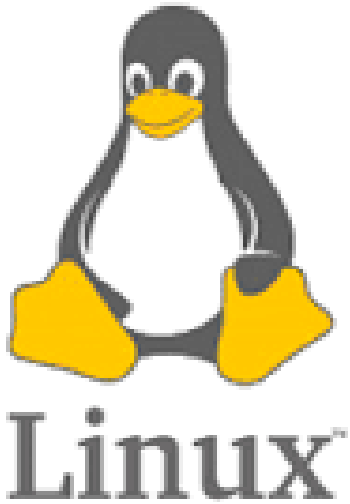
Proyecto de Automatización en Bash

MATEO MONTOYA Y MAXIMILIANO SILVA



Trabajo Final - Sistemas Operativos

Tecnicatura Superior en Desarrollo de Software

Instituto Superior IDRA

Problematica

En muchos casos, los usuarios **no cuentan con conocimientos técnicos** para mantener su sistema operativo en buen estado. Con el tiempo,los equipos comienzan a **acumular archivos innecesarios, falta de espacio en disco y errores por no realizar copias de seguridad**.
Además, la gestión manual de estas tareas suele ser **lenta, repetitiva y propensa a errores**, lo que afecta el rendimiento del sistema y la productividad del usuario.

backup.sh

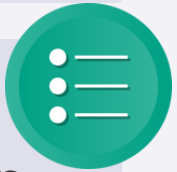
Crea una copia comprimida de seguridad del directorio elegido, con la fecha actual. También elimina los backups viejos para evitar acumulación de archivos.

limpiar_sistema.sh

Borra archivos temporales y carpetas de caché que ocupan espacio innecesario, mejorando el rendimiento general.

contador_archivos.sh

Analiza una carpeta y muestra cuántos archivos y subcarpetas contiene, ayudando al usuario a tener control de su organización.



menu.sh

Menú visual para acceder fácilmente a los otros scripts

Requisitos Técnicos del Proyecto

Sistema Operativo

El proyecto fue diseñado para ejecutarse en entornos basados en Linux o en Windows con WSL.

Es necesario contar con Git Bash o una terminal funcional que permita correr scripts en Bash.

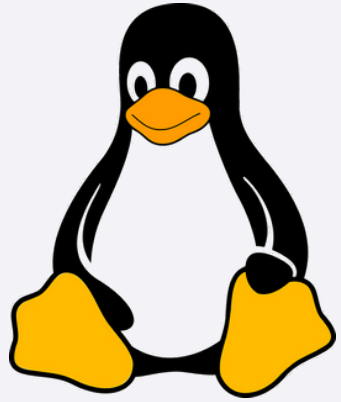
El sistema no requiere instalación adicional, solo acceso al terminal y permisos de usuario.

Permisos y Ejecución

Antes de ejecutar los scripts, es necesario otorgar permiso de ejecución a los archivos .sh con el comando: chmod +x *.sh

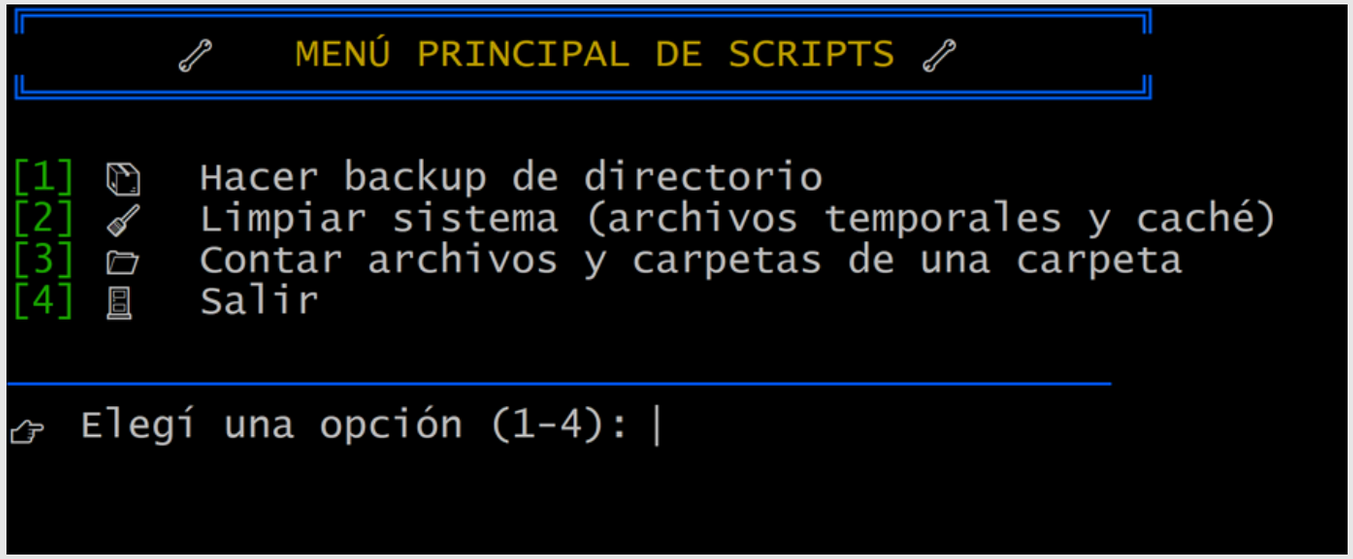
Luego, el programa principal se ejecuta mediante: ./menu.sh

