Soluciones taller de Shell Script

1. Realizar un script llamado '**01-hola-mundo.sh**' que muestre por pantalla "Hola mundo!".

```
#! /bin/bash

echo "Hola mundo!"
```

2. Ídem pero que en vez de "mundo" muestre los parámetros introducidos ('**02-hola-parametros.sh**').

```
#! /bin/bash

TODOS_LOS_PARAMETROS=$@
echo "Hola $TODOS_LOS_PARAMETROS!"
```

3. Ídem y que además verifique que al menos hayamos introducido un parámetro ('**03-hola-al-menos-1-parametro.sh**').

```
#! /bin/bash

NUMERO_DE_PARAMETROS=$#
if [ $NUMERO_DE_PARAMETROS -le 0 ]; then
    echo "Hay que introducir al menos un parámetro."
    exit 1
fi

TODOS_LOS_PARAMETROS=$@
echo "Hola $TODOS_LOS_PARAMETROS!"
```

4. Ídem y que además separe cada argumento por ", " ('**04-hola-parametros-separados.sh**').

```
#! /bin/bash

NUMERO_DE_PARAMETROS=$#
if [ $NUMERO_DE_PARAMETROS -le 0 ]; then
    echo "Hay que introducir al menos un parámetro."
    exit 1
fi

MENSAJE="Hola"
ES_PRIMERO=1
PRIMER_PARAMETRO=$1

# mientras haya parámetros
while [ -n "$PRIMER_PARAMETRO" ]; do

    if [ $ES_PRIMERO -eq 1 ]; then
        MENSAJE="$MENSAJE $PRIMER_PARAMETRO"
        ES_PRIMERO=0
    else
        MENSAJE="$MENSAJE, $PRIMER_PARAMETRO"
    fi

    # pasamos al siguiente parámetro
    shift
    PRIMER_PARAMETRO=$1
done

# mostramos la salida por pantalla
echo $MENSAJE"!"
```


5. Ídem y que además en caso de error muestra una ayuda ('**05-hola-con-ayuda.sh**').

```
#! /bin/bash

# función de ayuda
function ayuda() {
cat << DESCRPCION_AYUDA
SYNOPSIS
    $0 NOMBRE_1 [NOMBRE_2] ... [NOMBRE_N]

DESCRIPCION
    Muestra "Hola NOMBRE_1, NOMBRE_2, ... NOMBRE_N!" por pantalla.

CÓDIGOS DE RETORNO
    1 Si el número de parámetros es menor que 1

DESCRPCION_AYUDA
}

NUMERO_DE_PARAMETROS=$#
if [ $NUMERO_DE_PARAMETROS -le 0 ]; then
    echo "Hay que introducir al menos un parámetro."
    ayuda
    exit 1
fi

MENSAJE="Hola"
ES_PRIMERO=1
PRIMER_PARAMETRO=$1

# mientras haya parámetros
while [ -n "$PRIMER_PARAMETRO" ]; do

    if [ $ES_PRIMERO -eq 1 ]; then
        MENSAJE="$MENSAJE $PRIMER_PARAMETRO"
        ES_PRIMERO=0
    else
        MENSAJE="$MENSAJE, $PRIMER_PARAMETRO"
    fi

    # pasamos al siguiente parámetro
    shift
        PRIMER_PARAMETRO=$1
done

# mostramos la salida por pantalla
echo $MENSAJE"!"

exit 0
```


