Universidad de Valladolid – Departamento de Informática

Fundamentos de Informática II

Laboratorio - Programación del Shell de Unix

Guía de Prácticas

Sesión 2

Curso 2009/2010

1. Parámetros y variables

1. Crear un script que compruebe si una variable ha sido creada. Mostrar con ejemplos todas las aplicaciones posibles del uso de llaves para verificar la definición de variables.
   El programa debe mostrar los mensaje adecuados que expliquen claramente los ejemplos presentados.

2. Pruebas con variables de entorno.

   - Crear un script que cree una variable y la exporte como variable de entorno. Antes de salir del script verificar que la variable es efectivamente una variable de entorno.
   - Crear un segundo script que muestre el valor de la variable de entorno creada en el primer script. A continuación anular la definición de esa variable. Verificar que la variable ha eliminada.

3. Crear un script que nos muestre por pantalla sus dos primeros argumentos y el nombre del propio script. Para la siguiente entrada:

   ```
   ejemplo1 pruebo cualquier cosa
   ```

   el formato de la salida debe ser:

   ```
   ejemplo1 pruebo cualquier cosa
   El formato de la salida debe ser:
   El nombre del script es: ejemplo1
   El argumento 1 es pruebo
   El argumento 2 es cualquier
   ```

- Modificar el script anterior para que indique al principio cuántos argumentos ha recibido y que los muestre todos antes de mostrar de forma separada los dos primeros.
- Posteriormente, insertar la instrucción `shift` al final del guión anterior, y repetir todas las instrucciones que muestran los argumentos. ¿Qué ocurre? ¿Y si se hace `shift 2`?
- Insertar `shift` entre cada instrucción de escritura. ¿Qué ocurre?

4. Modificar el programa del anterior ejercicio para que dé un mensaje cuando no se introduzca ningún argumento y abandone la ejecución con un código de salida igual a 1.

5. Crear un script que muestre:

   - El número máximo de usuarios que pueden conectarse al sistema.
   - El número de usuarios conectados actualmente al sistema.

6. Crear un script que reciba un nombre de usuario como argumento.

   - Verificar que el script solo acepte un parámetro.
   - Verificar si el parámetro dado se corresponde con un usuario del sistema.
   - Si es un usuario del sistema:
     - Mostrar su directorio `HOME`
     - Mostrar si está conectado actualmente al sistema y finalizar el script con un código de salida igual a 0.
     - Si no está conectado actualmente al sistema finalizar el script con un código de salida igual a 6.
   - En caso contrario indicar que no se trata de un usuario del sistema y finalizar con un código de salida igual a 66.

7. Crear un guión que solicite al usuario el nombre de un directorio válido *dir*. A continuación debe hacer un listado de todos los archivos ordinarios que estén situados en ese directorio *dir* y convertirlos en los argumentos del guión.

   - Mostrar los nuevos argumentos utilizando las variables `$*` y `$@`.
   - Indicar cuántos argumentos hay utilizando `$#`.
   - Realizar una nueva versión donde sólo se muestren los directorios que estén en *dir*.

8. Modificar el script anterior para que el nombre del directorio *dir* se reciba como primer argumento.

   - Generar una nueva versión donde se compruebe que el argumento existe y además es un directorio válido.

9. Realizar un script que compruebe que el número de argumentos recibido sea exactamente dos y que si no es así termine dando un mensaje de error. El script debe tener un único punto de salida mediante la sentencia `exit`, devolviendo 0 si todo ha ido correctamente o 1 si se ha producido alguna incidencia.

10. Crear un script que reciba un único argumento, comprobando si es un directorio dentro del directorio actual. Si lo es, debe informar del número de ficheros que están bajo ese directorio. Si no era un directorio bajo el directorio actual, debe solicitar su ruta completa, y proseguir como se ha descrito anteriormente.

11. Crear un programa shell que muestre el contenido del directorio actual si hay más de 2 ficheros o directorios.

    - Modificar el programa para sea el usuario el que diga el número mínimo de ficheros o directorios que debe haber para que se muestren.
    - Modificar el programa anterior para que el número mínimo de ficheros o directorios se introduzca como argumento.

12. Crear un programa shell que compruebe si hay más de 3 ficheros (sólo ficheros) en el directorio actual y los muestre.

    - Crear una variación del programa que verifique directorios (sólo directorios).

2. Expresiones – Sentencias condicionales

En todos los casos a continuación si el script shell se realiza sin errores debe finalizar con un código de salida igual a cero. En otro caso debe finalizar con un código de salida distinto de cero.

1. Realizar un programa shell que lea dos variables y que presente los resultados de todas las operaciones aritméticas definidas en el comando `expr` aplicado a esas variables.

