Breve guía de programación en *shell* (Versión 1.5) *Pica Pica HackLab (PicaHack) https://picahack.org*

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):
```
declare 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: `$(comando)` `$(comando | comando | comando)`
Recoger el resultado del comando, asignándolo como el valor de una variable: `NombreVariable=$(comando)`
Conjuntos de comandos: Funciones:
```
NombreFunción () { comandos ; comandos ; comandos }
```

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

`for i in ... ; do comandos ; done`
 	
Nota: ***i*** es el nombre de variable (libremente elegido) que se asigna al texto entrante en cada iteración
	`for i in ...`
	`do`
	`  comandos`
	`done`

`case i in patrón ) comandos ;; esac`
 	
Nota: ***i*** es el nombre de variable (libremente elegido) que se asigna al texto entrante en cada iteración
	`case i in`
	`  patrón )`
	`  comandos`
	`  ;;`
	`esac`

`if condición ; then comandos ; fi`
`if condición ; then comandos ; else comandos ; fi`
`if condición ; then comandos ; elif condición ; then comandos ; else comandos ; fi`
	`if condición`
	`then`
	`  comandos`
	`elif condición`
	`then`
	`  comandos`
	`else`
	`  comandos`
	`fi`

`while condición ; do comandos ; done`
`until condición ; do comandos ; done`
	`while condición`
	`do`
	`  comandos`
	`done`

```
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:

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