6. Ídem y que además verifique que sean usuarios conectados al sistema ('**06-hola-usuario.sh**').

```
#! /bin/bash

# función de ayuda
function ayuda() {
cat << DESCRPCION_AYUDA
SYNOPSIS
    $0 NOMBRE_1 [NOMBRE_2] ... [NOMBRE_N]

DESCRIPCION
    Muestra "Hola NOMBRE_1, NOMBRE_2, ... NOMBRE_N!" por pantalla.

CÓDIGOS DE RETORNO
    1 Si el número de parámetros es menor que 1
    2 Si el usuario no está conectado
DESCRPCION_AYUDA
}

NUMERO_DE_PARAMETROS=$#
if [ $NUMERO_DE_PARAMETROS -le 0 ]; then
    echo "Hay que introducir al menos un parámetro."
    ayuda
    exit 1
fi

MENSAJE="Hola"
ES_PRIMERO=1
PRIMER_PARAMETRO=$1

# mientras haya parámetros
while [ -n "$PRIMER_PARAMETRO" ]; do

    ESTA_CONECTADO=`who | egrep ^$PRIMER_PARAMETRO' '`

    if [ -z "$ESTA_CONECTADO" ]; then
        echo "El usuario $PRIMER_PARAMETRO no está conectado"
        ayuda
        exit 2
    fi

    if [ $ES_PRIMERO -eq 1 ]; then
        MENSAJE="$MENSAJE $PRIMER_PARAMETRO"
        ES_PRIMERO=0
    else
        MENSAJE="$MENSAJE, $PRIMER_PARAMETRO"
    fi

    # pasamos al siguiente parámetro
    shift
    PRIMER_PARAMETRO=$1
done

# mostramos la salida por pantalla
echo $MENSAJE"!"
```

7. Realizar un script llamado '**07-usuarioconectado**' que retorna un SI si el primer parámetro coincide con algún usuario conectado o NO en caso contrario.

```
#! /bin/bash

function ayuda() {
cat << DESCRIPCION_AYUDA
SYNOPSIS
    $0 NOMBRE_USUARIO

DESCRIPCION
    Devuelve:
        SI si NOMBRE_USUARIO coincide con algún usuario conectado o
        NO si NOMBRE_USUARIO no coincide con ningún usuario conectado

CÓDIGOS DE RETORNO
  1 Si el número de parámetros es distinto de 1
DESCRIPCION_AYUDA
}

# si número de parámetros distinto 1
if [ $# -ne 1 ]; then
    echo "El número de parámetros debe de igual a 1"
    ayuda
    exit 1
fi

ESTA_CONECTADO=`who | grep $1`

if [ -z "$ESTA_CONECTADO" ]; then
    echo "NO"
else
    echo "SI"
fi
```

8. Modificar el fichero '**.bashrc**' para modificar el PATH y añadir la carpeta de estos ejercicios. Para ello añade la siguiente linea: **export PATH=$PATH":~/ruta_carpeta_taller"**

```
Con esto ponemos el comando en el PATH para que pueda ejecutarse desde
cualquier sitio.
```


9. Modificar el script '**06-hola-usuario.sh**' para que llame a 'usuarioconectado' y renombralo como '**09-hola-usuario.sh**'.

```
#! /bin/bash

# función de ayuda
function ayuda() {
cat << DESCRPCION_AYUDA
SYNOPSIS
    $0 NOMBRE_1 [NOMBRE_2] ... [NOMBRE_N]

DESCRIPCION
    Muestra "Hola NOMBRE_1, NOMBRE_2, ... NOMBRE_N!" por pantalla.

CÓDIGOS DE RETORNO
    1 Si el número de parámetros es menor que 1
    2 Si el usuario no está conectado
DESCRPCION_AYUDA
}

NUMERO_DE_PARAMETROS=$#
if [ $NUMERO_DE_PARAMETROS -le 0 ]; then
    echo "Hay que introducir al menos un parámetro."
    ayuda
    exit 1
fi

MENSAJE="Hola"
ES_PRIMERO=1
PRIMER_PARAMETRO=$1

# mientras haya parámetros
while [ -n "$PRIMER_PARAMETRO" ]; do

    ESTA_CONECTADO=`./07-usuario-conectado $1`

    if [ "$ESTA_CONECTADO" == "NO" ]; then
        echo "El usuario $1 no está conectado"
        ayuda
        exit 2
    fi

    if [ $ES_PRIMERO -eq 1 ]; then
        MENSAJE="$MENSAJE $PRIMER_PARAMETRO"
        ES_PRIMERO=0
    else
        MENSAJE="$MENSAJE, $PRIMER_PARAMETRO"
    fi

    # pasamos al siguiente parámetro
    shift
    PRIMER_PARAMETRO=$1
done

# mostramos la salida por pantalla
echo $MENSAJE"!"
```


