PRACTICAS DE S.O. UNIX

PRACTICA 1. El editor de pantalla "vi".

Objetivos. El objetivo final será conseguir un cierto dominio en el manejo de las herramientas que nos ofrece este editor que nos servirá en prácticas posteriores, sobre todo en programación donde se utilizará con frecuencia.

Herramientas. En esta práctica utilizaremos los comandos en general que nos ofrece este potente editor.

DESARROLLO DE LA PRACTICA.

1.− Movimientos del cursor.

1.1 Genere un fichero llamado "fich−1".Introduzca 10 líneas de texto.

vi fich−1

:a

[10 lineas]

1.2 Utilice los comandos de movimiento del cursor hacia arriba, a la izquierda, a derecha y abajo.

<esc>, [n] k

<esc>, [n] h,

<esc>, [n] <space>

<esc>, [n] j

1.3 Realice los siguientes apartados, utilizando órdenes del editor.

a) Sitúese al final del texto.

<esc> L

b) Colóquese al principio del texto

<esc> H

c) Sitúese al final de la quinta línea

<esc> 5 $

d) Vuelva al principio de la misma línea.

<esc> 0

1.4 En la situación actual, muévase dos palabras hacia la derecha. Vuelva ahora a la posición anterior.

<esc> 2 W

<esc> 2 B

2.− Adición de texto.

2.1 Inserte una nueva línea delante de la primera.

<esc> H

<esc> :i

[linea]

2.2 Sitúese en la segunda palabra de la tercera línea e inserte una nueva palabra delante de ella.

<esc> 3 G

<esc> 2 W

<esc> :i

2.3 Añada texto al final de la línea actual.

<esc> $

<esc> :a

3.− Borrado de texto.

3.1 Borre los cinco primeros caracteres de la línea actual.

<esc> 0

<esc> 5 x

3.2 Elimine la línea actual y la siguiente utilizando una sola orden.

<esc> 2 S

3.3 Sitúese en la segunda palabra de la línea actual y borre el resto de la línea.

<esc> 0

<esc> 2 W

<esc> D

4.− Opciones de entorno.

4.1 Numere las líneas.

<esc> : set nu

4.2 Establezca el retorno de carro en la columna 65. Compruébelo.

<esc> : set wm=65

5.− Búsqueda sustitución y eliminación.

5.1 Sustituya todas las ocurrencias del articulo "el" por "la".

: g/el/s//la/g

5.2 Borre todas las líneas que contengan la palabra "como".

<esc> :g/como/d

6.− Salida temporal al Shell.

6.1 Ejecute un shell sin salir del editor, luego elimínelo y vuelva al fichero editado.

:sh

$ exit <CR>

6.2 Ejecute una orden del unix sin cancelar su trabajo de edición.

:! [orden]

7.− Copiar y mover líneas de información.

7.1 Copie las líneas desde la tercera a la quinta incluidas y situe la copia a partir de la

septima línea.

<esc> H

<esc> 3 G

<esc> 3 Y

<esc> H

<esc> 7 G

<esc> P

7.2 Mueva las dos últimas líneas al principio del texto.

<esc> L

<esc> k

<esc> 2 Y

<esc> H

<esc> P

7.3 Añada un fichero de texto que ya exista, al final del fichero editado.

<esc> L

<esc> $

<esc> :r [fichero]

7.4 Añada a su fichero de pruebas el listado de las personas que actualmente estén

conectadas.

<esc> :r !who

PRACTICAS DE S.O. UNIX

PRACTICA 2. Introducción al UNIX.

Objetivos. Conocer de una forma práctica las herramientas de entrada y salida del sistema, la forma de obtener información acerca del sistema y órdenes relacionadas con la comunicación con los demás usuarios.

Herramientas. Los comandos a utilizar en esta práctica son:

logname uname id passwd who

write mesg exit news date

mail cal calendar finger lock

DESARROLLO DE LA PRACTICA.

1.− Entrada en el sistema y consulta de información básica del sistema y de los usuarios.

1.1. Indique los pasos para iniciar una sesión de trabajo UX, utilizando su identificativo (logname) y su

palabra de paso (password).

{Conectamos al servidor}

login: <nuestro login>

password: <nuestra password>

1.2 Visualización de su identificativo:

a) Visualizar el nombre de dos formas diferentes.

who am I

logname

b) Visualizar el número correspondiente a su identificativo.

id

1.3 Visualice la siguiente información sobre el sistema.

a) nombre

uname

b) versión del sistema operativo

uname −v

c) hardware que lo soporta.

uname −a

1.4 Modifique se palabra de paso. Verifique la acción anterior haciendo log−off (desconéctese) y vuelva a hacer log−in (conéctese).

passwd

<teclear password>

.0

1.5 Fechas:

a) Visualice la fecha y hora actual.

date

b) Obtenga la siguiente salida (tal cual está, en dos líneas):

Son las hh:mm:ss del día: dd

del mes: mm del año: aa.

date +Son las %T del día %e %n del mes %m del año %y.

c) Visualice el calendario completo del año.

cal 1996

d) Consulte el día de la semana de su cumpleaños en el año 2000.

cal 11 2000

1.6 Genere una agenda en la que se reflejen las actividades que debe recordar en los próximos días.Visualice las actividades de hoy y mañana.

1.7 Visualice todos los datos referentes al usuario de muestra.

finger <login>

1.8 Bloquear el terminal de forma temporal.

lock

2.− Comunicación con otros usuarios.

2.1 ¿Qué correo tiene pendiente? Orden que permita visualizarlo.

mail −h

2.2 Indique como conseguir con el comando 'mail':

a) Lectura del mensaje anterior al actual.

−

b) Lectura del mensaje posterior al actual.

+

b) Lectura de los mensajes en orden inverso.

c) Añadir los mensajes a un fichero.

s <fichero>

2.3 ¿Qué usuarios están conectados al sistema en este momento?.

who

2.4 Enviese correo en formato off−line a si mismo. También a un usuario de otro grupo.

mail f941162

mail f930158 [<mensajes>]

2.5 Envíe correo a un usuario no reconocido por el sistema.

a) ¿Qué ha ocurrido?.

Da un mensaje de error y graba el mensaje en un fichero.

b) ¿Dónde queda la información?.

En el fichero dead.lettter

c) Visualícela.

cat dead.letter | more

2.6 Capacitación/descapacitación de recepción de mensajes on−line:

a) Impida que otro usuario le envíe mensajes (on−line).

mesg −n

b) Visualice un listado en el que se recoja información que nos permita saber que usuarios tienen capacitado el terminal para recibir mensajes.

who −t

c) Capacite de nuevo la comunicación (on−line).

mesg −y

2.7 Visualice las noticias que haya dejado el administrador.

news

PRACTICAS DE S.O. UNIX

PRACTICA 3. Ordenes básicas.

