Bash Scripting: diversas formas de implementación de menús dinámicos

Desde simples lecturas de la entrada estándar hasta complejos algoritmos; con GNU/Bash, crear menús dinámicos puede convertirse en un entretenido arte.

Cuando el *scripting* en GNU/Bash no es nuestra actividad principal y solo lo utilizamos como herramienta complementaria para nuestros programas, es de lo más frecuente que le demos poca importancia al menú de nuestro *script*. Pero tarde o temprano nos encontraremos con la necesidad de mejorarlo y emplear un menú que realmente cumpla una función protagónica para el *script*.

Por suerte, GNU/Bash es un lenguaje muy completo que nos brinda un amplio abanico de opciones para crear menús que puedan satisfacer cada una de nuestras necesidades, cubriendo así una gran cantidad de alternativas posibles.

En este artículo veremos la forma de implementar **desde sencillos menús** que tan solo actúen leyendo la información de entrada estándar **hasta *hacks* que mediante el empleo de múltiples constructores del lenguaje, generen menús con una complejidad algorítmica que pueda contemplar hasta el más mínimo detalle**.

La forma más común y tradicional con la que la mayoría de los programadores suele trabajar en un *script*, es con la lectura de una entrada estándar:

```
#!/bin/bash

OPCIONES="Abrir Cerrar Editar Borrar Guardar Salir"
echo $OPCIONES
echo
echo -n 'Elija una opción: '; read opcion
echo "Usted eligió $opcion."
```

Básicamente, todo consiste en imprimir en pantalla una cadena de texto de la cual el usuario «copiará» una palabra (qué hace las veces de «opción»), la ingresará, nuestro *script* la leerá y en base a ella, hará determinada acción. Simple, sencillo y resuelve de forma rápida los requerimientos más primitivos de la

interacción con un usuario. A lo sumo, como gran complejidad, se le suele agregar una estructura de control condicional:

```
#!/bin/bash

OPCIONES="Abrir Cerrar Editar Borrar Guardar Salir"
echo $OPCIONES
echo
echo -n 'Elija una opción: '; read opcion

if [ "$opcion" = "Abrir" ]; then
    echo "Esto abre un archivo"
else
    echo "Usted eligió la opción $opcion"
fi
```

Sin embargo, **cuando la cantidad de opciones** a considerar en las estructuras de control condicionales **se van incrementando**, se comienza a **optar por constructores de selección más precisos**, como es el caso del **constructor case**:

```
#!/bin/bash

OPCIONES="Abrir Cerrar Editar Borrar Guardar Salir"
echo $OPCIONES
echo
echo -n 'Elija una opción: '; read opcion

case $opcion in
    Abrir)
        echo "Esto abre un archivo"
        ;;
    Cerrar)
        echo "Eso cierra un archivo"
        ;;
    Salir)
        echo "Esto sale del programa"
        exit
        ;;
    *)  # Cualquier opción no contemplada
        echo "No eligió ninguna opción correcta"
        ;;
esac
```

Pero en GNU/Bash, el constructor case es tan solo uno de los tantos constructores provistos por el lenguaje. También podemos encontrarnos con el constructor `select`.

El constructor select

El constructor `select`, adoptado por primera vez por Korn Shell, nos da resuelto en un solo paso:

- La enumeración de opciones posibles;
- La impresión ordenada de opciones en pantalla;
- La lectura estándar de la opción elegida por el usuario;
- La repetición constante del menú, en pantalla.

La sintaxis de este constructor es la siguiente:

```
select VARIABLE_NAME [in lista_de_opciones]; do
    # instrucciones
    [break]
done
```

Los corchetes [ ] indican «sintaxis opcional»

El constructor `select` reproduce el menú de forma permanente incluso después de haber finalizado con las instrucciones internas. La instrucción `break` romperá ese bucle.

```
#!/bin/bash

OPCIONES="Abrir Cerrar Editar Borrar Guardar Salir"

PS3="Elija una opción: "  # Si no se indica mostrará '#?' por defecto

select opcion in $OPCIONES; do
    echo "Usted Eligió la opción $opcion"
    break  # Rompe el bucle
done


# Ejecución en bucle (continuada) – no se indica el break
select opcion in $OPCIONES; do
    echo "Usted Eligió la opción $opcion"
done
```

El algoritmo anterior producirá una salida similar a la siguiente:

```
user@host:~$ ./menus-dnamicos.sh
1) Abrir
2) Cerrar
3) Editar
4) Borrar
5) Guardar
6) Salir
Elija una opción:
```

Como puede verse, nos ahorra muchísimo esfuerzo al momento de presentar las opciones en pantallas.

