Sistemas Operativos II [G4012227] [2023/2024]

Práctica 1 - Intérprete de comandos y Programación de Scripts

Resumen

Conocer el funcionamiento básico de los shells de unix y la programación de scripts.

1. Funcionamiento básico del Shell

1. Busca fuentes de información en internet que sirvan como guía de consulta.
2. Realiza las pruebas sobre los conceptos, comandos y scripts incluidos en el documento *material de apoyo al Tema 5: Uso del shell*.
3. Copia el contenido del fichero */etc/passwd* en tu directorio de trabajo *HOME* usando el comando `cat` y redireccionando la salida al fichero *passwd_copia*. Usa variables de entorno o expansión de tilde para tu directorio de trabajo, es decir, no usar */home/...* directamente.
4. Muestra el contenido del fichero */etc/passwd* y redirecciona la salida al fichero *passwd_copia* usando >>. ¿Cuál es el resultado?.
   - Comprueba que el número de líneas es el doble que antes con el comando `wc`.
5. Escribe un filtro que ordene alfabéticamente las líneas del fichero *passwd_copia*, elimine las duplicadas y guarde el resultado en un fichero *passwd_original* en el directorio temporal */tmp/*. Debe también indicar el número de líneas eliminadas.
   - Comprueba que */tmp/passwd_original* es igual a */etc/passwd* usando los comandos `diff` y `sort`.
6. Haz un script que calcule el tiempo pasado desde una fecha dada como parámetro hasta el momento actual en años, días y minutos.

Los dos últimos scripts deben ser entregados a través del correspondiente entregable del Campus Virtual con los nombres *passwd_ord.sh* y *tiempo.sh* respectivamente.

2. Programación de Shell Scripts

1. En una conferencia se han grabado en vídeo las charlas en distintas resoluciones (HD, Full HD, 4K y 8K) según las necesidades de cada presentación. Los nombres de los vídeos tienen un formato de tipo `sala_XX_fecha@hora.res`: *XX* es el número de sala en el rango 20 . . . 50, *fecha@hora* es la fecha y hora de la conferencia y *res* indica la resolución, por ejemplo `sala_20_2023-02-11@17:04.4K`.
   - Ordenar las distintas charlas en directorios con el formato `salaXX/fecha/res` y copiar cada vídeo a su carpeta correspondiente cambiando el nombre por `charla_hora`. Por ejemplo, el vídeo `sala_20_2023-02-11@17:04.4K` iría a la carpeta `sala20/2023-02-11/4K/` con el nombre `charla_17:04`.

- El script debe aceptar como parámetros el directorio donde se encuentran todos los vídeos y aquel en el que se van a crear las diferentes carpetas.
- Para probar el script tienes archivos de texto con distintos nombres en el campus virtual con el formato especificado para que puedas probar tu script. Ver carpeta *conference* en archivo *material_P1.zip* disponible en el campus virtual.
- Para resolver este ejercicio te puede interesar profundizar en *Shell Parameter Expansion*[1].

Requisitos del Script

- El script debe llamarse *conferencia.sh origen destino*. El *origen* y el *destino* pueden ser rutas relativas al directorio.
- No se debe usar el comando *awk*.
- Comprobar que se pasan sólo dos parámetros; en caso de que esto no se cumpla, el script debe terminar con un mensaje de error y un ejemplo de uso.
- Comprobar que el primer parámetro (`origen`) es un directorio y que tiene permisos de lectura; en caso de que esto no se cumpla, el script debe terminar con un mensaje de error y un ejemplo de uso.
- Comprobar que el segundo parámetro (`destino`) es un directorio y que tiene permisos de escritura; en caso de que esto no se cumpla, el script debe terminar con un mensaje de error y un ejemplo de uso. Si el directorio tiene algún subdirectorio con el nombre indicado en este enunciado, debe avisarse al usuario y salir del script.
- Crear todos lo directorios de salida, 20 . . . 50 y los subdirectorios con las distintas fechas, aunque estén vacíos.
- Las fechas se deben obtener a partir de la lista de ficheros de entrada.
- El directorio *conference* disponible en el Campus Virtual es un ejemplo con nombres de fichero. El script debe funcionar para cualquier fichero de conferencia con el formato indicado.

