Programación de shell-scripts: Práctica 2

NOTA IMPORTANTE: En todos los ejercicios deberá comprobarse que el número de parámetros es correcto. En caso contrario deberá indicarse el error y notificar al usuario con un mensaje en pantalla con el uso del shell-script. Por ejemplo:

"El número de parámetros es incorrecto"
"*USO: <nombre_shell-script> <argumentos>*"

sustituyendo <nombre_shell-script> por el nombre del shell-script en cada caso y <argumentos> por la lista de argumentos que espere con un nombre autoexplicativo del argumento y encerrado entre < > si es obligatorio o entre [ ] si es opcional.

1. Hacer un shell-script **NumeroUsuarios** que indique a la salida el número de usuarios que tiene el sistema. Utilizar para ello el comando ***getent passwd*** que vuelca a la salida estándar la base de datos de usuarios del sistema en formato CSV. El mensaje de salida debe ser exactamente el siguiente:

   El sistema tiene ?????? usuarios.

2. Hacer un shell-script **ListaUsuarios** que, usando el comando ***getent passwd***, vuelque a la salida los usuarios del sistema, ordenados por su UID (identificador de usuario).

3. Hacer un shell-script **MaxUID** que, usando el comando ***getent passwd***, vuelque a la salida el máximo UID del sistema.

4. Hacer un shell-script **UsuariosConShell** que, usando el comando ***getent passwd***, vuelque a la salida la lista de usuarios que tienen el *shell* indicado. La sintaxis sería:

   UsuariosConShell <shell>

   El mensaje de salida debe ser exactamente el siguiente:

   La lista de usuarios con el shell ???? es:
   <usuario1>
   <usuario2>
   .....
   <usuarioN>

5. Hacer un shell-script **InfoUsuario** que, usando el comando ***getent passwd,*** vuelque a la salida la información de dicho usuario. La sintaxis sería:

   InfoUsuario <usuario>

   El mensaje de salida debe ser exactamente el siguiente:

   Información del usuario <usuario>:
   UID: ?????
   GID: ?????
   Directorio personal: ?????
   Intérprete de comandos: ??????

6. Hacer un shell-script **InfoUsuarios** que, vuelque a la salida la misma información del ejercicio anterior, pero para varios usuarios. La sintaxis sería:

   InfoUsuarios <usuario1> <usuario2> …. <usuarioN>

7. Hacer un Shell-script **CrearUsuario** que añada un nuevo usuario al sistema. Para ello deberá añadir una línea al fichero *passwd* con el formato exigido por dicho fichero. La sintaxis de ejecución será:

   CrearUsuario <usuario> <UID> <GID> <comentario> <shell_usuario>

   El directorio personal del usuario estará bajo el directorio /home y se llamará como el propio usuario.

   Además, se deberá validar que el usuario y el UID no existen (son nuevos), utilizando para ello el comando *getent passwd* y realizando la búsqueda pertinente. También se deberá validar que el GID indicado existe, utilizando para ello el comando *getent group*. En caso de que no se cumplan estos requisitos se saldrá con error.

   NOTA IMPORTANTE: Para tener un fichero de ejemplo cópiese el fichero */etc/passwd* al directorio donde se vaya a realizar el Shell-script.

8. Hacer un shell-script **ProcesosUsuario** que, usando el comando ***ps uaxw***, vuelque a la salida solo los procesos del usuario indicado por parámetro. La sintaxis sería:

   ProcesosUsuario <usuario>

9. Hacer un shell-script **ExisteUsuario** que, usando el comando ***getent passwd***, indique si un usuario pertenece o no al sistema. La sintaxis sería:

   ExisteUsuario <usuario>

   Los mensajes deberán ser exactamente así:

   "El usuario <usuario> no pertenece al sistema" o bien

   "El usuario <usuario> pertenece al sistema" sustituyendo <usuario> por el nombre de usuario

10. Mejorar los shell-script realizados en los ejercicios 5 y 8 (llamar a los shell-scripts **InfoUsuarioMejorado** y **ProcesosUsuarioMejorado** respectivamente) de forma que, usando el shell-script realizado en el ejercicio 8, compruebe al principio de su ejecución si existe o no el usuario indicado por parámetro. En caso de que no exista deberá visualizar un mensaje de error indicándolo.

11. En el directorio ***/home/asignaturas/so/shell-scripts*** encontrarás un fichero CSV (**alumnos.csv**) con datos de alumnos. Realizar un Shell-script llamado **AlumnosGrupoPracticas** que muestre el nombre y apellidos de todos los alumnos que pertenecen al grupo de prácticas pasado como parámetro. La ruta al fichero CSV también se pasará como parámetro (no puede ponerse directamente **alumnos.csv** en el código del Shell-script, tiene que funcionar con cualquier fichero CSV que se indique). La sintaxis sería:

    AlumnosGrupoPracticas <fichero_CSV> <grupo_practicas>

    La salida del Shell-script deberá ser exactamente como la siguiente:

    La lista de alumnos que pertenecen al grupo de prácticas XXXX es:
    Alumno1
    Alumno2
    …..