> *El constructor select es el generador de menús de opciones por excelencia de GNU/Bash*

Combinando el constructor con una estructura de control condicional simple, es posible determinar si el usuario ha ingresado -o no- una opción válida:

```
#!/bin/bash

OPCIONES="Abrir Cerrar Editar Borrar Guardar Salir"

PS3="Elija una opción: "

select opcion in $OPCIONES; do
    if [ $opcion ]; then
        echo "Usted Eligió la opción $opcion"
        break
    fi
done
```

Esto nos da una gran ventaja ya que de tratarse de una opción válida *«switchear»* la elección del usuario no requeriría de validaciones adicionales:

```
#!/bin/bash

OPCIONES="Abrir Cerrar Editar Borrar Guardar Salir"

PS3="Elija una opción: "

select opcion in $OPCIONES; do
    if [ $opcion ]; then
        case $opcion in
            Abrir)
                echo "Esto abre un archivo"
                ;;
            Cerrar)
                echo "Eso cierra un archivo"
                ;;
            Salir)
                echo "Esto sale del programa"
                exit
                ;;
        esac
        break
    fi
done
```

SELECT HACK: CAPTURAR LA OPCIÓN INVÁLIDA

Frecuentemente, cuando en la bibliografía se habla del constructor `select` -o del tan conocido case-, solo se hace referencia a un conjunto de opciones posibles. Sin embargo, muchísimas veces una opción `select` inválida podría requerir un tratamiento distinto al de otra opción `select` inválida.

Imaginemos este escenario: el usuario se encuentra con un menú de opciones delante y la imposibilidad de elegir la indicada, puesto desconoce la utilidad de cada una. El *script* espera una entrada del usuario pero el usuario necesita conocer el número de versión del *script* antes de decidir qué opción elegir. ¿Cómo se entera del número de versión sin salir del *script*? Podría ofrecerse la misma como una opción más del menú pero sin embargo, se estaría desvirtuando la finalidad del programa. En cambio, un argumento ingresado en la entrada estándar, podría solucionar el problema.

> *Se puede capturar la respuesta del usuario accediendo a la variable de contexto $REPLY*

```
#!/bin/bash

OPCIONES="Abrir Cerrar Editar Borrar Guardar Salir"

PS3="Elija una opción: "

select opcion in $OPCIONES; do
    if [ $opcion ]; then
        case $opcion in
            Abrir)
                echo "Esto abre un archivo"
                ;;
            Cerrar)
                echo "Eso cierra un archivo"
                ;;
            Salir)
                echo "Esto sale del programa"
                exit
                ;;
        esac
        break
    else
        case $REPLY in
            -h|--help)
                echo "Ayuda sobre el programa"
                ;;
            -v|--version)
                echo "mi-programa versión 1.0.1"
                ;;
            -q|\q)
                exit
                ;;
            *)
                echo "Opción inválida"
        esac
    fi
done
```

ACERTIJO

¿Por qué un hombre nacido el 10 de agosto del año 15 AC a las 17:00 HS y fallecido el 10 de agosto del año 15 DC a las 16:59 HS, no llegó a cumplir los 30 años?

Ayuda: La respuesta no está en la matemática. Debes razonar de forma lateral.

Responde **ANTES** del **25/05/2014**
a través de **Twitter** utilizando el *hashtag* **#AcertijoTOH5**
El nombre de los ganadores será publicado en la siguiente edición
La respuesta correcta será publicada en The Original Hacker Nº6 junto con los ganadores