

Corso di Laboratorio di Sistemi Operativi

Lezione 2

Alessandro Dal Palù

email: `alessandro.dalpalu@unipr.it`

web: `www.unipr.it/~dalpalu`

Alias già visti:

1. `.` (directory corrente)
2. `..` (directory madre)

Anche l'alias `~user` sta per la directory home dell'utente `user`:

```
user> cd ~user          # equivale a cd /home/user
user> cd ~user/doc      # equivale a cd /home/user/doc
```

La shell scandisce la linea di comando impartita dall'utente sostituendo i caratteri alias prima di eseguire i comandi.

Il comando alias

Il comando `alias` serve per creare nuovi alias:

```
user> alias dir='ls -a'
user> dir
.  ..  .bash_history
user> alias
ls='ls -l'
```

Per rimuovere uno o più alias:

```
user> unalias dir ls
```

All'uscita dalla shell gli alias creati con il comando `alias` sono automaticamente rimossi.

Variabili in BASH

- Una variabile è un nome che può essere associato ad una **stringa**.
- Una variabile non definita è associata alla stringa vuota.
- Per definire una variabile: `nome_var=stringa`
- Per leggere il contenuto: `$nome_var`

Esempio:

```
>miofile = /home/studenti/nome.cognome/miofile.txt
```

```
>cat $miofile
```

- Il comando `set` visualizza l'elenco delle variabili definite

Variabili in BASH

- Le variabili hanno validità limitata nell'ambito della shell stessa.
- Per esportare le variabili e i loro valori anche ai processi generati dalla shell è necessario utilizzare il comando `export`.

Esempio:

```
> miofile = /home/studenti/nome.cognome/miofile.txt
```

```
> export miofile oppure
```

```
> export miofile =/home/studenti/nome.cognome/miofile.txt
```

Provare:

```
> test1=lsol # con e senza export
```

```
> bash
```

```
> echo $test1
```

- Per elencare le variabili esportate: `printenv`

Variabili

- Elenco di variabili a contenuto speciale:
- PATH Elenco di directory in cui vengono cercati gli eseguibili.
- TERM Denota il tipo di terminale usato per le sessioni interattive
- LANG stabilisce il quale lingua mostrare i messaggi.
- UID Numero dell'UID dell'utente
- HOME La directory Home dell'utente
- PWD La directory di lavoro dell'utente
- PS1 Prompt

Variabili di stato automatiche

Sono variabili speciali che servono per gestire lo *stato* e aggiornate *automaticamente* dalla shell. L'utente può accedervi solo in lettura.

Al termine dell'esecuzione di ogni comando, viene restituito un valore di uscita, **exit status**, uguale a 0, se l'esecuzione è terminata con successo, diverso da 0, altrimenti (codice di errore).

La variabile speciale `$?` contiene il valore di uscita dell'ultimo comando eseguito.

```
> cd
```

```
> echo $?
```

```
0
```

```
> cd abcdef
```

```
bash: cd: asdf: No such file or directory
```

```
> echo $?
```

```
1
```

Il comando `exit n`, dove `n` è un numero, usato all'interno di uno script, serve per terminare l'esecuzione e assegnare alla variabile di stato il valore `n`.

Variabili di stato automatiche

La `bash` mette a disposizione numerose variabili di stato; le principali sono:

Variabile	Contenuto
<code>\$?</code>	<i>exit status</i> dell'ultimo comando eseguito dalla shell
<code>\$\$</code>	PID della shell corrente
<code>\$!</code>	il PID dell'ultimo comando eseguito in background
<code>\$-</code>	le opzioni della shell corrente
<code>\$#</code>	numero dei parametri forniti allo script sulla linea di comando
<code>\$*</code> , <code>\$@</code>	lista di tutti i parametri passati allo script da linea di comando

In particolare `$$` viene usata per generare nomi di file temporanei che siano unici fra utenti diversi e sessioni di shell diverse, e.g., `/tmp/tmp$$`.

