Snippets en Bash para agilización de scripts de Shell

Los snippets son pequeñas fracciones de código reutilizables, que con un simple copiar y pegar solucionan un problema. Esta es una pequeña recopilación de snippets en Bash, que a diario debo utilizar para agilizar mis scripts de Shell. Tomé aquellos que utilizo con mayor frecuencia.

Escrito por: **Eugenia Bahit** (GLAMP Hacker & eXtreme Programmer)

Eugenia es **Arquitecta de Software**, docente e instructora de tecnologías **GLAMP** (GNU/Linux, Apache, MySQL, Python y PHP) y **Agile coach** (UTN) especializada en Scrum y **eXtreme Programming**. Miembro de la **Free Software Foundation**, **The Linux Foundation** y **Debian Hackers**. Creadora de python-printr, Europio Engine y colaboradora de Vim. Fundadora y Responsable Editorial de **Hackers & Developers Magazine**.

Webs:
Cursos de programación: www.cursosdeprogramacionadistancia.com
Web personal: www.eugeniabahit.com

Redes sociales:
Twitter / Identi.ca: **@eugeniabahit**

Bash es uno de los lenguajes para Shell Scripting más comunes y por lo tanto, más utilizados no solo por programadores, sino por sobretodo, por administradores de sistemas. Frecuentemente, los scripts de Shell suelen utilizarse para agilizar y por qué no, automatizar, tareas de administración del Sistema Operativo. Muchas veces, estos scripts son utilizados por el mismo administrador del sistema que los ha programado y otras tantas, los mismos están destinados a los usuarios del sistema.

Esta breve recopilación de Snippets, está destinada a cualquier tipo de scripts, ya sea para uso propio o de terceros. Vale aclarar que cada Snippet se encuentra liberado bajo licencia GNU GPL 3.0.

> *Los Snippets son pequeñas fracciones de código fuente reutilizable, que con solo copiar y pegar o a lo sumo, copiar, pegar y "customizar", pueden ser utilizados en cualquier programa.*

Snippet #1: Conocer cantidad de usuarios en el sistema

Objetivo posible: ejecución de una tarea programada que necesita correr sólo si no hay usuarios operando sobre el sistema.

```
script_a_ejecutar="./miscript.sh"

cantidad_usuarios=`who | wc -l`

if [ $cantidad_usuarios -eq 0 ]; then
   exec "$script_a_ejecutar"
fi
```

Snippet #2: Mantener un script en ejecución continua

Objetivo posible: demonio o proceso en segundo plano.

```
script_a_ejecutar="./miscript.sh"

while true; do
    exec "$script_a_ejecutar"
```

Snippet #3: Impedir ejecución de script si el usuario no es root

Objetivo posible: ejecución de cualquier tipo de tarea que requiera permisos de súper usuario.

```
if [ "$USER" != "root" ]; then
    echo "Permiso denegado."
    exit 0
fi

# ...
```

Snippet #4: Guardar y recuperar preferencias del usuario

Objetivo posible: archivo de configuración de aplicación propia.

```
app_name="myapp"
directorio_app="$HOME/.$app_name"
archivo_configuracion="$directorio_app/.config"


# Snippet 4.1:
# Crea un directorio de aplicación con el archivo de configuración si no existe
if [ ! -d "$myapp" ]; then
    mkdir $directorio_app
    touch $archivo_configuracion
fi


# Snippet 4.2:
# Solicita preferencias al usuario (a modo de ejemplo)
function pedir_preferencias() {
    echo -n "¿Color favorito? (r: rojo / v: verde) "
    read color_favorito

    if [ "$color_favorito" != "r" ] && [ "$color_favorito" != "v" ]; then
          echo "Opción inválida"
```

```
            pedir_preferencias
    fi
}


# Snippet 4.3:
# Guarda las preferencias del usuario
function guardar_preferencias() {
    pedir_preferencias
    echo "COLOR_FAVORITO = $color_favorito" > $archivo_configuracion
    echo "OTRA_VARIABLE = otro valor" >> $archivo_configuracion
}


# Snippet 4.4:
# Recupera las preferencias del usuario
function recuperar_preferencias() {
    cat $archivo_configuracion | while read varname asignation varvalue; do
           case "$varname" in
                  COLOR_FAVORITO) echo "El color favorito es $varvalue";;
                  OTRA_VARIABLE) echo "El valor de otra variable es $varvalue";;
           esac
    done
}
```

Snipet #5: Parsear argumentos

Objetivo posible: script con un menú de opciones en el cual, según la opción elegida por el usuario, sea la acción que el script realice.

```
echo -n "Elegir opción (a/b/c): "
read OPCION

case $OPCION in
    a) echo "se eligió la opción A";;
    b) echo "se eligió la opción B";;
    c) echo "se eligió la opción C";;
    *) echo "OPCIÓN INCORRECTA";;
esac
```

Snippet #6: Capturar el valor de retorno no numérico de una función

Objetivo posible: se necesita obtener un determinado valor no numérico el cual es establecido de forma dinámica por el llamado a una función y se lo necesita almacenar con un nombre de variable distinto al del definido por la función implementada en la llamada de retorno.

```
function foo() {
    # ...
    variable="Valor de retorno para el argumento $1"
    # en bash, el valor de retorno se imprime
    echo $variable
}


function bar() {
```

```
    argumento="Lorem Ipsum"
    nueva_variable=$(foo $argumento)
}
```

Snippet #7: Convertir una cadena de texto a mayúsculas

(incluyendo caracteres acentuados, diéresis y eñes)

```
#     Uso: strin2upper mi cadena de texto
# Retorna: MI CADENA DE TEXTO
string2upper() {
   echo $* | tr 'a-z|á|é|í|ó|ú|ñ|ü' 'A-Z|Á|É|Í|Ó|Ú|Ñ|Ü'
}
```

Snippet #8: Convertir una cadena de texto a minúsculas

(la inversa del anterior)

```
#     Uso: strin2lower MI CADENA DE TEXTO
# Retorna: mi cadena de texto
string2lower() {
   echo $* | tr 'A-Z|Á|É|Í|Ó|Ú|Ñ|Ü' 'a-z|á|é|í|ó|ú|ñ|ü'
}
```

Snippet #9: Verificar argumentos pasados al script

Objetivo posible: un script que pueda fallar si no le son pasados una determinada cantidad de argumentos.

```
if [ $# -ne 1 ] ; then
    echo "Uso ./script.sh <argumento1> <argumento2>..."
    exit 0
fi
```

Snippet #10: Leer un archivo y asignar números de línea

Objetivo posible: un script cuyo objetivo sea trabajar con código fuente o manejar diferencias entre diversos archivos.

```
function read_file() {
    archivo=$1
    contenido_con_numeros_de_linea=`awk '{print NR, " ", $0}' $archivo`
}
```