Esto permite que el usuario navegue por el menú y utilice las distintas funciones sin errores de permisos.



Espacio para Código MENU

MENU ./menu.sh

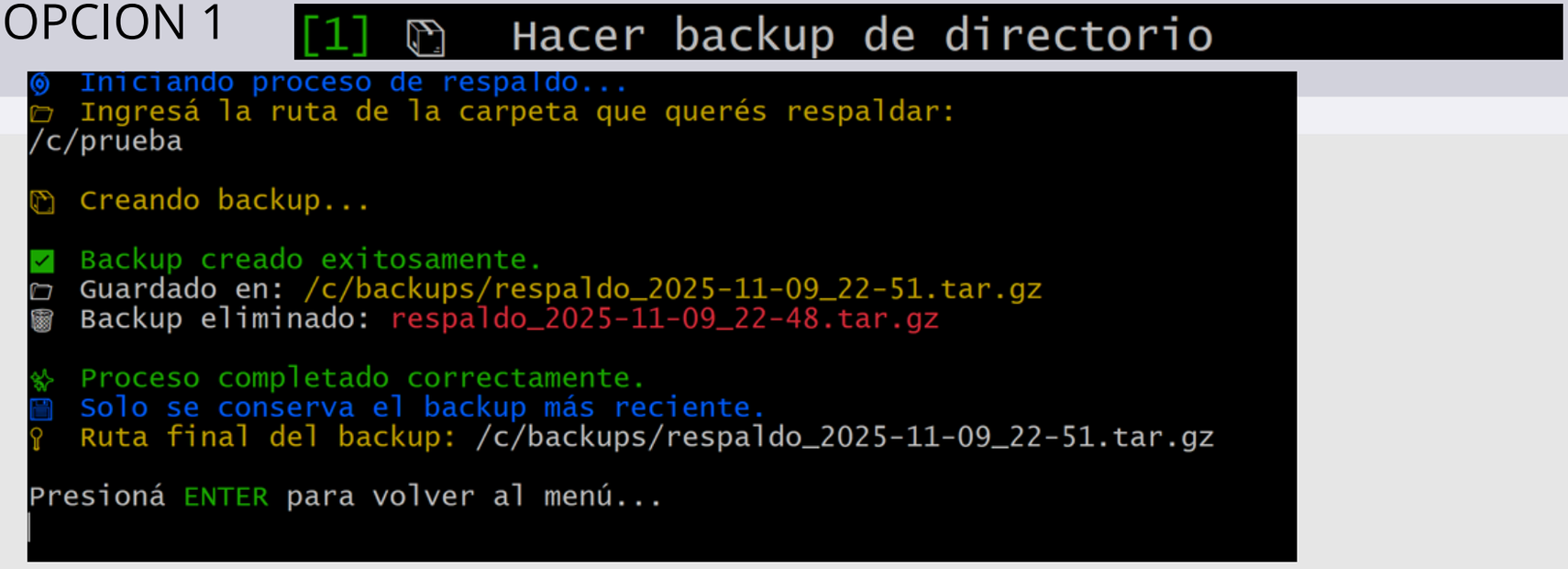


Al ejecutar el proyecto, aparece una pantalla de bienvenida con colores, un título grande que dice: “MENÚ PRINCIPAL DE SCRIPTS” y cuatro opciones numeradas.

