UNIDAD 5: SHELL SCRIPTING

Ejercicios básicos

B.1) Crear los directorios **tuia/edp/bin** en su home directory.

Posteriormente, crear un script en ese directorio que muestre por pantalla “hello world”. Agregar el directorio a la variable PATH para poder ejecutar el script como un comando corriente.

B.2) Crear un script que reciba exactamente dos argumentos, el primero corresponderá a un número de mes y el segundo a un año.
Deberá mostrar por pantalla el calendario del mes y año con el día destacado
Tip: analizar el comando cal
Nota: evitaremos por ahora la validación de la cantidad y valor de los argumentos enviados

B.3) Crear un script que reciba una cantidad de argumentos sin definir e indique si recibió una cantidad par o impar de argumentos. Emplear construcciones de test (corchetes [ ]) y operadores lógicos (&& por ejemplo).

B.4) Crear un script que reciba una cantidad de argumentos (no limitado) y los escriba (todos juntos) en un archivo llamado /tmp/args. Finalmente mostrar el archivo.

B.5) Crear un script que reciba como argumento tres números, indicando día, mes y año y determine si estos valores corresponden o no a una fecha válida.
Emplear construcciones de test (corchetes [ ]) y operadores lógicos (&& por ejemplo).
Fechas inválidas: son las que posean números no positivos y/o demasiado grandes para lo que representan, o bien no corresponden los días al mes indicado para el año establecido.
Tip: puede ser de ayuda el comando ***date***

Ejercicios con condicionales

C.1) Crear un script que reciba la ruta absoluta a un archivo e indique si existe o no y si se trata de un archivo regular o un directorio.

C.2) Crear un script que reciba una única cadena como argumento y determine si esa cadena es o no un palíndromo.
Tip: utilizar el comando rev

C.3) Crear un script que reciba dos rutas a archivos de texto y que indique si estos archivos tienen la misma cantidad de líneas. Validar que los archivos existen y se pueden leer.

C.4) Crear un script para determinar si un año es bisiesto mediante un if con una condición que responda a esto.

Un año es bisiesto si es:

- Divisible entre 4.
- No divisible entre 100.
- Pero si es divisible por 100 debe serlo entre 400
  (2000 y 2400 son **bisiestos** pues aún siendo divisibles entre 100 lo son también entre 400. Pero los **años** 1900, 2100, 2200 y 2300 no lo son porque sólo son divisibles entre 100).

C.5) Crear un programa que permita recibir el nombre completo de un mes del año (M) en inglés y muestre por pantalla el mensaje “El mes M tiene X días”. Emplear `case` para su resolución.
Nota: ignorar el caso de año bisiesto.

Ejercicios con construcciones iterativas

I.1) Crear un script para mostrar por pantalla los números impares del 1 al **n**.
n es un argumento recibido por la línea de comandos y debe ser un entero positivo.

I.2) Crear un script para mostrar por pantalla los números pares del **n** al 2.
n es un argumento recibido por la línea de comandos y debe ser un entero mayor a 2.

I.3) Crear un script que reciba como argumentos una cantidad de palabras.
El script debe determinar cuántas palabras son y cuántas de ellas tienen al menos 3 caracteres.
En caso de no suministrarse argumentos, deberá indicar que sin argumentos el script no puede generar resultados.

I.4) Crear un script que permita el ingreso de una serie de números (consideremos que la entrada es adecuada, que la persona que emplea el script ingresa solo números) y determinar cuántos de ellos son:

- pares
- positivos
- que tengan 3 cifras

I.5) Crear un script que permita recibir tres argumentos: una palabra **P**, un número positivo **n** y una ruta. Se debe validar la cantidad de argumentos, que **n** cumpla los requerimientos y se tenga permiso de escritura en la ruta indicada.

Deberá generarse con esta información un archivo llamado “palabra_**P**.dat” en la ruta especificada, que contenga **n** líneas numeradas y que cada línea el texto indique
“Se ingresó la palabra **P**”.

I.6) Crear un script que reciba como argumento un directorio, deberá validarse que exista y se tenga permisos para leerlo.
Luego debe mostrar las siguientes estadísticas acerca de su contenido:

- cantidad de archivos regulares
- cantidad de directorios
- cantidad de líneas de los archivos de texto

Ejercicios con Expresiones Regulares

E.1) Crear un script que indique cuántos archivos regulares con el nombre (exacto) README existen en el directorio /usr/share/doc.

