Introduzione alla shell

Ciro Mattia Gonano <ciro@winged.it>

0.1 Disclaimer

Vi sarei grato se riportaste qualsiasi errore, imprecisione, difficoltà trovaste nel testo. Contribuirete all'evoluzione della presente guida, e molta altra gente ve ne sarà grata ;-)

0.2 Ringraziamenti

La prima stesura del presente documento è avvenuta prendendo spunto per la sua quasi totale integrità dai lucidi del corso di "Laboratorio di Sistemi Operativi" del professor **Alberto Montresor**.

0.3 Nota alla presente versione

Questa versione (generata in data) non ha subito cambiamenti da due anni.

Questo implica che, come in ogni cosa che riguardi l'informatica, diverse cose potrebbero essere cambiate o migliorate; inoltre la presente guida è tutto fuorché finita.

Ho deciso di pubblicarla unicamente per dare un "la" a chi fosse interessato ad approfondire l'utilizzo della shell, ed eventualmente come spunto per chi volesse contribuire ad ampliare e completare la guida stessa.

Ovviamente i sorgenti sono disponibili su richiesta.

Indice

0.1 Disclaimer . . . 1
0.2 Ringraziamenti . . . 1
0.3 Nota alla presente versione . . . 1

1 Introduzione **4**
1.1 Cos'è la shell di Linux . . . 4
1.2 Prerequisiti . . . 4
1.3 Note tipografiche . . . 5

2 Nozioni di base **6**
2.1 Caratteristiche della shell . . . 6
2.2 Comandi built-in . . . 6
2.3 Metacaratteri . . . 7
2.4 Redirezione I/O . . . 7
Pipe, o catene di montaggio . . . 8
2.5 Wildcards . . . 8
2.6 Command substitution . . . 9
2.7 Sequenze . . . 9
Non condizionali . . . 9
Condizionali . . . 9
2.8 Raggruppamento di comandi . . . 9
2.9 Esecuzione in background . . . 9
2.10 Quoting . . . 10
2.11 Subshell . . . 10
2.12 Variabili . . . 10
2.13 Scripting . . . 11
2.14 Here document . . . 13

3 Programmazione della shell **14**
3.1 Valutazione delle espressioni . . . 14
3.2 Exit status . . . 14
3.3 Strutture di controllo . . . 15
3.4 Comandi built-in . . . 18
3.4.1 I/O . . . 18
3.4.2 variabili . . . 19
3.4.3 Script execution . . . 19
3.4.4 Comandi vari . . . 19
3.4.5 Directory stack . . . 20
3.5 File di configurazione . . . 20

3.5.1 Inizializzazione 20
3.5.2 Terminazione 21
3.6 Comandi esterni 21

A GNU Free Documentation License **22**
1. APPLICABILITY AND DEFINITIONS 22
2. VERBATIM COPYING 24
3. COPYING IN QUANTITY 24
4. MODIFICATIONS 25
5. COMBINING DOCUMENTS 26
6. COLLECTIONS OF DOCUMENTS 27
7. AGGREGATION WITH INDEPENDENT WORKS 27
8. TRANSLATION 27
9. TERMINATION 28
10. FUTURE REVISIONS OF THIS LICENSE 28
ADDENDUM: How to use this License for your documents 28

Capitolo 1

Introduzione

1.1 Cos'è la shell di Linux

La shell di Linux costituisce l'interfaccia tra il sistema e l'utente umano.

Consente di:

- poter interagire con il sistema a svariati livelli
- richiamare altri programmi
- programmare degli script[1]

In ambiente Linux abbiamo a disposizione svariati tipi di shells: *sh, ash, csh, bash, esh, lsh, kiss, pdksh, sash, psh, tcsh, zsh* tanto per citarne solo alcune. Alcune sono derivate (e potenziate) da altre[2], mentre altre differiscono in maniera notevole, sia nell'uso che nell'impostazione verso l'utente.

Per la grande varietà presente, qui ci concentreremo sull'utilizzo di *bash*, che è ormai lo standard de facto sui sistemi Linux[3].