El usuario puede elegir qué tarea quiere realizar escribiendo un número del 1 al 4.

Hacer Backup del Directorio

- [ ] TOCA OPCION 1



Paso a paso del script de backup

1. **Pantalla limpia y mensaje inicial**
   - Limpia la terminal y muestra:
   - "🌀 Iniciando proceso de respaldo..."
2. **Define la carpeta de destino**
   - Todos los backups se guardan en **/c/backups.**
   - Si la carpeta no existe, la crea automáticamente.

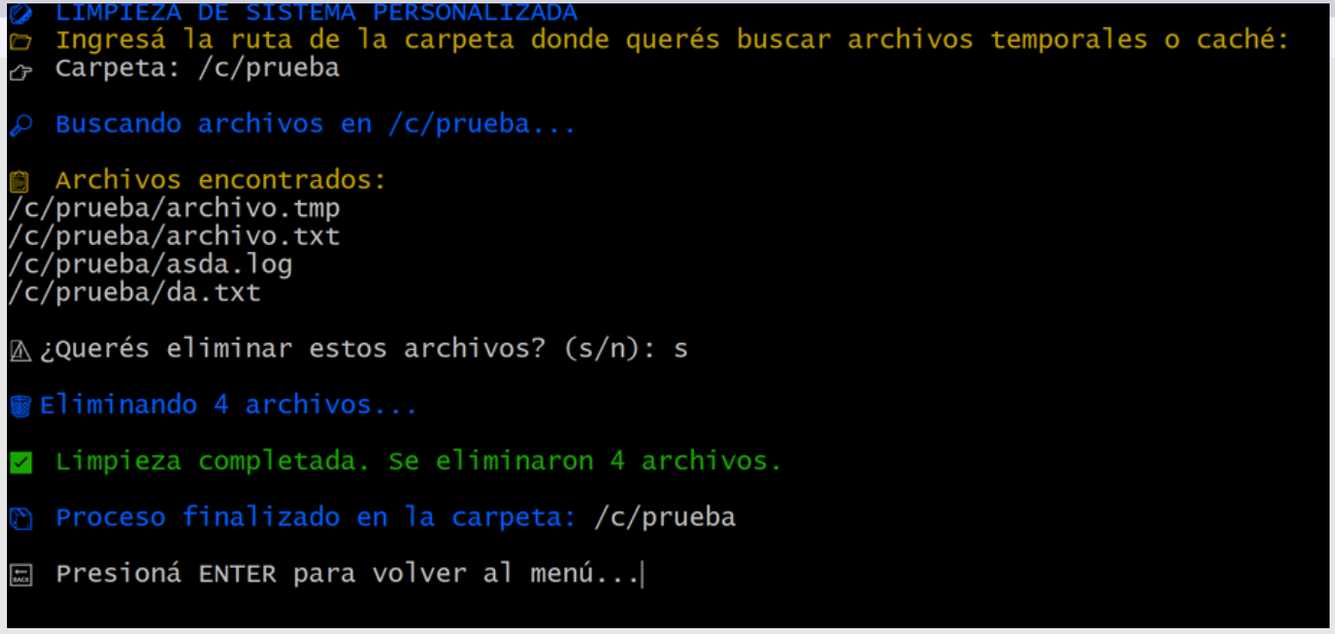
3. El usuario elige la carpeta a respaldar. Pone la carpeta que quiere

4. Una vez puesto si existe se guarda en la carpeta de BACKUPS con Fecha y Hora.

5. Si llega a ver un backup viejo se borra y queda el nuevo.

Limpiar SISTEMA

- TOCA OPCION 2 `[2] 🧹 Limpiar sistema (archivos temporales y caché)`



🧹 **Función: Limpiar sistema**

1. Inicio de la opción:
2. Cuando el usuario elige la opción “Limpiar sistema”, el programa explica que sirve para borrar archivos que ya no son necesarios, como los temporales o de caché.