2. El archivo *access.log* disponible en el Campus Virtual tiene información de los accesos a un servidor web Apache con el siguiente formato:

   `Dirección IP - - [Fecha Hora Zona] "GET/POST URL" Respuesta Bytes enviados.`
   Por ejemplo:

   `127.0.0.1 - - [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0"200 2326`

   - Crea un script para procesar el fichero de log que realice diferentes tareas en función de los parámetros de entrada:
   - `-c`: Muestra los diferentes códigos de respuesta sin repetición. Indicará, además, el número de veces que se produce cada uno de dichos códigos.
   - `-t`: Muestra el número de días para los que no hay ningún acceso al servidor, desde la fecha del primer acceso hasta la fecha del último.
   - `GET/POST`: Cuenta el total de accesos con una petición tipo `GET/POST` con una respuesta `200`, mostrando por pantalla el resultado. Ten en cuenta que los dígitos de la respuesta, `200`, pueden formar parte de la dirección IP, la zona horaria, la URL o de los Bytes enviados. La salida debe indicar la fecha y hora de ejecución del script seguida por la información solicitada tener el formato del siguiente ejemplo:

     `Feb 12 11:10:31. Registrados 101422 accesos tipo POST con respuesta 200.`

[1] https://www.gnu.org/software/bash/manual/html_node/Shell-Parameter-Expansion.html

- `-s`: Resume el total de Datos enviados en *KiB* por cada mes. Los `Datos enviados` registrados en el log están representados en bytes. Ten en cuenta que los meses en el fichero de log quedaron registrados en inglés (Dec, Jan, Feb). La salida debe ser como la del siguiente ejemplo:
  `1764100 KiB sent in Jan by 1626 accesses`.
- `-o`: Ordena las líneas del fichero *access.log* en orde decreciente del número de bytes enviados. El resultado lo almacenará en el fichero *access_ord.log*.

Requisitos del Script

- El script debe llamarse *accesos.sh [-c, GET/POST, -s] ruta/al/archivo/de/log/access.log*.
- Comprobar que se pasan sólo dos parámetros; en caso de que esto no se cumpla, el script debe terminar con un mensaje de error y un ejemplo de uso.
- Comprobar que el segundo parámetro es un archivo de tipo regular y que tiene permisos de lectura; en caso de que esto no se cumpla, el script debe terminar con un mensaje de error y un ejemplo de uso.
- El archivo *access.log* disponible en el Campus Virtual es un ejemplo de fichero de log. El script debe funcionar para cualquier fichero de log con el mismo formato.

3. **OPCIONAL.** Crea un script denominado *telefonos.sh* que simule una agenda telefónica en un archivo de texto que se le debe pasar como argumento. En cada una de las líneas dicho archivo se deben almacenar el nombre y el número de teléfono correspondiente usando como separador el carácter @. Las principales acciones de dicho script serán crear una nueva agenda, registrar una nueva entrada en la agenda, buscar por nombre, buscar por número de teléfono, modificar una entrada y borrar una entrada.

4. **OPCIONAL.** Crea un script denominado *ordenacion.sh* que muestre el contenido de un directorio ordenándolo en función de los siguientes criterios:

   - `-a`: Según el número de caracteres de los nombres de los ficheros.
   - `-b`: Según el orden alfabético de sus nombres escritos al revés.
   - `-c`: Según el valor de los últimos 4 dígitos de su *inode*, de menor a mayor.
   - `-d`: Según su tamaño y organizados en grupos 8 diferentes asociados al valor de los permisos *rwx* del propietario.
   - `-e`: Según su tamaño y organizados en grupos definidos por el mes de su último acceso.