1.2 Prerequisiti

Lo shell scripting non è materia astrusa, ma nemmeno banale. Nel corso di queste pagine daremo per scontate alcune nozioni che riteniamo fondamentali per un utente Linux, su cui non ci soffermeremo per non appesantire il documento. In particolare daremo per scontata la conoscenza di:

- nozioni di login/logout;
- organizzazione del filesystem (file e directory, gerarchie, pathname assoluti e relativi);
- comandi di base:
  - listare il contenuto di una directory (`ls`);

[1] gli script sono programmi in formato testuale formati da sequenze di comandi, fondamentali per ottimizzare e semplificare la vita all'amministratore del sistema

[2] come *bash* è una versione potenziata di *sh*

[3] attenzione: NON Unix/BSD, per i quali lo standard è in genere *csh* o una sua derivata

  - maneggiare i file (`cp`, `mv`, `rm`, `mkdir`, `rmdir`);
  - visualizzare il contenuto di un file (`cat/more/less/head/tail`);

- attributi dei file (`read/write/execute`)
- conoscenza del significato di permesso (`chmod`)
- gestione dei gruppi e dei proprietari dei file (`chgrp/chown`)

per cui, se non vi sentite sicuri di questi concetti, vi consigliamo di rivederli prima di avvicinarvi al bash scripting.

1.3 Note tipografiche

Per facilitare la lettura e la comprensione del testo, sono state utilizzate le seguenti convenzioni tipografiche:

- lo stile `macchina da scrivere` è utilizzato per i comandi, le variabili, le parole chiave e gli script;
- in *corsivo* le definizioni di nuovi termini, gli argomenti dei comandi e i nomi dei file;

Capitolo 2

Nozioni di base

2.1 Caratteristiche della shell

Ogni shell (non solo bash) presenta una serie di caratteristiche, raggruppabili in macrocategorie:

- comandi built-in;
- metacaratteri;
- redirezione di I/O;
- command substitution;
- sequenze condizionali e non;
- raggruppamento di comandi;
- esecuzione in background;
- quoting;
- subshell;
- variabili (locali/d'ambiente);
- scripting.

Passiamo a vederli in generale.

2.2 Comandi built-in

Quando richiedete l'esecuzione di un comando esterno, la shell si occupa di richiamare il corrispondente file eseguibile, caricarlo in memoria ed eseguirlo (ad esempio, quando chiamiamo il comando `ls`, *bash* si occupa di trovare e lanciare `/bin/ls`).

Al contrario, quando richiedete un comando che possa essere riconosciuto ed eseguito direttamente dalla shell, fate uso di un comando *interno*, detto anche *built-in*.
Esempi:

```
echo          # stampa tutti i suoi argomenti sullo standard output
cd            # cambia directory
```

2.3 Metacaratteri

La shell dispone di una serie di caratteri che non sono considerati nella loro forma a video, ma servono a specificare serie di funzionalità interne alla shell. Per questo motivo quando volete digitare uno di tali caratteri dovete solitamente farne l'*escape* con il carattere *backslash* ("\").

I metacaratteri possono essere trattati in maniera leggermente diversa a seconda della shell utilizzata, ma in generale sono questi[1]:

- `>, >>, <` redirezione I/O
- `|` pipe
- `*, ?, [...]` wildcards
- `` `command` `` command substitution
- `;` esecuzione sequenziale
- `||, &&` esecuzione condizionale
- `(...)` raggruppamento comandi
- `&` esecuzione in background
- `", '` quoting
- `#` commento
- `$` espansione di variabile
- `\` carattere di backslash
- `<<` "here documents"

2.4 Redirezione I/O

Ogni processo è associato a tre "stream": lo *standard input* (`stdin`), lo *standard output* (`stdout`) e lo *standard error* (`stderr`).

