ACTIVIDAD PRÁCTICA: BASH SCRIPTING

Introducción:

Este conjunto de ejercicios está diseñado para que los estudiantes se familiaricen con la escritura de scripts en Bash, el lenguaje de comandos utilizado en sistemas Linux. Cada sección está organizada por tema, con una explicación de los objetivos pedagógicos y una serie de ejercicios con su respectiva solución.

Sección 1: Creación de scripts y comandos básicos

Objetivo: Esta primer sección se enfoca en la introducción a la escritura de scripts en Bash y la ejecución de comandos fundamentales. Los ejercicios buscan que el estudiante se familiarice con la estructura básica de un script, cómo imprimir mensajes en la pantalla, listar el contenido de un directorio y realizar operaciones simples de manipulación de archivos, como crear un directorio y copiar archivos.

Ejercicio 1.1: Crear un script llamado saludo.sh que muestre en pantalla el mensaje "Y si, es nuestro primer programa".

```
#!/bin/bash
echo " Y si, es nuestro primer programa"
```

Ejercicio 1.2: Escribir un script que liste todos los archivos y directorios del directorio actual en formato largo.

```
#!/bin/bash
ls -l
```

Ejercicio 1.3: Crear un script que cree un directorio llamado backup y copie en él todos los archivos .txt del directorio actual.

```
#!/bin/bash
mkdir -p backup
cp *.txt backup/
```

Sección 2: Variables y operadores aritméticos

Objetivo: Los ejercicios están diseñados para que el estudiante aprenda a declarar variables, asignarles valores, realizar cálculos matemáticos y mostrar los resultados en la pantalla. Se introduce el uso de la expansión aritmética y, opcionalmente, el uso de bc para cálculos de mayor precisión.

Ejercicio 2.1: Crea un script que defina dos variables numéricas y muestre la suma, resta, multiplicación y división entre ellas.

```
#!/bin/bash
a=10
b=5
echo "Suma: $((a + b))"
echo "Resta: $((a - b))"
echo "Multiplicación: $((a * b))"
echo "División: $((a / b))"
```

Ejercicio 2.2: Escribe un script que calcule el área de un rectángulo a partir de dos variables (base y altura).

```
#!/bin/bash
base=8
altura=4
area=$((base * altura))
echo "Área: $area"
```

Ejercicio 2.3: Define tres variables: nombre, edad y ciudad, y luego imprime un mensaje con esos datos concatenados en una oración.

```
#!/bin/bash
nombre="Ana"
edad=25
ciudad="Córdoba"
echo "$nombre tiene $edad años y vive en $ciudad."
```

Sección 3: Condicionales

Objetivo: Esta sección se centra en la implementación de la lógica condicional en los scripts de Bash, utilizando las estructuras if, then, else y elif. El objetivo es que el estudiante aprenda a escribir scripts que tomen decisiones basadas en diferentes condiciones, como la edad del usuario, la existencia de un archivo o el valor de una nota.

Ejercicio 3.1: Escribir un script que pida al usuario ingresar su edad y muestre si es mayor o menor de edad.

```
#!/bin/bash
read -p "Ingresa tu edad: " edad
if [[ $edad -ge 18 ]]; then
  echo "Eres mayor de edad"
else
  echo "Eres menor de edad"
fi
```

Sección 4: Bucles

Objetivo: Se introducen las estructuras de control de bucles: for, while y until. Los ejercicios buscan que el estudiante aprenda a repetir una serie de comandos múltiples veces, ya sea un número fijo de veces o hasta que se cumpla una determinada condición. Se practican la iteración sobre rangos de números, la acumulación de valores y la creación de bucles que se ejecutan hasta que el usuario ingresa una contraseña específica.

Ejercicio 4.1: Crear un script que imprima los números del 1 al 10 usando un bucle for.

```
#!/bin/bash
for i in {1..10}; do
  echo "Número: $i"
done
```

Ejercicio 4.2: Escribir un script que sume todos los números del 1 al 100 utilizando un bucle while.

```
#!/bin/bash
suma=0
contador=1
while [[ $contador -le 100 ]]; do
  suma=$((suma + contador))
  contador=$((contador + 1))
done
echo "Suma total: $suma"
```

Ejercicio 4.3: Crear un script que pida al usuario ingresar contraseñas hasta que escriba "secreto", usando un bucle until.

```
#!/bin/bash
clave="secreto"
input=""
until [[ $input == $clave ]]; do
  read -p "Ingresa la contraseña: " input
done
echo "Contraseña correcta."
```

Sección 5: Entrada del usuario

Objetivo: El foco está en cómo interactuar con el usuario, solicitando y leyendo datos desde la entrada estándar. Los ejercicios cubren la lectura de nombres, palabras y contraseñas, incluyendo el manejo de contraseñas ocultas y la verificación de la coincidencia entre ellas.

Ejercicio 5.1: Crear un script interactivo que solicite nombre y apellido

```
#!/bin/bash
read -p "Nombre: " nombre
read -p "Apellido: " apellido
echo $nombre  $apellido
```

Ejercicio 5.2: Crear un script que solicite al usuario una contraseña oculta con “read -s” y luego confirme su ingreso con un mensaje

```
#!/bin/bash
read -s -p "Contraseña: " clave
echo Contraseña guardada.
echo Acaba de tipear $clave
```

Sección 6: Scripts combinando conceptos

Objetivo: se proponen ejercicios que integran varios de los conceptos aprendidos en las secciones anteriores. Esto permite al estudiante aplicar un enfoque más completo y complejo en la resolución de problemas, combinando la lógica condicional, los bucles, la entrada del usuario y la manipulación de cadenas para crear scripts más útiles y funcionales.

