Solución Script-práctica 1

Validación/analisis de parámetros

El método que he seguido para validar y analizar los parámetros ha sido con el comando “getopts”, contenido dentro de un bucle while.

Con la cabezera que se muestra en la imagen:

```
while getopts ":u:gdf" opcion; do
  case $opcion in
```

Como podemos comprobar, **-u** espera que se le pase un parámetro…

Para poder solucionar el procesamiento de los parámetros cuando el usuario introduce **-u**

Seguido de **-g ,-d** o **-f**, tuve que apoyarme en algunas sentencias condicionales dentro de la opcion **-u**, para que **getopts** no los tomara como parámetros

```
#Comprobamos si existe el usuario (silenciamos errores del sistema)
comprueba=$(cat /etc/passwd|grep -w "$OPTARG" 2>/dev/null|cut -d ":" -f1)

 if  [ -n "$comprueba" ]
   then
     usuario=$comprueba #si existe,guardamos el nombre de usuario
     declare  -i existe=1 #true

   #sino, si se introduce algun parámetro sin nombre de usuario,se procesa
   elif [ "$OPTARG" = "-g" ]
     then

      parametros[g]=1 # si es = 1,se especificó como parámetro...
     ((numParams++))  # para llevar la cuenta de los parámetros
      existe=0 #no existe (false)

   elif [ "$OPTARG" = "-d" ]
     then

      parametros[d]=1 #*
     ((numParams++))
      existe=0 #no existe (false)

   elif [ "$OPTARG" = "-f" ]
     then

      parametros[f]=1 #*
     ((numParams++))
      existe=0 #no existe (false)
```

Lo primero que hago dentro de la opcion **-u** es comprobar si el usuario introducido existe,y,en ese caso,lo guardo en la variable **usuario** ,y además declaro una variable booleana **existe** a true (1),todo esto para más adelante…

Si el usuario introducido como parámetro no existe ,la variable **existe** será false (0),y además declaro 3 sentencias **elif** para poder procesar los tres parámetros en el caso de que el usuario quiera ver la información de todos los usuarios del sistema,ya que sino **getopts** los tomaría como parámetro de **-u** y el script produciría errores.

En la siguiente imagen podemos ver que si el usuario introdujera **-u -d** o cualquiera de los tres,

El array **parametros[d]** contendría 1 (true) y la variable **numParams,**encargada de contabilizar el número de parámetros introducidos,sumaría 1 (++).

Esto habria que hacerlo con los tres parámetros especificados en la cabezera de **getopts**, y además **existe**=false en cada una de ellas.

```
elif [ "$OPTARG" = "-d" ]
  then

   parametros[d]=1 #*
  ((numParams++))
   existe=0 #no existe (false)
```

El resto de opciones,quedarían de la siguiente manera:

```
g)
    parametros[g]=1 # si es = 1,se especificó como parámetro...
   ((numParams++))  #para llevar la cuenta de los parámetros


  ;;

d)
    parametros[d]=1 #*
   ((numParams++))

  ;;

f)
    parametros[f]=1 #*
    ((numParams++))
```

Sería practicamente igual que en las sentencias **elif**,porque en el caso de que se introduzca **-u** seguido de un nombre de usuario introducido existente,y,al menos un parámetro,**getopts** entraría en cada una de las opciones.

Para cerrar **getopts**,lo hago con estas dos opciones que ejecutan una función llamada **ayuda** para mostrar la respectiva ayuda del script en caso de no introducir a **-u** ningún parámetro o uno inexistente en **getopts.**

```
159
160       ;;
161
162      \?)
163
164           #Llamada a la función ayuda,si se especifica un parámetro inexistente
165           ayuda
166           exit 1
167         ;;
168
169        *)
170           #si no se especificó ningún parámetro para (-u) se muestra ayuda.
171           ayuda
172           exit 1
173         ;;
174
175
176
177   esac
178
179 done
180
181
```

Función **ayuda.**

```
 6 function ayuda {
 7
 8 cat << TEXTO
 9
10         ¡Uso incorrecto del comando $0!
11         ========================================
12
13 Se esperaba la siguiente sintaxis:
14
15   $0 [-u NombreDeUSuario] [-g] [-d] [-f]
16
17
18     -g --> Muestra los grupos a los que pertenece el usuario
19
20     -d --> Muestra el número de directorios del directorio de usuario
21
22     -f --> Muestra el número de archivos del directorio de usuario
23
24 Debiéndose especificar al menos uno de los 3 parámetros.
25
26  "De no especificar el nombre de usuario,se obtendrá la información de
27     todos los usuarios del sistema".
28
29               Author:David Pérez Pardo
30
31 TEXTO
32 }
33
```

