Chuleta básica de Bash (Linux)

Variables en Bash

En Bash, las variables se utilizan para almacenar valores. Aquí tienes cómo declarar y acceder a variables en Bash:

```
# Declaración de variables
mi_variable="Hola, mundo"

# Acceso a variables
echo $mi_variable
```

En el ejemplo anterior, hemos declarado una variable llamada mi_variable y la hemos inicializado con el valor "Hola, mundo". Luego, utilizamos echo para mostrar el contenido de la variable en la consola.

Bucles en Bash

Los bucles son útiles para repetir tareas en Bash. Aquí tienes ejemplos de bucles for y while:

Bucle for para iterar sobre una lista de elementos

```
frutas=("manzana" "pera" "uva")

for fruta in "${frutas[@]}"
do
    echo "Fruta: $fruta"
done
```

En el bucle for anterior, hemos definido una lista de frutas y luego hemos recorrido cada elemento de la lista utilizando la variable fruta.

Bucle while para contar hasta 5

```
contador=1
while [ $contador -le 5 ]
do
    echo "Contador: $contador"
    ((contador++))
done
```

El bucle while se ejecuta mientras se cumple una condición. En este caso, aumentamos el contador en cada iteración y salimos cuando alcanzamos el número 5.

Condicionales en Bash

Los condicionales permiten tomar decisiones en función de ciertas condiciones. Aquí tienes un ejemplo de una estructura if en Bash:

```
numero=12

if [ $numero -ge 10 ]
then
    echo "El número es mayor o igual a 10"
else
    echo "El número es menor que 10"
fi
```

En el ejemplo anterior, hemos evaluado si la variable numero es mayor o igual a 10. Dependiendo de la condición, se muestra un mensaje apropiado.

Lectura de Archivos en Bash

En Bash, puedes leer y procesar archivos. Aquí tienes ejemplos de cómo leer archivos línea por línea y mostrar el contenido completo:

Leer un archivo línea por línea

```
while IFS= read -r linea
do
    echo "Línea leída: $linea"
done < archivo.txt
```

Utilizamos un bucle while y el comando read para leer cada línea del archivo archivo.txt y mostrarla en la consola.

Leer contenido completo de un archivo

```
cat archivo.txt
```

El comando cat se utiliza para mostrar el contenido completo de un archivo.

Funciones en Bash

Las funciones permiten reutilizar lógica y estructurar mejor los scripts.

```
saludar() {
    echo "Hola, $1"
}

saludar "Jesús"
```

En este ejemplo, $1 es el primer argumento pasado a la

función.

Parámetros del script (argumentos) y variables especiales

Cuando ejecutas un script, puedes pasarle argumentos. Bash ofrece variables especiales para manejarlos.

```
echo "Nombre del script: $0"
echo "Primer argumento: $1"
echo "Número de argumentos: $#"
echo "Todos los argumentos: $@"
```

Ejemplo de uso:

```
./mi_script.sh Juan 25
```

Entrada del usuario (read)

Para pedir datos al usuario desde la terminal:

```
read -p "Introduce tu nombre: " nombre
echo "Hola, $nombre"
```

Si quieres leer una contraseña sin mostrarla:

```
read -s -p "Contraseña: " pass
echo
```

Arrays en Bash (listas)

Tu ejemplo de frutas ya es un array. Aquí van operaciones

típicas:

```
frutas=("manzana" "pera" "uva")

echo "Primera: ${frutas[0]}"
echo "Total: ${#frutas[@]}"

frutas+=("naranja")   # añadir elemento
unset frutas[1]       # eliminar elemento en posición 1
```

Operadores y tests de condición

En Bash, las condiciones suelen escribirse con [ ] (test) o con [[ ]] (más flexible).

- Números: -eq -ne -lt -le -gt -ge
- Cadenas: = != -z (vacía), -n (no vacía)
- Archivos:
    - -f existe y es fichero
    - -d existe y es directorio
    - -r/-w/-x permisos lectura/escritura/ejecución

Ejemplo:

```
archivo="archivo.txt"

if [ -f "$archivo" ]; then
    echo "Existe el fichero $archivo"
else
    echo "No existe"
fi
```

Redirecciones y tuberías (entrada/salida)

```
echo "Hola" > salida.txt          # sobrescribe
echo "Otra línea" >> salida.txt   # añade

cat archivo.txt | grep "error"    # tubería (pipe)
```

Capturar la salida de un comando en una variable:

```
fecha=$(date)
echo "Fecha: $fecha"
```

Control básico de errores (exit codes)

Los comandos devuelven un código: 0 es éxito; distinto de 0 es error.

```
mkdir carpeta_prueba
if [ $? -eq 0 ]; then
    echo "Carpeta creada"
else
    echo "Error al crear carpeta"
fi
```

Más directo:

```
mkdir carpeta_prueba || echo "No se pudo crear"
```

Buenas prácticas rápidas

1. Citar variables para evitar problemas con espacios:

```
echo "$mi_variable"
```

2. Modo “estricto” (recomendable en scripts):

```
set -euo pipefail
```

3. Cabecera típica de script:

```
#!/usr/bin/env bash
set -euo pipefail
```

Depuración rápida (debug)

Para ver qué está ejecutando Bash:

```
bash -x mi_script.sh
```

O dentro del script:

```
set -x # activa debug
# …
set +x # desactiva debug
```