Shell scripting es un lenguaje de programación que se usa en la terminal Bash para automatizar tareas y crear scripts complejos en sistemas Linux o Unix

Cron es el administrador de tareas de linux para ejecutar tareas de forma programada

Vamos a poder programar dentro de la terminal, entonces podemos llamar a estos comandos de la terminal de forma programática, ¡Tremendo!

La extensión de un fichero no determina el fichero que es
Necesitamos permisos de ejecución para poder ejecutarlos
./nombredearchivo.sh o bash nombredearchivo.sh
Primera linea de nuestros documentos:
#!/bin/bash

echo “Hola mundo, estoy en el terminal”

para ejecutar

./ nombre.sh si da permiso denegado es porque no tiene permisos de ejecucion, los colores de los archivos ej verde es permisos de ejecucion, para eso utilizamos chmod u+x nombredeachivo.sh

ejecutamos
./ nombredearchivo.sh

#!/bin/bash

echo “hola mundo inicia programa”
ls -l
echo “fin del programa”

al ejecutar este archivo ejecutara la cadena de caracteres

Si hay valores que piden datos, para esto podemos ejecutar comando interactivos para que no nos pidan datos mientras los estamos ejecutando

otra forma de ejecutar es llamar a bash, seria
bash nombredelachivo.sh

ejemplos para ver componentes, si luego queremos aprender más debemos especializarnos en shellscripting

Componentes

Comandos
Variables
Comentarios
Constantes
Strings
Caracteres comodines ~
input de entrada

argumentos de entrada
operaciones matemáticas
control de flujo
funciones con parámetros y argumentos
Expansiones

bash Copy code

```bash
#!/bin/bash
nombre="Juan"
echo "Bienvenido, $nombre"
```

Ejemplo 2: condicionales

bash Copy code

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

```bash
#!/bin/bash
for i in {1..5}; do
    echo "Número: $i"
done
```

Ejemplo 4: iteraciones, while

```bash
#!/bin/bash
contador=0
while [ $contador -lt 5 ]; do
    echo "Contador: $contador"
    ((contador++))
done
```

Ejemplo 5: petición de valores con read

```bash
#!/bin/bash
echo "¿Cuál es tu nombre?"
read nombre
echo "Hola, $nombre"
```

Ejemplo 7: procesar archivos

bash Copy code

```bash
#!/bin/bash
procesar_archivos() {
    directorio="$1"
    for archivo in "$directorio"/*; do
        # Realizar operaciones en cada archivo
        echo "Procesando $archivo"
    done
}

procesar_archivos "/ruta/a/procesar"
```

Ejemplo 8: procesar archivos

bash Copy code

```bash
#!/bin/bash
agregar_usuario() {
    usuario="$1"
    contraseña="$2"
    useradd "$usuario"
    echo "$usuario:$contraseña" | chpasswd
}

agregar_usuario "nuevo_usuario" "contraseña123"
```

Ejemplo 9: empaquetado

bash — Copy code

```bash
#!/bin/bash
respaldar_archivos() {
    directorio_origen="$1"
    directorio_destino="$2"
    fecha=$(date +"%Y%m%d")
    tar -czf "$directorio_destino/respaldo_$fecha.tar.gz" "$directori
}

respaldar_archivos "/ruta/a/respaldar" "/ruta/donde/guardar"
```

Ejemplo 10: sincronización y backup

bash — Copy code

```bash
#!/bin/bash

# Directorios fuente y destino
directorio_fuente="/ruta/del/directorio/fuente"
directorio_destino="/ruta/del/directorio/destino"

# Ejecutar rsync para sincronizar los directorios
rsync -av --delete "$directorio_fuente" "$directorio_destino"

# Verificar el código de salida de rsync
if [ $? -eq 0 ]; then
    echo "Sincronización completada con éxito"
else
    echo "Error durante la sincronización"
fi
```

- rsync -av --delete "$directorio_fuente" "$directorio_destino" realiza la sincronización, copiando los archivos del directorio fuente al directorio destino de manera recursiva (-a), mostrando el progreso (-v) y eliminando los archivos en el destino que no existen en el origen (--delete).
- El comando rsync devuelve un código de salida que indica si la operación fue exitosa ($?). Se verifica y se muestra un mensaje adecuado en consecuencia.

Cron es un servicio de programador de tareas, para ejecutar tareas, scripts, comando cuando nosotros queramos.

Comandos relacionados

- crontab -e: Permite editar el archivo crontab del usuario actual.
- crontab -l: Muestra el contenido del archivo crontab del usuario actual.
- crontab -r: Elimina el archivo crontab del usuario actual.
- /etc/crontab: El archivo crontab a nivel de sistema que define tareas para todos los usuarios.

Campos de crontab

Minuto (0-59): El primer campo especifica el minuto en el que se ejecutará la tarea. Puede ser un número entre 0 y 59.

- Hora (0-23): El segundo campo especifica la hora en la que se ejecutará la tarea. Puede ser un número entre 0 y 23, donde 0 representa la medianoche.
- Día del Mes (1-31): El tercer campo especifica el día del mes en el que se ejecutará la tarea. Puede ser un número entre 1 y 31, dependiendo del mes. Tenga en cuenta que no todos los meses tienen 31 días, por lo que si especifica un valor mayor que el número de días en un mes, la tarea no se ejecutará.