3. Luego pide que el usuario escriba la dirección de la carpeta que quiere revisar.

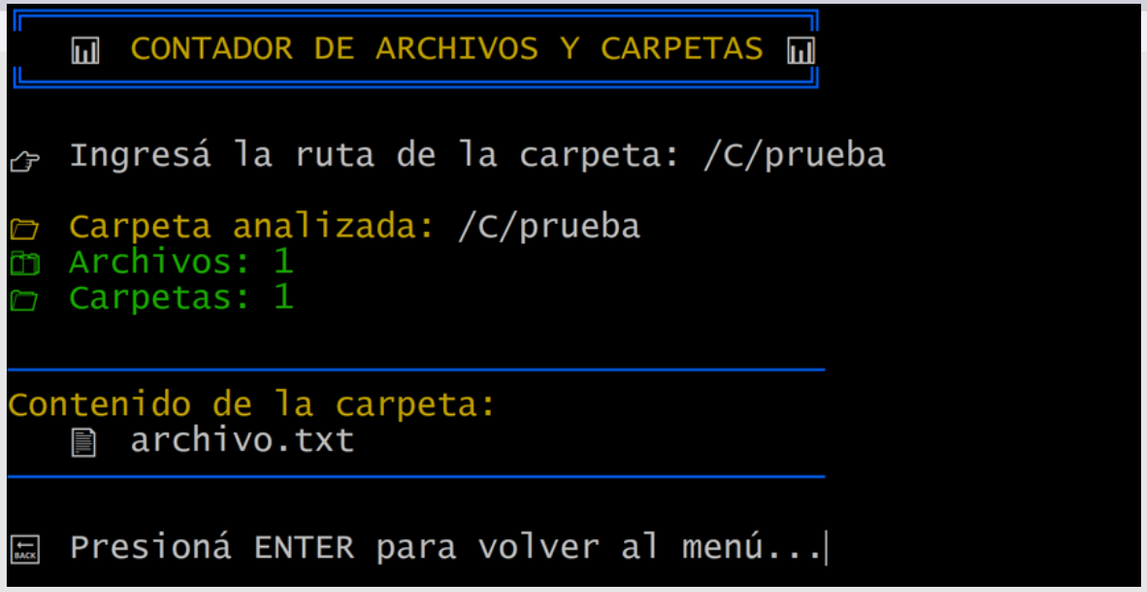
4. El programa busca dentro de esa carpeta y muestra todos los archivos con extensión .txt **(aunque podrían ser también .log, .tmp o . cache, que son tipos comunes de archivos temporales del sistema o de navegadores).**

5. Después pregunta si el usuario quiere borrarlos:
   - Si responde “s”, el programa elimina los archivos.
   - Si responde “n”, no borra nada y vuelve al menú.
6. Si se confirma la limpieza, muestra un mensaje diciendo que los archivos fueron eliminados correctamente

