Ejercicios de Shell Script

01 - Básicos

1. Realizar un script llamado '**01-hola-mundo.sh**' que muestre por pantalla "Hola mundo!".
2. Ídem pero que en vez de "mundo" muestre los parámetros introducidos ('**02-hola-parametros.sh**').
3. Ídem y que además verifique que al menos hayamos introducido un parámetro ('**03-hola-al-menos-1-parametro.sh**').
4. Ídem y que además separe cada argumento por ", " ('**04-hola-parametros-separados.sh**').
5. Ídem y que además en caso de error muestra una ayuda ('**05-hola-con-ayuda.sh**').
6. Ídem y que además verifique que sean usuarios conectados al sistema ('**06-hola-usuario.sh**').
7. Realizar un script llamado '**usuarioconectado**' que retorna un SI si el primer parámetro coincide con algún usuario conectado o NO en caso contrario.
8. Modificar el fichero '**.bashrc**' para modificar el PATH y añadir la carpeta de estos ejercicios. Para ello añade la siguiente linea: **export PATH=$PATH":~/ruta_carpeta_ejercicios"**
9. Modificar el script '**06-hola-usuario.sh**' para que llame a 'usuarioconectado' ('**09-hola-usuario.sh**').
10. Realizar un script llamado '**usuariosistema**' que retorna un SI si el primer parámetro coincide con algún usuario del sistema o NO en caso contrario.
11. Modificar el script '**09-hola-usuario.sh**' para que llame a 'usuariosistema' ('**11-hola-usuario.sh**').

02 - Calculadora

12. Realizar un script llamado '**suma**' que realice la suma de 2 parámetros introducidos (tendrá que poder sumar números decimales, como 2.2 + 3).
13. Realizar un script llamado '**resta**' que realice la resta de 2 parámetros introducidos (tendrá que poder sumar números decimales, como 2.2 - 3).
14. Realizar un script llamado '**multiplica**' que multiplique los 2 parámetros introducidos (tendrá que poder multiplicar números decimales, como 2.2 * 3).
15. Realizar un script llamado '**division**' que realice la división de 2 parámetros introducidos (tendrá que poder sumar números decimales, como 2.2 / 3).
16. Realizar un script llamado '**calc01.sh**' que realice operaciones básicas entre 2 números llamando a cada uno de los scripts anteriormente creados (suma, resta, multiplicación y división).
17. Ídem pero sin llamar a los scripts ('**calc02.sh**').
18. Realizar un script llamado '**calc03.sh**' que calcule el valor una expresión numérica pasada por parámetro.

Autor: Adolfo Sanz De Diego (asanzdiego - Blog|GitHub|LinkedIn|SlideShare|Twitter)

19. Realizar a mano un fichero '**notas.csv**' con los siguientes datos:

```
Pepito      3.1   4.4   5.7
Fulanito    4.2   6.5   8.8
Menganito   5.3   5.6   5.0
```

20. Realizar un fichero '**notas.awk**' y su correspondiente interfaz '**notas.sh**' para que al final obtengamos algo parecido a esto:

```
+-----------------------------+-------+------+
| NOMBRE      EX1   EX2   EX3   |  MED  | APTO |
+-----------------------------+-------+------+
| Pepito      3.1   4.4   5.7   |  4.4  |  NO  |
| Fulanito    4.2   6.5   8.8   |  6.5  |  SI  |
| Menganito   5.3   5.6   5.0   |  5.3  |  SI  |
+-----------------------------+-------+------+
| TOTAL       4.2   5.5   6.5   |  5.4  |   2  |
+-----------------------------+-------+------+
```

03 - Banco

21. Realizar un script llamado '**banco**' para añadir, buscar y listar movimientos bancarios, y calcular el saldo de la cuenta.

22. Realizar un script llamado '**banco-menu.sh**' que sirva de interfaz del anterior.

23. Realizar un script llamado '**banco-flags.sh**' para poder usar el script '**banco**' mediante CLI.

04 - Demonios

24. Realizar un demonio llamado '**alerta**' que escriba la fecha cada X segundos en un log llamado '**~/alerta.log**'.

25. Realizar las interfaces del demonio 'alerta' con las opciones básicas: start, stop, restart y status ('**servicio-alerta.sh**').

05 - Copias

26. Realizar un script llamado '**copia-total**' que empaquete y comprima el contenido de la carpeta '**~/carpeta_a_copiar**' en un fichero llamado '**total-aaaa.mm.dd-HH.MM.SS.tar.zip**' en la carpeta '**~/copia_seguridad**'.

27. Realizar un script llamado '**copia-diferencial**' que empaquete y comprima los ficheros de la carpeta '**~/carpeta_a_copiar**' modificados desde la última copia total (si no existe copia total no hacer nada) en un fichero llamado '**diferencial-aaaa.mm.dd-HH.MM.SS.tar.zip**' en la carpeta '**~/copia_seguridad**'.

28. Realizar un script llamado '**copia-incremental**' que empaquete y comprima los ficheros de la carpeta '**~/carpeta_a_copiar**' modificados desde la última **copia incremental** en un fichero llamado '**incremental-aaaa.mm.dd-HH.MM.SS.tar.zip**' en la carpeta '**~/copia_seguridad**'.

29. Modificar el fichero '**miCrontab**' para que imprima la fecha en el fichero '**~/ultimo-crontab.txt**' cada minuto, y ejecutarlo con crontab.

06 - Varios

30. Crear un script llamado '**array.sh**' que declare un array, lo rellene con datos y luego itere sobre el mismo para mostrar los datos.