La redirezione dell'I/O permette di associare questi stream a sorgenti/destinazioni[2] diverse dalle loro attuali. Possiamo in questo modo salvare l'output di un processo in un file di testo, oppure possiamo salvare il suo log di errore, attraverso semplici comandi si questo tipo:

```
ls > list.txt     # salva l'output di ls in un file list.txt
ls >> list.txt    # aggiunge l'output di ls in coda
                  # al file list.txt
rm fileinutile >& /dev/null  # redireziona stdout e stderr
                             # di rm su /dev/null
```

[1] e *bash* segue questo elenco

[2] ovviamente nel caso di `stdin` avremo una sorgente, mentre per `stdout` e `stderr` avremo due destinazioni – che *possono* coincidere

Notate che, se non specificato, "`>`" ridirige lo `stdout`, e "`<`" lo `stdin`. "`>&`" ridirige *tutti* i flussi di output (quindi anche lo `stderr`). Notate anche il diverso significato di "`>`" e "`»`": il primo se rediretto in un file lo tronca al suo inizio, cancellando il suo contenuto prima di scrivere il flusso, mentre il secondo conserva il contenuto del file e accoda ad esso il flusso.

Pipe, o catene di montaggio

Attraverso le pipe, potete utilizzare l'output di un processo come input di un altro processo:

```
$ ls
  a.c b.c a.o b.o dir1 dir2
$ ls | wc -w
  6
```

`wc` (un programma che conta parole, linee e molto altro) prende come `stdin` lo `stdout` di `ls`, ritornando il numero di parole listate da quest'ultimo: 6, appunto.

Potrebbe sembrare una grossa limitazione poter unicamente ridirigere il flusso, e non poterlo duplicare: a questo supplisce il comando `tee` [3] che copia lo `stdin` sul file specificato E sullo `stdout`:

```
$ who | tee who_capture.txt | sort
```

Se `tee` viene chiamato con l'opzione `-a`, viene fatto l'append al file argomento, invece del troncamento.

2.5 Wildcards

Le wildcards sono utilizzate per specificare pattern comprendenti più file: i file vengono passati in rassegna, e quelli che corrispondono[4] vengono restituiti. Vengono utlizzati più spesso delle *regular expressions* per la loro semplicità, ad ogni modo sono un sottoinsieme di queste ultime.

- `*` match di qualsiasi stringa (anche vuota)
- `?` match di qualsiasi carattere singolo
- `[...]` match di qualsiasi carattere inserito nelle parentesi

Esempi:

```
$ ls *.c
 prova001.c prova004.c prova125.c prov.c
$ ls prova00?.c
 prova001.c prova004.c
$ ls prova[0-9][0-9][0-9].c
 prova001.c prova004.c prova125.c
```

[3] il nome deriva dai "giunti a T", utilizzati dagli idraulici
[4] in gergo, "*to match*"

2.6 Command substitution

Si può sostituire ad un argomento un comando del tipo `‘comando‘`: il suo standard output verrà sostituito al comando stesso:

```
$ echo Data odierna: ‘date‘
$ echo Utenti collegati: ‘who | wc -l‘
$ tar czvf src-‘date‘.tar.gz src/
```

2.7 Sequenze

Non condizionali

Si possono eseguire sequenzialmente dei comandi utilizzando il metacarattere ";":

```
$ date ; pwd ; ls
```

Condizionali

Le sequenze condizionali specificano la sequenza da eseguire in base all'*exit code* del comando precedente all'interno della sequenza.

"||" viene utilizzato per eseguire il comando nel caso l'*exit code* del precedente sia diverso da 0 (il che indica un fallimento).

"&&" invece esegue il comando se il precedente esce con un *exit code* uguale a 0 (cioè esce con successo).

Esempi:

```
$ gcc prova.c && ./a.out
$ gcc prova.c || echo Compilazione fallita
```

2.8 Raggruppamento di comandi

È possibile raggruppare dei comandi all'interno di "(" e ")": verranno eseguiti in una subshell, e condivideranno gli stessi `stdin`, `stdout` e `stderr`:

```
$ date ; ls ; pwd > out.txt
$ (date ; ls ; pwd) > out.txt
```