Contar Archivos y Carpetas

- TOCA OPCION 3

```
[3]   Contar archivos y carpetas de una carpeta
```

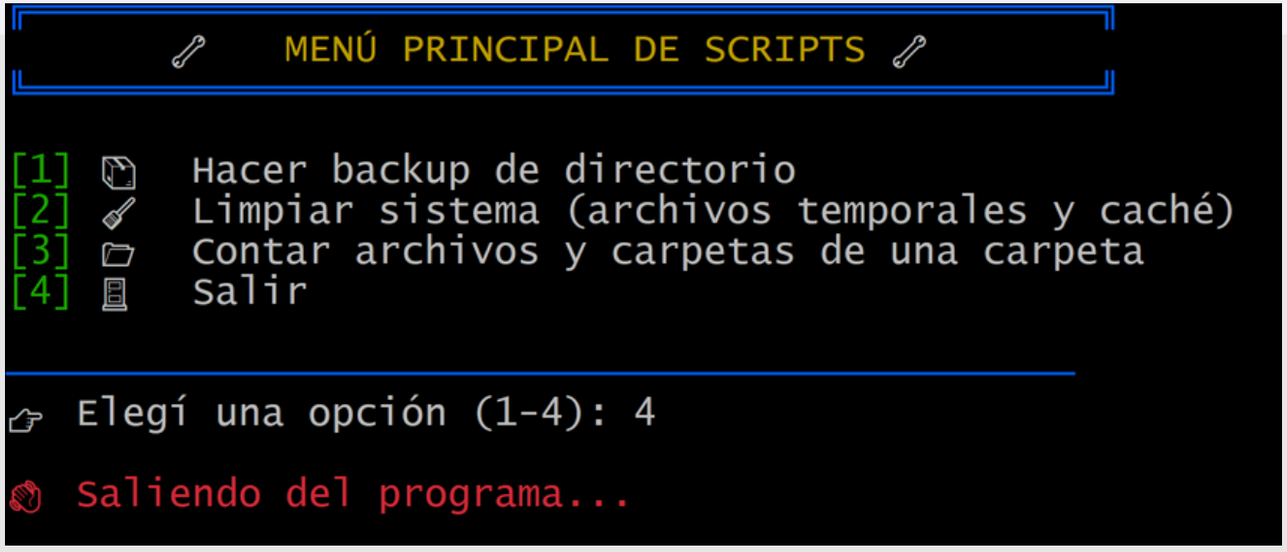


Opción 3 — Contar archivos y carpetas

1. Primero pide en pantalla: “Ingresá la ruta de la carpeta:” (ej.: /c/prueba).
2. Una vez ingresada la ruta, aparece un recuadro con:
   - “Carpeta analizada: /c/prueba”
   - “Archivos: N” (número total de archivos)
   - “Carpetas: M” (número total de subcarpetas)
   - (estos contadores aparecen destacados, con colores o íconos).
3. Debajo aparece un título “Contenido de la carpeta:” y la lista de los nombres que contiene, por ejemplo:
   - archivo.txt
   - imagenes/
4. Al final muestra: “Presioná ENTER para volver al menú...”.

SALIDA

- TOCA OPCION 4



Opción 4 — Salir

1. Muestra un mensaje de despedida: “👋 Saliendo del programa...”.
2. El programa espera 1 segundo y luego cierra la ejecución, volviendo al prompt de la terminal.

Desarrollo, explicación detallada del código.

MENU

Menú principal (menu.sh)

```
while true; do
  clear
```

- **while true** crea un bucle infinito, lo que significa que el menú se va a mostrar continuamente hasta que el usuario elija salir.
- **clear** limpia la terminal, dejando solo el menú visible.

```
  echo -e "${azul}╔═══════════════════════════════════════════════════════════════════════════════╗ ${reset}"
  echo -e "      🔧 ${amarillo}MENÚ PRINCIPAL DE SCRIPTS${reset} 🔧"
  echo -e "${azul}╚═══════════════════════════════════════════════════════════════════════════════╝ ${reset}"
```

- **Muestra un título decorado con colores y emojis.**
- **${azul} y ${amarillo} son variables de color definidas al inicio.**
- **${reset} vuelve al color normal después del mensaje.**

```
  echo ""echo -e "${verde}[1]${reset} 📦 Hacer backup de directorio"
  echo -e "${verde}[2]${reset} 🧹 Limpiar sistema (archivos temporales y caché)"
  echo -e "${verde}[3]${reset} 📂 Contar archivos y carpetas de una carpeta"
  echo -e "${verde}[4]${reset} 🚪 Salir"
```

- **Lista las opciones del menú con números, colores y emojis.**
- **Cada opción corresponde a un script diferente.**

```
  echo ""echo -e "${azul}──────────────────────────────────────────────────────────────────────────────${reset}"
  read -p "👉 Elegí una opción (1-4): " opcion
  echo ""
```

- **Separa visualmente las opciones.**
- **read -p captura la opción que escribe el usuario y la guarda en la variable opcion.**

```
  case $opcion in
    1) bash backup.sh ;;
    2) bash limpiar_sistema.sh ;;
    3) bash contador_archivos.sh ;;
    4) echo -e "${rojo}👋 Saliendo del programa...${reset}"; sleep 1; exit 0 ;;
    *) echo -e "${rojo}⚠️ Opción inválida, intentá de nuevo.${reset}"; sleep 2 ;;
  esac done
```

- **case:** selecciona qué hacer según la opción del usuario:
  - **1 → Ejecuta backup.sh.**
  - **2 → Ejecuta limpiar_sistema.sh.**
  - **3 → Ejecuta contador_archivos.sh.**
  - **4 → Muestra mensaje de salida, espera 1 segundo y cierra el programa.**
  - *** → Captura cualquier opción inválida y muestra un mensaje de error por 2 segundos.**
- **Resumen: Este menú permite acceder fácilmente a los diferentes scripts sin escribir rutas complicadas.**

Desarrollo, explicación detallada del código.

