utilizzo e si consiglia di impararne i rudimenti (`man vi`)
- Ci soffermiamo invece su un potente editor `emacs`.
- Questo editor è programmabile e noi lo usiamo nella sua forma più semplice.

Emacs

- Per creare/aprire un file da editare:
  `> emacs nome_file`

- Per caricare un file `Ctrl-x Ctrl-f` e scrivere il nome file

- Per modificare usare i tasti standard. Notare che funzionano le sequenze Ctrl per l'editing della riga di comando.

- Per salvare il file corrente `Ctrl-x Ctrl-s`

- Per uscire `Ctrl-x Ctrl-c`

Prepariamo le directory per le lezioni

- Utilizzando i comandi di gestione dei file e directory, creare la struttura richiesta per la lezione del giorno (sul sito del corso)

- Copiare i files per la documentazione, come spiegato nel file latex.txt

- Utilizzare Emacs per creare in latex un report degli esercizi svolti.