2.9 Esecuzione in background

Se il comando lanciato da shell viene seguito dal metacarattere "&", viene creata una subshell, il comando viene avviato in background e il controllo torna alla shell immediatamente, che prosegue l'esecuzione concorrentemente con il processo lanciato. Quest'ultimo abbandonerà la sorgente di `stdin` come tastiera (quindi non sarà possibile fornirgli input in tale modo), ritornadolo alla shell che vi permetterà di proseguire il lavoro; ovviamente è una funzionalità molto utile se dovete eseguire attività lunghe che non necessitano il vostro input:

```
$ find / -name passwd -print &
 [34256]
$ /etc/passwd
$ find / -name passwd -print &> results.txt &
 [34263]
$
```

2.10 Quoting

Esiste la possibilità di disabilitare *command substitution*, *wildcards* e *variable substitution* (per non essere costretti a fare necessariamente l'escape di ogni carattere) attraverso il *quoting*.

Con gli apici singoli (’’) vengono inibiti *wildcards*, *command substitution* e *variable substitution*:

```
$ echo 3 * 4 = 12
$ echo ’3 * 4 = 12’
```

Con i doppi apici (“), invece, vengono disabilitate le sole *wildcards*:

```
$ export nome="mionome"
$ echo "Il mio nome: $nome - la data: ‘date‘"
$ echo ’Il mio nome: $nome - la data: ‘date‘’
```

2.11 Subshell

Quando aprite un terminale (ad esempio quando fate login) viene eseguita una *shell*. Viene creata una *child shell* (o *subshell*) quando

- usate il raggruppamento di comandi (vedi 2.8)
- eseguite un processo in background (vedi 2.9)
- eseguite uno script

Le *subshell* hanno la propria directory corrente, la propria area di variabili indipendenti dalla *shell* madre (vedi 2.12 per le variabili).

2.12 Variabili

Le variabili supportate da qualsiasi shell sono di due tipi: *globali* e *locali*.

Le variabili *locali* non vengono passate da una shell alle altre subshell.

Al contrario, le variabili *globali* (chiamate anche variabili *d’ambiente*) vengono passate da una shell alle subshell che essa crea, e vengono utilizzate per comunicazioni tra shell madre e figlie.

Entrambi i tipi di variabile contengono dati di tipo *stringa*. Ogni shell ha una serie di variabili d’ambiente che vengono inizializzate all’avvio della shell stessa (in genere attraverso i file */etc/profile*, */home/nomeutente/.bash_profile*, etc.), come:

```
$HOME, $PATH, $USER, $SHELL, $TERM, ...
```

Un elenco completo delle variabili attualmente in uso si ottiene attraverso il comando `env`.

Per accedere al contenuto di una variabile, si può utilizzare il comando $; $VARIABILE è la forma abbreviata di ${VARIABILE} (a volte quest'ultima forma è necessaria):

```
$ echo $SHELL
 /bin/bash
```

Per assegnare un valore ad una variabile, nel caso si tratti di una variabile locale, basta il nome della variabile seguito dal valore[5]:

```
$ mionome=Ciro\ Mattia\ Gonano   # problemi con gli spazi
$ mionome="Ciro Mattia Gonano"   # nessun problema ;)
```

Per trasformare una variabile locale dichiarata in questo modo, utilizzare il comando `export`[6]:

```
$ mionome="Ciro Mattia Gonano"
$ export mionome
```

Un esempio più lungo per capire come funzionano le variabili locali e d'ambiente:

```
$ nome="Ciro Mattia"
$ cognome="Gonano"
$ echo $nome $cognome
$ export nome
$ bash
$ echo $nome $cognome
$ exit
$ echo $nome $cognome
```

2.13 Scripting

Uno script è una sequenza di comandi che devono essere eseguiti (e, ovviamente, può contenere anche sequenze condizionali), memorizzata all'interno di un file di testo e poi richiamato dalla shell. Risulta molto utile per automatizzare procedure lunghe che ripetete frequentemente.

Per scrivere uno script, è sufficiente scrivere la sequenza di comandi nel file di testo, renderlo eseguibile (con `chmod`), e lanciarlo[7]; la shell si occuperà di decidere la shell con cui eseguire lo script, di creare la subshell, e di passare il contenuto del file come `stdin` della subshell.

