ASO: EJERCICIOS UD01 / UD02

1.- Varias personas de nuestra organización consultan de forma habitual el fichero **que contiene las cuentas de usuario**.

Nos han dicho que les resultaría más cómodo verlo en otro formato en lugar del mostrado por diversos comandos como cat, more, etc. El formato idóneo sería html, el cual podría ser abierto con cualquier navegador.

Realizar un script que muestre, en formato tabla de HTML la siguiente información**: login, la información en general, directorio de conexión y shell** de los usuarios del sistema con un **ID que se encuentre entre el 1000 y el 1999**, es decir, que muestre las cuentas de usuario consideradas normales (asumimos que no vamos a tener más de 2000 usuarios en el sistema).

Nuestro script, ***passwdToHTML.sh*** debe generar, cuando se ejecute, el fichero **passwd.html (en el mismo directorio en el que se ejecuta el script)**, el cual debe mostrar la información solicitada en forma de tabla como se muestra en la imagen 1ª.

Imagen 1ª: passwd.html generado por el script abierto con un navegador:

antonio	antonio,,,	/home/antonio	/bin/bash
alberto	,,,	/home/alberto	/bin/bash
miguelangel	,,,	/home/miguelangel	/bin/bash
test		/home/test	/bin/sh
test2	,,,	/home/test2	/bin/bash

Imagen nº 2: contenido del fichero que contiene las cuentas de usuario.

```
antonio:x:1000:1000:antonio,,,:/home/antonio:/bin/bash
vboxadd:x:999:1::/var/run/vboxadd:/bin/false
postgres:x:114:126:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
odoo:x:117:134::/var/lib/odoo:/usr/sbin/nologin
sshd:x:118:65534::/run/sshd:/usr/sbin/nologin
alberto:x:1001:1001:,,,:/home/alberto:/bin/bash
miguelangel:x:1002:1002:,,,:/home/miguelangel:/bin/bash
test:x:1003:1003::/home/test:/bin/sh
test2:x:1004:1004:,,,:/home/test2:/bin/bash
```

Como se puede observar, el script selecciona los UIDs adecuados.

Observación: recordemos que una tabla básica (con un borde) creada con HTML consta de la etiqueta <table border="1"> </table>. Cada fila dentro de una tabla se define con la etiqueta <tr> </tr> y cada columna dentro de una fila con <td></td>

Para este script no hacen falta más elementos para generar un fichero html (<html>, <head>, etc.).

Código script debajo

-
-
-
-
-
-
→

```
#!/bin/bash
FILE=passwd.html
uid=
#creamos la cabecera del fichero html y las pimera lineas de la tabla
echo "<!DOCTYPE html>
      <html>
      <head>
       <title>Información de Usuarios</title>
      </head>
      <body>
       <h1>Información de Usuarios</h1>
       <table border="1">
        <tr>
         <th>Login</th>
         <th>Información General</th>
         <th>Directorio de Conexión</th>
         <th>Shell</th>
        </tr>
      " > $FILE
#aqui va el contenido de las tablas del html con un bucle
while read LINEA ; do  #hacemos el buble while read
      U=$(echo "$LINEA" | cut -d":" -f3)  #declaramos la variable U con el
contenido para filtra por el UID de los usuarios
      echo "$U"       #esto es una simple comprobacion parar comprobar
quefunciona
      if [ "$U" -ge 1000 ] && [ "$U" -le 2000  ]; then   #si el UID es mayor o igual
a 1000 y  menor o igual a 2000 entonces hará las columnas
            echo "          <tr>
                  <td>$(echo $LINEA | cut -d":" -f1)</td>
                  <td>$(echo $LINEA | cut -d":" -f5)</td>
                  <td>$(echo $LINEA | cut -d":" -f6)</td>
                  <td>$(echo $LINEA | cut -d":" -f7)</td>
               </tr>" >> $FILE   #se redirecciona el echo a el archivo html con
doble mayor para que no se carge lo anterior escrito
      fi  #se cierra el if

done < /etc/passwd   #se cierra el bucle while con la redicrección del read
donde leíamos las lineas antes



echo "  </table>
      </body>
      </html>
      " >> $FILE #cerramos el html y finaliza el script
```

2.- En una organización vamos a controlar el acceso a Internet de determinados equipos mediante la MAC. Un ejemplo de una dirección MAC es 30-65-EC-6F-C4-58, es decir, 6 grupos de 2 dígitos **hexadecimales** separados, en este caso, por un guion.

