Escuela técnica superior
Facultad de Ingeniería Informática

Práctica 7
Scripts de bash

Ignacio Fernández Contreras

1 Parte 1

Resuelva los siguientes ejercicios de clase.

1.1 Ejercicio 1.

Escriba un script que haga lo siguiente:

- Compruebe que se le pasa al menos un parámetro, en caso contrario termina (return) el script con código 1.
- Compruebe que se le pasa un directorio como argumento, en caso contrario termina con código 2.
- Si se le pasa un directorio, debe comprobar que no es el directorio actual y que tiene permisos de búsqueda, en caso contrario devuelve el código 3. En caso afirmativo, nos desplazamos hacia él.

Notas:

- Ejecuta este script con source o '.'
- La variable $PWD nos devuelve el directorio actual.

```bash
#!/bin/bash

# Comprueba que se le pasa al menos un parámetro
if [ $# -lt 1 ]; then
    echo "Error: Debes proporcionar al menos un parámetro."
    return 1
fi

# Obtiene el directorio pasado como argumento
directorio=$1

# Comprueba que se le pasa un directorio como argumento
if [ ! -d "$directorio" ]; then
    echo "Error: El argumento no es un directorio válido."
    return 2
fi

# Comprueba que el directorio no es el directorio actual
if [ "$directorio" == "$PWD" ]; then
    echo "Error: El directorio proporcionado es el directorio actual."
    return 3
fi

# Comprueba que tiene permisos de búsqueda en el directorio
if [ ! -x "$directorio" ]; then
    echo "Error: No tienes permisos de búsqueda en el directorio proporcionado."
    return 3
fi

# Cambia al directorio proporcionado
cd "$directorio"

# Imprime el directorio actual después de cambiar
echo "Directorio actual: $PWD"
```

```
user@user:~$ source comprobar_directorio.sh
Error: Debes proporcionar al menos un parámetro.
user@user:~$ source comprobar_directorio.sh /home/user
Error: El directorio proporcionado es el directorio actual.
user@user:~$ source comprobar_directorio.sh /home/user/test
Directorio actual: /home/user/test
```

1.2 Ejercicio 2.

Extienda el script anterior para que liste todos los ficheros ejecutables que haya en el directorio pasado como argumento.

```bash
#!/bin/bash

# Comprueba que se le pasa al menos un parámetro
if [ $# -lt 1 ]; then
    echo "Error: Debes proporcionar al menos un parámetro."
    return 1
fi

# Obtiene el directorio pasado como argumento
directorio=$1

# Comprueba que se le pasa un directorio como argumento
if [ ! -d "$directorio" ]; then
    echo "Error: El argumento no es un directorio válido."
    return 2
fi

# Comprueba que el directorio no es el directorio actual
if [ "$directorio" == "$PWD" ]; then
    echo "Error: El directorio proporcionado es el directorio actual."
    return 3
fi

# Comprueba que tiene permisos de búsqueda en el directorio
if [ ! -x "$directorio" ]; then
    echo "Error: No tienes permisos de búsqueda en el directorio proporcionado."
    return 3
fi

# Cambia al directorio proporcionado
cd "$directorio"

# Imprime el directorio actual después de cambiar
echo "Directorio actual: $PWD"

# Lista todos los archivos ejecutables en el directorio
echo "Archivos ejecutables en el directorio:"
for archivo in $(find . -maxdepth 1 -type f -executable); do
    echo "$archivo"
done
```

```
user@user:~$ source comprobar_directorio_2.sh /home/user/test
Directorio actual: /home/user/test
Archivos ejecutables en el directorio:
./archivo_ejecutable
```

1.3 Ejercicio 3.

Sabiendo que el operador ${#var} devuelve el tamaño de una cadena, y utilizando el operador para obtener una subcadena ${var : offset : longitud}, cree una función Reverse que devuelva a través de variable global una cadena de caracteres invertida.

```bash
#!/bin/bash

# Función para invertir una cadena y almacenar el resultado en una variable global
Reverse() {
    local original="$1"
    local longitud=${#original}
    local invertida=""

    for (( i=longitud-1; i>=0; i-- )); do
        invertida="${invertida}${original:i:1}"
    done

    # Almacena el resultado en la variable global
    cadena_invertida=$invertida
}

# Comprueba que se le pasa un parámetro
if [ $# -eq 0 ]; then
    echo "Error: Debes proporcionar una cadena entre comillas para invertir."
    exit 1
fi

# Llama a la función Reverse con el primer argumento
Reverse "$1"

# Imprime la cadena original y la invertida
echo "Cadena original: $1"
echo "Cadena invertida: $cadena_invertida"
```

```
user@user:~$ source invertir.sh "Administracion de sistemas operativos"
Cadena original: Administracion de sistemas operativos
Cadena invertida: sovitarepo sametsis ed noicartsinimdA
user@user:~$ source invertir.sh
Error: Debes proporcionar una cadena entre comillas para invertir.
```

