PRACTICAS DE SISTEMA OPERATIVO UNIX

PRACTICAS DE UNIX

PRACTICA 5. El entorno del shell KORN (ksh).

Objetivos
Manejar variables del entorno, variables shell, alias, operaciones con variables de tipo cadena (sustitución de parámetros). Personalizar el entorno de trabajo. Primer contacto con los guiones (guion=shell script).

Herramientas
Las herramientas a utilizar serán:
env set export sh csh alias chmod ksh
read shift operador "." history let

DESARROLLO DE LA PRACTICA

1. Variables.
   1.1 Variables de entorno y variables shell:
      a. Defina indicando la diferencia.
      b. Como se puede convertir una variable shell en variable de entorno. Ponga un ejemplo.
         `export`
      c. Orden que permita visualizar las variables de entorno.
         `env`
      d. Visualice las variables shell.
         `set`

   1.2 Indique la utilidad de cada una de las siguientes variables del entorno y especifique si son reconocidas por el shell estándar (bourne):

a. PATH	b. CDPATH	c. HOME	d. LOGNAME	e. PS1
f. PS2	g. SHELL	h. RANDOM	i. EDITOR	j. TERM
k. IFS	l. SECONDS	m. OLDPWD	n. HISTFILE	ñ. HISTSIZE
o. ENV	p. PWD			

      a. La ruta de ejecutables.
      b. Ruta de búsqueda para cd.
      c. El directorio de presentación.
      d. El nombre de conexión.
      e. Símbolo del PROMPT principal.
      f. Símbolo del PROMPT secundario.
      g. El shell usado.
      h. Un número aleatorio.
      i. El editor por defecto (ed, vi, emacs...)
      j. Tipo de terminal.
      k. Separador por defecto de campos de entrada.
      l. Segundos desde el comienzo de la sesión.
      m. Directorio anterior.
      n. Fichero del histórico de comandos.
      Ñ. Tamaño del histórico.
      o. Ruta al archivo de entorno inicial.
      p. Directorio actual.

   1.3 Utilice variables de entorno para:
      a. visualizar su directorio de conexión. `echo $HOME`
      b. visualizar su directorio de trabajo (directorio actual). `echo $PWD`
      c. cambiar el prompt principal por 1>. `PS1="1>"`

   1.4 Ejecute la siguiente instrucción: `$ echo "hola"`
      a. ¿Qué muestra el sistema y que nos indica ese símbolo?
         `El prompt secundario.`
      b. Cambie ese prompt por 2>.
         `PS2="2>"`

c. Verifique el contenido de la variable modificada.
```
echo $PS2
```

1.5 Abandone su sesión de trabajo (salga del sistema). Inicie una nueva sesión.

a. Verifique el valor de las variables anteriormente modificadas.
```
echo "$PS1, $PS2"
```

b. ¿Qué ha sucedido?
```
Que están como al ppo.
```

c. ¿Cómo podría conseguir que los cambios realizados estuvieran al iniciar la nueva sesión de trabajo?
```
Almacenandolos en el .profile.
```

1.6 Realice las siguientes operaciones:

a. Genere dos variables llamadas nombre y apellido1 y asigne sus datos correspondientes a dichas variables.
```
NOMBRE=Andres
APELLIDO1=Purriños
```

b. Visualice el contenido de las dos variables.
```
echo "$NOMBRE, $APELLIDO1"
```

c. ¿Cuántas variables definidas tiene ahora en su shell y cuántas son del entorno?.
```
Se han creado dos más: nombre y apellido1.
```

1.7 Genere un sub-shell.
```
ksh
```

a. ¿Cuántas variables definidas tiene en el sub-shell y cuántas de entorno? ¿Porqué hay diferencia con los números del ejercicio anterior?
No tiene las que hemos creado.

b. Abandone el sub-shell.
```
exit
```

1.8 Repita el ejercicio anterior pero convirtiendo anteriormente las variables nombre y apellido1 en variables de entorno.
```
export NOMBRE
export APELLIDO1
```

1.9 En su shell actual, realice las siguientes operaciones:

a. Asigne el camino absoluto de su directorio de trabajo a una variable llamada dirtra.
```
DIRTRA=$HOME
```

b. Cambie al directorio /etc.
```
cd /etc
```

c. Regrese a su anterior directorio de trabajo utilizando la variable dirtra.
```
cd $DIRTRA
```