Objetivos. Manejo de órdenes básicas para mostrar información por la pantalla y gestionar

directorios y ficheros.

Herramientas: Los nuevos comandos a utilizar en esta práctica son:

echo banner op. grave pwd dircmp

wc dirname basename cat pg

pr more tail ls mv

head mkdir cd rmdir stty

rm cp tr ln

DESARROLLO DE LA PRACTICA

1.− Manejo de la pantalla.

1.1.− Recordando que el comando "echo", acepta los caracteres de control de

escritura, imprima la siguiente salida.

"Como esta frase es bastante larga

vamos a utilizar dos líneas."

echo "Como... larga \n vamos... lineas."

1.2.− Dadas las siguientes órdenes escritas en un shell script:

echo "Texto de la primera línea"

echo "y de la segunda."

Inserte un caracter de control en la primera instrucción para que el resultado de

las dos instrucciones se visualice en una única línea.

echo "Texto...\c"

echo ...

1.3..− Modifíquense las teclas con significados especiales (Retroceso, Interrupción, etc.)

cambiándolas por otras personalizadas.

setkey 1 ´cat .profile | more´

2. Ejecución de órdenes como parte de otra orden.

2.1.− Utilizando el comando "echo" y sin utilizar variables, visualice las siguientes salidas:

a) "Mi grupo es <nombre> que se corresponde con el número <nº>"

echo "Mi grupo es\c";id −g −n;echo "que se corresponde con el numero\c";id −g

b) "Mi directorio actual es: <directorio> ".

echo "Mi directorio actual es: \c";pwd

c) "Listado de las personas actualmente conectadas:

< listado >"

Echo "Listado de las personas conectadas: "; finger

d) "Hay <nº> usuarios conectados en este momento."

echo "Hay `who | grep #users= | cut −f2 −d=` usuarios conectados actualmente.";

2.2.− Visualice en letras grandes:

"Mi identificativo es <id>"

bannner Mi id es `logname`

2.3.− Muestre por pantalla la siguiente información:

a) el nombre del directorio actual (sólo el nombre, sin la ruta de acceso)

basename $PWD

b) la ruta de acceso al directorio actual (sólo la ruta, sin el directorio actual)

dirname $PWD

3. Gestión de directorios.

3.1.− ¿Qué se entiende por camino absoluto y por camino relativo?

3.2.− Cuando iniciamos una sesión de trabajo, el sistema nos sitúa en un directorio llamado de conexión.

a) ¿En qué archivo se establece cuál es el directorio de conexión en el que debe situar el sistema al usuario de forma inicial?

/etc/passwd

b) ¿Qué variable almacena dicho directorio?

$HOME

3.3.− Obtenga un listado del contenido de su directorio de conexión en el cual solo aparezcan los nombres de los archivos y/o directorios.

ls $HOME

3.4.− Cree la siguiente estructura de directorios teniendo en cuenta que los identificadores que comienzan por "dir" son directorios y el resto son archivos ordinarios:

ÚÄÄÄÄÄÄÄ¿

³ $HOME ³

ÀÄÄÄÂÄÄÄÙ

ÚÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÅÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿

ÚÄÄÄÁÄÄÄÄÄÄ¿ ÚÄÄÄÁÄÄÄÄÄ¿ ÚÄÄÄÁÄÄÄÄÄ¿

³ dirdatos ³ ³ diruno ³ ³ dirdos ³

ÀÄÄÄÄÄÄÄÄÄÄÄÙ ÀÄÄÄÂÄÄÄÄÄÙ ÀÄÄÄÄÄÄÄÄÄÙ

ÚÄÄÄÄÄÄÄÄÄÄÄÂÄÄÄÁÄÄÄÄÄÄÄÄÄÄÄ¿

ÚÄÁÄÄ¿ ÚÄÄÄÄÄÁÄÄÄÄÄ¿ ÚÄÄÄÄÄÄÁÄÄÄÄÄ¿

³ f1 ³ ³ dirtres ³ ³ dircuatro ³

ÀÄÄÄÄÙ ÀÄÄÄÄÂÄÄÄÄÙ ÀÄÄÄÄÄÄÂÄÄÄÄÙ

ÚÄÄÄÄÁÄÄÄÄ¿ ³

ÚÄÄÁÄ¿ ÚÄÄÁÄ¿ ÚÄÁÄÄ¿

³ f2 ³ ³ f3 ³ ³ f4 ³

ÀÄÄÄÄÙ ÀÄÄÄÄÙ ÀÄÄÄÄÙ

mkdir dirdatos

mkdir diruno

mkdir dirdos

cd diruno

mkdir dirtres

mkdir dircuatro

echo > f1

cd dirtres

echo > f2

echo > f3

cd ../dircuatro

echo f4

3.5.− Desde el directorio HOME, determine la estructura completa del árbol con raíz en el subdirectorio diruno.

ls −R diruno

3.6.− Desde el directorio dircuatro obtenga un listado del directorio datos utilizando tanto camino absoluto como relativo.

ls ../../dirdatos

ls $HOME/dirdatos

3.7.− Directorios:

a) Supuesto que nos encontramos en el directorio de conexión, cambie su situación al directorio "dirtres".

cd diruno/dirtres

b) Indique dos métodos (dos órdenes) diferentes para regresar al directorio de conexión.

cd $HOME

cd ../..

3.8.− Liste de su directorio de conexión todos los nombres de archivos y/o directorios que comiencen por "d".

ls $HOME/d*

3.9.− Realice un listado largo de todos los ficheros de su directorio de conexión e identifique:

− tamaño

− cuántos enlaces (links) tiene cada fichero

− qué usuarios pueden acceder al fichero ".profile", razone la respuesta

− como se puede saber el tipo de archivo de cada archivo del listado.

ls −l

Está despues del nombre.

Está antes del nombre.

Por defecto, .profile solo es accesible por el propietario (y el administrador, claro está).

La 1ª letra, antes de los atributos, indica el tipo de fichero.

3.10.− Agrege lo necesario a la orden anterior para conseguir que en lugar de que aparezcan varios espacios en blanco consecutivos en el listado, solo ararezca uno.

ls −l | tr − s

3.11.− Compare los contenidos de los directorios dirtres y dircuatro.

dircmp dirtres dircuatro

4.− Gestión de ficheros.

