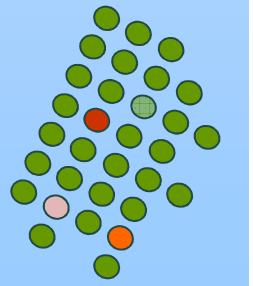
BASH:
Bourne Again Shell

(3)



Personalizzare l’ambiente

- bash fornisce 4 importanti strumenti

1. File speciali
   - *.bash_profile*, *.bash_logout, .bashrc* che sono letti da bash quando avviene il log-in o il log-out o quando viene aperta una nuova shell
2. Alias
   - Sinonimi per comandi
3. Opzioni
   - Controllano vari aspetti dell’ambiente
4. Variabili
   - Contengono valori che possono essere cambiati così che la shell o altri programmi possono avere comportamenti diversi secondo il contenuto di tali variabili

Variabili di ambiente

- Controllano il modo in cui la shell si comporta
- Forniscono informazioni ai processi

```
HOME= usr/local/home/rescigno
USER=rescigno
HOSTNAME=poseidon.dia.unisa.it
SHELL=/bin/bash
PATH=/bin:/usr/bin:/usr/local/bin/:.
EDITOR=vi
...
```

- Alcune sono assegnate automaticamente
- Possiamo definire nuove variabili
- Possiamo cambiare il valore delle variabili

File speciali

- File di inizializzazione globali: **/etc/profile**
- .bash_profile, .bashrc, .bash_logout
- .bash_profile:
  - eseguito ad ogni login

```
PATH= bin:/usr/bin:/usr/local/bin/:.
EDITOR=/usr/bin/vi
PS1=’\h:\w> ’
PS2=’> ’
export EDITOR
......
```

  - Qualsiasi modifica ha effetto al prossimo login
  - source .bash_profile
  - . .bash_profile

File speciali

- .bashrc:
  - eseguito per ogni subshell
- .bash_profile deve contenere comandi che devono essere eseguiti solo al login
- .bashrc deve contenere i comandi che vogliamo in ogni subshell
- .bash_logout
  - eseguito al logout

Alias

- alias nome=comando
  - bash esegue una sostituzione testuale quando incontra il nome di un alias

```
alias cerca=grep
    cerca adele numeri_telefonici.txt
alias lf='ls –F'
alias printall='cat *.txt | lpr'
alias mali=mail
alias emcas=emacs
```

- Non si possono usare parametri. Esistono le funzioni per questo

Alias

- alias
  - Senza argomenti restituisce la lista di tutti gli alias definiti
- alias nome
  - Senza il segno di = restituisce il valore di nome

Opzioni

- Cambiano il comportamento della shell

- set -o *nome* opzione diventa ON
- set +o *nome* opzione diventa OFF

- Alcuni esempi:
  - set -o emacs editor di linea: emacs
  - set -o ignoreeof CTRL-D non termina la shell
  - set -o noclobber non sovrascrive file
    - sort file1 > file2
  - set -o noblog non espande le wildcard nei nomi dei file
  - set -o nounset da errore se variabile non definita

Opzioni

- Per verificare lo stato delle opzioni dare
  - set -o

- shopt permette di controllare altre opzioni
  - shopt –s *nome* setta l'opzione
  - shopt –u *nome* unsetta l'opzione

  - Esempio: shopt –s cdable_vars

Variabili

- Definire una variabile: *nome =valore*
  - non ci devono essere spazi intorno al segno =
  - se *valore* contiene spazi bisogna usare "*valore*"

- Usare una variabile: ${*nome* }
  - le { } possono essere omesse se non ci sono ambiguita'
  - OK se il nome della variabile non e' seguito da
    - lettera, numero o underscore
    - $UID_pippo non va bene se vogliamo riferirci a UID
    - ${UID}_pippo

Variabili

- Cancellare una variabile: unset *nome*
  - una variabile che non e' definita vale "" (stringa nulla)
  - con l'opzione nounset, l'uso di una variabile non definita causa un errore

Variabili

```
bash> PATH=/bin:/usr/bin:/usr/local/bin/:.
bash> echo “il path e’ $PATH”
il path e’ /bin:/usr/bin:/usr/local/bin/:.
bash> echo ’il path e $PATH’
il path e’ $PATH
bash>
bash> nome=pippo.
bash> echo ${nome}txt
pippo.txt
bash> echo $nometxt
bash> set –o nounset
bash> echo $nometxt
bash: nometxt: unbound variable
bash>
```


Variabili di ambiente

Sono variabili predefinite (che possiamo cambiare)

- HOME — home directory
- PATH — ricerca dei comandi
- EDITOR — vi o emacs (normalmente)
- PS1 — prompt
- HISTFILE — nome del file per l'history (default .bash_history)
- PWD — current working directory (path)
- OLDPWD — directory precedente
- SECONDS — contati dall'inizio della shell
- IFS — separatori (internal field separator)
- ...

Variabile per il prompt

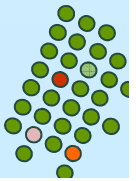
- Personalizzare il prompt
  - \d data
  - \h,\H hostname
  - \s nome della shell
  - \t,\T orario
  - \u nome utente
  - \v,\V versione di bash
  - \w,\W cwd
  - \!,\# il numero del comando (history)
  - \\ backslash
  - \$ se effective UID è 0 setta #, altr. $
- PS1="[\u@\h]> "
  - rescigno@udsab>

Usare piu' linee

- Comando = una linea
- Per usare piu' linee basta usare \ per indicare che si vuole continuare sulla linea successiva
- Se ci sono delle virgolette aperte e' automatico

```
bash> echo comando che  \
>usa due linee.
comando che usa due linee.
bash> echo "comando che
>usa due linee."
comando che
usa due linee.
bash>
```

- PS2 e' il prompt per le linee successive alla prima
  - di default è >



Variabili

- I processi che hanno come padre la shell possono accedere alle variabili di ambiente
  - HOME
  - PATH
  - PWD ...
- Altre variabili per essere variabili di ambiente devono essere "esportate" altrimenti non sono visibili
- export *nome*
- export *nome =valore*
- *nome =valore comando*
  - TERM=vt100 emacs file1

Script

- Uno script e' un file con comandi shell

```
#!/bin/bash
echo Questo e\' un script
echo I file in \/bin sono:
ls /bin
```

- Per eseguirlo
  - source *filename*
  - oppure
    - chmod +x *filename*
    - *./filename*
  - Il secondo metodo lancia una subshell ed esegue lo script nella subshell
- Prima riga #!*comando*, opzionale
  - viene usato *comando* per interpretare lo script

Parametri

- Quando gli script vengono invocati possono richiedere dei parametri
  - I parametri sono passati in variabili speciali: 0,1,2,....
- $1, $2, ..., ${10}, ${11}, ...
  - $0 e' il nome dello script
- $* e' una stringa contente tutti i parametri separati con il primo carattere contenuto nella variabile IFS
- $@ = "$1" "$2" ... "${N}“
  - cioè N stringhe separate da spazio
- $# = numero di parametri
  - ${$#} ultimo parametro
- Tali parametri sono a sola lettura; il loro valore non può essere modificato nello script