# Trabajo Práctico N°8

## Scripts del shell

**Objetivos:** Desarrollar habilidades para escribir y estructurar scripts en Bash, aplicando conceptos como manejo de variables, operadores, control de flujo, funciones y gestión de archivos.
Recuerda utilizar el comando **`man <comando>`**, o bien, **`<comando> -- help`**, para obtener ayuda de los comandos.

1. Escribe un script llamado `hola_mundo.sh` que imprima "Hola, mundo" en la terminal.
   a) Escribe una línea que muestre cuál es el directorio de trabajo actual.
   b) Escribe otra línea que muestre el contenido de dicho directorio.
   c) Agrega un comentario que explique cada línea del script.

2. Crea un script llamado `variables.sh` que declare las siguientes variables locales: nombre, apellido, y edad.
   a) Muestra sus valores por pantalla.
   b) Modifica el script para mostrar el usuario actual, su UID y directorio de trabajo actual, usando las variables de entorno correspondientes.
   c) Agrega las líneas necesarias para acceder, a un directorio y luego, retornar al directorio anterior, con la variable de entorno correspondiente.

3. Escribe un script llamado `operaciones_basicas.sh` que declare dos variables numéricas, `a` y `b`.
   a) Realiza la suma, resta, multiplicación y división entre esas variables.
   b) Muestra los resultados en pantalla sin utilizar variables que almacenen los resultados.
   c) Declara dos variables más, `c` y `d` y realiza las operaciones:
      i) `(a+b)*(c-d)`.
      ii) `((a*b)+c)/d`.

4. Crea un script llamado `comparación.sh` que declare dos variables numéricas.
   a) Verifica si las variables son iguales, cuáles es mayor o menor.
   b) Controla si alguna de las variables es positiva o negativa y muestra un mensaje por pantalla.

5. Escribe un script llamado `cadenas.sh` que haga lo siguiente:
   a) Muestre en pantalla un saludo para el usuario actual.
   b) Presente en pantalla el texto literal `La variable de entorno usada es $USER`.
   c) Muestre el saludo en dos líneas, pero en un solo comando: en la primera línea el saludo del apartado **a)**, y en una segunda línea, muestre cuál es su directorio de trabajo por defecto.
   d) Repite el apartado anterior, pero muestra la ruta entre comillas.

6. Crea un script que, con un número entero, muestre un mensaje por pantalla:
   a) Si es positivo.
   b) Si lo es, si está entre 5 y 10. Si no, si está entre 1 y 5. Sino, si es mayor que 10.
   c) Si no es positivo, si es cero.
   d) Si es negativo, si está entre -5 y -1. Si no, si es menor que -5.

7. Escribe un script llamado `case.sh` que permita ingresar un carácter, y muestre por pantalla un mensaje según su valor.
   a) Si es una `a` o una `A`.
   b) Si es una `b`.
   c) Si es una `c`.
   d) Si es cualquier otro carácter.

8. Crea un script, llamado `for_bash.sh`, que muestre:
   a) Los elementos de un arreglo de cadenas: “Juan”, “Paco”, “Pedro”, “de la Mar”.
   b) Los elementos de un arreglo de números: 354, 443, 3128, 789, 802.
   c) Los N primeros números enteros.
   d) Además del tipo de datos del arreglo, ¿qué otra diferencia hay en el script?

9. Escribe un script llamado `while_bash.sh` que permita ingresar un carácter por teclado, lo muestre por pantalla y:
   a) Se repita el ingreso mientras sea un número.
   b) Si no es un número, muestre el mensaje que no es un número y termine el script.

10. Repite el script anterior, en un nuevo script llamado `until_bash.sh`, donde se repita el ingreso mientras el carácter no sea una letra.

11. Escribe un script, llamado `funciones.sh`, que implemente las siguientes funciones:
   a) Reciba un nombre y muestre por pantalla un saludo a ese nombre.
   b) Reciba dos números, los sume y devuelva el resultado.

12. Crea un script, llamado `entrada_teclado.sh`, que permita ingresar datos por teclado:
   a) Pida al usuario su nombre y apellido, y los muestre por pantalla.
   b) Ingresa los mismos datos, pero en un solo comando.
   c) Repite lo anterior, pero el mensaje que pide el dato incluido en el comando de ingreso.

13. Escribe un script, llamado `archivos1.sh`, que realice las siguientes tareas:
   a) Pida al usuario su nombre y la ruta de un archivo (que no exista).
   b) Verifique la existencia del archivo. Si no existe, pregunte al usuario si quiere crearlo.
   c) Si el archivo existe, verifique que no está vacío y que el usuario tiene permiso de lectura y escritura.
   d) Luego, que pida al usuario ingresar una contraseña, que no debe ser visible al ingresarla, y escribir en el archivo una línea con el texto: `<nombre>; <contraseña>`.
   e) Pregunte al usuario si quiere repetir todas estas tareas, y hacerlo mientras responda que sí.
   f) Probar el script:
      i) Con un archivo que no existe y que el script lo cree.
      ii) Con el archivo creado.

14. Descarga los archivos auxiliares para el práctico.
   a) Analiza el script `create_resources.sh`, ¿qué realiza? Ejecuta el script en el directorio de trabajo por defecto del usuario actual.

**b)** Con el archivo `app_events.log`:

**i)** Escribe un script que reciba como argumento el nombre de un usuario y extraiga todas las líneas asociadas a ese usuario. Los resultados deben guardarse en un archivo llamado `eventos_<usuario>.log`. El script debe verificar que el usuario existe en la carpeta `resources`. Si el usuario no existe, debe mostrar un mensaje de error.

**ii)** Escribe un script que lea el archivo y genere un resumen de cuántos eventos hay por cada tipo de error. La salida debe ser algo como:

```
102: 15 errores
456: 20 errores
...
```

Guarda el resumen en un archivo llamado `resumen_errores.log`.

**iii)** Escribe un script que lea el archivo y, para cada evento, copie el archivo mencionado desde su subdirectorio en `resources` a una carpeta llamada `processed_files`, que debe ser creada automáticamente. Si el archivo no existe en `resources`, el script debe registrar el evento fallido en un archivo llamado `errores_movimiento.log`. La carpeta `processed_files` debe quedar organizada con subdirectorios por usuario.

**iv)** Escribe un script que genere un informe por cada usuario presente en la carpeta `resources`. Cada informe debe contener:

1. El número total de eventos del usuario en `app_events.log`.
2. Los archivos del usuario que están relacionados con esos eventos.
3. Los tipos de errores más comunes en sus eventos.

Los informes deben guardarse en un archivo `informe_<usuario>.log` en el subdirectorio del usuario.

**v)** Escribe un script que determine cuál es el archivo que aparece más veces en eventos con errores graves (definidos como errores 456 o 963). La salida debe ser algo como:

```
Archivo con más errores graves: richard3.txt (25 veces)
```

El resultado debe mostrarse en pantalla y guardarse en un archivo llamado `archivos_errores_graves.log`.