4.1.− Visualíce los ficheros "f1" y "f2" con una única instrucción "cat".

cat f1 f2

4.2.− Visualícelos de nuevo:

a) deteniendo la pantalla cada 20 líneas, con tres órdenes diferentes

cat f1 f2 | more −20

pg −20 f1 f2

more −20 f1 f2

b) a partir de la tercera línea

pg +3 f1 f2

c) sólo las 10 primeras líneas.

head f1 f2

d) sólo las 10 últimas líneas.

tail f1 f2

4.3.− Sitúese en el directorio "dircuatro".

a) Copie todos los archivos del directorio "/bin" que comiencen por "c" en el directorio actual.

cp /bin/c*

b) ¿Quién es el propietario de los archivos "/bin/c*" y quién es el propietario de los archivos "./c*"?

De los /bin/c* es bin, y de los ./c* el usuario.

4.4.− Borre los ficheros del directorio actual cuyo nombre comience por "ch".

rm ch*

4.5.− Enlaces.

a) Cree un enlace al fichero "datos" llamándole "infor".

ln datos infor

b) Indique una orden que permita ver el número de enlaces que tiene el fichero "datos".

ls −l datos

c) ¿Qué sucede si se borra el archivo "datos"?

Se elimina un link, pero no se borra el fichero mientras algún otro link siga exisitiendo.

4.6.− Copie el contenido completo de "diruno" bajo "dirdos".

cp $HOME/diruno/* $HOME/dirdos/*

4.7.− Visualice con una orden el número de líneas que tienen los archivos "f2" y "f3" de

manera que se aprecie que número corresponde a que fichero.

wc −l f2 f3

4.8.− Visualice un número (solo un número) que será la suma de las líneas de los archivos

"f1" y "f3".

echo `wc −l | f1 f3 | grep total ` | cut −f2 −d

PRACTICAS DE S.O. UNIX

PRACTICA 4. Comodines y caracteres de sustitución. Gestión de ficheros y selección de información.

Objetivos. En esta práctica se realizarán ejercicios de redirección, tuberías, filtros, caracteres de sustitución, selección de información y salidas divididas. Se pretende familiarizarse con el uso de caracteres especiales de redirección de información así como conseguir soltura en la selección de información de un fichero o entrada de información, teniendo en cuenta algún campo o información a partir de alguna columna.

Herramientas. Los herramientas a utilizar serán :

tee cut find grep egrep fgrep uniq cmp diff comm sort

DESARROLLO DE LA PRACTICA

1.− Redirección y canalización de información.

1.1.− Introduzca nuevas líneas al final de un fichero de texto que ya tenga, conservando el texto original.

cat >> fichero

1.2.− Utilice la siguente sentencia " $ cat f1 f2 ". EL fichero f1 debe existir y el f2 no debe existir. Observe la salida por el monitor.

a) Consiga que la salida de errores vaya a un fichero llamado "errores".

cat f1 f2 2> errores

b) Haga ahora lo necesario para que el mensaje de error se desprecie.

cat f1 f2 2> /dev/nul

1.3.− Visualice por pantalla un listado de los usuarios, que estén conectados en este momento al sistema y el número de ellos. Así mismo se debe almacenar el listado obtenido en un archivo llamado "presentes".

who | grep −n tty > presentes

1.4.− Obtener un listado por pantalla de los usuarios conectados actualmente al sistema y su hora de conexión.

who

1.5.− Visualice y almacene en un archivo llamado "TTYS" el nombre de los usuarios conectados, así como el tty asociado a cada usuario.

who | tee TTYS

2.− Caracteres de sustitución y selección de información.

2.1.− Liste del directorio "/usr/bin":

a) aquellos ficheros cuyo nombre empiece por "c"

find /usr/bin −name c* −print

b) los ficheros que comiencen por alguna letra comprendida entre la "a" y la "p"

find /usr/bin −name [a−p]* −print

c) todos aquellos ficheros cuya letra inicial del nombre no esté comprendida entre la

"d" y la "p"

find /usr/bin ! −name [d−p]*−print

d) todos aquellos ficheros cuyo nombre tenga cuatro letras.

find /usr/bin −name ???? −print

2.2.− Consiga un listado en el que aparezca únicamente el nombre y el tamaño de todos sus ficheros a partir de su directorio de conexión.

2.3.− Cree un fichero que esté formado por el identificativo (UID), el nº de usuario y el nº de grupo de los usuarios dados de alta.

cut −d −f1,3,4 /etc/passwd > users

2.4.− Obtenga un listado en el que aparezcan los nombres de los grupos de usuarios dados de alta en el sistema, así como los usuarios que formen parte de cada grupo.

cut −d: −1,4 /etc/group

2.5.− Cree un fichero llamado "agenda1" cuyo contenido sea nombre, apellidos, prefijo y teléfono de una serie de personas (como separador de campos se utilizará el signo dos puntos ":").

a) Liste los datos de las personas que vivan en Madrid (prefijo 91).

grep 91 agenda1

b) Liste los datos de aquellos cuyo nombre empiece por "A" y su teléfono acabe por "00".

grep ^A agenda1 | grep 00$

c) Liste el número de personas que no sean de Madrid.

grep −v Madrid agenda1

d) Liste los datos de todas las personas de "agenda1", menos aquellos que se llamen "juan".

grep −v juan

2.6.− Suponiendo que los números de los usuarios dados de alta van desde el 100 en

adelante, visualice el nombre de las cuentas dadas de alta que no pertenezcan a usuarios.

grep 0[0−9][0−9$]

3.− Búsqueda de ficheros.

3.1.− Liste todos los nombres de ficheros y directorios a partir de su directorio de conexión:

a) utilizando el comando "find"

find $HOME −print

b) utilizando "ls".

ls −R $HOME

3.2.− Visualizar el contenido de todos los ficheros que se encuentren a partir de su directorio de conexión y que empiecen por "fich". Repita lo anterior pero pidiendo confirmación antes de visualizar.

find $HOME −name fich* −exec cat {} \;

find $HOME −name fich* −ok cat {} \;

3.3 Liste la ubicación y nombre de todos aquellos ficheros ordinarios del sistema de ficheros cuyo nombre empiece por "p" y ocupen más de un bloque.

find / −name p* −size +1 −type f −print

3.4 Liste por pantalla el nombre los ficheros ordinarios que cuelguen de su directorio de conexión.

find $HOME −type f −print

3.5 Visualizar un listado en dos columnas con el nombre y número de nodo−i de todos los directorios de su propiedad.

find / −type d −user f941162 −exec ls −di {} \;

3.6 Envíe a su propio correo un listado largo de todos los archivos de su propiedad que tengan únicamente permisos de ejecución.

find / −user f941162 −perm 111 | ls −li | mail f941162

3.7 Orden que permita borrar, solicitando confirmación, todos los archivos ordinarios de su propiedad que tenga en la estructura de directorios.

find / −user f941162 −type f −ok rm {} \;

3.8 Orden que busque a partir de cada uno de los caminos incluidos en su PATH, los ficheros que incluyan en su nombre la cadena "core".

for a in $PATH ^j do ^j find $a −nanme *core* −print ^j done

4.− Manipulación de ficheros.