31. Realizar a mano un fichero '**roles.csv**' con los siguientes datos:

```
Pepito:Jefe,Sistemas
Fulanito:Jefe,Desarrollo
Menganito:Operario,Sistemas,Desarrollo
```

32. Realizar un script '**roles-sin-awk.sh**', que, sin utilizar awk, al final obtengamos algo parecido a esto:

```
Desarrollo
 -> Fulanito Menganito
Operario
 -> Menganito
Sistemas
 -> Pepito Menganito
Jefe
 -> Pepito Fulanito
```

33. Realizar un fichero '**roles.awk**' y su correspondiente interfaz '**roles-con-awk.sh**' para que al final obtengamos lo mismo que el ejercicio anterior.

34. Realizar un script llamado '**ordena**' que liste el contenido del directorio actual ordenado por tamaño del archivo de menor a mayor. El listado sólo mostrará el nombre de los archivos y el número de línea correspondiente. En el caso de que se introduzca algún parámetro se mostrará el siguiente mensaje de error: "No se permiten parámetros." y retornará un código de retorno igual a 1.

35. Realizar un script llamado '**jaula**' que cree, sólo si no existe, el directorio **. jaula** en la $HOME del usuario y mueva los ficheros pasados por parámetro a dicho directorio. En el caso de que no se le pase ningún parámetro se mostrará el siguiente mensaje de error: "Hay que introducir al menos un parámetro." y retornará un código de retorno igual a 1. En el caso de que algún fichero introducido por parámetro no exista se mostrará el siguiente mensaje de error: "El fichero '$FICHERO' no existe." y retornará un código de retorno igual a 2. Si el fichero **. jaula** existe en la $HOME del usuario pero no es un directorio mostrará el siguiente mensaje de error: "El fichero '$HOME/.jaula' no es un directorio." y retornará un código de retorno igual a 3.

36. Realizar un script llamado '**calendario**' al que si pasamos el parámetro **-c** o el parámetro **--corta** mostrará la fecha de hoy con el formato "$DIA/$MES/$AÑO" y si le pasamos el parámetro **-l** o **--larga** mostrará la fecha de hoy con el formato "Hoy es el día '$DIA' del mes '$MES' del año '$AÑO'.". En el caso de que no se introduzca ningún parámetro se mostrará el calendario del mes actual. En el caso de que el número de parámetros introducidos sea distinto de 1 se mostrará el siguiente mensaje de error: "Sólo se admite un parámetro." y retornará un código de retorno igual a 1. Si pasamos otra cosa que no sea **-c**, **--corta**, **-l** o **--larga** mostrará el siguiente mensaje de error: "Opción incorrecta." y retornará un código de retorno igual a 2.

37. Realizar un script llamado '**elevado**' que calcule "a^b", osea "a elevado a b", donde "a" será el primer parámetro y "b" el segundo parámetro. En el caso de que el número de parámetros introducidos sea menor que 2 se mostrará el siguiente mensaje de error: "Para ejecutar este script se necesitan 2 números." y retornará un código de retorno igual a 2. Nota: se deberá realizar con una iteración.

38. Realizar un script llamado '**citas**' en el que se puedan utilizar las siguientes opciones:

```
-h --help   Para mostrar un texto de ayuda.
-a --add    Para añadir una cita con HORA_INICIO, HORA_FINAL, y NOMBRE_PACIENTE.
-s --search Para buscar los pacientes que contengan PATRÓN.
-i --init   Para buscar las citas que empiecen a HORA_INICIO.
-e --end    Para buscar las citas que terminen a HORA_FINAL.
-n --name   Para listar todas las citas ordenadas por NOMBRE_PACIENTE.
-o --hour   Para listar todas las citas ordenadas por HORA_INICIO.
```

- Para cada una de las opciones se comprobará que se introducen el número de parámetros correctos y con el formato correcto.

- HORA_INCIO y HORA_FINAL serán números enteros comprendidos entre 00 y 23.

- Al introducir una cita nueva se comprobará que no se solape con otra ya introducida.

- Se comprobará también que no se repita ningún nombre de paciente.

39. Realizar un script llamado '**citas-menu.sh**' que sea una interfaz del script 'citas' motrando un menú con las siguientes opciones:

```
1.Añadir cita nueva.
2.Buscar por nombre del paciente.
3.Buscar citas por hora inicial.
4.Buscar citas por hora final.
5.Listar las citas ordenadas por nombre del paciente.
6.Listar las citas ordenadas por hora inicial.
7.Salir del programa.
```

40. Realizar un script llamado '**citas-flags.sh**' para poder usar el script '**citas**' mediante CLI.

07 - Bonus

41. Ver y entender los scripts de https://github.com/asanzdiego/markdownslides, en particular:
    1. https://github.com/asanzdiego/markdownslides/blob/master/build.sh
    2. https://github.com/asanzdiego/markdownslides/blob/master/git-push.sh
    3. https://github.com/asanzdiego/markdownslides/blob/master/git-pages.sh
42. Ver y entender los scripts de http://asanzdiego.blogspot.com.es/2014/09/el-making-of-del-mapa-de-la-evolucion-del-no2-en-Madrid-del-hack4good.html, en particular:
    1. https://github.com/asanzdiego/mapa-evolucion-contaminacion-aire-madrid/blob/master/estaciones-madrid-toarray.sh
    2. https://github.com/asanzdiego/mapa-evolucion-contaminacion-aire-madrid/blob/master/parsea.sh
    3. https://github.com/asanzdiego/mapa-evolucion-contaminacion-aire-madrid/blob/master/filtra.sh
    4. https://github.com/asanzdiego/mapa-evolucion-contaminacion-aire-madrid/blob/master/categoriza-no2.sh

