Ejercicios de Shell Script y AWK

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 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
```

13. 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  |
+-----------------------------+-------+------+
```

14. Realizar un script llamado '**calc-simple-awk.sh**' que realice operaciones básicas entre 2 números (suma, resta, multiplicación y división) utilizando el comando awk.

15. Ídem pero utilizando el comando bc ('**calc-simple-bc.sh**').

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

16. Realizar un script llamado '**calc-avanzada-awk.sh**' que calcule el valor una expresión numérica pasada por parámetro utilizando el comando awk.

17. Realizar un script llamado '**calc-avanzada-bc.sh**' que calcule el valor una expresión numérica pasada por parámetro utilizando el comando bc.

03 - Banco

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

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

04 - Demonios

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

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

05 - Copias

22. 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**'.

23. 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**'.

24. 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**'.

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

06 - Varios

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

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

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

28. Realizar un script '**roles-sin-awk.sh**', que, sin utilizar awk, al final obtengamos algo parecido a esto (tanto roles como personas están ordenados alfabéticamente:

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


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

30. 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.

31. 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.

32. 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.

33. 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.

34. (Oposiciones Madrid 2018 INF) Realizar un script llamado '**primos.sh**' que imprima un listado de los números primos en el intervalo [A, B]. A y B serán pasados como parámetros.

Ejemplo de ejecución y salida generada para el intervalo [5,14]:

```
$ ./primos.sh 5 14
5
7
11
13
```

NOTA: Se puede comprobar si un número es primo analizando la salida de la utilidad factor incluida en el paquete GNU coreutils (que suponemos estará instalado en su sistema). Dicha utilidad factoriza números enteros.

Ejemplo de uso y salida generada de la llamada a la utilidad factor:

```
$ factor 2018

2018: 2 1009
```

35. (Oposiciones Madrid 2018 SAI) Realizar un script llamado '**buscar**' que simule el comando find, pero sin usar ni el comando find ni el comando ls.

Podrá recibir de uno a tres parámetros:

- En caso de ser uno, será el fichero a buscar en el directorio actual.

- En caso de ser dos, será una de las siguientes opciones:

  - o directorio (ruta completa) y fichero a buscar en ella.

  - o opción '-s' y fichero y buscará el fichero en la ruta actual y sus subdirectorios.

- En caso de ser tres serán opción '-s', directorio (ruta completa), fichero y buscará el fichero en la ruta y sus subdirectorios.

Ejemplos:

```
$ buscar fichero

$ buscar directorio fichero

$ buscar -s fichero

$ buscar -s directorio fichero
```

Nota: Se valorará la eficiencia y el control de errores.

36. (Oposiciones Madrid 2016 SAI) Tenemos un pc con una versión de Linux server sin interfaz gráfica en el que disponemos de un fichero llamado empleados.txt que contiene información de los empleados de nuestra empresa, una línea por cada empleado. Cada línea contiene información de un empleado separando cada campo por el carácter #. Los campos almacenados en cada línea son; DNI, primer apellido, segundo apellido, nombre, día nacimiento, mes nacimiento, año nacimiento, marca de fecha de nacimiento, edad, departamento al que pertenece, si tiene o no carnet de conducir, número de hijos que tiene, estado civil y sueldo.

```
DNI#APE_1#APE_2#NOM#DIA#MES#ANO#F_NAC#EDAD#DEPART#C_CONDUCIR#N_HIJOS#ESTADO_CIVIL#SUELDO
45.302.157-T#HAANAN#MIMUN#NADIA#7#1#1990#32880#26#INFORMATICA#NO#5#DIVORCIADO#1743
43.094.614-R#ABAD#BARRIO#FUENCISLA#9#8#1968#25059#47#SERVICIO TÉCNICO#NO#2#SOLTERO#2044
32.843.994-V#ABAD#VÁZQUEZ#TATIANA PALOMA#10#1#1962#22656#54#INFORMATICA#SI#1#SOLTERO#2774
13.138.768-H#ABAJO#HERNANDO#INMACULADA#26#9#1957#21089#58#RRHH#NO#5#SOLTERO#912
71.555.763-A#ABANADES#FERNÁNDEZ#LIBERTAD#16#11#1989#32828#26#RRHH#NO#2#SOLTERO#2095
45.293.901-R#ABDELKADER#MIMUN#KARIMA#7#11#1985#31358#30#SERVICIO TÉCNICO#NO#2#DIVORCIADO#2243
45.283.008-X#ABDEL-LAH#MOHAMED#CANDILA#23#3#1983#30398#32#CONTABILIDAD#SI#4#SOLTERO#2026
45.281.108-L#ABDERRAMAN#MOHAMED#LEILA#10#8#1982#30173#33#I+D#NO#2#DIVORCIADO#2520
12.383.679-L#ABRIL#FERNÁNDEZ#ELENA#21#10#1936#13444#79#RRHH#SI#5#CASADO#2488
71.547.481-R#ACEDO#PEREZ#ADORALINA#20#5#1959#19499#62#RRHH#SI#3#DIVORCIADO#1413
45.282.525-X#ACOSTA#BERNABE#M. ANGELES#24#10#1035#13081#80#INVESTIGACIÓN#NO#1#VIUDO#1555
80.159.645-E#ACOSTA#FERNÁNDEZ#MARIA#5#2#1947#17203#69#INVESTIGACIÓN#NO#1#SOLTERO#2066
51.929.697-K#ACOSTA#GONZALEZ#M. CARMEN#7#11#1977#28436#38#I+D#SI#1#SOLTERO#3049
06.243.735-V#ADEVA#RODRÍGUEZ#AFRICA#1#11#1936#13455#79#RRHH#NO#3#CASADO#973
53.567.648-G#AGRO-MARTÍN#JUAREZ#LUZDIVINA#21#11#1955#20414#60#RRHH#NO#3#DIVORCIADO#2666
12.760.394-V#AGUADO#SAN MARTÍN#JUAN CARLOS#11#7#1970#25760#45#CONTABILIDAD#SI#5#SOLTERO#2690
46.451.841-Y#AGUAYO#FERRERAS#DAMARIS#22#10#1980#29516#35#RRHH#NO#4#VIUDO#1303
44.097.101-Y#AGUDELO#LONDOÑO#NEILA ESLEIDI#27#3#1954#19810#61#RRHH#NO#5#SOLTERO#1778
```