4.1 Liste por pantalla el número de personas que almacena el fichero "agenda1" sin repeticiones.

sort −u agenda1

4.2 Liste en orden alfabético de apellidos el fichero "agenda1".

soet +1 −d agenda1

4.3 Cree otro fichero con la estructura de "agenda1" y llámelo "agenda2". Clasifique en orden inverso los ficheros "agenda1" y "agenda2", colocando la salida en el fichero "ageninv" y utilizando el segundo campo como clave de clasificación.

sort 1 +1 −r agenda1 agenda2 > ageninv

4.4 Visualice las líneas del fichero "agenda2" ordenadas, suprimiendo todas menos la primera ocurrencia de las líneas que tengan el mismo apellido.

sort −d agenda2 | uniq −d −n0

4.5 Visualice el fichero de palabras de paso, ordenado por identificativo de usuario (recuerde que se trata del tercer campo y que están separados por ":").

sort −d −t:/etc/passwd

4.6 Partiendo de "agenda1" y "agenda2", obtenga un fichero llamado "telefonos" cuyo contenido sea la fusión de los dos anteriores sin repetir elementos y clasificado en primer orden por prefijo y en segundo por nombre.

sort +2 +0 −u agenda1 agenda2 > telefonos

4.7 Genere un par de ficheros de texto "fich1" y "fich2" que tengan semejanzas. Utilice el comando "cmp" para ver cual es la primera diferencia entre ambos ficheros y en qué posición se ha producido.

cmp f1 f2

4.8 Utilizando los mismos ficheros del ejemplo anterior, muestre por pantalla todas las diferencias que existan entre ambos.

cmp −l f1 f2

4.9 Haga un listado:

a) de todas las líneas del primero que no estén en el segundo y todas las líneas comunes

comm −2 f1 f2

b) todas las líneas del segundo que no estén en el primero.

comm −l3 f1 f2

PRACTICAS DE SS.OO. 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 conexion.

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 linea de comandos.

b) $1

Primer parámetro de la linea 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:`

7 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

***

PRACTICAS DE SS.OO. UNIX

PRACTICA 6. Órdenes de programación shell.

Objetivos. Tener un primer contacto con las herramientas de programación shell en cuanto al control de flujo de ejecución en los guiones shell.

Herramientas. Las herramientas a utilizar serán:

if test case while until for

return true/false function break/continue

DESARROLLO DE LA PRACTICA

1.− Sentencia IF.

1.1.− Codifique un shell script llamado "p6_1−1" que reciba un parámetro posicional, y realice la siguiente gestión:

a) Verifique que se le ha pasado un argumento como parámetro posicional. Si no es correcto se mostrará el mensaje:

"La llamada debe ser p6_1−1 argumento"

b) Si el argumento no es un fichero ordinario debe mostrar el siguiente mensaje:

"<parámetro> no es un fichero ordinario."

Y si existe un fichero ordinario en el directorio actual que tenga ese nombre, se indicará si es o no ejecutable, legible, si se puede escribir en él o no y si su tamaño es cero o mayor de cero.

c) Si existe en el directorio de trabajo un subdirectorio con el mismo nombre indicado en el parámetro, se visualizará un listado con las entradas que existan en el subdirectorio, y si no es un directorio mostrará el mensaje:

"<parámetro> no es un directorio."

```
if [ $# −lt 1 ]
then
echo "Introduce un parametro, por favor."
else
var=`find . −type f −name $1 −print`
if [ −z "$var" ]
then
echo "$1 no es un fichero ordinario"
else
var=`find . −type d −name $1 −print`
if [ ! −z "$var" ]
then
echo "$1 ees un directorio"
ll $1
else
echo "$1 no es un directorio"
fi
fi
fi
```

1.2.− Codifique un shell script que visualice el texto:

"Se encontró algún archivo llamado prueba."

si existe al menos un fichero ordinario llamado "prueba" en el sistema de ficheros.

```
if find / −depth −name prueba −print
then
echo Se encontró algún archivo llamado prueba.
fi
```

1.3.− Genere un shell script llamado "y" que simule el comportamiento del metacaracter "&&", es decir:

El shell "y" debe recibir dos parámetros posicionales que serán dos instrucciones UNIX. (Si en las instrucciones indicadas como parámetros existen espacios en blanco no olvidar dar el parámetro entre apóstrofos o dobles comillas.)

Si la primera instrucción se ejecuta de forma satisfactoria (consultaremos la variable $?) se ejecutará a continuación la segunda instrucción. En caso contrario no se procesa el segundo parámetro.

El guión devolverá un valor cierto (0) si se ejecutan correctamente las dos instrucciones indicadas como parámetros. Cuando no se ejecute de forma correcta alguna de las dos instrucciones devolverá un valor falso (1).

```
if [ $# −lt 2 ]
then
echo y necesita dos parámetros para su ejecución.
else
valor=1
$1
if [ $? −eq 0 ] then
$2
if [ $? −eq 0 ] then
valor=0
fi
fi
fi
echo $valor
exit $valor
```

1.4.− Ahora cree otro shell script llamado "o" que simule el comportamiento del metacaracter "||".

El guión devolverá un valor cierto (0) si se ejecuta correctamente al menos una de las dos

instrucciones indicadas como parámetros. En otro caso devolverá falso (1).

```
if [ $# −lt 2 ]

then

echo o necesita dos parámetros para su ejecución.

else

valor=1

$1

if [ $? −eq 0 ] then

valor=0

fi

$2

if [ $? −eq 0 ] then

valor=0

fi

fi

echo $valor

exit $valor
```

1.5.− Realizar un shell script que copie el fichero indicado como primer parámetro posicional de manera que la copia tenga el nombre indicado en el segundo parámetro posicional. Hay que controlar:

a) Que se indiquen dos parámetros.

b) Que exista y sea archivo ordinario el primer parámetro.

c) Que no exista un identificador (fichero ordinario, directorio, etc.) con el mismo nombre que el indicado en el segundo parámetro.

