



Memoria TÉCNICA

Instituto Superior IDRA

Tecnicatura Superior en Desarrollo de
Software

Materia: Sistemas Operativos

Trabajo Final Integrador

Integrantes: Soldo, Miranda, Baviello y
Mangioni

Docente: Lucas Treser

INTRODUCCIÓN

El presente trabajo final tiene como objetivo integrar y aplicar los conocimientos adquiridos durante la cursada de Sistemas Operativos, mediante el desarrollo de un proyecto práctico en Bash.

El proyecto consiste en un conjunto de scripts automatizados que ejecutan tareas de mantenimiento y administración del sistema.

Se buscó crear una herramienta funcional, modular y fácil de usar, que permita simplificar tareas rutinarias del administrador de sistemas.

Las tres tareas principales desarrolladas fueron:

1. Automatización de backups con retención de copias antiguas.
2. Generación de un reporte del sistema con información del estado actual.
3. Limpieza de archivos temporales y caché para liberar espacio en disco.

Además, se desarrolló un menú interactivo en Bash que permite ejecutar cada una de estas tareas de forma simple, con colores, validaciones y mensajes claros.

DESCRIPCIÓN GENERAL DEL PROYECTO

El proyecto se compone de **cuatro scripts principales**:

Archivo	Descripción
menu.sh	Script principal con menú interactivo para ejecutar las tareas.
backup.sh	Crea backups comprimidos y elimina copias antiguas.
reporte_sistema.sh	Genera un informe del sistema (disco, procesos, logs).
limpieza.sh	Elimina archivos temporales y caché del sistema.

- Los scripts comparten un archivo de configuración común llamado config.env, donde se definen las variables necesarias (rutas, días de retención, opciones de limpieza, etc.).
- Cada ejecución genera su propio archivo de log con información detallada, almacenado dentro de la carpeta logs/.
- El enfoque general fue mantener un código seguro, legible y portable, utilizando buenas prácticas de scripting, validación de entrada, mensajes de estado y manejo de errores controlado.

REQUISITOS TÉCNICOS Y HERRAMIENTAS UTILIZADAS

- Sistema operativo: Windows 10
- Entorno de ejecución: Git Bash (emulación de Bash para Windows)
- Lenguaje: Bash shell scripting
- Dependencias utilizadas: bash 4+, tar, gzip, df, ps, ls, head, awk. En entornos Linux, los scripts también utilizan free, top y uptime, pero en esta versión fueron reemplazados o adaptados para funcionar en Git Bash



DESARROLLO DEL PROYECTO

• **SCRIPT PRINCIPAL - MENU.SH**

Este script constituye la interfaz principal del proyecto.

Muestra un menú interactivo con colores y opciones numeradas, desde donde el usuario puede elegir qué tarea ejecutar.

Entre sus características más importantes:

- Valida dependencias mínimas antes de iniciar (tar, gzip, df, date, etc.).
- Verifica que la opción ingresada sea válida.
- Llama a los scripts correspondientes según la elección del usuario.
- Muestra mensajes de confirmación o error al finalizar cada tarea.
- Permite salir del programa de forma controlada.

El uso de colores (tput o secuencias ANSI) mejora la experiencia visual, destacando los estados de éxito, error o advertencia.

DESARROLLO DEL PROYECTO

• **BACKUP AUTOMÁTICO - BACKUP.SH**

Este script automatiza el respaldo de un directorio. Lee las variables desde el archivo config.env, incluyendo:

- BACKUP_SOURCE_DIR → ruta del directorio a respaldar.
- BACKUP_DEST_DIR → carpeta donde se guardarán los backups.
- BACKUP_RETENTION_DAYS → cantidad de días a conservar.
- LOG_DIR → ubicación de los logs.

El flujo de ejecución es el siguiente:

- 1.Verifica que el directorio de origen exista.
- 2.Genera un archivo comprimido .tar.gz con fecha y hora en el nombre.
- 3.Registra toda la actividad en un log con timestamp.
- 4.Busca y elimina automáticamente los backups que superen la antigüedad configurada.

De esta manera, el script mantiene un equilibrio entre seguridad y limpieza, evitando acumulación de copias innecesarias.