backup.sh

Backup (backup.sh)

```
clear
echo -e "${azul}🌀 Iniciando proceso de respaldo...${reset}"
sleep 1

destino="/c/backups"mkdir -p "$destino"

echo -e "${amarillo}📂 Ingresá la ruta de la carpeta que querés respaldar:${reset}"
read -e origen

if [ ! -d "$origen" ]; thenecho -e "\n${rojo}❌ La carpeta ingresada no existe.${reset}"
    exit 1
fi

fecha=$(date +%Y-%m-%d_%H-%M)
archivo="$destino/respaldo_$fecha.tar.gz"

echo -e "\n${amarillo}📦 Creando backup...${reset}"
sleep 1
tar -czf "$archivo" "$origen" &> /dev/null

if [ -f "$archivo" ]; thenecho -e "\n${verde}✅ Backup creado exitosamente.${reset}"
    echo -e "📂 Guardado en: ${amarillo}$archivo${reset}"
elseecho -e "\n${rojo}❌ Error: no se pudo crear el backup.${reset}"
    exit 1
ficd "$destino" || exitfor viejo in respaldo_*.tar.gz; doif [ "$viejo" != "$(basename "$archivo")" ]; thenrm -f "$viejo"
        echo -e "🗑️  Backup eliminado: ${rojo}$viejo${reset}"
    fidoneecho -e "\n${verde}✨ Proceso completado correctamente.${reset}"
echo -e "${azul}💾 Solo se conserva el backup más reciente.${reset}"
echo -e "${amarillo}📍 Ruta final del backup:${reset} $archivo"
echo -e "\nPresioná ${verde}ENTER${reset} para volver al menú..."
read
```

Explicación detallada:

- **Limpia pantalla y muestra mensaje de inicio con colores y emojis.**
- **Carpeta de destino fija: /c/backups. Si no existe, se crea con mkdir -p.**
- **read -e → usuario ingresa la carpeta a respaldar, con autocompletado.**
- **Verifica existencia de la carpeta; si no existe, termina con error.**
- **fecha → crea un timestamp para nombrar el archivo.**
- **tar -czf → comprime la carpeta en .tar.gz.**
- **Verifica si el archivo se creó correctamente.**
- **Elimina todos los backups antiguos, dejando solo el más reciente.**
- **Muestra mensaje final con la ruta del backup y espera ENTER para volver al menú.**

Desarrollo, explicación detallada del código.

LIMPIAR SISTEMA.sh

```
      Limpiar sistema (limpiar_sistema.sh)

clear
echo -e "${azul}🧼 LIMPIEZA DE SISTEMA PERSONALIZADA${reset}"
sleep 1

echo -e "${amarillo}📂 Ingresá la ruta de la carpeta donde querés buscar archivos temporales o caché:${reset}"
read -p "👉 Carpeta: " carpeta

if [ ! -d "$carpeta" ]; thenecho -e "${rojo}❌ Error: la carpeta no existe.${reset}"
   read -p "Presioná ENTER para salir..." _
   exit 1
fiecho -e "${azul}🔎 Buscando archivos en ${carpeta}...${reset}"
sleep 1

archivos=$(find "$carpeta" -type f \( -name "*.tmp" -o -name "*.log" -o -name "*.cache" -o -name "*.dat" -o -name "*.txt" \))

if [ -z "$archivos" ]; thenecho -e "${verde}✨ No se encontraron archivos temporales, de caché ni .txt.${reset}"
   read -p "🔙 Presioná ENTER para volver al menú..." _
   exit 0
fiecho -e "${amarillo}📋 Archivos encontrados:${reset}"
echo "$archivos"
read -p "⚠️ ¿Querés eliminar estos archivos? (s/n): " confirmar

if [[ "$confirmar" == "s" || "$confirmar" == "S" ]]; then
   cantidad=$(echo "$archivos" | wc -l)
   echo -e "${azul}🗑️ Eliminando $cantidad archivos...${reset}"
   find "$carpeta" -type f \( -name "*.tmp" -o -name "*.log" -o -name "*.cache" -o -name "*.dat" -o -name "*.txt" \) -delete
   echo -e "${verde}✅ Limpieza completada. Se eliminaron $cantidad archivos.${reset}"
elseecho -e "${rojo}❌ Limpieza cancelada por el usuario.${reset}"
fiecho -e "${azul}📦 Proceso finalizado en la carpeta:${reset} $carpeta"
read -p "🔙 Presioná ENTER para volver al menú..." _
```

Explicación detallada:

- **Limpia pantalla y muestra título con colores y emojis.**
- **Usuario ingresa la carpeta donde buscar archivos temporales y de caché.**
- **Verifica que la carpeta exista.**
- **find busca archivos con extensiones .tmp, .log, .cache, .dat, .txt.**
- **Muestra los archivos encontrados y pregunta si quiere eliminarlos.**
- **Si confirma, los elimina con find ... -delete.**
- **Muestra mensaje final y vuelve al menú.**

Desarrollo, explicación detallada del código.