Si se produce alguna de las situaciones anteriores se visualizará un mensaje de error indicativo de tal error.

```
if [[ ! $# −eq 2 ]]
```

```
then
echo Debes indicar dos parámetros.
else
if [[ −f $1 ]]
then
if [[ ! −a $2 ]]
then
cp $1 $2
else
echo Ya existe un id. llamado $2.
fi
else
echo $1 no es fichero ordinario.
fi
```

2.− Sentencias CASE.

2.1.− Utilizando una sentencia "case", construya un shell script que, tomando como valores los ficheros de su directorio de conexión, organice una estructura de subdirectorios, de la forma:

Ficheros con extensión ".c" se mueven al directorio "prog_c".

Ficheros con extensión ".f" se mueven al directorio "prog_for".

Ficheros con extensión ".p" se mueven al directorio "prog_pas".

El resto de ficheros no se mueven.

Los directorios indicados únicamente se crearán si existen archivos para ser movidos a tales directorios.

```
for archivo in $HOME/*
do
```

```
case of $archivo
*.c) if [ ! test −d $HOME/prog_c ]
then
mkdir $HOME prog_c
fi
mv $HOME/$archivo $HOME/prog_pas/$archivo
*.f) if [ ! test −d $HOME/prog_for ]
then
mkdir $HOME prog_for
fi
mv $HOME/$archivo $HOME/prog_c/$archivo
*.p) if [ ! test −d $HOME/prog_pas]
then
mkdir $HOME prog_pas
fi
mv $HOME/$archivo $HOME/prog_pas/$archivo
esac
done
```

2.2.− Generar un shell script que muestre el siguiente menú de opciones y ejecute los tratamientos correspondientes:

1. Mostrar un listado con los usuarios de su grupo.

2. Mostrar el nombre de los usuarios de su mismo grupo junto al directorio de conexión que tienen asociado.

3. Hacer un listado con el nombre de los grupos dados de alta y el número de usuarios que tiene cada grupo.

4. Salir.

Se abandonará el shell script cuando se seleccione la opción 4.

```
function menu
{
echo Usuarios de FT−22
echo =============
echo 1.− Lista users
echo 2.− Dirs. $HOME
echo 3.− Lista Grupos
echo 4.− Salir al shell\n\n
echo Opción: \c
read opcion
case opcion in
1) listado;;
2) directorios;;
3) grupos;;
}
function listado
{
echo  Listado usuarios del FT22
echo ====================
for i in ´grep /etc/group ft22$ | cut −f4 −d: | cut −d,´
do
echo $i\n
done
}
function directorios
```

```
{
echo  Listado usuarios del FT22 con directorios de conexión
echo ====================================
for i in ´grep /etc/group ft22$ | cut −f4 −d: | cut −d,´
do
dir=echo `grep /etc/passwd ft22$ | cut −f6 −d:
echo $i\t\t$dir\n
done
}
function grupos
{
echo  Listado grupos con # de usuarios
echo ========================
for i in ´cat /etc/group´
do
echo $i | cut −f1 −d:\c
echo $i | wc | −f4 −d:\n
done
}
opcion=
until [$opcion=4]
do
menu
done
```

4.− Sentencias WHILE y UNTIL.

4.1.− Construya un shell script que lea un valor desde teclado, hasta que se obtenga la

cadena "fin".

until [$cadena=fin]

do

read cadena

done

4.2.− Genere un shell script que admita varios, uno o ningún argumento.

Si se indica algún argumento, este será el nombre de algún fichero. Mientras exista un argumento se mostrará el siguiente menú de opciones para realizar el tratamiento correspondiente con el argumento:

1. Comprobar si el fichero existe en su directorio actual.

2. Mostrar su nombre y tamaño.

En caso de no existir ningún argumento, visualizará el mensaje:

"no existen más argumentos".

function menu

{

clear

echo Fichero $1

echo 1. Comprobar existencia del fichero.

echo 2. Mostrar nombre y tamaño.\n\n

read opcion

case opcion in

1) buscar

2) mostrar

}

function buscar

{

```
if [ −f $1 ]
then
echo Existe $1.
else
echo No existe $1.
fi
}
function mostrar
{
ls −l $1 | tr −s  | cut −f2,5 −d
}
while [ ! −z $# ]
do
menu
shift
done
echo No existen más argumentos.
```

4.3.− Escriba un shell script que procese los números enteros del 1 al 20 y liste en tres columnas: el número, su cuadrado y su cubo.

```
clear
num=1
while [$num −le 20]
do
((cua=$num*$num))
((cub=$cua*$num))
echo $num\t$cua\t$cub
```

done

4.4.− Cree un shell script que borre todos aquellos ficheros que se hayan introducido como

argumentos. Se borrarán todos los ficheros de su propiedad que tengan el nombre especificado. Si los argumentos son directorios, se visualizará su nombre junto con el mensaje "es un directorio".

```
for i in $*
do
if [ −d $i ]
then
echo $i es un directorio.
else
rm $i
fi
done
```

4.5.− Escriba un shell script que lea números hasta encontrar un cero y escriba si el número

leido es primo o no.

5.− Sentencia FOR.

5.1.− Utilice un bucle "for" para eliminar de su directorio actual todos los ficheros que comiencen por "a". Adicionalmente, lleve el nombre de cada fichero borrado a un fichero llamado "borrados".

```
for i in ´ls a*´
do
rm $i
echo $1 >> borrados
done
```

5.2.− Busque en el file system completo todos los ficheros que usted especifique como

argumentos de su actual shell script y visualice por cada argumento un listado de la

siguiente manera:

El archivo ordinario <argumento> se encuentra en los directorios:

<directorio1>

<directorio2>

.....

donde <directorio?> será la ruta de acceso absoluta a dicho archivo (sin el nombre del archivo).

```
for nombre in $*
do
echo El archivo ordinario $nombre se encuentra en los directorios:
for dir in `find / −depth −type f $nombre`
do
echo \t`dirname $dir`
done
done
```

5.3.− Utilizando un bucle "for", obtenga los cuadrados de todos los números que especifique como parámetros del guión.

```
for i in $*
do
let a= $i * $i
echo $a
done
```

5.4.− Utilizando un bucle "for", distinga cuales de las anotaciones de su directorio actual, corresponden con un directorio y visualice su nombre.

```
for i in ls −1 *
do
if [ −d $i ]
then
```

```
echo "$i es un dir"

fi

done
```

5.5.− Genere un shell script que muestre por pantalla los nombres y números de líneas que

contiene cada fichero ordinario de su directorio de trabajo de la forma:

nombre: <NOMBRE>

contiene: <NUMERO> líneas

```
for i in ls −1 *

do

if [ −f $i ]

then

echo "nombre: $i"

echo "contiene: `cat $i | wc −l` lineas

fi

done
```

5.6.− Genere un shell script que permita borrar aquellos ficheros cuyos nombres especifique

como argumentos. Verifique anteriormente que existen y que son ordinarios. (Como argumento se especificará el directorio junto con el nombre del fichero)

```
if [ $# −lt 1 ]

then

echo "Introduce un parametro, por favor."

else

if [ !−f $1 ]

then

echo "$1 no es un fichero ordinario"

else

rm $1
```

fi

fi

5.7.− Genere un shell script que permita enviar un mensaje con "mail" a los usuarios que estén conectados al sistema en este momento, saludándolos.

for cuenta in `who | tr −s  | cut −f1 −d `

do

echo ¡Hola! | mail $cuenta

done

PRÁCTICAS DE SS.OO. UNIX

PRACTICA 7. Procesos.