2 Parte 2

Explica paso a paso el siguiente script de bash:

```
function ZZZ {
IFS='
'
        ficheros=$(ls -1 $1)
        for fichero in $ficheros
        do
                path_fichero="$1/$fichero"
                if [ -x $path_fichero ]; then
                        echo $path_fichero
                fi
        done
        IFS=':'
}

IFS=':'
for dir in $PATH
do
        if [ -z "$dir" ]; then
                echo "ENCONTRADO UN DIRECTORIO VACIO"
                exit 1
        elif ! [ -d "$dir" ]; then
                echo "$dir NO ES UN DIRECTORIO VALIDO"
                exit 1
        else
                ZZZ $dir
        fi
done
```

- Función ZZZ
  - Se añade una simple línea delante del for para que no trate los espacios como separador y obtengamos el listado como queremos
  - Obtiene la lista de archivos en el directorio proporcionado ($1) y la almacena en la variable ficheros.
  - Itera sobre cada archivo en ficheros y verifica si es ejecutable.
  - Imprime la ruta completa de los archivos ejecutables encontrados.
- Bucle principal
  - añadir una simple línea delante del forpara que no trate los : como separador y obtengamos el listado como queremos
  - Itera sobre cada directorio en PATH.
  - Verifica si el directorio es una cadena vacía y, si es así, imprime un mensaje de error y sale del script con el código de salida 1.
  - Verifica si el elemento en PATH no es un directorio válido, imprime un mensaje de error y sale del script con el código de salida 1.
  - Si el directorio es válido, llama a la función ZZZ con el directorio como argumento.

3 Ej 3

La sucesión alícuota es una sucesión de números que resulta de la suma de los divisores propios del término anterior. La sucesión da comienzo en un término k mayor que 0 y termina cuando se alcanza el término 1. En caso de que se repita el término anterior o se alcancen más de 10 términos, se considera una sucesión alícuota infinita y se para la serie. A continuación, se muestra varios ejemplos de sucesiones alícuotas:

k=12 Div_{12} = 1+2+3+4+6 = **16,** Div_{16} = 1+2+4+8 = **15,** Div_{15} = 1+3+5 = **9,** Div_{9} = 1+3 = **4,** Div_{4} = 1+2 = **3,** Div_{3} = **1**	*k*=6 Div_{6} = 1+2+3 = **6** **Sucesión alícuota infinita**	*k=220* *Div_{220}=* *1+2+4+5+10+11+20+22+44+55+110=**284*** *Div_{284}= 1+2+4+71+142=**220*** *Div_{220}=* *1+2+4+5+10+11+20+22+44+55+110=**284*** *Div_{284}= 1+2+4+71+142=**220*** ... **Sucesión alícuota infinita**

Escriba un script de bash que lea un término k mayor que 0 desde teclado y muestre la sucesión alícuota de dicho término. En el caso de una sucesión alícuota infinita, cuando se detecte, se mostrará un mensaje indicándolo. A continuación, se muestran varios ejemplos de la ejecución del programa:

```
El termino k es: 12
La sucesion alicuota es:
16 15 9 4 3 1

El termino k es: 6
La sucesion alicuota es:
La secuencia alicuota es infinita

El termino k es: 220
La sucesion alicuota es:
284 220 284 220 284 220 284 220 284 220 La secuencia alicuota es
infinita
```

```bash
#!/bin/bash

# Función para calcular la sucesión alícuota de un término
function sucesion_alicuota {
    local k=$1
    local contador=1

    while [ $k -ne 1 ] && [ $contador -le 10 ]; do
        echo -n "$k "
        k=$(divisores_propios $k)
        ((contador++))
    done

    if [ $contador -gt 10 ]; then
        echo -e "\nSucesión alícuota infinita detectada."
    else
        echo ""
    fi
}

# Función para obtener la suma de los divisores propios de un número
function divisores_propios {
    local num=$1
    local suma=0

    for (( i=1; i<=$num/2; i++ )); do
        if [ $((num % i)) -eq 0 ]; then
            ((suma += i))
        fi
    done

    echo "$suma"
}

# Lectura del término k desde teclado
read -p "El término k es: " k

# Verifica que k sea mayor que 0
if [ $k -le 0 ]; then
    echo "El término k debe ser mayor que 0."
    exit 1
fi

# Muestra la sucesión alícuota en la misma línea
echo -n "La sucesión alícuota es: "
sucesion_alicuota $k
```

```
user@user:~$ source alicuota.sh
El término k es: 12
La sucesión alícuota es: 12 16 15 9 4 3
user@user:~$ source alicuota.sh
El término k es: 6
La sucesión alícuota es: 6 6 6 6 6 6 6 6 6 6
Sucesión alícuota infinita detectada.
user@user:~$ source alicuota.sh
El término k es: 220
La sucesión alícuota es: 220 284 220 284 220 284 220 284 220 284
Sucesión alícuota infinita detectada.
```