    - El programa debe dar mensajes muy descriptivos de modo que el usuario entienda la naturaleza de todos los resultados presentados.
    - El programa debe validar los datos que el usuario introduce, es decir, no debe aceptar datos nulos o alfanuméricos, todos los datos deben ser numéricos.
    - En caso de que el segundo dato sea un cero, lo cual implicaría realizar una operación no válida en el caso de la división o del módulo, el programa debe mostrar el mensaje `"No es posible dividir por cero"`. El resto de operaciones deben ser realizadas.

2. Realizar el anterior programa, pero en este caso aceptando los datos como argumentos del script. Si hay algún problema con los argumentos estos debe indicarse al usuario, para a continuación permitir su lectura mediante un comando `read`

3. Realizar un programa que lea dos datos alfanuméricos y presente los resultados de todas las operaciones de cadena que pueden realizarse mediante el uso de variables y del comando `expr`. En este caso se considerará como dato no valido la cadena nula o vacía.

4. Realizar el programa de manipulación de cadenas anteriormente descrito, pero en este caso aceptando las cadenas como parámetros del script. Si se detecta algún problema en los parámetros se deben leer las cadenas mediante los comandos `read` que sean necesarios.

5. Realizar un programa que lea un nombre de usuario y que indique el número de sesiones abiertas que tiene actualmente ese usuario. (El programa debe estar implementado con sentencias `if`.

   - Si tiene una sola sesión abierta, mostrar los procesos lanzados por ese usuario.
   - Si tiene más de una sesión abierta mostrar la información detallada de esas sesiones.
   - Si no tiene ninguna sesión abierta mostrar su directorio `HOME`

6. Crear una nueva versión del anterior programa empleando sentencias `case`.

7. Realizar un script que reciba como único argumento una cadena de caracteres y que a continuación muestre todos los usuarios que están conectados al sistema cuyo nombre contenga la cadena que se pasa como argumento.

8. Realizar un script que reciba exactamente tres argumentos. Los dos primeros argumentos identifican un fichero: cuyo nombre es el primer argumento, y se encuentra en el script que se pasa como segundo argumento. El tercer argumento deberá ser una cadena `TODO` ó `PASO`, bien en mayúsculas o en minúsculas, para indicar que se debe mostrar todo el contenido de una vez o parándose tras mostrar cada pantalla de información, respectivamente.

9. Realizar el script anterior utilizando al menos una vez sentencias `case`.

10. Realiza un script del shell llamado fecha que nos permita sacar distintos tipos de información de la fecha actual. El script debe utilizar la salida del comando `date`. El script tendrá un argumento, que será opcional. Su sintaxis será:

    ```
    fecha [a|A|m|M|d|D|h|H]
    ```

    Si recibe más de un argumento debe informar que hubo un error y terminar. Si no recibe ningún argumento, debe dar toda la información de la fecha actual, en el formato siguiente:

    ```
    Son las XX:YY h del DD del mes MM de AAAA
    ```

    Si recibe exactamente un argumento, debe analizar caso por caso y mostrar mensajes similares a las del párrafo anterior, indicando sólo la hora (h), día (d), mes (m) o año(a). Por ejemplo:

    ```
    $ fecha a
    Estamos en el año 2008
    $ fecha M
    Estamos en el mes de abril
    ```

Y así sucesivamente.

11. Realizar una nueva versión del algoritmo anterior utilizando el esquema condicional `case`.

12. Crear un script que reciba un único argumento y nos indique si es un directorio válido y en ese caso, si en ese directorio hay algún subdirectorio que no sean él mismo y su directorio padre.

13. Realizar un script que reciba como único parámetro un directorio y que genere un fichero en el directorio de trabajo actual, cuyo nombre será `ejecutables`, que contenga los nombres de los archivos ordinarios en ese directorio que además sean ejecutables.

14. Realizar un script que tenga un mínimo de dos argumentos. El primer argumento será el nombre de un directorio. El resto de argumentos serán nombres de ficheros. El script debe incluir el contenido de todos los ficheros en un nuevo fichero llamado `todos.txt`, siempre que sean ficheros ordinarios. `todos.txt` debe residir en el directorio destino que se ha pasado como primer parámetro. Informar si se ha producido alguna incidencia o si todo el proceso se ha desarrollado sin errores.

15. Realizar un script que tenga exactamente tres argumentos. El primer argumento será el nombre de un directorio. El script debe copiar los ficheros de ese directorio en otro, que se pasa como segundo argumento. Sólo deben copiarse los ficheros ordinarios que contengan en su nombre la cadena que se pasa como tercer argumento.