La selezione della shell da usare per lo script è basata sulla prima riga dello stesso script:

- se contiene solo un simbolo #, verrà utilizzata la shell corrente;

[5] fate attenzione, perché variabile e valore devono essere inframezzate da un segno di uguale (=), ma tra il segno e i caratteri **non ci devono essere spazi**

[6] `export` funziona solo se usate una shell di tipo *bourne* (come *bash*), mentre se utilizzate una *c shell* (come *csh*) dovete usare il comando `setenv`

[7] state attenti al PATH: se la directory corrente non è all'interno della vostra variabile `$PATH`, dovrete lanciare qualcosa di simile a `./mioscript`

- se contiene `#!`*`pathname`*, verrà utilizzata la shell identificata da *pathname*: per eseguire uno script con la Korn shell scriveremo nella prima riga dello script `#!/bin/ksh`. Questa è la forma che vi consiglio di utilizzare sempre, perché risulta non ambigua e non dipende dalla configurazione su cui lo script viene lanciato.
- se non contiene nessuno di questi due, viene interpretato da una Bourne shell (*bash*, *sh*, etc.).

Alcuni comportamenti interessanti per farvi capire come viene eseguito uno script[8]...

```
#!/bin/cat
#!/bin/rm
```

Esistono alcune variabili *built-in* studiate apposta per gli script (di seguito sono indicate se valgono per le Bourne shell (*sh*), per le C shell (*csh*) o per entrambe):

- `$$` l'identificatore di processo della shell (*sh*)
- `$0` il nome dello shell script (*sh,csh*)
- `$1-$9` l'n-esimo argomento della riga di comando (*sh,csh*)
- `${n}` l'n-esimo argomento della riga di comando (*sh,csh*)
- `$*` lista di tutti gli argomenti della riga di comando (*sh,csh*)
- `$#` numero degli argomenti della riga di comando (*sh*)
- `$?` valore di uscita dell'ultimo comando eseguito (*sh*)

Un piccolo script per cominciare a capire...

```
#!/bin/bash
a=23
echo $a
b=$a
echo $b
# Questo e‘ un commento!
# Proviamo qualcosa di piu‘ eccitante!
a=‘echo Ciao mondo\!‘ # assegna il risultato di echo ad a
echo $a
a=‘ls -l‘  # adesso a e‘ uguale all’output
           # di ls -l
echo $a
exit 0
```

[8] eseguiteli senza paura, non causeranno danni al vostro sistema, se li eseguite da normale utente

2.14 Here document

L'ultimo metacarattere che ci rimane da vedere è il cosiddetto "Here document" ("`<<`"). I comandi

```
<comando> << <parola>

<comando> <</ <parola>
```

servono a copiare il proprio `stdin` fino alla linea che inizia con `<parola>` (esclusa), e quindi eseguire `<command>` utilizzando questi dati copiati come `stdin`.

La versione con la *slash* ("`<</`") non esegue variable substitution.

Esempio:

```
$ cat << ENDOFTEXT > nota
> Ricordarsi di specificare che
> con << le variabili vengono sostituite
> e con <</ no.
>                  $mionome
> ENDOFTEXT
$
```

Capitolo 3

Programmazione della shell

3.1 Valutazione delle espressioni

La shell non supporta direttamente (come faceva, per chi lo ricorda, il caro vecchio **C=64**, avendo il *BASIC* integrato) la valutazione delle espressioni; a questo supplisce il comando `expr` *`espressione`*, che ci ritorna il risultato della formula introdotta. Tutti i componenti dell'espressione devono essere necessariamente separati da spazi tra di loro, e tutti i metacaratteri devono essere ovviamente escaped[1], altrimenti la shell li espanderà secondo il loro metasignificato.

Il risultato può essere numerico o una stringa, e può venire assegnato ad una variabile con un uso opportuno del *command substitution*.

