Escuela Politécnica Superior de Orihuela
Grado en Ingeniería Informática
en Tecnologías de la Información

Sistemas Operativos
Cuaderno de practicas

Programación del Shell

Objetivos

El objetivo de este cuaderno es familiarizarse con la programación del Shell Scripts.

Para ello se irán proponiendo pequeños ejercicios que permitan poner en práctica y afianzar los conocimientos adquiridos en cuadernos anteriores.

Desarrollo

Los scripts creados en esta sesión, junto con los archivos necesarios para su ejecución y los archivos que se puedan generar por su ejecución, podrán ser almacenados en un directorio con el nombre del ejercicio, dentro del directorio **bin** que cada usuario deberá tener en su home.

$HOME/bin/ejercicio0/

$HOME/bin/ejercicio1/

$HOME/bin/ejercicio2/

etc...

Todos los scripts deberán estar debidamente documentados por medio de comentarios en su código.

Preparación

El objetivo de este punto es preparar el entorno de ejecución para disponer de utilidades que faciliten el trabajo de edición y ejecución de scripts.

Lo primero que haremos será modificar nuestro **.bashrc** para añadirle al PATH el directorio de programas.

Esto lo haremos añadiendo una línea al final del fichero que indique modifique la variable de entorno PATH

```
PATH=$PATH:~/bin
```

Añadiremos otra línea que modifique el PATH para que incluya siempre el directorio de trabajo.

```
PATH=$PATH:.
```

Crearemos un fichero en el directorio de **bin** con el nombre **x**

x se utilizará para dar permisos de ejecución al usuario para el fichero que se le pasa como parámetro. Este es en realidad nuestro primer script.

x necesitará permisos de ejecución el mismo, esto lo haremos con chmod.

Ejercicio0: Creación archivo documentación

Crear un archivo en el directorio de este ejercicio cuyo contenido sea la explicación de porque hemos preparado de este modo el entorno de trabajo, es decir, el porque de cada una de las cosas que hemos hecho en el apartado de preparación.

Ejercicio 1: Lineas

Seleccionar un archivo de texto que ya exista (o en su defecto crearlo) que contenga más de 10 lineas.

Crear un Script llamado **lineas_a** que acepte como único parámetro el nombre del fichero.

El script obtendrá las líneas 5 a 8 del archivo pasado como parámetro y las guardara en un archivo cuyo nombre sea el mismo que el del archivo pasado como parámetro concatenando con '.lines'.

El script mostrará el error 'Numero incorrecto de parámetros' si se pasan 0 o más de 1 parámetro y en las siguientes líneas la sintaxis del script, de esta forma :

Error lineas_a: Número incorrecto de parámetros

Sintaxis: lineas_a <file_name>

donde:

file_name: es el nombre del fichero de entrada

Ejercicio 2: Mejora Lineas

Mejorar el script **lineas_a** creando el script **lineas_b** de forma que además de comprobar que el numero de parámetros es 1, compruebe que el fichero existe.

Si el fichero no existe entonces mostrar el error:

Error lineas_b: El fichero no existe

Sintaxis: lineas_b <file_name>

donde:

file_name: es el nombre del fichero de entrada

Ejercicio 3: Copia Seguridad

Crear un script de copia de seguridad del contenido de nuestro directorio raíz.

Para ello utilizar el comando tar (ver man tar) con las opciones –cPf.

El script se llamará **seg** y tendrá dos modos de ejecución excluyentes. Cada uno de los modos aceptará un único parámetro, significando cosas distintas en cada uno, según se indica a continuación.

El primer modo realizará una copia de seguridad, para lo cual aceptará un parámetro para indicar el directorio de destino de la copia de seguridad. Si el parámetro se omite, creará el fichero de copia de seguridad en el directorio de trabajo. Si el directorio de destino se indica como primer parámetro, (siempre dentro de nuestro home) pero no existe lo creará y generará el fichero de copia de seguridad en el. Si existe no indicará nada y generará el fichero de copia de seguridad en el.

El nombre del fichero generado será **yyyymmdd_home.tar.gz**, donde yyyy es el año, mm el mes y dd el dia. Sugerencia (man date). El fichero de copia de seguridad estará comprimido con la utilizad **gzip** (por eso la extensión gz)

El segundo modo de ejecución utilizará el primer parámetro como opción **–l** de forma que **seg –l** mostrará por la salida estándar un listado con el contenido del fichero de copia de seguridad. Tras el uso de este modo el fichero comprimido debe de seguir existiendo sin modificación.

Notas:

Atención: usar las opciones --exclude-backups y --exclude=".*" para no hacer copia de seguridad de los backups ni de los ficheros ni carpetas ocultas de configuración del Shell del usuario

Para crear el fichero será necesario utilizar primero **tar** y luego **gzip**. (No poner opción de compresión al usar tar). No deben salir mensajes de error por la salida estándar, redirigirlos a **/dev/null**

Para crear la salida del contenido del fichero habrá que descomprimir el fichero con **unzip** y luego utilizar la opción de listado de contenido de **tar.** Recordar que tras este uso el fichero debe quedar igual que antes.

Ejercicio 4: Propiedades del fichero

Realizar un script que reciba el nombre de un fichero. Si el fichero existe, mostrar por pantalla el directorio donde se ubica, indicando también si el usuario tiene permisos de escritura y ejecución.

Modificar el ejercicio para mostrar otra información del archivo.

Ejercicio 5: Ficheros ocultos

Realizar un script que muestre únicamente los ficheros ocultos del directorio que se pasa como parámetro, no los ocultos que están en carpetas anidadas, sólo los de primer nivel.