Ejercicio 6.1: Escribir un script que solicite al usuario su nombre y edad, y luego le diga si puede votar (mayor de 16).

```
#!/bin/bash
read -p "Nombre: " nombre
read -p "Edad: " edad
if [[ $edad -ge 16 ]]; then
  echo "$nombre puede votar."
else
  echo "$nombre no puede votar."
fi
```

Ejercicio 6.2: Crear un script que lea una lista de nombres desde un archivo de texto (nombres.txt) e imprima un saludo personalizado para cada uno.

```
#!/bin/bash
while read nombre; do
  echo "Hola, $nombre!"
done < nombres.txt
```

Ejercicio 6.3: Generar codigo bash para calcular el promedio entre 5 números utilizando el bucle for.

```
#!/bin/bash
# Script para calcular el promedio de 5 números
suma=0
echo "Ingresa 5 números:"
for i in {1..5}
do
  read -p "Número $i: " numero
  suma=$((suma + numero))
done
promedio=$((suma / 5))
echo "El promedio de los 5 números es: $promedio"
```

Anexo - Ayudas para la resolución

Sección 1: Creación de scripts y comandos básicos

Datos relevantes:

- **Estructura básica de un script:** Todo script de Bash generalmente comienza con #!/bin/bash. Esta línea le indica al sistema que ejecute el script con el intérprete Bash.
- **echo:** Este comando se utiliza para mostrar texto en la pantalla. El texto que quieres mostrar va entre comillas dobles (") o simples ('). La diferencia principal es que dentro de las comillas dobles se pueden expandir variables, mientras que con las simples se toman literalmente.
- **ls:** Este comando lista los archivos y directorios en el directorio actual.
    - La opción -l muestra la información en formato largo, incluyendo permisos, propietario, tamaño, etc.
- **mkdir:** Este comando crea un nuevo directorio.
    - La opción -p crea los directorios padre necesarios si no existen.
- **cp:** Este comando copia archivos y directorios.
    - *.txt es un comodín que representa todos los archivos que terminan con la extensión .txt en el directorio actual.
    - backup/ indica el directorio de destino.

Sección 2: Variables y operadores aritméticos

Datos relevantes:

- **Variables:** En Bash, se asignan valores a las variables simplemente escribiendo el nombre de la variable seguido de un signo igual (=) y el valor. No se necesita declarar el tipo de variable.
    - Ejemplo: nombre="Juan"
- **Expansión de variables:** Para usar el valor de una variable, se debe preceder su nombre con un signo de dólar ($).
    - Ejemplo: echo "Hola, $nombre"

- **Expansión aritmética:** Para realizar operaciones matemáticas, se utiliza la sintaxis $((expresión)).
  - Ejemplo: suma=$((a + b))
- **bc (opcional para mayor precisión):** Es una calculadora de precisión arbitraria. Si necesitas cálculos más complejos o con decimales, puedes usar bc.
  - Ejemplo: resultado=$(echo "10 / 3" | bc)

Sección 3: Condicionales

Datos relevantes:

- **read:** Este comando se utiliza para leer la entrada del usuario desde el teclado.
  - La opción -p permite mostrar un mensaje (prompt) antes de la lectura.
  - Ejemplo: read -p "Ingresa tu nombre: " nombre
- **Estructura if:**

```
Bash
if [[ condición ]]; then
   # Comandos a ejecutar si la condición es verdadera
elif [[ otra_condición ]]; then # Opcional
   # Comandos a ejecutar si otra_condición es verdadera
else # Opcional
   # Comandos a ejecutar si ninguna condición anterior es verdadera
fi
```

- **Operadores de comparación numérica:**
  - -eq: igual a
  - -ne: no igual a
  - -gt: mayor que
  - -ge: mayor o igual que
  - -lt: menor que
  - -le: menor o igual que
- **Operador de verificación de archivos:**
  - -e archivo: True si el archivo existe.

Sección 4: Bucles

Objetivo: Aprender a repetir comandos múltiples veces con for, while y until.

Datos relevantes:

- **Bucle for:** Se utiliza para iterar sobre una secuencia de elementos.
    - **Iteración sobre un rango:** for i in {inicio..fin}; do ... done
        - **Iteración sobre una lista:** for elemento in item1 item2 item3; do ... done
- **Bucle while:** Se ejecuta mientras una condición sea verdadera.
  while [[condición]]; do
  # Comandos a ejecutar
  done
- **Bucle until:** Se ejecuta hasta que una condición sea verdadera (es decir, mientras la condición sea falsa).
  until [[condición]]; do
  # Comandos a ejecutar
  done

Sección 5: Entrada del usuario

Datos relevantes:

- **read (ya explicado):** Lee la entrada del usuario.
- **read -s:** Lee la entrada del usuario de forma silenciosa (sin mostrar los caracteres en la pantalla), útil para contraseñas.

Sección 6: Scripts combinando conceptos

Datos relevantes:

- **Reutilización de comandos y estructuras:** Combina comandos como read, if/else, for, while y manipulación de cadenas para lograr la funcionalidad deseada.
- **Lectura desde archivos:** El operador < se utiliza para redirigir la entrada estándar de un comando desde un archivo.
    - Ejemplo: while read linea; do ... done < archivo.txt lee cada línea del archivo archivo.txt y la asigna a la variable linea.