Il prompt

- Il Prompt è definito con la variabile PS1 (es.: > PS1 = "Prompt>")

- Alcune stringhe tipiche:

\$

"\u@\h \W> "

"[\t][\u@\h:\w]\\$ "

Escape Char	Significato
\d	la data (e.g., Tue May 26)
\h	l'hostname fino al primo '.'
\H	l'hostname
\s	il nome della shell, il nome base di \$0
\t	l'ora corrente nel formato 24-ore HH:MM:SS
\u	lo username dell'utente corrente
\w	la directory di lavoro corrente
\W	il nome di base della directory di lavoro corrente
!\	il numero cronologico (history number) di questo comando
\#	il numero di questo comando
\\$	se l'UID effettivo 0, un #, altrimenti un \$

Impostazioni della Bash

- BASH interattiva di login:
/etc/profile +
/etc/profile.d/*.sh +
~/.bash_profile (include spesso ~/.bashrc) |
~/.bash_login |
~/.profile
(Al logout: ~/.bash_logout)
- BASH Interattiva non di login: ~/.bashrc
- La shell di login viene lanciata dopo la procedura di accesso e ha generalmente il nome di `-bash` (`>echo $0`).
Le shell successive (interattive e non) avranno il nome `bash` (senza trattino)

Personalizzare la shell

- Modificare con emacs il file `~.bashrc`
- Aggiungere alcuni alias
- Modificare in modo permanente il prompt
- Impostare una maschera di permessi (`umask`)

Informazioni sui files

- Il comando `whereis` cerca la locazione di un programma tra i sorgenti, eseguibili e i manuali:
`whereis [-bms] comando`
- Il comando `which` cerca il path completo o l'alias di un comando.
- Altri comandi: `locate`, `type`, `file`.
- Esercizio: provare i comandi (es. `> whereis ls`) e consultare manuali/google per il loro utilizzo.

Archivi e compressione

- Il comando tar consente di gestire archivi:
tar -cf <nomearchivio> file(s)_da_archiviare # crea
tar -tf <nomearchivio> # mostra il contenuto
tar -xf <nomearchivio> # estrae un archivio
- Esempio: tar -xf miobackup.tar restore/
- I comandi gzip e gunzip permettono di comprimere/decomprimere un file (utilizzando la codifica Lempel-Ziv):
- Esempi: gzip myfile # genera myfile.gz,
gunzip myfile.gz # ripristina myfile
- Il comando tar -czf crea un archivio e lo comprime con gzip (.tar.gz)
- Il comando tar -xzf estrae un archivio compresso
- Esercizio: creare un backup della directory della lezione corrente

I Metacaratteri della Shell

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

Quando l'utente invia un comando, la shell lo scandisce alla ricerca di eventuali metacaratteri, che processa in modo speciale.

Una volta processati tutti i metacaratteri, viene eseguito il comando.

Esempio:

```
user> ls *.java
```

```
Albero.java          div.java      ProvaAlbero.java  
AreaTriangolo.java  EasyIn.java  ProvaAlbero1.java
```

Il **metacarattere** `*` all'interno di un pathname è un'**abbreviazione** per un nome di file. Il pathname `*.java` viene espanso dalla shell con tutti i nomi di file che terminano con la *stringa* `.java`.

Abbreviazione del Pathname

I seguenti metacaratteri, chiamati **wildcard** sono usati per **abbreviare** il nome di un file in un pathname, o per identificarne molti con un'unica stringa:

Metacarattere	Significato
*	stringa di 0 o più caratteri
?	singolo carattere
[]	singolo carattere tra quelli elencati
{ }	stringa tra quelle elencate

Esempi:

```
user> cp /JAVA/Area*.java /JAVA_backup
```

copia tutti i files il cui nome inizia con la stringa Area e termina con l'estensione .java nella directory JAVA_backup.

```
user> ls /dev/tty?
```

```
/dev/ttya /dev/ttyb
```

... esempi

```
user> ls /dev/tty?[234]
```

```
/dev/ttyp2 /dev/ttyp4 /dev/ttyq3 /dev/ttyr2 /dev/ttyr4  
/dev/ttyp3 /dev/ttyq2 /dev/ttyq4 /dev/ttyr3
```

```
user> ls /dev/tty?[2-4]
```

```
/dev/ttyp2 /dev/ttyp4 /dev/ttyq3 /dev/ttyr2 /dev/ttyr4  
/dev/ttyp3 /dev/ttyq2 /dev/ttyq4 /dev/ttyr3
```

```
user> mkdir /user/studenti/rossi/{bin,doc,lib}
```

crea le directory bin, doc, lib .