Objetivos. En esta práctica se realizarán ejercicios de control de procesos en las modalidades foreground y background, así como tratamiento de prioridades, evaluación de tiempos y planificación de trabajos.

Herramientas. Las herramientas a utilizar serán:

ps jobs bg fg kill

nice sleep time at nohup

wait stty stop

DESARROLLO DE LA PRACTICA

1.− Procesos foreground.

1.1.− Averigüe cual es la actividad actual del sistema. Para ello visualice un listado completo

del estado de todos los procesos que se están ejecutando en el sistema.

ps −e

1.2.− Obtener un listado con los siguientes datos de los procesos de su shell actual.

PID PPID PRIORIDAD NICE COMANDO

... ...

... ...

El listado debe salir ordenado por pid.

ps −l | tr −s" " | cut −f4,5,7,8,14 −d" " | sort

1.3.− Cree un alias llamado "prio" que informe del PID, PPID, prioridad, valor nice, TTY y

comando asociados a todos los procesos del sistema cuyo propietario sea usted.

alias prio=pl −l | tr −s" " | cut −f4,5,7,8,12,14 −d" "

1.4.− Indique la/s diferencia/s en el resultado de las siguientes instrucciones. Explique el por

qué.

$ (who && who | wc −l) | tee quien

$ who && who | wc −l | tee quien

La primera manda al archivo 'quien' y a pantalla la lista de usuarios conectados y el n£mero de ellos, la segunda manda a pantalla todo, pero solo el n£mero de usuarios conectados al fichero

La causa, es que el entubamiento hacia tee del segundo caso, solo afecta al segundo operando del &&.

1.5.− Indique las diferencias en la ejecución de las siguientes instrucciones:

a) $ (cd .. ; pwd)

b) $ { cd .. ; pwd }

(después de las llaves hay un espacio en blanco.)

c) $ {

cd ..

pwd

}

b y c se ejecutan en un subshell.

1.6.− La orden "kill $$":

a) ¿Cuál es su objetivo?

Eliminar el shell actual.

b) ¿Lo consigue?

No.

c) Modifique la instrucción para conseguir dicho fin.

kill −9 $$

1.7.− Con una única instrucción, obtenga el proceso o procesos que tenga(n) mayor valor de

nice.

ps −l | sort −t +8 | head −1

2.− Prioridad subordinada. Procesos background.

2.1.− Cree un shell script "shell_71" cuyo contenido sean las dos instrucciones siguientes:

sleep 100

find / −name "passwd" −exec grep `logname` {} \; >> resultado 2> /dev/null

Ejecútelo como proceso subordinado (background).

shell_71&

2.2.− Ejecute el guión otras dos veces más modificando la prioridad en cada ejecución, primero disminuyéndola en 5 unidades y después en 10. Verifique los diferentes valores nice asignados, para lo cual ejecute la orden "ps −l".

nice −5 shell_71&

nice −10 shell_71&

ps −l

2.3.− Interrumpa el proceso correspondiente al segundo shell lanzado. Utilice inmediatamente después, el comando "ps" para ver el grado de éxito obtenido.

kill −9 <PID>

2.4.− ¿Qué diferencias encuentra al ejecutar las siguientes cuatro instrucciones?

a) (sleep 50 ; sleep 55)&

b) sleep 60 ; sleep 65 &

c) sleep 70& sleep 75&

d) sleep 80&;sleep 85&

a, c y d mandan los sleep en background. El primer sleep de b se ejecuta en foreground.

2.5.− Ejecute el fichero "shell_71" tres veces cada una en background pero utilizando una sola línea.

shell_71&;shell_71&;shell_71&;

2.6.− Ejecute el comando:

find / −name ".profile" − user `logname` −exec cat {} \;

calculando el tiempo que tarda en ejecutarse.

Ahora haga lo mismo en background dejando el resultado en un fichero llamado "miprofile", evitando posibles mensajes de error.

2.7.− Cree un guión "shell_72" que almacene en una variable "ord" el número de archivos ordinarios que tenga en la estructura de directorios y en otra variable "dir" el número de directorios. Las instrucciones para conseguir lo anterior se deben lanzar en background.

A continuaciòn se visualizarán los mensajes:

Número de ficheros ordinarios <ord>

Número de directorios <dir>

Total ... <xxx>

Asegúrese de que no se visualizarán los mensajes hasta que no se hayan terminado de ejecutar las instrucciones subordinadas.

```
ord={ find / −type f −depth }& > $i1
dir={ find / −type d −depth }& > $i2
until [ ! `ps $i1` && ! `ps $i2` ]
do
sleep 1
done
echo "Ficheros ordinarios: $ord"
echo "Directorios : $dir"
echo "TOTAL : `expr $ord + $dir`"
```

2.8.− Ejecute el shell anterior pero evitando paradas imprevistas. Pruebe a abandonar el sistema y retornar, comprobando que el proceso activado prosigue su ejecución.

Indique en qué fichero la orden "nohup" almacena los resultados y los posibles errores.

nohup.out

2.9.− Cierre la sesión de trabajo, inicie una nueva y verifique que solo tiene asociados los procesos correspondientes a su nueva sesión. Ejecute las siguientes instrucciones:

$ sleep 600 &

$ sleep 700 &

$ ps −f > pslist1

$ kill −9 $$

Abra una nueva sesión de trabajo. ¿Qué ha sucedido con los dos procesos hijos del proceso de conexión anterior, generados con las órdenes sleep?

Compruébelo de la siguiente manera:

1_) Ejecute la orden "ps −f −u `logname` > pslist2"

2_) Compare los archivos "pslist1" y "pslist2" y saque conclusiones.