1.10 Indique el resultado de las siguientes acciones:

a. Almacene en una variable llamada grupo el nombre del grupo en el cual está inscrito. (Haga la asignación obteniendo el nombre de grupo con una instrucción Unix.)
```
MIGRUPO= `cat /etc/group | grep $LOGNAME | cut -f1 -
d:`
```

b. Cree otra variable llamada identidad cuyo contenido sea la línea del fichero "passwd" donde quedó registrado como usuario del sistema.
```
IDENTIDAD= `cat /etc/passwd | grep $LOGNAME `
```

c. Con una sola instrucción almacene en otra variable llamada total los siguientes datos separados cada uno por dos puntos (:):
- su identificativo y su número de grupo extrayéndolo de la variable identidad.
- el nombre del grupo utilizando la variable grupo.

```
TOTAL= `$IDENTIDAD:$MIGRUPO`
```

1.11 Una de las características del shell Korn es la capacidad para tratar variables como si fueran arrays unidimensionales.

a. Declare un array de tres elementos almacenando en cada uno su nombre y sus dos apellidos (cada dato en un elemento del array).

```
ARRAY[1]="Andres Javier"
ARRAY[2]=Purriños
ARRAY[3]=Blázquez
```

b. Visualice utilizando el array su nombre y segundo apellido.

```
for i in 1 2 ^j do ^j echo "$ARRAY[$i]\c"
```

1.12 Variables de conmutación:

a. ¿Qué son?
Las que toman valores lógicos true y false.

b. ¿Cuál es la utilidad de las variables de conmutación `noclobber` e `ignoreeof`?
`noclobber` a 1 impide que al redireccionar la salida se sobreescriban archivos, `ignoreeof` impide salir del shell con break.

c. Active dichas variables.

```
set noclobber
set ignoreeof
```

d. Desactive dichas variables.

```
unset noclobber
unset ignoreeof
```

1.13 Indique que valor almacenan las siguientes variables shell:

a. `$#` Número de parámetros en línea de comandos.
b. `$1` Primer parámetro de la línea de comandos.
c. `$0` Nombre del shell-script.
d. `$*` Todos los parametros.
e. `$$`
f. `$!`
g. `$?`

1.14 Ejecute la instrucción: `$ find $HOME -name "z1x1y1.aeiou"`

a. Visualice el valor de retorno de dicha instrucción.
Es 0.

b. ¿Por qué devuelve este valor?
Porque no se encuentra el fichero.

2. Sustitución de parámetros:

2.1 Sustitución de parámetros de forma general: `$ {<parámetro>:<caracter><valor>}`
Rellene el cuadro indicando qué sucede en los casos expuestos:

	Parámetro	
	Es nulo o no existe	Existe o no es nulo
`echo ${LOGNAME:=`logname`}`	Le da valor permanentemente.	Deja el valor existente.
`echo ${LOGNAME:+existe}`		
`echo ${LOGNAME:-existe}`	Le da valor temporalmente.	Deja el valor existente.
`echo ${LOGNAME:?no existe}`	Muestra el mensaje "no existe".	Muestra el valor.

2.2 ¿Qué diferencia existe entre utilizar $nom y ${nom} para hacer referencia al contenido de una variable?

2.3 Explique qué muestra la siguiente orden: `echo ${#PATH}`
La longitud de $PATH.

3. Alias.

3.1 Cree un alias llamado datos que pueda ser utilizado de la siguiente manera: `$datos<nombre_usuario>` y muestre la línea del archivo `/etc/passwd` correspondiente al usuario indicado. (ej. de llamada: `$ datos pepe`)
```
alias datos="grep $1 /etc/passwd"
```

3.2 Cree otro alias que tenga alguna utilidad e indique su utilidad.
```
alias cls="cat screen.cls"
```
Lleva el contenido de screen.cls (normalmente vacío, para realizar un borrado) a pantalla.

3.3 Visualice los alias definidos. `alias`

3.4 Elimine los alias definidos.
```
alias NOMBRE=
alias MIGRUPO=
...
```

4. Personalización del entorno.

4.1 ¿Qué dos ficheros ejecutará el shell inicial cuando un usuario comience una sesión de trabajo con el shell Korn?
```
.profile
.kshrc
```