DESARROLLO DEL PROYECTO

• **REPORTE DEL SISTEMA - REPORTE_SISTEMA.SH**

El objetivo de este script es recopilar información general del sistema y almacenarla en un archivo de texto dentro del directorio de logs.

Datos incluidos:

- Fecha y hora de ejecución.
- Nombre del host y sistema operativo.
- Espacio en disco (df -h).
- Procesos activos (ps | head -n 10).
- Últimos logs de backups generados.

La salida se guarda en un archivo `reporte_<fecha>.txt`.

El script fue diseñado para funcionar tanto en Linux como en Git Bash para Windows, garantizando compatibilidad multiplataforma.

DESARROLLO DEL PROYECTO

- **LIMPIEZA DEL SISTEMA - LIMPIEZA.SH**
- Este script automatiza tareas de mantenimiento y limpieza, eliminando archivos innecesarios para liberar espacio.
- Entre las tareas implementadas:
 1. Eliminar archivos de /tmp y /var/tmp con más de 7 días.
 2. Limpiar la caché del usuario (~/.cache).
 - El comportamiento se controla mediante variables de configuración:
CLEAN_USER_CACHE=true
CLEAN_APT_CACHE=false
CLEAN_JOURNAL=false
- Cada acción se documenta en un log con mensajes como “inicio”, “directorio limpiado”, “saltado” o “requiere permisos”.

PRUEBAS Y VALIDACIONES

DURANTE LA ETAPA DE PRUEBAS, SE VERIFICÓ QUE CADA SCRIPT FUNCIONE DE FORMA INDEPENDIENTE Y TAMBIÉN DESDE EL MENÚ PRINCIPAL.

Nº	Prueba	Resultado esperado	Estado
1	Ejecutar menú principal	Mostrar opciones con colores	Paso
2	Elegir opción “1” (backup)	Crear archivo .tar.gz y log	Paso
3	Modificar BACKUP_SOURCE_DIR a ruta inválida	Mostrar error y salir con código 2	Paso
4	Ejecutar limpieza	Eliminar archivos viejos en /tmp y ~/.cache	Paso
5	Generar reporte	Crear archivo en logs/reportes_*.txt	Paso
6	Opción inválida en el menú	Mostrar mensaje de error	Paso
7	Ejecución en Git Bash (Windows)	Compatible y funcional	Paso

PRUEBAS Y VALIDACIONES

DURANTE LA ETAPA DE PRUEBAS, SE VERIFICÓ QUE CADA SCRIPT FUNCIONE DE FORMA INDEPENDIENTE Y TAMBIÉN DESDE EL MENÚ PRINCIPAL.

Nº	Prueba	Resultado esperado	Estado
1	Ejecutar menú principal	Mostrar opciones con colores	Paso
2	Elegir opción “1” (backup)	Crear archivo .tar.gz y log	Paso
3	Modificar BACKUP_SOURCE_DIR a ruta inválida	Mostrar error y salir con código 2	Paso
4	Ejecutar limpieza	Eliminar archivos viejos en /tmp y ~/.cache	Paso
5	Generar reporte	Crear archivo en logs/reportes_*.txt	Paso
6	Opción inválida en el menú	Mostrar mensaje de error	Paso
7	Ejecución en Git Bash (Windows)	Compatible y funcional	Paso

Reflexiones finales y mejoras futuras

- Durante el desarrollo se presentaron dificultades relacionadas con las diferencias entre entornos Linux y Git Bash, ya que algunos comandos (free, top, uptime) no existen en Windows. Para solucionarlo, se implementaron versiones simplificadas compatibles con Git Bash, manteniendo la lógica y funcionalidad esperadas

Posibles mejoras

- Añadir soporte multiplataforma automático (detectar si el entorno es Linux o Windows).
- Implementar notificaciones automáticas al completar el backup.
- Incorporar un sistema de restauración de backups desde el menú principal.

El proyecto permitió aplicar conceptos clave de administración del sistema, automatización de tareas y scripting en Bash, demostrando un entendimiento práctico de los fundamentos de los sistemas operativos