Se pide realizar un script bash que reciba un único argumento de entrada que podrá ser “dpto” o “estadocivil”. Si no se introduce ningún argumento o el número de argumentos es mayor de uno o el único argumento es distinto de los dos valores citados, se deberá mostrar un mensaje de error por pantalla y se terminará el script.

Si el primer argumento es:

- dpto. Creará una cuenta por cada empleado que haya dentro del fichero poniendo como comentario su nombre y apellidos, el Shell que se utilizará será /bin/bash, la cuenta caducará en 2017 el día y mes de su nacimiento. Su grupo principal será el departamento al que pertenece pero anteponiendo un “g” y el login será su primera letra del nombre seguido de un “.” Seguido del primer apellido seguido de un “.” Y su segundo apellido. Si tomamos de ejemplo al primer empleado se pondrá como comentario “nadia aanan mimun”, la cuenta caducará el 7/1/2017 el grupo principal será ginformatica y el login será n.aanan.mimun. Se pondrá a todos los empleados la clave Op0s2016

- estadocivil. Creará una cuenta por cada empleado que haya dentro del fichero poniendo como comentario su nombre y apellidos, el Shell que se utilizara será /bin/bash, la cuenta caducará en 2017 el día y mes de su nacimiento. Su grupo principal será su estado civil pero anteponiendo una “g” y el login será su primera letra del nombre seguido de un “.” seguido del primer apellido seguido de un “.” y su segundo apellido. Si tomamos de ejemplo al primer empleado se pondrá como comentario “nadia aanan mimun”, la cuenta caducará el 7/1/2017 el grupo principal será gdivorciado y el login será n.aanan.mimun. Se pondrá a todos los empleados la clave Op0s2016

Al final del script, se mostrará un resumen donde aparezcan; número de departamentos diferentes, número de empleados creados correctamente y número de empleados que no se han podido crear.

La salida tendrá el siguiente formato:

```
Número de departementos/estados: XXXX
Número de empleados creados correctamente: XXXX
Número de empleados no creados: XXXX
```

Los errores que se produzcan al crear el usuario deberán almacenarse en el fichero llamado “log_creacion.txt”

Consideraciones.

- Todos los datos del fichero empleados.txt se convertirán a minúsculas (nombre, apellidos, estado civil, departamento, etc)

- No se permite la utilización de ficheros temporales.

- Los departamentos y estado civil pueden contener espacios, de forma que se sustituirán por guión bajo _

- Los nombres, primer apellido y segundo pueden contener espacios. Se deben eliminar dichos espacios.

- No se debe considerar la primera línea del fichero de empleados ya que solo contiene un encabezado y no información de un empleado.

- El número y nombre de departamentos y estados civiles deben obtenerse del fichero empleados.txt

- Se eliminarán acentos y eñes.

37. 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.

38. Realizar un script llamado '**citas-menu.sh**' que sea una interfaz del script 'citas' mostrando 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.
```

07 - Bonus

39. Ver y entender los scripts de https://github.com/asanzdiego/markdownslides, en particular:
    1. https://github.com/asanzdiego/markdownslides/blob/master/build.sh

40. 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

41. Ver y entender los scripts de https://github.com/asanzdiego/xmlprocessor/, en particular:
    1. https://github.com/asanzdiego/xmlprocessor/blob/master/xmlprocessor