Ejercicio 6: Paises

Se dispone (crearlo con un editor de textos) de un fichero con varias columnas, separadas por tabulador o espacios, siendo éstas:

<Numero Pasaporte o DNI> <Nombre y Apellidos> <Nacionalidad>

Se pide realizar un script que genere tantos ficheros como nacionalidades distintas cuyo contenido será las tres columnas restantes siendo el nombre del fichero el de la Nacionalidad.

Ejercicio 7: Tick

Escribir un script denominado "tick" que nos muestra la fecha y hora cada segundo durante un minuto y después termine. CONSEJO: utilizar una variable para contar los segundos.

Ejercicio 8: Enlaces

Escribir un script llamado enlaces, que recibe como parámetro un nombre de directorio. Mostrará todos los enlaces simbólicos de dicho directorio si se utiliza la opción –s y todos los enlaces hard si se usa la opción –h, mostrando el directorio enlazado.

Utilizar getopts para el procesamiento de las opciones y parámetros y una función llamada usage() que mostrará la sintaxis del script en caso de introducir parámetros errónos.

Ejercicio 9: Tick60

Escribir un fichero script denominado "tick60" que una vez ejecutado en background (&), podamos seguir trabajando con el shell y nos muestra la fecha y hora actual cada minuto, en un bucle sin fin. Además, cada vez nos informará en una línea diferente desde cuando (fecha y hora) se está ejecutando el script. En este caso, los mensajes que se deberán mostrar serán:

```
Fecha / hora inicio: Sat Nov 2 17:45:16 CET 2002
Fecha / hora actual: Sat Nov 2 17:50:24 CET 2002
```

CONSEJO: necesitaremos una variable de ambiente para almacenar la fecha y hora de inicio.

NOTA: podemos ejecutar en segundo plano (background) el script de forma manual al invocarlo, o bien, que se haga de forma automática desde dentro del script. Si ejecutamos el script en background, para cancelarlo tendremos que utilizar el comando kill, matando un proceso de nombre "sleep" y otro "bash", con cuidado de no matar el primer proceso "bash", puesto que es nuestro shell. El "bash" que tenemos que matar es el shell hijo que se ha creado para ejecutar el script.

Ejercicio 10: Saludo

Escribir un script llamado "saludo" que nos diga "Buenos días" si la hora actual está comprendida entre las 08:00 y las 13:59, "Buenas tardes" si está entre las 14:00 y las 20:59 y "Buenas noches" en el resto de casos (desde las 21:00 hasta las 7:59).

Ejercicio 11: Diasemana

Escribir un script llamado "diasemana" que nos diga el día de la semana actual.

Ejercicio 12: Cuentausuarios

Escribir un script llamado "cuentausuarios" para que cuente el número de usuarios conectados actualmente y nos muestre por pantalla únicamente un valor numérico.

CONSEJO: para contar el número de usuarios podemos utilizar el comando "who", junto con el comando "wc".

Ejercicio 13: Saludo Sistema

Basándonos en los tres scripts anteriores, modificar el archivo script del sistema necesario para que cuando nos conectemos al sistema nos muestre los siguientes mensajes:

```
Buenos dias ... pepe
```

```
Hoy es Martes
Número de usuarios: ?? (Servidor atareado...)
```

En este ejemplo suponemos que nuestro nombre de usuario es "pepe". Deberemos escribir un script de forma genérica, esto es, que sirva para cualquier usuario sin necesidad de modificar nada. Además, mostraremos un mensaje en función del número de usuarios conectados N:

Si 0 < N <= 10: "Número de usuarios: ?? (Servidor ocioso ...)"

Si 10 < N < 20: "Número de usuarios: ?? (Servidor atareado ...)"

Si 20 < N: "Número de usuarios: ?? (Servidor saturado ...)"

CONSEJO: para mostrar los dos mensajes, podemos invocar a los scripts anteriores, en vez de copiar el código. Para el primer mensaje, deberemos de modificar ligeramente el script "saludo" para que imprima un parámetro ($1), y al invocarlo le pasaremos el texto que queremos mostrar justo después del saludo, en este caso, el nombre del usuario.

NOTA: para invocar un script desde dentro de otro debemos escribir lo mismo que si estuviéramos desde el shell, es decir, "./nombre", o bien, modificar la variable de ambiente PATH para que incluya nuestro directorio actual (un punto ".").

Ejercicio14: Avisame

Escribir un script llamado "avisame" que recibe un único parámetro desde la línea de órdenes. Este parámetro es el nombre de un usuario del sistema, y el script nos avisará cuando dicho usuario se conecte al sistema mientras seguimos trabajando, es decir, que se ejecutará en segundo plano.

Si el usuario ya está conectado nos avisa de inmediato y termina su ejecución.

Realizaremos la comprobación cada 20 segundos. En el momento en que detectemos que el usuario está conectado, saldremos del script, o cuando hayan pasado 2 minutos desde su lanzamiento.

CONSEJO: utilizaremos el comando "who" para conocer los usuarios conectados, y "grep" para buscar el usuario que hemos especificado.

Ejercicio15: Menu

Escribir un script llamado "menu" que nos muestre un menú con 3 opciones. Por ejemplo:

```
   1. Mostrar fecha y hora (date)
   2. Ejecutar editor "nano"
   0. Salir
   Escriba la opcion (0-2): _
```

Nos debe pedir la opción por teclado, comprobar que es correcta y ejecutar el comando correspondiente. En caso de no ser correcta nos deberá mostrar un mensaje. El menú aparecerá una y otra vez hasta que tecleemos la opción 0, pasa salir del script