> `ls c?t` \Rightarrow `ls cat cbt cct cdt`

> `ls c*t` elenca tutti i file che iniziano per c e finiscono per t.

> `ls c[au]t` \Rightarrow `ls cat cut`

> `ls c[!a-z]t` elenca tutti i files di 3 caratteri che iniziano per c e terminano per t escludendo le lettere alfabetiche minuscole.

`c{ha,oa}t` rappresenta le stringhe chat e coat.

Il “quoting”

Il meccanismo del **quoting** è utilizzato per inibire l'effetto dei metacaratteri. I metacaratteri a cui è applicato il quoting perdono il loro significato speciale e la shell li tratta come caratteri ordinari.

Ci sono tre meccanismi di quoting:

- il metacarattere di **escape** `\` inibisce l'effetto speciale del metacarattere che lo segue:

```
user> cp file file\  
user> ls file*  
file      file?
```

- tutti i metacaratteri presenti in una stringa racchiusa tra **singoli apici** perdono l'effetto speciale.
- i metacaratteri per l'abbreviazione del pathname in una stringa racchiusa tra **doppi apici** perdono l'effetto speciale (tranne `$`, `'` e `\`)

```
user> echo *  
user> echo "*"  
user> cp prova "prova 1"
```

Esempio pratico quoting

- Un nome di file contiene uno spazio: es `file 1`
- ```
>cat file 1
cat: file: No such file or directory
cat: 1: No such file or directory
```
- ```
>cat "file 1"
...
```
- ```
>cat file\ 1
...
```

# Ridirezione dell'I/O

Ogni processo possiede un certo numero di file aperti con cui pu' leggere o scrivere dati. Ad ogni file aperto corrisponde un descrittore che e' identificato da un numero.

Esistono sempre almeno tre descrittori per ogni processo:

- 0 = standard input;
- 1 = standard output;
- 2 = standard error.

Per i processi interattivi (shell):

standard input  $\Rightarrow$  tastiera

standard output  $\Rightarrow$  video

standard error  $\Rightarrow$  video.

Le shell supportano la possibilita di redirigere questi canali di I/O verso file.

# Ridirezione dell'I/O

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   |
| 2>            | ridirezione dei messaggi di errore (bash Linux) |

Esempi:

```
user> ls LABS0 > temp
```

```
user> more temp
```

```
lezione1.aux lezione1.log lezione1.tex lezione2.dvi lezione2.tex
lezione1.dvi lezione1.ps lezione2.aux lezione2.log lezione2.tex
```

## ... esempi

```
user> echo ciao a tutti >file # ridirezione dell'output
```

```
user> more file
```

```
ciao a tutti
```

```
user> echo ciao a tutti >>file # ridirezione dell'output (append)
```

```
user> more file
```

```
ciao a tutti
```

```
ciao a tutti
```

Il comando `wc` (**word counter**) fornisce numero di linee, parole, caratteri di un file:

```
user> wc <progetto.txt
```

```
21 42 77
```

```
user> wc <<delim # here document
```

```
? queste linee formano il contenuto
```

```
? del testo
```

```
? delim
```

```
2 7 44
```

```
user> cat file_non_esistente
```

```
cat: file_non_esistente: No such file or directory
```

```
user> cat file_non_esistente 2>temp
```

# Pipe

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

La sequenza di comandi

```
user> ls /usr/bin > temp
```

```
user> wc -w temp
```

```
459
```

ha lo stesso effetto della pipeline:

```
user> ls /usr/bin | wc -w
```

```
459
```

I comandi `ls` e `wc` sono eseguiti in parallelo: l’output di `ls` è letto da `wc` mano a mano che viene prodotto.

