# -> Comentarios.

Siempre es aconsejable poner en la primera línea para que tipo de terminal está hecho el script. Por ejemplo, si es para bash.

#!/bin/bash

También es una buena práctica poner un comando de salida para cuando pulsemos Ctrl + C. Su código sería.

Function ctrl\_c(){

Echo -e “\n\n[!] Saliendo..\n”

Exit 1

}

Se brinda una paleta de colores para dar una mejor visión a la salida de texto por el terminal:

#Colours

greenColour="\e[0;32m\033[1m"

endColour="\033[0m\e[0m"

redColour="\e[0;31m\033[1m"

blueColour="\e[0;34m\033[1m"

yellowColour="\e[0;33m\033[1m"

purpleColour="\e[0;35m\033[1m"

turquoiseColour="\e[0;36m\033[1m"

grayColour="\e[0;37m\033[1m"

Modo de empleo. Antes del texto que vayamos a imprimir debemos poner -e. Para iniciar el color debemos poner ${color} Y SIEMPRE debemos finalizar el color con ${endColour}. Ej:

**Variables:**

Para declarar una variable podemos simplemente poner el nombre de la variable y asignarle un valor.

NombreVariable=”hola”

También podemos declararla con declare

Declare –Tipo NombreVariable=ValorVariable

Declare –i VariableInt=0

$RANDOM %37-> Nos muestra un número random del 0 al 37.

Tput cuvus -> elimina la visibilidad del cursor.

Tput cnorm -> Nos muestra de nuevo el cursor.

**Condicionales:**

If [ parametros ]; then -> Si hay más de un parámetro if [ parámetros ] && [ parámetros]; then

Acciones

Elif [parametros]; then

Acciones

Else

Acciones

Fi -> Con fi cerramos el IF.

Podemos tener como operadores el &&, ||…

Como parámetros –lt -> lower than. –gt -> grater than

Case parametro in

Acciones

Esac -> Con esac cerramos el case.

**Bucles:**

While parametros; do

Acciones

Done

Es común añadir los parametros que queremos para nuestro script con un while getopts. Ej.

While getopts “m:h” args; do

Case $arg in

m) Acciones para m

h) Acciones para h

Esto hace que cuando ejecutamos nuestro script añadiendo -m o –h, nos realice la acción para ese caso.

**Funciones:**

Declarar funcion:

Function NombreFuncion(){

Acciones

}

Una vez se declara la función, cuando se utiliza ese nombre de función se llama a esta.

**Arrays:**

Declare –a NombreArray=(1 2 3 4) -> el –a indica al sistema que estamos declarando un array.

Echo $NombreArray -> Te muestra el primer elemento del array.

Echo ${NombreArray[@]} -> Te muestra todos los elementos de un array.

For element in ${NombreArray[@]}; do

Echo “[+] Elemento $element

Done

Echo ${#NombreArray[@]} -> Te muestra cuantos elementos tiene el array.

Echo ${NombreArray[-1]} -> Te muestra el último elemento del array.

NombreArray+=(5) -> Añadimos el elelemto 5 al array.

Unset NombreArray[0] -> Elimina el elemento de la posición 0.

*Conceptos $*

En Bash se pueden usar argumentos desde la línea de comandos, los cuales son enviados a los scripts como variables. Estos quedarían representados de la siguiente forma:

**[$0]**: Representa el nombre del script que se invocó desde la terminal.

**[$1]**: Es el primer argumento desde la línea de comandos.

**[$2]**: Es el segundo argumento desde la línea de comandos y así sucesivamente.

**[$#]**: Contiene el número de argumentos que son recibidos desde la línea de comandos.

**[$\*]**: Contiene todos los argumentos que son recibidos desde la línea de comandos, guardados todos en la misma variable.

Es decir, que si creamos un script donde pongamos por ejemplo echo $1, $1 lo tomará del primer argumento que le pasemos cuando ejecutemos el script.

El resultado será que imprimirá Hola por pantalla.