10. Realizar a mano un fichero '**10-notas.csv**' con los siguientes datos:

```
Pepito      3.1   4.4   5.7
Fulanito    4.2   6.5   8.8
Menganito   5.3   5.6   5.0
```

11. Realizar un fichero '**11-notas.awk**' y su correspondiente interfaz '**11-notas.sh**' para que al final obtengamos algo parecido a esto:

```
+------------------------------+-------+------+
| NOMBRE      EX1   EX2   EX3  |  MED  | APTO |
+------------------------------+-------+------+
| Pepito      3.1   4.4   5.7  |  4.4  |  NO  |
| Fulanito    4.2   6.5   8.8  |  6.5  |  SI  |
| Menganito   5.3   5.6   5.0  |  5.3  |  SI  |
+------------------------------+-------+------+
| TOTAL       4.2   5.5   6.5  |  5.4  |   2  |
+------------------------------+-------+------+
```

11-notas.awk:

```
# esto se ejecutará solo una vez al principio
BEGIN {
    print "+----------------------------+-----+------+"
    print "| NOMBRE      EX1  EX2  EX3 | MED | APTO |"
    print "+----------------------------+-----+------+"
}

# esto se ejecutará para cada una de las líneas del fichero
{
    suma2+=$2
    suma3+=$3
    suma4+=$4
    mediaFila=($2+$3+$4)/3
    apto="NO"
    if ( mediaFila >= 5 ) {
        apto="SI"
        aptos++
    }
    print "| "$0" | "mediaFila" |  "apto"  |"
}

# esto se ejecutará solo una vez al final
END {
    media2=suma2/3
    media3=suma3/3
    media4=suma4/3
    media=(media2+media3+media4)/3
    print "+----------------------------+-----+------+"
    print "| MEDIAS      "media2"  "media3"  "media4" | "media" |   "aptos"
|"
    print "+----------------------------+-----+------+"
}
```

11-notas.sh:

```
awk -f 11-notas.awk 10-notas.csv
```

12. Realizar un demonio llamado '**12-alerta**' que escriba la fecha cada X segundos en un log llamado '**~/12-alerta.log**'.

```
#! /bin/bash

# función de ayuda
function ayuda() {
cat << DESCRIPCION_AYUDA
SYNOPIS
    $0 [SEGUNDOS]
DESCRIPCION
    Escribe la fecha cada X segundos en el log '~/alerta.log'
CODIGOS DE RETORNO
    0 Si no hay ningún error
DESCRIPCION_AYUDA
}

# si primer parámetro == '-h' o == '--help'
if [ "$1" == "-h" -o "$1" == "--help" ]; then
    ayuda
    exit 0
fi

function main() {

    DEFAULT=2
    SEGUNDOS=$1

    # comprobar que SEGUNDOS es un número
    if [ "$SEGUNDOS" != "0" -a "`echo $SEGUNDOS | awk '{ print $1 * 1 }'`" !=
"$SEGUNDOS" ]; then
        echo "El parámetro '$SEGUNDOS' no es un número. Se cogerá el valor por
defecto ($DEFAULT)"
        SEGUNDOS=$DEFAULT
    fi

    # reinicio alerta.log
    echo "" > ~/alerta.log

    while [ true ]; do
        date +%d/%m/%Y" "%H:%M:%S >> ~/alerta.log
        sleep $SEGUNDOS
    done
}

main $1
```

