Documentación técnica bash script con funcionalidades varias

Por Federico Rodriguez

Índice del Documento

1. Introducción

El presente proyecto consiste en el desarrollo de un **script** en **Bash** (bash script) destinado a automatizar tareas de mantenimiento y gestión del sistema. Su objetivo es optimizar y simplificar tareas repetitivas, en este caso: limpieza de caché, respaldo de archivos y actualización del sistema. Como añadido a la propuesta ideada por el profesor, decidí hacer que el script funcione tanto en Linux como en Windows.

Tareas automatizadas:

- Detección del sistema operativo y adaptación de comandos.
- Limpieza de directorios temporales.
- Respaldo de archivos.
- Verificación e instalación de actualizaciones del sistema.

2. Requisitos Técnicos

Sistema operativo compatible:

- Linux o Windows (con Bash 4 o superior).
- Posibilidad de expandirse a macOS

Herramientas y dependencias:

- Bash.
- (En Windows) **GIT** o WSL (subsistema no testeado)
- Permisos de usuario con sudo.

Variables y rutas utilizadas:

- **$CACHE_DIR:** directorio de caché a limpiar.
- **$BACKUP_DIR:** destino de respaldos.
- **$SOURCE_DIR:** carpeta de origen de archivos.

3. Desarollo del script

3.1. Detección del sistema operativo

Esta parte identifica el sistema operativo en el que se ejecuta el script. Utiliza comandos estándar para determinar si se trata de Linux o Windows, adaptando el comportamiento según corresponda. En caso de detectar un sistema no compatible, el script finaliza su ejecución mostrando un mensaje informativo. Dependiendo de si detecta un sistema u otro, las variables de direcciones se encontrarán en lugares distintos dependiendo del sistema de archivos por lo que las variables cambiarán.

3.2. Limpieza de caché

En este módulo se eliminan archivos temporales o innecesarios del sistema. Utilizando el comando **rm** o **remove** con unos parámetros de **rf.** La **r** es por recursivo, el remove eliminará tanto carpetas como subcarpetas hasta que no haya nada, la **f** es para **forzar** para que el proceso se realice de manera segura, redirigiendo posibles mensajes de error para evitar interrupciones si el directorio ya está vacío o protegido.

3.3. Respaldo de archivos

Esta sección se encarga de copiar los archivos importantes desde un directorio de origen hacia una ubicación de respaldo. Se utiliza una herramienta de sincronización que conserva la estructura de carpetas, los permisos y las fechas de modificación, garantizando que el respaldo sea fiel al original. Usamos **mkdir** solo por si la carpeta **destino** no existe para crearla. Luego guardamos el backup comprimido con el nombre **backup_fecha**.tar.gz

3.4. Módulo de limpieza de archivos temporales y caché

Esta sección se encarga de eliminar los archivos temporales y cache dentro del sistema. Debido a que Windows y Linux tienen carpetas distintas para estos archivos, usaremos las variables que declaramos en **Detección del sistema operativo.**

3.5. Opciones y menú interactivo

El script incluye un menú que permite al usuario seleccionar qué tareas ejecutar. Se usa do-while para el ciclo del menú y una variable **opcion** que controla el switch. Incluye una validación por si el usuario ingresa una opción no valida.

ADVERTENCIA: Es muy importante que el menú esté luego de los módulos de funcionalidades porque de otra manera el script no podrá encontrar las funciones al ejecutarse.

4. Pruebas y Validación

4.1. Procedimiento de prueba

Las pruebas se realizaron ejecutando el script en un entorno Windows, validando que cada módulo funcione de manera independiente. Se verificó que:

- Los archivos temporales fueran eliminados correctamente.
- Los respaldos se generaran en el destino esperado.
- El informe del sistema se registre correctamente.

4.2. Resultados esperados

- Confirmación visual del sistema operativo detectado.
- Limpieza exitosa de los directorios temporales.
- Respaldo de archivos con estructura y permisos preservados.
- Mensajes informativos en consola sin errores.

5. Reflexiones Finales

5.1. Dificultades encontradas

Durante el desarrollo surgieron algunos desafíos, principalmente relacionados con la compatibilidad con Windows. Los permisos necesarios para ejecutar determinadas acciones, y la manipulación segura de archivos del sistema. Tuve problemas al entender los diferentes sistemas de archivos como para entender como poder usarlos en los módulos.

5.2. Posibles mejoras futuras

- Implementar una interfaz gráfica o basada en texto más amigable.
- Permitir respaldos automáticos en la nube.
- Mejorar la detección de errores mediante validaciones más robustas.
- Implementación formal de macOS.

5.3. Conclusiones

El script permitió automatizar tareas esenciales del mantenimiento del sistema, reduciendo el tiempo requerido y minimizando el riesgo de errores humanos. Ademas, su diseño modular facilita futuras ampliaciones.