Per visualizzare l’output di `ls` pagina per pagina

```
user> ls | more
```

# Metacaratteri comuni a tutte le shell (I)

| Simbolo | Significato                                                        | Esempio d'uso                    |
|---------|--------------------------------------------------------------------|----------------------------------|
| >       | Ridirezione dell'output                                            | <code>ls &gt;temp</code>         |
| >>      | Ridirezione dell'output (append)                                   | <code>ls &gt;&gt;temp</code>     |
| <       | Ridirezione dell'input                                             | <code>wc -l &lt;text</code>      |
| <<delim | ridirezione dell'input da linea di comando (here document)         | <code>wc -l &lt;&lt;delim</code> |
| *       | Wildcard: stringa di 0 o più caratteri, ad eccezione del punto (.) | <code>ls *.c</code>              |
| ?       | Wildcard: un singolo carattere, ad eccezione del punto (.)         | <code>ls ?.c</code>              |
| [...]   | Wildcard: un singolo carattere tra quelli elencati                 | <code>ls [a-zA-Z].bak</code>     |
| {...}   | Wildcard: le stringhe specificate all'interno delle parentesi      | <code>ls {prog,doc}*.txt</code>  |

# Metacaratteri comuni a tutte le shell (II)

| Simbolo | Significato                                                                                         | Esempio d'uso                                                        |
|---------|-----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| ;       | Sequenza di comandi                                                                                 | <code>pwd;ls;cd</code>                                               |
|         | Pipe                                                                                                | <code>ls   more</code>                                               |
| &       | Esecuzione concorrente                                                                              | <code>cmd1 &amp; cmd2</code>                                         |
|         | Esecuzione condizionale.<br>Esegue un comando se il precedente fallisce.                            | <code>cc prog.c    echo errore</code>                                |
| &&      | Esecuzione condizionale.<br>Esegue un comando se il precedente termina con successo.                | <code>cc prog.c &amp;&amp; a.out</code>                              |
| (...)   | Raggruppamento di comandi                                                                           | <code>(date;ls;pwd)&gt;out.txt</code>                                |
| #       | Introduce un commento                                                                               | <code>ls # lista di file</code>                                      |
| \       | 1 Protezione metacarattere seguente<br>2 Spezza riga a capo<br>(se è l'ultimo carattere sulla riga) | <code>ls file.\*</code><br><code>123 456\</code><br><code>789</code> |
| !       | Ripetizione di comandi memorizzati nell'history list                                                | <code>!ls</code>                                                     |



# Esercizi

- Scrivete un unico comando (con pipe) per
  - contare quanti studenti hanno un account sulla macchina (guardare le home)
  - contare quanti file sono presenti nella vostra home
  - fornire la lista dei file in `/dev` che sono composti da due caratteri alfanumerici seguiti da una cifra 0-9
- Qual è la differenza tra i seguenti comandi?  
`ls > ls`  
`ps | ls`  
`ls | ps`
- Quale effetto producono i seguenti comandi?
  - `cat file | sort | uniq`, dove `file` è il nome di un file;
  - `who | wc -l ;`
  - `ps -e | wc -l .`

# Esercizi (I)

- Definire il comando `ll` in modo che venga chiamato `ls -l`
- Quale stringa devo impostare per avere un prompt che indichi il path completo?
- Stampare a video l'ultimo comando lanciato (senza eseguirlo!).
- Scrivere un comando che fornisca il numero dei comandi distinti contenuti nella history list.

## Esercizi (II)

- Come si lancia un eseguibile solo se la compilazione ha avuto successo?
- Elencare tutti i vostri file che contengono una 'a'.
- Qual è l'effetto dei seguenti comandi?
  - `ls -R || (echo file non accessibili > tmp)`
  - `cat /etc/issue > temp.txt; cat /etc/issue >> temp.txt;`  
`cat temp.txt | uniq`
  - `(cd /bin ; pwd ; ls | wc -l )`