13. Realizar las interfaces del demonio '12-alerta' con las opciones básicas: start, stop, restart y status y llamarlo '**13-servicio-alerta.sh**'.

```
#! /bin/bash

# función de ayuda
function ayuda() {
cat << DESCRIPCION_AYUDA
SYNOPIS
    $0 start|stop|restart|status

DESCRIPCIÓN
    Muestra que arraca, para, relanza y nos muestra el estado de 'alerta'.

CÓDIGOS DE RETORNO
    0 Si no hay ningún error.
DESCRIPCION_AYUDA
}

DAEMON=12-alerta
PIDFILE=/tmp/$DAEMON.pid

# función que arranca 'alerta'
function do_start() {
    # si exite el fichero
    if [ -e $PIDFILE ]; then
         echo "El proceso ya se está ejecutando."
         exit 0;
    fi
    ./$DAEMON &
    echo $! > $PIDFILE
    echo "Ejecutandose..."
}

# función que para 'alerta'
function do_stop() {
    # si exite el fichero
    if [ -e $PIDFILE ]; then
        kill -9 `cat $PIDFILE`
        rm $PIDFILE
    fi
    echo "Parado."
}

# función que para y arrance 'alerta'
function do_restart() {
    do_stop
    do_start
}
```

```
# función que muestra el estado de 'alerta'
function do_status() {
    # si exite el fichero
    if [ -e $PIDFILE ]; then
        echo "Ejecutandose..."
    else
        echo "Parado."
    fi
}

# si primer parámetro == '-h' o == '--help'
if [ "$1" == "-h" -o "$1" == "--help" ]; then
    ayuda
    exit 0
fi

case $1 in
    start)
        do_start ;;
    stop)
        do_stop ;;
    restart)
        do_restart ;;
    status)
        do_status ;;
    *)
        echo "Parámetro '$1' incorrecto." ;;
esac
```

14. Crear un script llamado '**14-array.sh**' que declare un array, lo rellene con datos y luego itere sobre el mismo para mostrar los datos.

```
declare -a ARRAY;

ARRAY=("cero" "uno" [3]="tres")
ARRAY[2]="dos"

LENGTH=${#ARRAY[*]}

for (( i=0; i<LENGTH; i++ )); do
    echo $i=${ARRAY[i]}
done
```

15. Realizar a mano un fichero '**15-roles.csv**' con los siguientes datos:

```
Pepito:Jefe,Sistemas
Fulanito:Jefe,Desarrollo
Menganito:Operario,Sistemas,Desarrollo
```

16. Realizar un script '**16-roles-sin-awk.sh**', que, sin utilizar awk, al final obtengamos algo parecido a esto:

```
Desarrollo
 -> Fulanito Menganito
Operario
 -> Menganito
Sistemas
 -> Pepito Menganito
Jefe
 -> Pepito Fulanito
```

```
ROLES_FILE=./15-roles.csv

ROLES=`cut -d : -f 2 $ROLES_FILE | sed 's/,/\n/g' | sort | uniq`

for ROL in $ROLES; do
    echo $ROL
    NAMES=`grep -E $ROL $ROLES_FILE | cut -d : -f 1`
    echo " -> "$NAMES
done
```

17. Realizar un fichero '**17-roles.awk**' y su correspondiente interfaz '**17-roles-con-awk.sh**' para que al final obtengamos lo mismo que el ejercicio anterior.

17-roles.awk:

```
# esto se ejecutará solo una vez al principio
BEGIN {
    FS = ",|:"
}

# esto se ejecutará para cada una de las líneas del fichero
{
    nombre=$1
    for (N=2; N<=NF; N++) {
        rol=$N
        roles[rol]=""roles[rol]" "nombre
    }
}

# esto se ejecutará solo una vez al final
END {
    for ( rol in roles) {
        print rol
        print " ->" roles[rol]
    }
}
```

17-roles-con-awk.sh:

```
awk -f 17-roles.awk 15-roles.csv
```