CONTAR ARCHIVO Y CARPETAS

Contador de archivos y carpetas (contador_archivos.sh)

Pantalla inicial

```
clear
echo -e "CONTADOR DE ARCHIVOS Y CARPETAS"
```

- **Limpia pantalla y muestra el título.**

Seleccionar carpeta

```
read -p "👉 Ingresá la ruta de la carpeta: " carpeta
echo “ ”
```

- **Usuario ingresa carpeta a analizar.**

Verificar existencia

```
if [ ! -d "$carpeta" ]; then
   echo -e "${rojo}❌ La ruta no existe. Verificá e intentá de nuevo.${reset}"
   read -p "Presioná ENTER para salir..." _
   exit 1
fi
```

- **Comprueba que la carpeta exista.**

Contar archivos y carpetas

```
archivos=$(find "$carpeta" -type f | wc -l)
carpetas=$(find "$carpeta" -type d | wc -l)
```

- **archivos = cantidad de archivos en la carpeta.**
- **carpetas = cantidad de carpetas.**

Mostrar contenido

```
echo -e "${azul}————————————————————————————————————————————————————————————————————————————————${reset}"
echo -e "${amarillo}Contenido de la carpeta:${reset}"
for item in "$carpeta"/*; do
   if [ -f "$item" ]; then
      echo -e "  📄 $(basename "$item")"
   elif [ -d "$item" ]; then
      echo -e "  📁 $(basename "$item")"
   fi
done
```

- **Muestra todos los archivos y carpetas dentro de la ruta seleccionada.**

Mensaje final

```
read -p "🔙 Presioná ENTER para volver al menú..." _
```

- **Espera ENTER para regresar al menú principal.**

Reflexiones Finales

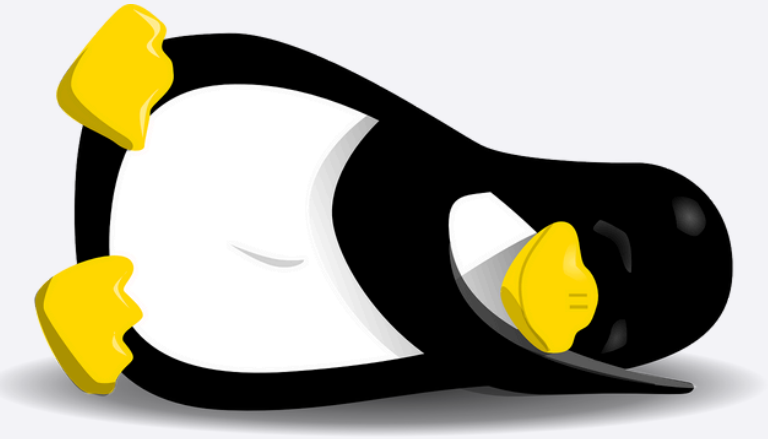
Aprendizajes Clave

- Estructura de scripts en Bash
- Automatización de tareas del sistema
- Diseño de interfaces de usuario en terminal

Dificultades Encontradas

La implementación de creación de usuarios y informe del sistema del CPU, RAM Y MEMORIA resultó compleja y generaba conflictos en sistemas Windows o entornos no administrativos.

Solución: Se reemplazó por tareas más estables y compatibles con cualquier entorno Linux.



Posibles Mejoras

Historial de backups:

Guardar un registro con las fechas de los backups realizados, el tamaño de cada uno y cuánto espacio se liberó al limpiar el sistema.

Interfaz Más Interactiva

Añadir confirmaciones de usuario, barras de progreso y mensajes de estado más descriptivos

Validación de Entradas

Mejorar la verificaciónde datos ingresados por el usuario para prevenir errores

Conclusión

El proyecto cumplió exitosamente con los objetivos planteados, desarrollando un conjunto de herramientas de automatización funcionales y compatibles con diversos entornos Linux.

Los scripts demuestran comprensión sólida de los conceptos de sistemas operativos y programación en Bash, ofreciendo

soluciones prácticas para tareas cotidianas de administración del sistema.