(No tarde más de 10 minutos en realizar esta prueba ya que 600/60=10)

2.10.− Ejecute la instrucción "$find / −name "fichero" 2> /dev/null &"

Recuerde el número de proceso que se ha asociado a la instrucción anterior.

Convierta el proceso background anterior en un proceso foreground.

stop <PID>

fg <PID>

2.11.− Verifique si tiene activada la suspensión de trabajos para lo cual utilice la orden "$ stty" y busque en el listado que se presenta la cadena "susp" para ver que secuencia de caracteres tiene asociada (por ejemplo puede aparecer "susp = ^Z" lo cual indica que pulsando ctrl+z se suspende la ejecución de un trabajo foreground).

Si no tiene asociada una combinación de teclas asigne una, por ejemplo con la orden "$ stty susp ^Z".

a) Ejecute la orden "$ sleep 400" en foreground.

b) Suspenda su ejecución con la combinación de teclas en cuestión.

c) Mire cuál es el número de proceso del trabajo suspendido y convierta dicho proceso

en un proceso subordinado (background).

bg <PID>

3.− Planificación de trabajos.

3.1.− Indique una instrucción para que visualice dentro de diez minutos el mensaje:

"Han pasado diez minutos."

{sleep 600;echo "Han pasado 10 minutos";}&

3.2.− Planifique un trabajo para que se ejecute el día 5 de diciembre a las 9 de la mañana.

Este trabajo será la ejecución de un guión que se encontrará en su directorio de

conexión y cuya misión será informar de que los días 6 y 8 de diciembre son festivos.

Cree el guión e indique la instrucción para su planificación.

at 0900 12,05

aviso

wall "Los dias 6 y 8 de diciembre son festivos."

3.3.− Planifique otro trabajo que le recuerde la fecha de su cumpleaños lo cual conseguirá

enviando en dicha fecha a su correo un mensaje de "felicidades".

at 0000 11,07

mail f941162 < echo "Felicidades!!"

3.4.− Visualice los trabajos planificados con la herramienta "at".

at −l

3.5.− Con una única instrucción, elimine el trabajo planificado anteriormente para el día 5 de

diciembre, sin efectuar una consulta previa de la referencia del trabajo en cuestión.

at −r

PRÁCTICAS DE SS.OO. UNIX

PRÁCTICA 8. Programación shell general.

Objetivos. Realizar supuestos de programación de guiones shell utilizando todas las herramientas tratadas hasta la fecha.

Herramientas. Las herramientas a utilizar serán las vistas y, además, df.

DESARROLLO DE LA PRÁCTICA

1.− Crear un guión shell llamado "renom" que permita renombrar ficheros. Se debe comprobar

que el primer argumento sea un fichero y que el segundo no exista, visualizándose mensajes de

error oportunos.

```
if [ −f $1 ]
then
if [ ! −a $2 ]
then
mv $1 $2
else
echo Error, el fichero $2 ya existía anteriormente.
fi
else
echo Error, $1 no es un fichero.
fi
```

2.− Hacer un guión llamado "borrar" que admita un número indeterminado de parámetros y

realice lo siguiente:

− Si el parámetro especificado es un fichero ordinario y ocupa más de 50 bytes se

borrará el fichero.

− Si es un directorio solicitará confirmación antes de proceder a borrarlo. Se debe tratar

también el caso de que el directorio tenga ficheros lo cual se notificará antes de la

solicitud de confirmación.

− En cualquier otro caso se visualizará:

"No se procesa el argumento: <argumento>"

Sólo se procesará el directorio actual.

```
for i in $*
do
if [ −f $i ]
then
if [ −s $i > 50 ]
then
rm $i
fi
else
if [ −d $i ]
then
if (find −name $i/*)
then
echo $i es un directorio que contiene archivos. ¿Eliminar? (S/N)
else
echo $i es un directorio vacío. ¿Eliminar? (S/N)
fi
read opcion
if $opcion −eq S
then
rmdir $i
fi
else
echo $i no se procesa.
fi
```

done

3.− Generar un guión shell para manejar un fichero "agenda" que estará formado por los campos nombre, apellidos y teléfono, utilizando como separador de campos el carácter dos puntos (:). El shell debe presentar el siguiente menú y realizar los tratamientos oportunos con el fichero "agenda":

MENÚ DE OPCIONES

==========

[A]ltas

[B]ajas

[C]onsultas

[S]alir

Elija opción (A,B,C,S):

En caso de teclear otra opción se notificará del error y se volverá a presentar el menú.

```
function menu
{
clear
echo MENU DE OPCIONES
echo ==========
echo [A]ltas
echo [B]ajas
echo [C]onsultas
echo [S]alir
echo Opción:
read opcion
case $opcion in
A) alta;;
```

```
B) baja;;

C) consulta;;

esac

}

function alta

{

clear

echo Nuevo telefono

echo ==========\n\n

echo Nombre: \c

read nombre

echo Apellido: \c

read apellido

echo Telefono: \c

read telefono

echo $nombre:$apellido:$telefono >> agenda

}

function baja

{

clear

echo Baja telefono

echo ==========\n\n

echo Apellido: \c

read apellido

for n in cat agenda

do
```

```
ape=echo $n | cut −f2 −d:
if ! $ape=$apellido
then
echo $n >> temp
fi
rm agenda
mv temp agenda
done
}
function consulta
{
clear
echo Buscar telefono
echo ==========\n\n
echo Apellido: \c
read apellido
echo `grep $apellido agenda | cut f1,2 −d:` −−> `grep $apellido agenda | cut f3 −d:`
}
opcion=
until opcion=S
do
menu
done
```

4.− Guión shell que elimine los procesos de su propiedad que lleven ejecutándose más de cinco minutos, a excepción del proceso de conexión, el proceso del shell inicial y el correspondiente a este guión shell.

```
for i in ps
do
proceso=echo $i | tr −s  | cut −f2 −d
if [ `time $proceso` −gt 300 ]
then
if [ ! `$i | tr −s  | cut −f1 −d ` = O ]
then
kill −9 $i
fi
fi
done
```

5.− Guión shell que genere un listado con información de los grupos dados de alta en el sistema. El listado tendrá el siguiente formato.

```
NOMBRE DEL GRUPO: xxxxxxxx
nombre de usuario
nombre de usuario
. . .
NOMBRE DEL GRUPO: xxxxxxxx
. . .
```

```
for i in cat /etc/group
do
grupo=´echo $i | cut −f1 −d,´
usuarios=´echo $i | cut −f4 −d,´
echo NOMBRE DEL GRUPO: $grupo\n
for j in $usuarios
do
```

```
echo  $j\n
done
done
```