ANEXO: Shell-scripts

Ficheros en formato CSV

Son **ficheros de texto** que almacenan la información de una tabla de una base de datos. Existen dos caracteres especiales en el fichero que son muy importantes para estructurar la información:

1. **El carácter delimitador de registros**, que, normalmente, es el carácter de fin de línea. Este carácter separa unos registros de otros en la tabla. Si, como es habitual, es el carácter fin de línea entonces cada línea del fichero es un registro.
2. **El carácter delimitador de campos** que normalmente es alguno de los siguientes: , (coma) ; (punto y coma) : (dos puntos), pero puede ser cualquier carácter que quiera el creador del fichero (eso sí una vez decidido debe mantenerse en todo el fichero). Este carácter separa unos campos de otros dentro de un registro.

Ya sabemos de bases de datos que los campos de un registro son LOS MISMOS para todos los registros del fichero (en la misma cantidad y en el mismo orden).

El fichero CSV puede tener OPCIONALMENTE una PRIMERA LÍNEA describiendo cada campo de los registros y separando cada descripción por el delimitador de campos. Esta primera línea no formaría entonces parte de la información y habría que omitirla al tratar el fichero.

Hay comandos UNIX que están preparados para extraer información de los ficheros CSV. Estos comandos UNIX asumen por defecto que el separador de registros es el fin de línea. También asumen un separador de campos por defecto (normalmente dos puntos) pero **todos tienen una opción para indicar el separador de campos si fuera otro**. También **suelen tener opciones para acceder a un campo concreto del registro o a varios de ellos**. Ejemplos de comandos UNIX que soportan CSV son: ***cut*** y ***sort***

Ejecutar el código producido en la salida estándar de un programa

Al ejecutar un programa podemos hacer que todo lo que genere por la salida estándar forme parte del código del Shell-script. Para ello debe "envolverse" el programa bien entre $() o bien entre apóstrofes inversas.
Así, por ejemplo, podemos almacenar en una variable el resultado de contar las líneas de un fichero:
n=$(wc -l /etc/passwd) o bien
n=`wc -l /etc/passwd`

El intérprete PRIMERO **ejecuta el programa entre $( ) ó ` `**. En este caso: **wc -l /etc/passwd**

En SEGUNDO LUGAR, **sustituye toda la expresión $() ó ´ ´ por el texto generado por el programa a la salida estándar**. Si, por ejemplo, el número de línea del fichero ***/etc/passwd*** fuera **23**, quedaría el siguiente código:
n=23

Finalmente, en TERCER LUGAR, **el intérprete ejecuta la línea completa resultante** por lo que asignará el texto **23** a la variable **n**

A partir de ese momento podremos visualizar el número de líneas del fichero a través de la variable que lo contiene:
echo $n

Otro ejemplo:
Razona que haría la siguiente orden:
echo "El número de usuarios es $(getent passwd | wc -l)"

Expresiones regulares y el comando grep

El comando **grep** por defecto busca, en cualquier lugar de la línea, el texto indicado por parámetro. La búsqueda es "por contención" y esto puede hacer que la búsqueda sea poco precisa.
Para hacer búsquedas más precisas **se pueden utilizar expresiones regulares. Sobre todo, si el texto en el que estamos buscando tiene formato CSV y queremos buscar por un campo concreto**.
El comando **grep** reconoce ciertos caracteres como símbolos especiales para indicar una expresión regular:

^ indica comienzo de línea
$ indica fin de línea
. indica cualquier carácter
* indica que se repita la expresión inmediatamente anterior cero o más veces
+ indica que se repita la expresión inmediatamente anterior una o más veces

Hay más símbolos especiales (leer documentación del **grep** con **man**) pero con los anteriores se puede ya buscar bien por un campo concreto de un fichero CSV. Por ejemplo, este comando **grep** buscaría el número 23 en el tercer campo del fichero CSV generado por el comando **getent passwd**:

getent passwd | grep "^.*:.*:23:.*$"

Obsérvese que la expresión regular va entre comillas. El primer **.^*:** SE SALTA los caracteres correspondientes al primer campo, el segundo **.*:** SE SALTA los caracteres correspondientes al segundo campo. A continuación, ponemos literalmente **23:** para que busque exactamente 23 como tener campo. Finalmente ponemos **.*$** para que SE SALTE todos los caracteres que quedan en la línea (y por tanto en el registro).