

# Bash

## Comandi

Comando	Spiegazione
<code>[[ condizioni ]]</code>	<a href="#">Conditional Expressions</a>
<code>read A B ...</code>	Legge una riga dallo <b>stdin</b> . <a href="#">Altro</a>
<code>while read A B ...</code>	Per leggere da file in modo comodo. <a href="#">Esempio</a>
<code>cat file1 file2 ...</code>	Stampa i file uno dopo l'altro
<code>grep "substring"</code>	Cerca la substring dentro lo <b>stdin</b> e restituisce solo le righe che la contengono ( <code>grep -- "substring"</code> se nella substring c'è il <code>-</code> ). <a href="#">Altro</a>
<code>wc</code>	"word count", conta <code>\n</code> , <b>numero di parole</b> e <b>bytes</b> (ovvero caratteri se sono tutti ASCII). <a href="#">Altro</a>
<code>head</code>	Scriva le prime 10 (di default) righe dallo STDIN. <a href="#">Altro</a>
<code>tail</code>	Scriva le ultime 10 (di default) righe dallo STDIN. <a href="#">Altro</a>
<code>sort</code>	Sorta le righe dell'STDIN in ordine crescente alfabeticamente. <a href="#">Altro</a>

## Variabili speciali

Variabile	Spiegazione
<code>\$RANDOM</code>	Variabile speciale che ti da un numero random
<code>\$#</code>	Ottieni il numero di argomenti passati allo script
<code>\$?</code>	Ottieni il codice di errore dell'ultima operazione eseguita

Variabile	Spiegazione
<code>\$\$</code>	Ottieni il PID del processo che sta eseguendo lo script corrente

## Scripting

Comando	Spiegazione
<code>#!/bin/bash</code>	Prima riga di uno script bash. Indica con quale interprete lo script si aspetta di essere seguito. ( <code>/bin/bash</code> in questo caso).
<code>comando1 \   comando2</code>	È la pipe; ficca nello <b>stdin</b> della roba di destra l'output della roba di sinistra

## Conditional Expressions

Sono quelle che usi in una `[[ condizione ]]`

Vedi la lista completa nel manuale

```
man bash | grep -B 94 -- -eq
```

## Parameter Expansions

### Substring

`${variabile:indice_inizio:lunghezza}`

Sia `indice_inizio` che `lunghezza` sono una espressione matematica (quelle con le doppie parentesi), quindi ci si possono fare i conti.

Se `indice_inizio` è negativo fa effetto pac-man, quindi l'indice è considerato dalla fine della stringa.

Se `lunghezza` è negativo, la stringa viene presa tutta fino a `lunghezza` caratteri dalla fine

Importante: Se si vuole mettere uno di questi due valori in negativo, bisogna mettere uno spazio dopo i :, in quanto esiste un'altra expansion che si fa con :- che però non abbiamo trattato.

## Esempio

Script:

```
variabile="Ciao sono Andrea"
echo ${variabile: -6: 6}
echo ${variabile: 0: -7}
```

Output:

```
Andrea
Ciao sono
```

## Length

Per avere la lunghezza della stringa contenuta in una variabile: `${#variabile}`

## Search e Replace

Sintassi	Cosa fa
<code>\${variabile#qualcosa}</code>	Cerca l'occorrenza più corta di <code>qualcosa</code> nel valore di <code>variabile</code> partendo dall'inizio e lo toglie. Se gli hashtag sono due (##) cerca l'occorrenza più lunga.

## Ulteriori spiegazioni

`read`

Può avere dei parametri in cui mette le sottostringhe della riga che ha trovato in base ai separatori definiti in **\$IFS** (**spazio**, **\t** e **\n** di default).

Esempio: `echo "Ciao-sono-andrea" | IFS="-" read A B C ; echo $C`

Variabili attese: `$A = Ciao $B = sono $C = andrea`

Output: `andrea`

### `while read`

Per leggere da file in modo comodo

```
while read A B ...;  
do  
    ...  
done < percorso-al-file
```

### `wc`

Questo comando senza nessun argomento legge dallo **stdin**.

Passandogli un file come argomento `wc file` fornirà un input del tipo `Numero Nomefile`.

Per estrarre il numero quando si utilizza la modalità col parametro, si utilizzano le operazioni sulle stringhe in questo modo:

```
Appoggio=$(wc -l file)  
Appoggio=${Appoggio%% *}}
```

Flag utili	Cosa fa
<code>-c</code>	Restituisce il numero di <b>byte</b>
<code>-l</code>	Restituisce il numero di <b>\n</b>
<code>-w</code>	Restituisce il numero di <b>parole</b>

## head

Stampa le **prime** 10 righe dall'alto

Flag utili	Cosa fa
-n <numero>	Stampa le <numero> righe dall'alto
-n -<numero>	Stampa tutto tranne le <b>prime</b> <numero> righe <b>dal basso</b>

## tail

Stampa le **ultime** 10 righe (le prime 10 dal basso)

Flag utili	Cosa fa
-n <numero>	Stampa le <numero> righe dal basso
-n +<numero>	Stampa tutto tranne le <b>ultime</b> <numero> righe (le prime <b>all'alto</b> )

## sort

Flag utili	Cosa fa
-r	Sorta al contrario
-n	Compara numericamente

## grep

WIP

Occorrenze

Esempio