Comprobación de usuario y obtención de su directorio personal

La comprobación de usuario (en caso de ser especificada) la realizo abriendo el fichero que contiene toda la información de los usuarios del sistema con el comando **cat**, y mediante tuberias me apoyo en **grep –w** para contar las líneas que contengan el nombre de usuario especificado (silenciando los posibles errores del sistema si un usuario no existe) y obteniendo con **cut –d** el trozo con delimitador **:** que contiene el nombre de usuario en la posición 1 de la línea (ahí se guardan los nombres de usuarios).

```
comprueba=$(cat /etc/passwd|grep -w "$OPTARG" 2>/dev/null|cut -d ":" -f1)
```

Para la obtención del directorio del usuario introducido,me apoyo en los mismos comandos,cambiando la posición en la que “corto” el trozo delimitado por **:**,siendo este la 6,son se almacena la ruta del directorio del usuario.

```
dirUsuario=$( cat /etc/passwd|grep "$i"|cut -d ":" -f6 )
```

En el caso de no especificarse un parámetro con nombre de usuario,se mostraría la información de todos los usuarios del sistema,de la siguiente manera:

```
function usuarios {

  if [ $# -eq 0 ]
    then
      local users=$(cat /etc/passwd|cut -d ":" -f1)
    else
      users="$1"
  fi



  #Obtiene el nombre de cada usuario del sistema


  for i in $users
  do

    #obtenemos el directorio de cada usuario del fichero /etc/passwd
    dirUsuario=$( cat /etc/passwd|grep "$i"|cut -d ":" -f6 )
```

La función usuarios evalúa si se le pasa como parámetro un nombre de usuario y lo guarda en una variable **users.**

En el caso de que no se le pasen parámetros,la variable **local users** almacenaria la instrucción

```
local users=$(cat /etc/passwd|cut -d ":" -f1)
```

Dentro de la función tenemos un bucle for,al que según el caso,la variabla contador **i** tendría como valor un nombre de usuario,o si no se especificó nombre alguno,tomaría el valor de **local users**,haciendo que el bucle recorra cada línea del archivo passwd obteniendo todos los nombres de usuarios.

Obtención del numero de directorios,archivos y grupos del usuario

Mediante 3 ifs,comprobamos qué parámetros se especificaron en **getopts,**y con ello vamos obteniendo toda la información.

```
#si se especifica -d (directorios)
if  [ ${parametros[d]} -eq 1 ]
  then

    #se cuentan directorios del directorio del usuario (no se muestran los ocultos)
    echo  "Nº de directorios- del directorio de usuario $i: $( ls -l $dirUsuario 2>/dev/null|grep "^d"|wc -l)"
fi
```

Para obtener el numero de directorios,utilize el comando **ls -l dirUsuario** obtenido anteriormente y silenciando los errores,comando **grep y** contamos todas las lineas que empiezen por **d** (directorios) con **wc –l.**

```
#obtenemos el directorio de cada usuario del fichero /etc/passwd
dirUsuario=$( cat /etc/passwd|grep "$i"|cut -d ":" -f6 )

#si se especifica -g (grupos)
if  [ ${parametros[g]} -eq 1 ]

  then
    echo  "Grupos del usuario $(groups $i)"

fi


#si se especifica -d (directorios)
if  [ ${parametros[d]} -eq 1 ]
  then

    #se cuentan directorios del directorio del usuario (no se muestran los ocultos)
    echo  "Nº de directorios- del directorio de usuario $i: $( ls -l $dirUsuario 2>/dev/null|grep "^d"|wc -l)"
fi

#si se especifica -f (ficheros)
if  [ ${parametros[f]} -eq 1 ]
  then

  #se cuentan archivos del directorio del usuario (no se muestran los ocultos)
  echo "Nº de -archivos- del directorio de usuario $i: $( ls -l $dirUsuario 2>/dev/null|grep "^-"|wc -l)"

fi
```

Lo mismo sería para los archivos,cambiando la **d** por **–** (archivos).

Para obtener los grupos,basta con el comando **groups** seguido del nombre del usuario.

Desde aquí es donde se realiza la llamada a la función **usuarios** que realiza todo el trabajo.

Todo al finalizar **getopts.**

Si el usuario es introducido,existe, y al menos hay un parámetro,lo pasamos como parámetro a la funcion.

Si no se especifica un usuario,y se especifica la menos un parámetro,se llama a la función sin más.

En caso de no especificarse ningún parámetro,el script muestra la **ayuda** y **exit 1.**

```
  esac

done


#Si no se especificó ningúno de los 3 parámetros,salimos y mostramos la ayuda.
if  [ $numParams -eq 0 ]
  then
    ayuda #funciona que muestra la ayuda
    exit 1
  elif  [ "$existe" -eq 1 ] #si se pasa al menos un parámetro,y existe el usuario,l
    then
      usuarios $usuario #se pasa el nombre de usuario obtenido a la funcion "usuari
  elif  [ "$existe" -eq 0 ]
    then
      usuarios


fi
```