Gli operatori disponibili sono:

- Aritmetici: `+ - * / %`
- Confronto: `= != > < >= <=`
- Logici: `& | !`
- Parentesi[2]: `( )`
- Stringhe:
  - `match` *`string regexp`*
  - `substr` *`string start length`*
  - `length` *`string`*
  - `index` *`string charList`*

3.2 Exit status

Ogni comando lanciato dalla shell ritorna un *exit status*. Per convenzione UNIX, un *exit status* pari a 0 indica "uscita con successo" (pari al valore `TRUE` nella valutazione delle espressioni booleane), mentre un valore diverso indica "uscita (spesso prematura) con fallimento". L'*exit status* torna molto utile negli script

[1] come dicevamo prima, apponendo una *backslash* (\) al carattere (ad esempio: *)
[2] devono essere prefissate dalla *backslash*

per controllare il flusso dell'esecuzione dei comandi, modificandolo in base ai risutlati d'uscita prodotti. Proprio per questo, la shell ci offre il comando `exit` *`nn`* per far terminare il nostro script con *exit status* pari a *nn*, e la variabile *built-in* `$?`, che contiene l'*exit status* dell'ultimo comando eseguito:

```
$ cat script.sh
#!/bin/bash
echo hello
echo $?     # "echo hello" ha tornato exit status 0 (successo)
lskdf       # comando inesistente
echo $?     # "lskdf" torna exit status diverso da 0 (fallimento)
exit 113    # il nostro script esce tornando exit status
            # 113 (attenzione: viene considerato fallimento!)
$ chmod 755 script.sh
$ ./script.sh
hello
0
./script.sh: lskdf: command not found
127
```

3.3 Strutture di controllo

Come dicevamo prima, gli *exit status* vengono utilizzati per le espressioni condizionali che governano il programma:

```
if cmp a b > /dev/null   # ridirigiamo l'output su /dev/null
then echo "I file a e b sono identici."
else echo "I file a e b sono diversi."
fi
```

Le condizioni sono prevalentemente controllate con il comando `test` *`expression`*, che torna 0 se l'espressione è vera, 1 altrimenti. Molti sono gli argomenti di confronto che può accettare `test`, e sarebbe inutile elencarli qui, quindi vi rimando alla manpage **test(1)**. È utile comunque ricordare che gli operatori di confronto sono diversi per interi e stringhe, e che esistono operatori di test per file e directory (ad esempio, per sapere se un file esiste ed e' eseguibile).

La shell offre il comando built-in `[ ]` e le keyword `[[ ]]` per valutare la condizione (usando il valore di ritorno di `test`) al loro interno. La differenza tra i due sta nel fatto che all'interno di `[[ ]]` non viene effettuato filename expansion, inoltre operatori come &&, ||, > e < vengono interpretati correttamente.

Ad esempio, per controllare l'esistenza di un argomento:

```
if [ -n "$1" ]
then
  lines=$1
fi
```

Questo esempio ci introduce il cosiddetto *blocco condizionale*, comunemente chiamato *blocco if*. Un blocco condizionale permette di eseguire comandi condizionati, ovvero di scegliere, a seconda che una condizione sia vera o falsa, una lista di comandi piuttosto che un'altra. La struttura di un blocco condizionale è la seguente:

```
if list1
then
         list2
elif list3
then
         list4
else
         list5
fi
```

Nel listato sopra, vengono innanzitutto eseguiti i comandi di `list1`, e se l'exit status dell'ultimo comando della lista è `0` (successo), il blocco passa ad eseguire i comandi in `list2` e quindi esce; in caso negativo (ovvero `list1` fallisce), vengono eseguiti i comandi in `list3`, e ancora come prima, se la lista esce con successo vengono eseguiti `list4` e quindi il blocco esce, altrimenti vengono eseguiti i comandi in `list5`.

È importante notare che un blocco if può contenere zero o più sezioni `elif`, e che la sezione `else` è sempre opzionale.

