Operatori su stringhe

Minimale...

Accesso alle variabili

- Operatori disponibili:

  `$<var>` o `${<var>}`

  ritorna il valore di **`<var>`**

  `${<var>:-<val>}`

  se **`<var>`** esiste ed ha valore non vuoto, ritorna il suo valore, altrimenti ritorna **`<val>`**

  `${<var>:=<val>}`

  se **`<var>`** esiste ed ha valore non vuoto, ritorna il suo valore, altrimenti assegna **`<val>`** a **`<var>`** e ritorna **`<val>`**

  `${<var>:?<message>}`

  se **`<var>`** esiste ed ha valore non vuoto, ritorna il suo valore, altrimenti stampa il nome e **`<message>`** su **`stderr`**

Accesso alle variabili (2)

- Operatori disponibili (cont.):

  `${<var>:+<val>}`

  se `<var>` esiste ed ha valore non vuoto, ritorna il valore `<val>`

- Esempi:

```
bash:~$ echo $PIPPO
                    -- variabile non definita
bash:~$ echo ${PIPPO:-ii}
ii
bash:~$ echo $PIPPO
                    -- ancora non definita
bash:~$
```

Accesso alle variabili (3)

- Esempi:

```
bash:~$ echo $PIPPO
iiii                      -- variabile definita
bash:~$ echo ${PIPPO:+kk}
kk
bash:~$ echo $PIPPO
iiii                      -- ancora stesso valore
bash:~$
```

Accesso alle variabili (4)

- Esempi:

```
bash:~$ echo $PIPPO
iiii                    -- variabile definita
bash:~$ echo ${PIPPO:+kk}
kk
bash:~$ echo $PIPPO
iiii                    -- stesso valore
bash:~$ echo ${PIPPO:=kkll}
iiii                    -- non modificata
bash:~$
```

Sottostringhe

`${<var>:<offset>}`

`${<var>:<offset>:<length>}`

ritorna la sottostringa di **`<var>`** che inizia in posizione **`<offset>`** (NOTA: il primo carattere è in posizione 0)

Nella seconda forma la sottostringa è lunga **`<length>`** caratteri. Esempio:

```
bash:~$ A=armadillo
bash:~$ echo ${A:5}
illo
bash:~$ echo ${A:5:2}
il
bash:~$
```

Lunghezza

`${#<var>}`

consente di ottenere la lunghezza (in caratteri) del valore della variabile **`<var>`** (NOTA: la lunghezza è comunque una stringa)

– Esempio:

```
bash:~$ A=armadillo
bash:~$ echo ${#A}
9
bash:~$ echo ${A:$(({#A}-4))}
illo
bash:~$ B=${A:3:3}
bash:~$ echo ${#B}              -- $B=adi
3
bash:~$
```

Pattern matching

- È possibile selezionare parti del valore di una variabile sulla base di un pattern (modello)
- I pattern possono contenere *,?, e [] e sono analoghi a quelli visti per l'espansione di percorso
- occorrenze iniziali

`${<var>#<pattern>}`

`${<var>##<pattern>}`

se **`<pattern>`** occorre all'inizio di **`$<var>`** ritorna la stringa ottenuta eliminando da **`$<var>`** la più corta / la più lunga occorrenza *iniziale* di **`<pattern>`**

Pattern matching (2)

- Occorrenze finali

  `${<var>%<pattern>}`

  `${<var>%%<pattern>}`

  se **`<pattern>`** occorre alla fine di **`$<var>`** ritorna la stringa ottenuta eliminando da **`$<var>`** la più corta / la più lunga occorrenza *finale* di **`<pattern>`**

- esempi:
  - **`outfile=${infile%.pcx}.gif`**
    - rimuove l'eventuale estensione **`.pcx`** dal nome del file e ci aggiunge **`.gif`** **`(pippo.pcx`** → **`pippo.gif)`**

Pattern matching (3)

- Esempi (cont):
  - **`basename=${fullpath##*/}`**
    - rimuove dal **`fullpath`** il prefisso più lungo che termina con '`/`' (cioè estrae il nome del file dal path completo)
  - **`dirname=${fullpath%/*}`**
    - rimuove dal **`fullpath`** il suffisso più corto che inizia per '`/`' (cioè estrae il nome della directory dal path completo)

```
bash:~$ fullpath=/home/s/susanna/myfile.c
bash:~$ echo ${fullpath##*/}
myfile.c
bash:~$ echo ${fullpath%/*}
/home/s/susanna
```

Sostituzione di sottostringhe

- È possibile sostituire le occorrenze di un pattern nel valore di una variabile

  ```
  ${<var>/<pattern>/<string>}
  ${<var>//<pattern>/<string>}
  ```

  - l'occorrenza più lunga di **`pattern`** in **`var`** è sostituita con **`string`**.
  - La prima forma sostituisce solo la prima occorrenza, la seconda le sostituisce tutte
  - se **`string`** è vuota le occorrenze incontrate sono eliminate
  - se il primo carattere è **`#`** o **`%`** l'occorrenza deve trovarsi all'inizio o alla fine della variabile
  - se var è **`*`** o **`@`** l'operazione è applicata ad ogni parametro posizionale, e viene ritornata la lista risultante

Sostituzione di sottostringhe (2)

- Esempi:

```
bash:~$ echo $A
unEsempioDiSostituzione
bash:~$ echo ${A/e/eee}
unEseeempioDiSostituzione
bash:~$ echo ${A//e/eee}
unEseeempioDiSostituzioneee
bash:~$ echo ${A/%e/eee}
unEsempioDiSostituzioneee
bash:~$ ${A/#*n/eee}
eeeEsempioDiSostituzione
bash:~$
```