4.2 Consiga que cuando se conecte se lleven a cabo las siguientes tareas:

a. Borrar la pantalla
```
echo clear >> .profile
```

b. Saludar indicando en letras grandes hola <nº UID>
```
echo "`banner Hola ` `id | cut -f1 -d" " | cut -f2 -
d"=" `" >> .profile
```

c. Situarnos en un directorio llamado trabajo que se encontrará colgado del directorio de conexión.
```
echo "cd $HOME/trabajo"
```

d. Establecer una máscara para la protección de los archivos.
```
umask 744
```

4.3 Ejecute el archivo correspondiente para que los cambios anteriores afecten al shell actual.
```
.profile
```

5. Primer contacto con los guiones.
   5.1 Realice los siguientes guiones:
      a. Cree un guión que muestre el calendario correspondiente al mes actual.
      ```
      cal | date | cut -f -f5,6
      ```

      b. Ejecute dicho guión de dos formas diferentes.
      ```
      ksh calen

      chmod 700 calen
      calen
      ```

   5.2 Cree un shell script que reciba como parámetro posicional el identificativo de un usuario (UID) y visualice el número de usuario correspondiente al identificativo así como su grupo (GID) y el nº de grupo.
   ```
   echo "`grep /etc/passwd $1 |cut -f3 -d":"`  \c"
   echo "grep /etc/group $1 | cut -f1,3 -d":""
   ```

   5.3 Almacene en un archivo llamado telef los teléfonos de algunos individuos. Dicho archivo tendrá la siguiente estructura:
   ```
   nombre:apellidos:telefono
   ```

   Almacene también en otro archivo llamado direc la dirección de los anteriores. Este último archivo tendrá como estructura:
   ```
   nombre:apellidos:dirección
   ```

   Realice un guión que solicite el nombre y apellidos de un individuo y visualice su número de teléfono y su dirección.
   ```
   echo "Nombre: \c"
   read nombre
   echo "Apellidos: \c"
   read apellidos
   echo "`grep telefono nombre:apellidos |cut -f3 -d":"`"
   echo "`grep direccion nombre:apellidos | cut -f3 -d":"`"
   ```

   5.4 Genere un guión que solicite un nombre y almacene en un archivo llamado resultado el nombre, apellidos, teléfono y dirección de todos los individuos cuyo nombre coincida con el tecleado, extrayendo dichos datos de los archivos del punto anterior. El archivo resultante tendrá la estructura:
   ```
   apellidos, nombre:telefono:dirección
   ```

   ```
   read nombre
   for apellido in (grep telef $nombre | cut -f2 -d:)
   do
       echo "$apellido, $nombre:`grep telef "$nombre:
       $apellido" -f3 -d:`:`grep direcc "$nombre:$apellido" -
       f3 -d:`" >> resultado
   done
   ```

   5.5 Realice las siguientes operaciones:
      a. Genere un guión llamado lfich que visualice un listado largo por pantalla solamente de los ficheros ordinarios de su directorio actual.
      ```
      for file in `find . -type f *`
      do
        ls -l $file
      done
      ```

      b. Ahora cree un shell script llamado ldir que liste solo los ficheros directorios que dependan directamente del directorio actual.
      ```
      for file in `find . -type d`
      ```

```
do
  ls -l $file
done
```

5.6 Cree un guión que reciba como parámetro posicional el nombre de un usuario y genere un fichero llamado sugrupo cuyo contenido sea el identificativo y el directorio de conexión de los usuarios que pertenezcan a dicho grupo.

```
grupo=`grep $1 /etc/passwd | cut -f4  -d:`
echo `grep /etc/group $grupo | cut -f4 -d:
```

5.7 Guión que reciba como parámetros posicionales varios nombres. Como salida visualizará los valores recibidos y el número de ellos.

```
echo $*
echo $#
```

5.8 Crear un shell script llamado cargar123 que almacene, con una instrucción en el interior del shell, en las variables $1, $2 y $3 las cadenas logname, pwd y cal 11 2000. Ejecute las instrucciones almacenadas en dichas variables haciendo uso de ellas.

5.9 Modifique el shell anterior para que se ejecuten las instrucciones utilizando únicamente la variable $1.

6 Histórico de comandos.

6.1 Visualice las últimas órdenes que ha tecleado en su terminal.

```
history
```

6.2 Visualice y después duplique el valor del número de órdenes que se pueden reejecutar en el histórico de comandos.

```
echo $HISTSIZE
HISTSIZE=36
```

6.3 Ejecute la antepenúltima orden tecleada de tres formas diferentes.

```
r -3
```

6.4 Muestre por pantalla el contenido de todos los ficheros con extensión txt de su directorio. Utilice el editor de línea de órdenes para aplicar el proceso anterior a los ficheros de extensión doc.

```
cat *.txt
***
```