Antes de introducirlas debemos comprobar la validez de estas. En un fichero tenemos todos los nombres de los equipos a revisar, así como su MAC. Su estructura es esta:

Nombre_equipo;MAC

Ejemplo:

W10_01;30-65-EC-6F-C4-58

El fichero que contiene esta información se llama ***eq_mac.txt***.

Nuestro script debe almacenar en ***eq_mac_ok*** aquellas direcciones correctas y en ***eq_mac_ko*** aquellas direcciones incorrectas. Asimismo, se debe crear un fichero ***mac.log*** que contenga la fecha en el que se ejecuta el script y una pequeña estadística que diga el número total de direcciones procesadas, el número de direcciones correctas y el número de direcciones incorrectas.

Observación: asumimos que el nombre del equipo siempre es correcto y que la **mac** está presente, pero puede que no sea correcta.

Observación: el control de la MAC debe hacerse en una función.

Probar el script con estos datos

EQ1;30-65-EC-6F-C4-58
EQ2;30-65-EC-6F-C4-5H
EQ3;30-65-99-6F-C4-58
EQ4;30-65-EC-11-22-C8

Contenido de **eq_mac_ok**
EQ1; 30-65-EC-6F-C4-58
EQ3; 30-65-99-6F-C4-58
EQ4; 30-65-EC-11-22-C8

Contenido de **eq_mac_ko**
EQ2; 30-65-EC-6F-C4-5H

Contenido de ***mac.log***
Comprobación de MACs: 07/12/2023
Líneas procesadas: 4
Direcciones OK:3
Direcciones KO:1

```
#!/bin/bash
F="eq_mac.txt"
TOT=0
FA=0
CR=0
for linea in $(cat $F | cut -d";" -f2); do
      echo "Linea $linea"
      if [[ $linea =~ ^([0-9A-F]{2}-){5}[0-9A-F]{2}$ ]]; then
            echo "$linea">>eq_mac_ok
            ((CR++))
      else
            echo "$linea">>eq_mac_ko

            ((FA++))
      fi
      ((TOT++))

done
echo "Comprobación de logs $(date +%F)" >>mac.log
echo "      TOTAL MACS $TOT
       MACS incorrectas $FA
       MACS correctas $CR" >>mac.log
```

3.- En un sistema Linux, sabemos que, de momento, vamos a trabajar con los servicios samba, ssh y cups.

Crear un script que almacene los nombres de estos servicios para quien los quiera consultar. Este script debe permitir añadir nuevos servicios. Cuando se ejecute, debe mostrar el siguiente menú:

Servicios

1.- Consultar servicios

2.- Añadir servicio.

3.- Salir.

```
#!/bin/bash
servicios=(Samba SSH Cups)
echo "servicios"
echo "1.-Consultar Servicios"
echo "2.-Añadir Servicio"
echo "3.-Salir"
read -p "Introduce una opción: " O
if [ $O = 1 ]; then
      echo "Servicios actuales:"
      echo ${servicios[*]}
else
      if [ $O = 2 ]; then
            echo ${servicios[*]}

            read -p "Si quieres intoducir un dato escriba si:" R
            while [ $R = si -o $R = SI -o $R = Si -o $R = sI ];
do
```

```
                read -p "Intorduce el servicio nuevo:" SER
                T=${#servicios[*]}
                echo $T
                servicios[$T]=$SER
                echo ${servicios[*]}
                R=0
                read -p "Si quieres intoducir un dato escriba
si:" R
          done
          read -p "Quieres guardar los ficheros si/no:" G
          if [ $G = si ]; then
                for S in  ${servicios[*]}
                      echo "$S" >> ./bin/nombre.servicios
                done
                echo "${servicios[*]}" >
/usr/local/bin/nombre.servicios
          fi
     else

          if [ $O = 3 ]; then
                exit 4
          else
                echo "Itroduce un valor 1, 2 o 3"
                echo "Saliendo..."
                exit 5
          fi
     fi
fi
echo "Termina el script"
exit 6
```

4.- Mejorar el script anterior, permitiendo al usuario que ejecuta el script, guardar los cambios en un fichero (*nombre.servicios*).

El anterior ya está con esas modificaciones