Un esempio un po' più lungo per capire meglio il funzionamento:

```
#!/bin/bash
stop=0
while [[ $stop -eq 0 ]]
do
        cat << ENDOFMENU
        1: sysadmin
        2: user
        3: guest
ENDOFMENU
echo "Your choice?"
read ch
if [[ "$ch" = "1" ]]; then
        sysadmin
elif [[ "$ch" = "2" ]]; then
        user
elif [[ "$ch" = "3" ]]; then
        guest
else
        echo error
fi
```

Per evitare molteplici utilizzi di `elif`, che possono risultare scomodi e rendere meno leggibile il codice, un'altra struttura di controllo è disponibile: il cosiddetto `case - in - esac`. La struttura è la seguente:

```
case expression in
        value1)
                list1
                ;;
        value2)
                list2
```

```
                ;;
esac
```

`expression` viene valutata, e il suo risultato viene confrontato con i vari `value`, dal primo all'ultimo; quando il primo `value` che corrisponde al risultato viene trovato, si esegue la `list` corrispondente, e finita la sequenza di comandi si salta al `esac` del blocco. È importante notare che i `value` possono contenere wildcards, quindi mettere ad esempio `*)` come ultimo value ha senso per prendere tutti i risultati che non rispecchiano i valori specificati sopra. Ad esempio:

```
#!/bin/bash
# Stampa gli utenti che consumano piu‘ spazio su HD
case "$1" in
  "")
    lines=50
    ;;
  *[!0-9]*)
    echo "Usage: ‘basename $0‘ usersnum";
    exit 1
    ;;
  *)
    lines=$1
    ;;
esac
du -s /home/* | sort -gr | head -$lines
```

Per ripetere più volte una lista di comandi finché una data condizione non sia vera, la shell ci mette a disposizione tre tipi di comandi:

- `while - do - done`
- `until - do - done`
- `for - in - do - done`

Le strutture sono cosìdefinite:

```
while list1
do
        list2
done

until list1
do
        list2
done
```

per while e until, molto simili fra loro: nel while, `list1` viene eseguita, e se l'ultimo comando esce con successo viene eseguita `list2`, e si ritorna quindi all'inizio del ciclo con una nuova esecuzione di `list1`, finché quest'ultima non esce con status fallimentare. Nell'until funziona alla stessa maniera, salvo il fatto che il ciclo viene ripetuto finché `list1` è falsa, e quando invece esce con successo, il ciclo termina.

A questi due cicli si applicano due istruzioni: `break`, che causa l'immediata uscita dal ciclo, e `continue`, che riporta immediatamente il ciclo all'inizio.

Ad esempio, possiamo utilizzare un ciclo until e un while per stampare le tabelline...

```
#!/bin/bash
if [ "$1" -eq "" ]; then
  echo "Usage: ‘basename $0‘ max"
  exit
fi
x=1
until [ $1 -gt $x ]
do
  y=1
  while [ $y -le $1 ]
  do
    echo ‘expr $x \* $y‘ ""
    y=‘expr $y + 1‘
  done
  echo
  x=‘expr $x + 1‘
done
```

Il ciclo for, invece, ha la seguente struttura:

```
for name in words
do
        list
done
```

Ciascuna parola nella lista `words` viene caricata nella variabile `name` ad ogni iterazione del ciclo, e quindi viene eseguita `list`. Se la clausola `in` viene omessa, al suo posto viene utilizzata la variabile `$*`. Ancora un esempio:

```
#!/bin/bash
for color in red yellow blue green
do
  echo One color is $color
done
```

3.4 Comandi built-in

3.4.1 I/O

`echo`

Stampa i suoi argomenti su `stdout`. L'opzione `-n` viene utilizzata per evitare che la stampa includa il newline finale. Utilizzato con command substitution, può servire per settare una variabile:

```
a=‘echo "HELLO" | tr A-Z a-z‘
```

Per abilitare l'interpretazione delle sequenze di escape, si specifica l'opzione `-e`.

`printf`

Versione migliorata di `echo`, la sintassi è *simile* a quella della funzione C `printf(3)`. L'interpretazione delle sequenze di escape è abilitata di default.

