SISTEMAS OPERATIVOS

Actividad - Semana 5

Objetivos

El objetivo de esta actividad es familiarizaros con los conceptos básicos de la programación de la shell BASH. En concreto con los temas abordados hasta la semana 9 de prácticas.

Tareas

Te proponemos que realices las siguiente tareas para la próxima semana:

1. Leer los contenidos de la Lección de la semana 9

En el aula podrás encontrar un enlace al documento Manejo de errores, depuración y comandos adicionales. Te sugerimos que lo leas y hagas los ejemplos que allí se proponen por ti mismo. Te recordamos que también dispones de la presentación BASH Scripting: Una introducción mediante ejemplos.

Al terminar conocerás algunos nuevos elementos de la programación en BASH y será la hora de utilizarlos en tu script **sysinfo_page** a medio hacer.

2. Mejorar el procesamiento de errores

Incorpora a tu script la función de salida con error **error_exit**. Úsala para salir con error si se pasa al script una opción no soportada, después de mostrar la ayuda sobre el uso del programa de la función **usage**.

Además, antes de procesar la línea de comandos comprobar si los comandos **df**, **du** y **uptime** existen. Si no es así salir con un error indicando que falta un programa básico para el funcionamiento del script. La ruta del archivo que implementa un comando se puede obtener ejecutando **which uptime**, por ejemplo. Mientras que el comando **test** dispone de una opción para comprobar si el archivo indicado existe y es ejecutable.

3. Mejorar home_space()

Vamos a mejorar nuestra función **home_space** con algunas funciones adicionales.

1. Si el usuario actual es el root, procesaremos CADA directorio en /home por separado.
   Es decir, en este caso hay que mostrar la información para cada directorio en /home (p.ej /home/ubuntu, /home/jmtorres ,etc) por separado, no la información global para todo /home
2. Mientras que si el usuario actual no es root, procesaremos sólo el directorio personal del usuario.

Para cada directorio a procesar usaremos **du** y **find** para calcular:

1. El espacio total usado, como hasta ahora.
2. El número total de archivos en el directorio y los subdirectorios de éste.
3. El número total de subdirectorios en éste.

Mostrar un cabecera como la siguiente:

```
Directorios Archivos Usado Directorio
```

y a continuación una línea para cada directorio procesado con el número de directorios, el número de archivos, el espacio total usado y la ruta del directorio. Por ejemplo así:

```
Directorios Archivos Usado Directorio
4           123      1200  /home/jmtorres
5           67       4678  /home/jjtoledo
```

¿Sabrías alinear las columnas usando printf para imprimir la información?

4. Desarrollar open_files()

El comando **lsof** se utiliza para listar todos los archivos abiertos en el sistema. El comando es complejo y admite diversas opciones. Una de las más interesantes es **-u**, que permite conocer todos los archivos abiertos por un usuario concreto:

$ lsof -u root

Incorpora a tu script **sysinfo_page** una función **open_files** que muestre una lista ordenada y sin duplicados de **cuentas de usuario que tienen algún archivo abierto** junto con el número de dichos archivos abiertos por cada usuario. Es decir, algo como esto:

```
<h2>Número de archivos abiertos</h2>
<pre>
Usuario    Nº Archivos
root       90
jmtorres   18
jttoledo   25
</pre>
```

Recuerda que existen muchas más cuentas de usuario que la que usas para autenticarte. Tendrás que obtener una lista de todas las existentes y para cada una contar el número de archivos abiertos.

NOTA: El comando tail +n te permite eliminar las primeras n líneas de la entrada ¿para qué lo podrías usar?

6. Revisa y organiza tu script

En este punto podemos dar por terminado el desarrollo del script **sysinfo_page**. Revisa tu código y límpialo para asegurarte que tiene suficiente calidad como para que se note que es de un buen desarrollador:

1. **Recuerda facilitar la lectura de tu código**:
    1. Usa comentarios en los elementos más complejos del programa para que te sea más sencillo entenderlo cuando haya pasado un tiempo y no te acuerdes
    2. Usa funciones para compartimentar el programa, así como variables y constantes (variables en mayúsculas) que hagan tu código más legible.
2. **Incluye código para procesar la línea de comandos**.
    1. Se debe mostrar ayuda sobre el uso y las opciones que soporta si el usuario emplea la opción -h o --help.
    2. Se debe indicar el error y mostrar ayuda sobre el uso si el usuario emplea uno opción no soportada
3. **Ojo con la sustitución de variables y las comillas**. En caso de problemas piensa en cómo quedarían las sentencias si las variables no valieran nada ¿tendría sentido para BASH el comando a ejecutar?
4. **Maneja adecuadamente los errores**.
    1. En caso de error muestra un mensaje y sal con código de salida distinto de 0. Recuerda la función **error_exit** que hemos desarrollado y úsala donde haga falta.
    2. Trata como un error que el usuario emplee opciones no soportadas
    3. Detecta si los comandos de los que depende tus script, como **lsof**, están instalados antes de usarlo. Si no lo está, indica el error.
    4. Haz lo mismo con las otras posibles condiciones de error que se te ocurran ¿has probado a invocar tu programa opciones absurdas a ver si lo haces fallar?

5. Evaluación