E.2) Crear un script que reciba un directorio como entrada y determine cuántos archivos dentro del mismo poseen extensiones .sh o .txt y si comienzan su nombre con dos letras minúsculas.

E.3) Crear un script donde se ingrese una serie de números enteros y se devuelva la suma de los mismos, se permitirá el ingreso de un máximo de 10 números y solo se deben sumar los argumentos válidos (los que representen un número).

Tips: utilizar shift y esta expresión regular

```
ES_NUMERO='^-?[0-9]+$'
```

E.4) Construya un script que reciba una cadena, el script debe validar si la cadena ingresada es una dirección IP válida.
Ejemplo de dirección IP válida: 192.168.0.1 (son cuatro enteros entre 0 y 255 separados por punto)

Ejercicios varios

V.1) Comando tac
a) Interpretar la página de manual del comando *tac*, realizar pruebas con el comando.

b) Crear un script que imprima el contenido de un archivo de texto de manera invertida, es decir primero la última línea, luego la penúltima y así sucesivamente hasta imprimir en último lugar la primer línea del archivo, pero SIN usar el comando tac.

V.2) Día de cumpleaños y días vividos
a) Deberá ingresarse en formato dd-mm-aaaa (guión incluido) la fecha en que nació y el script retornará el día de la semana de dicha fecha.
Nota: La fecha introducida debe ser válida, el script verificará esto, en caso de fecha inválida el script aborta comentando dicha situación. Puede ser útil el comando date.

b) Días vividos
Deberá ingresarse en formato dd-mm-aaaa la fecha en que nació, el script deberá retornar la cantidad de días transcurridos hasta la fecha.
Nota: La fecha introducida debe ser válida, en caso contrario el programa aborta.

V.3) Verificar passwords
a) Crear un script que tome como entrada dos archivos de texto *usuarios.txt* y *claves.txt*, ambos deberán contener una única palabra por línea. El primero contendrá nombres de usuario, mientras que el segundo claves de acceso en formato plano. Ninguno puede ser vacío y deben contener la misma cantidad de líneas (ambas condiciones deben ser validadas por el script).
El script deberá generar un nuevo archivo *credenciales.txt* que contendrá registros de la forma usuario:clave, donde usuario proviene del archivo *usuarios.txt* y clave del archivo *claves.txt*.

b) Crear un script que emule el proceso de login de un usuario, solicitando nombre de usuario y luego la clave, la clave introducida no deberá mostrarse por pantalla mientas de tipea (tip: opción de *read*) y debe cotejarse con la clave correspondiente al usuario en el archivo *credenciales.txt* creado en el apartado a).
Indicar claramente si el usuario no está registrado o si la clave ingresada no corresponde.

V.4) Front-end para el comando *chmod*
Crear un script que permita cambiar permisos a un archivo de forma interactiva. Es decir, permitir que la persona vaya indicando qué permisos y para quien quiere conceder o revocar. Posteriormente aplicar estas acciones al archivo o directorio indicado.

Consideraciones
. Validar que el archivo o directorio existe y se tengan derechos suficientes para aplicar los cambios previo a la aplicación de los mismos.
. Permitir la posibilidad de aplicar cambios recursivamente

V.5) Front-end para el comando *find*
Mediante las man pages u otro recurso investigar el funcionamiento del comando find y crear una interfaz de usuario que permita buscar archivos según los siguientes criterios:

- Tipo (directorio, archivo, pipe con nombre, symbolic link)
- Tamaño (bytes, kbytes, megabytes)
- Permisos (lectura, escritura, ejecución, combinaciones de los anteriores)
- Por número de inodo

Ejemplo:
`$ bash my_find.sh`
deberá mostrar un menú similar al siguiente

```
Interfaz para comando find
1 - Búsqueda por tipo de archivo
2 – Búsqueda por tamaño de archivo
3 – Búsqueda según los permisos del archivo
4 – Búsqueda por inodo
Introduzca criterio de búsqueda:
```

En este punto se solicita la ruta desde donde se comenzará la búsqueda y las entradas correspondientes a la selección realizada.

Comentarios:

Para la interfaz de usuario utilizar la estructura de control select
Utilizar case para determinar y tratar cada criterio de búsqueda
Considere el uso de funciones para reducir código