

Metacarácter: | & ; ( ) < > *espacio tabulador nueva-línea* \* @ # ? - \$ ! \ [ ]

Operador de control: || && ; ; ;& ;;& ( ) | |& *nueva-línea*

Operador de redirección: < > >| << >> <& >& <<- <>

Palabras reservadas: ! case coproc do done elif else esac fi for function if in select then until while { } time [[ ]]

Declarar variable (ojo, diferencia entre mayúsculas y minúsculas):

**de**clare NombreVariable=ValorContenidoVariable

NombreVariable=ValorContenidoVariable

P. ej. Lugar=Olivar

Para referirse al valor o contenido almacenado dentro de la variable nombrada "NombreVariable" es **\$NombreVariable**

P. ej. El valor o contenido almacenado dentro de la variable de nombre "Lugar" puede mostrarse con **echo "\$Lugar"**

Comando bien ("verdad" o "verdadero"): "\$?" = 0

Comando mal ("falso"): "\$?" > 0

Recoger el resultado de un comando: **\$(comando)**

Recoger el resultado del comando, asignándolo como el valor de una variable: **Nombre=\$(comando)**

Conjuntos de comandos: Funciones:

**nombre-función () { comandos }**

Control del flujo del programa (bucles y derivaciones, oficialmente estructuras de "selección" e "iteración"):

<pre>for i in ... ; do comandos ; done</pre> <p>Nota: <b>i</b> es el nombre de variable (libremente elegido) que se asigna al texto entrante en cada iteración</p>	<pre>for i in ... do   comandos done</pre>
<pre>case i in patrón ) comandos ;; esac</pre> <p>Nota: <b>i</b> es el nombre de variable (libremente elegido) que se asigna al texto entrante en cada iteración</p>	<pre>case i in   patrón )     comandos   ;; esac</pre>
<pre>if condición ; then comandos ; fi if condición ; then comandos ; else comandos ; fi if condición ; then comandos ; elif condición ; then comandos ; else comandos ; fi</pre>	<pre>if condición then   comandos elif condición then   comandos else   comandos fi</pre>
<pre>while condición ; do comandos ; done until condición ; do comandos ; done</pre>	<pre>while condición do   comandos done</pre>

```
echo "¿Lugar donde estamos?"
read RESPUESTA
if [ "$RESPUESTA" = "Olivar" ] ; then
  echo "Bien"
else
  echo "Mal"
fi
```

```
ls | while read linea ; do echo rm "$linea" ; done
```

Condiciones:

[ "... " **comparador** "... " ] [ **comprobador** "... " ]

cumple condición 1 Y condición 2: [ **condición** ] && [ **condición** ]

cumple condición 1 O condición 2 ("o" inclusivo, puede cumplir ambas): [ **condición** ] || [ **condición** ]

Inversa de la condición: [ **! condición** ]

## Comparadores y comprobadores:

<b>-a archivo</b>	Verdad si archivo existe (es mejor usar <b>-e</b> ).
<b>-b archivo</b>	Verdad si archivo existe y es un archivo especial de bloques.
<b>-c archivo</b>	Verdad si archivo existe y es un archivo especial de caracteres.
<b>-d archivo</b>	Verdad si archivo existe y es un directorio.
<b>-e archivo</b>	Verdad si archivo existe.
<b>-f archivo</b>	Verdad si archivo existe y es un archivo regular.
<b>-g archivo</b>	Verdad si archivo existe y tiene el bit SGID.
<b>-h archivo</b>	Verdad si archivo existe y es un enlace simbólico o blando (se usa más <b>-L</b> ).
<b>-k archivo</b>	Verdad si archivo existe y tiene el bit 'pegajoso' ('sticky').
<b>-p archivo</b>	Verdad si archivo existe y es una tubería con nombre (FIFO).
<b>-r archivo</b>	Verdad si archivo existe y se puede leer.
<b>-s archivo</b>	Verdad si archivo existe y tiene un tamaño mayor que cero.
<b>-t fd</b>	Verdad si el descriptor de archivo fd está abierto y se refiere a una terminal.
<b>-u archivo</b>	Verdad si archivo existe y tiene el bit SUID.
<b>-w archivo</b>	Verdad si archivo existe y se puede modificar.
<b>-x archivo</b>	Verdad si archivo existe y es ejecutable.
<b>-G archivo</b>	Verdad si archivo existe y su grupo es el GID efectivo.
<b>-L archivo</b>	Verdad si archivo existe y es un enlace simbólico o blando.
<b>-N archivo</b>	Verdad si archivo existe y ha sido modificado desde que se leyó la última vez.
<b>-O archivo</b>	Verdad si archivo existe y su propietario es el UID efectivo.
<b>-S archivo</b>	Verdad si archivo existe y es un zócalo (socket).
<b>archivo1 -ef archivo2</b>	Verdad si archivo1 y archivo2 tienen los mismos números de nodo-i y de dispositivo.
<b>archivo1 -nt archivo2</b>	Verdad si archivo1 es más reciente (según la fecha de modificación) que archivo2, o si archivo1 existe mientras que archivo2 no.
<b>archivo1 -ot archivo2</b>	Verdad si archivo1 es más antiguo que archivo2, o si archivo2 existe mientras que archivo1 no.
<b>-v nombre-variable</b>	Verdadero si está definida la variable de la shell NombreVariable (se le ha asignado un valor).
<b>-R nombre-variable</b>	Verdadero si la variable de la shell NombreVariable está definida y es una referencia de nombre.
<b>-z cadena</b>	Verdad si la longitud de cadena es cero.
<b>-n cadena</b>	Verdad si la longitud de cadena no es cero.
<b>cadena1 = cadena2</b>	Verdadero si las cadenas son iguales.
<b>cadena1 != cadena2</b>	Verdad si las cadenas no son iguales.
<b>cadena1 &lt; cadena2</b>	Verdad si cadena1 se ordena lexicográficamente antes de cadena2.
<b>cadena1 &gt; cadena2</b>	Verdad si cadena1 se clasifica lexicográficamente tras cadena2.
<b>núm1 -eq núm2</b>	Verdad si número-entero-1 es igual a número-entero-2.
<b>núm1 -ne núm2</b>	Verdad si número-entero-1 es no igual a número-entero-2.
<b>núm1 -lt núm2</b>	Verdad si número-entero-1 es menor que número-entero-2.
<b>núm1 -le núm2</b>	Verdad si número-entero-1 es menor que o igual a número-entero-2.
<b>núm1 -gt núm2</b>	Verdad si número-entero-1 es mayor que número-entero-2.
<b>núm1 -ge núm2</b>	Verdad si número-entero-1 es mayor que o igual a número-entero-2.

Autoría (2013-2025): Alexis Puente Montiel

Puede usar libremente este documento con la condición de preservar siempre (incluido derivados) el reconocimiento de la autoría original y enlace a <https://picahack.org> como la fuente.