6.− Se tiene en un fichero la información sobre los jugadores de fútbol que han conseguido goles en cada jornada durante la liga. El formato del fichero es el siguiente:

jornada:nombre del jugador:equipo del jugador:n_ goles:equipo goleado

Crear un guión shell que genere un listado en el que aparezca el número de goles que ha conseguido cada equipo en cada una de las jornadas.

```
for equipo in `cat goleadores | cut −f3 −d:`
do
goles=0
for i in `grep $equipo goleadores`
do
if [ echo $i | cut −f3 −d: −eq $equipo ]
then
(( $goles=$goles + `echo $equipo | cut −f4 −d:` ))
fi
done
echo $equipo −−−> $goles
done
```

7.− Teniendo en cuenta un fichero llamado "alumnos" con una estructura como la indicada al final del presente ejercicio, escriba un guión shell que genere un fichero "medias" con las notas medias de cada curso. (El fichero puede estar desordenado y no se conocen a priori cuantos ni cuales son los cursos de alumnos.)

N_EXPEDIENTE:CURSO:ASIGNATURA:NOTA

```
for curso in `cat alumnos | cut −f2 −d:`
```

```
do
nota_sum=0
n=0
for i in `grep $curso alumnos`
do
if [ echo $i | cut −f2 −d: −eq $curso ]
then
(( $nota_sum=$nota_sum + `echo $nota_sum | cut −f4 −d:` ))
(( $n = $n +1 ))
fi
done
(( $media= $nota_sum / $n ))
echo $curso −−−> $media
done
```

8.− Supuesto que el nombre de los usuarios del sistema comienza por "p", "r" o "f", obtenga el número de usuarios dados de alta. Visualice también un listado con el nombre de usuario y su directorio de conexión.

```
num=0
for i in `grep /etc/passwd [p,r,f]$`
do
echo $i | cut −f1,5 −d:
(($num=$num+1))
done
echo \nTOTAL: $num
```

9.− Crear un guión que liste el nombre de los usuarios que no tienen fichero ".profile" y el número de ellos, supuesto que el nombre de los usuarios comienza por "p", "r" o "f", y se

tiene permiso de lectura en sus directorios de conexión.

```
for usuario in grep /etc/passwd [p,r,f]$ | cut −f5 −d:
do
if [ ! −f $usuario/.profile ]
then
echo `grep $usuario /etc/passwd | cut −f1 −d:` no tiene .profile.
fi
done
```

10.− Sin utilizar el comando "sleep", crear un guión shell que admita un número del 1 a 60 como parámetro posicional. Este número será el número de segundos que tardará en aparecer en video inverso el mensaje "TIEMPO CONSUMIDO".

Indicar la llamada el guión shell para no necesitar esperar a que termine su ejecución para tener el control del terminal.

```
{ guion_ej10 nn }&
```

11.−

(1) Crear un guión shell que nos permita contar el total de bloques ocupados por los siguientes tipos de ficheros: a) ficheros de bloques, b) ficheros especiales de caracteres, c) directorios, d) tuberías, e) ficheros ordinarios, f) enlaces.

(2) Generar un guión shell que nos permita contar los ficheros de unos directorios pasados como argumentos, mostrando por la salida estándar el nombre del directorio y el total de ficheros del directorio. La totalización de ficheros deberá ser recursiva en cada caso.

(3) Programar un guión shell que simule el comportamiento del comando df, produciendo una salida en megabytes del tipo siguiente:

```
Nombre Espacio Espacio Porcentaje
sist.ficheros ocupado total ocupacion
−−−−−−−−−−−− −−−−−−− −−−−−−− −−−−−−−−−−
```

... ...

... ...

Total espacio disco −−−−−−− −−−−−−− −−−−−−−−−−

tot_tam=0

tot_ocu=0

echo Nombre Espacio Espacio Porcentaje

echo fichero ocupado total ocupacion

echo −−−

for f in ls −R −l /*

do

tamaño=echo $f | tr −s | cut −f −d

ocupa= echo $f | tr −s | cut −f −d

((tot_tam=tot_tam + tamaño))

((tot_ocu= tot_ocu + tamaño))

((porcen=tot_ocu))

done

(4) Crear un guión shell que permita realizar las siguientes operaciones sobre un

fichero de texto tipo base de datos, que contengan los campos abajo indicados:

(a) Actualizaciones (altas, bajas, modificaciones) y consultas, dando a elegir al

usuario la clave de búsqueda.

(b) Clasificación por cualquier campo.

Estructura de la Base de Datos:

NIF(9):NOMBRE(10):APELLIDOS(20):DIRECCIÓN(20): COD_POST(5):TELEF(9)

function menu

{

clear

```
echo 1− Altas\n2−Bajas\n3−Modificaciones\n4−Clasificación\nX−Salir\n
read opcion
clear
case opcion in
1) altas;;
2) bajas;;
3) modificaciones;;
4) clasificacion;;
esac
}
function altas
{
echo NIF: \c
read nif
echo Nombre: \c
read nombre
echo Apellidos: \c
read apellidos
echo Dirección: \c
read direccion
echo C.P.: \c
read cp
echo Tlf: \c
read telefono
echo $nif:$nombre:$apellidos:$direccion:$cp:$telefono >> agenda
}
```

```
function bajas
{
echo Introduzca Clave para Baja (1−NIF, 2−Nombre, 3−Apellidos, 4−Direccion, 5−CP, 6−Tlf): \c
read clave
echo Valor de la Clave de Baja: \c
read valor
for unidad in cat agenda | cut −f$clave −d:
do
if [ ! $unidad −eq $valor ]
then
echo grep agenda $unidad >> temp
fi
done
rm agenda
mv temp agenda
}
function modificaciones
{
echo Introduzca Clave Modificación (1−NIF, 2−Nombre, 3−Apellidos, 4−Direccion, 5−CP, 6−Tlf): \c
read clave
echo Valor de la Clave a Modificar: \c
read valor
mv agenda temp
for unidad in cat temp | cut −f$clave −d:
do
if [ $unidad −eq $valor ]
```

```
then
echo grep temp $unidad
altas;;
else
echo grep temp $unidad >> agenda
fi
done
rm temp
}
function clasificacion
{
echo Introduzca Clave Clasificación (1−NIF, 2−Nombre, 3−Apellidos, 4−Direccion, 5−CP, 6−Tlf): \c
read clave
sort −t: +$clave agenda agenda
}
until [ opcion −eq X ]
do
menu
done
```