arrays bash

linux, bash, pendiente, wiki

/via: http://www.thegeekstuff.com/2010/06/bash-array-tutorial/

> **más información de interés (arrays indirection)**
> http://wiki.bash-hackers.org/syntax/arrays?rev=1534920675

declaración y asignación

```
array=(valor0 'valor uno' 'valor dos')
```

```
array[0]='valor0'
array[1]='valor uno'
array[2]='valor dos'
```

inicialización durante declaración

```
declare -a array=(valor0 'valor uno' 'valor dos')
```

+ asignaciones

```
array[n]=valor
array[cadena]=valor
array=valor # se asigna al índice 0 (en asociativos, a "0")
array=(valor1 valor2 valor3)
array=([0]=valor1 1=valor2 2=valor3)
array=([cadena]=valor1 [cadena2]=valor2)
array+=(valor5 valor6) # añade
array=("${otro_array[@]}")
```

recuperación

Syntax	Description
${ARRAY[N]}	Expands to the value of the index N in the **indexed** array ARRAY. If N is a negative number, it's treated as the offset from the maximum assigned index (can't be used for assignment) - 1
${ARRAY[S]}	Expands to the value of the index S in the **associative** array ARRAY.
«${ARRAY[@]}« ${ARRAY[@]} «${ARRAY[*]}« ${ARRAY[*]}	Similar to mass-expanding positional parameters, this expands to all elements. If unquoted, both subscripts * and @ expand to the same result, if quoted, @ expands to all elements individually quoted, * expands to all elements quoted as a whole.

Syntax	Description
«${ARRAY[@]:N:M}« ${ARRAY[@]:N:M} «${ARRAY[*]:N:M}« ${ARRAY[*]:N:M}	Similar to what this syntax does for the characters of a single string when doing substring expansion, this expands to M elements starting with element N. This way you can mass-expand individual indexes. The rules for quoting and the subscripts * and @ are the same as above for the other mass-expansions.

metada

Syntax	Description
${#ARRAY[N]}	Expands to the **length** of an individual array member at index N (**stringlength**)
${#ARRAY[STRING]}	Expands to the **length** of an individual associative array member at index STRING (**stringlength**)
${#ARRAY[@]} ${#ARRAY[*]}	Expands to the **number of elements** in ARRAY
${!ARRAY[@]} ${!ARRAY[*]}	Expands to the **indexes** in ARRAY since BASH 3.0

destrucción

The unset builtin command is used to destroy (unset) arrays or individual elements of arrays.

Syntax	Description
unset -v ARRAY unset -v ARRAY[@] unset -v ARRAY[*]	Destroys a complete array
unset -v ARRAY[N]	Destroys the array element at index N
unset -v ARRAY[STRING]	Destroys the array element of the associative array at index STRING

usar siempre comillas para encerrar la variable para evitar problemas con las variables glob

mostar array/elementos/subcadenas

- todo el array: `echo ${array[@]}`
- un elemento: `echo ${array[n]}`
- del elemento n al m: `echo ${array[@]:n:m}`
- del elemento n, subcadena a-b: `echo ${array[n]:a:b}`

longitud

- del array: `echo ${#array[@]}`
- del primer elemento: `echo ${#array}`
- del tercero: `echo ${#array[2]}`

adición / eliminación / sustitución elementos

- añadir elemento(s) a un array existente: `array=(«${array[@]}» «valor 3» «valor 4»)`
- eliminar un elemento n: `unset array[n]`
  - existe forma de eliminar el índice o eliminar por patrón
- eliminar un array: `unset array`
- cambiar string1 por string2: `${array[@]/string1/string2}`

- copiar un array: `new_array=(«${array[@]}»)`
- concatenar arrays: `new_new_array=(«${array[@]}» «${new_array[@]}»)`

cargar contenido de un fichero

- `array=(` cat file `)`

funky constructs

- todos los items del array: `${array[*]}`
- todos los índices del array: `${!array[*]}`
- número de items: `${#array[*]}`
- vía: http://www.linuxjournal.com/content/bash-arrays

ejemplos

quoted "*", quoted "@", unquoted

```
#!/bin/bash

array=("first item" "second item" "third" "item")

echo "Number of items in original array: ${#array[*]}"
for ix in ${!array[*]}
do
    printf "   %s\n" "${array[$ix]}"
done
echo

arr=(${array[*]})
echo "After unquoted expansion: ${#arr[*]}"
for ix in ${!arr[*]}
do
    printf "   %s\n" "${arr[$ix]}"
done
echo

arr=("${array[*]}")
echo "After * quoted expansion: ${#arr[*]}"
for ix in ${!arr[*]}
do
    printf "   %s\n" "${arr[$ix]}"
done
echo

arr=("${array[@]}")
echo "After @ quoted expansion: ${#arr[*]}"
for ix in ${!arr[*]}
do
```

```
    printf "   %s\n" "${arr[$ix]}"
done
```

resultado:

```
Number of items in original array: 4
   first item
   second item
   third
   item

After unquoted expansion: 6
   first
   item
   second
   item
   third
   item

After * quoted expansion: 1
   first item second item third item

After @ quoted expansion: 4
   first item
   second item
   third
   item
```