`read` *var*

Legge l'input da stdin e lo archivia nella variabile specificata (`var`). L'opzione `-p` evita la stampa dell'input, `-nN` accetta solo N caratteri in input, e `-p` stampa una stringa alla chiamata di `read`.

```
read -s -n1 -p "Premi un tasto..." keypress
echo; echo "Il tasto premuto e' \"$keypress\"."
```

3.4.2 variabili

`declare`, `typeset`

Permettono di restringere le proprietà di una variabile, una forma debole di gestione dei tipi. L'opzione `-r` dichiara una variabile come read-only (come `readonly`, `-i` come intera, `-a` come array e `-x` esporta la variabile (come `export`).

`let`

Esegue operazioni aritmetiche sulle variabili, in molti casi può funzionare come una versione semplificata di `expr`.

```
let a=11       # assegna ad a il valore 11
let "a <<= 3"  # shift a sinistra di 3 posizioni
let "a /= 4"   # a viene diviso per 4 e il risultato viene riassegnato
let "a -= 5"   # sottrazione di 5 da a e riassegnamento
```

`unset`

Cancella il contenuto di una variabile

3.4.3 Script execution

`source`

Conosciuto anche come *dot command* (`.`). Esegue uno script senza aprire una subshell. Corrisponde alla direttiva `#include` del preprocessore C.

```
$ source ~/bash_profile
$ . ~/bash_profile
```

3.4.4 Comandi vari

`true`, `false`

Ritornano sempre rispettivamente 0 e 1.

`type` [*cmd*]

Stampa un messaggio che indica se `cmd` è una keyword, un comando built-in oppure un comando esterno.

`shopt` [*options*]

Setta alcune opzioni della shell (ad esempio, `shopts -s cdspell` permette il mispelling in `cd`).

`alias`, `unalias`

Un *alias* è una scorciatoia per abbreviare lunghe sequenze di comandi:

```
alias dir="ls -l"
alias rd="rm -r"
unalias dir
```

`history`

Visualizza l'elenco degli ultimi comandi eseguiti.

3.4.5 Directory stack

La shell Bash permette di manipolare una pila di directory con alcuni comandi utili per visite ad albero.

`dirs`

Stampa il contenuto del directory stack.

`pushd` *dirname*

Esegue il push della directory *dirname* nello stack; successivamente si sposta nella directory *dirname* ed esegue `dirs`.

`popd`

Esegue il pop dallo stack, si sposta sull'attuale top element ed esegue `dirs`.

`$DIRSTACK`

Contiene il top element dello stack.

3.5 File di configurazione

3.5.1 Inizializzazione

La shell, all'avvio, fa il parsing di alcuni file di configurazione:

/etc/profile

Settings sistem-wide validi per tutte le shell, non solo Bash.

/etc/bashrc

Configurazione sistem-wide valida unicamente per Bash, contenente funzioni, alias e modelli di comportamento.

$HOME/.bash_profile

Settings specifici per l'utente relativi a Bash (se Bash non trova questo file, cercherà il file `$HOME/.profile`.

$HOME/.bashrc

Configurazione di Bash specifica per l'utente, contiene funzioni e alias. Viene letto solo se la shell è di tipo interattivo o in esecuzione di script.

3.5.2 Terminazione

$HOME/.bash_logout

Al logout, Bash esegue lo script contenuto in questo file.

3.6 Comandi esterni

Appendice A

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document free in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of copyleft, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated

herein. The **Document**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **you**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **Modified Version** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **Secondary Section** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **Invariant Sections** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **Cover Texts** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **Transparent** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not Transparent is called **Opaque**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **Title Page** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, Title Page means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **Entitled XYZ** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **Acknowledgements**, **Dedications**, **Endorsements**, or **History**.) To **Preserve the Title** of such a section when you modify the Document means that it remains a section Entitled XYZ according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an

Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled History, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled History in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network

locations given in the Document for previous versions it was based on. These may be placed in the History section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled Acknowledgements or Dedications, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled Endorsements. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled Endorsements or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled Endorsements, provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled History in the various original documents, forming one section Entitled History; likewise combine any sections Entitled Acknowledgements, and any sections Entitled Dedications. You must delete all sections Entitled Endorsements.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an aggregate if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a

disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled Acknowledgements, Dedications, or History, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License or any later version applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the with...Texts. line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.