Trabajo Final

Sistemas Operativos

Integrantes:

Lisandro Manuel Frende

Mariano Scarcella

DOCUMENTACION TÉCNICA. Deberán acompañar el desarrollo del script con un informe de memoria técnica en formato PDF con base en los siguientes puntos:

- Introducción, descripción del proyecto y las tareas automatizadas.
- Requisitos técnicos, herramientas y dependencias necesarias para ejecutar los scripts.
- Desarrollo, explicación detallada del código.
- Pruebas y validación, cómo probar las funcionalidades del script, incluyendo capturas de pantalla si es posible.
- Reflexiones finales sobre el desarrollo, dificultades encontradas y posibles mejoras.

Documentación Técnica

1. Introducción

- **Objetivo del Proyecto**: Este proyecto consiste en la creación de un script en Bash para automatizar tareas comunes en sistemas operativos Linux. Las tareas elegidas para la automatización son:
    - Respaldo de un directorio.
    - Generación de un informe de uso de CPU, memoria y disco.
    - Gestión de usuarios en el sistema.
- **Descripción**: El script presenta un menú interactivo que permite al usuario seleccionar una tarea para ejecutar, mejorando la gestión y el mantenimiento del sistema de manera automatizada. Utiliza

funciones para organizar cada tarea y permite al usuario navegar de manera intuitiva entre las opciones.

2. Requisitos Técnicos

- **Entorno**: Sistema operativo Linux o Windows con WSL.
- **Herramientas Necesarias**:
    - ***tar***: Para la compresión de archivos en el respaldo.
    - ***top***, ***free***, ***df***: Para obtener el uso de CPU, memoria y disco respectivamente.
    - ***useradd*** y ***chmod***: Para la creación de usuarios y configuración de permisos (requiere permisos de ***sudo***).
- **Dependencias**: No se requieren paquetes adicionales; todas las herramientas utilizadas son básicas en sistemas Linux y están preinstaladas en la mayoría de las distribuciones.

3. Desarrollo y Explicación del Código

- **Estructura General del Código**: El script está organizado en **funciones** para mejorar la legibilidad y la modularidad. Cada tarea principal se implementa en una función específica que el usuario puede activar desde un menú interactivo.
- **Función *backup()*:**
    - **Propósito**: Realiza un respaldo comprimido del directorio principal del usuario en un archivo ***.tar.gz*** y gestiona la eliminación de respaldos antiguos, manteniendo solo los últimos cinco.
    - **Descripción**:
        1. Crea un directorio para guardar los respaldos si aún no existe.
        2. Genera un nombre único para el archivo de respaldo utilizando la fecha y hora actual.
        3. Comprueba si el número de archivos de respaldo supera el límite configurado (5). Si es así, elimina el archivo más antiguo.

4. Muestra un mensaje de éxito o error según el resultado de la operación.

- **Función *system_report()*:**
    - **Propósito**: Genera un archivo de log que contiene el uso actual de CPU, memoria y disco.
    - **Descripción**:
        1. Crea un archivo de log con un nombre único basado en la fecha y hora actuales.
        2. Utiliza ***top*** para capturar el uso de la CPU, ***free*** para la memoria y ***df*** para el disco.
        3. Guarda los resultados en el archivo de log y muestra un mensaje de confirmación.
- **Función *manage_users()***:
    - **Propósito**: Permite crear un nuevo usuario en el sistema y configurar permisos básicos para su directorio personal.
    - **Descripción**:
        1. Solicita un nombre de usuario al operador.
        2. Utiliza el comando ***useradd*** para crear el usuario junto con su directorio ***home***.
        3. Configura permisos de acceso estándar (755) en el directorio del usuario.
        4. Devuelve un mensaje de éxito o error según el resultado.
- **Menú Interactivo**:
    - **Propósito**: Facilita la navegación del usuario entre las diferentes funciones del script.
    - **Descripción**:
        1. Muestra un menú de opciones numeradas con colores para mejorar la experiencia de usuario.

2. Permite al usuario seleccionar una opción e invoca la función correspondiente.
3. Valida que la entrada del usuario sea válida y muestra un mensaje de error en caso contrario.
4. Incluye una opción para salir del script.

4. Pruebas y Validación

- **Procedimiento de Pruebas**:
    1. **Función de Respaldo**:
        - Ejecuta ***backup()*** y verifica que se haya creado un archivo ***.tar.gz*** en el directorio de backups.
        - Comprueba que se mantenga un máximo de 5 archivos en el directorio, eliminando los más antiguos.
    2. **Generación de Informe**:
        - Ejecuta ***system_report()*** y revisa que se haya creado un archivo de log con los datos de CPU, memoria y disco.
        - Abre el archivo para confirmar que los datos son correctos y están en el formato esperado.
    3. **Gestión de Usuarios**:
        - Ejecuta ***manage_users()*** e intenta crear un usuario nuevo.
        - Verifica que el directorio del usuario fue creado en ***/home***.
        - Asegúrate de que los permisos del directorio estén configurados como ***755***.
- **Capturas de Pantalla**:
    - Incluye capturas de pantalla del menú interactivo y de los resultados de cada función.
    - Captura la salida de la terminal que muestra la creación de un respaldo, la generación de un informe de sistema y la creación de un usuario.

5. Reflexiones Finales

- **Dificultades Encontradas**:
    - En algunos sistemas, es posible que se requieran permisos ***sudo*** para ejecutar comandos como ***useradd***, lo que puede provocar errores si no se tiene la configuración adecuada.
    - Validar la entrada del usuario en el menú fue un aspecto importante para asegurar que no se produzcan errores por opciones incorrectas.
- **Posibles Mejoras**:
    - **Archivo de Configuración**: Implementar un archivo de configuración para que el usuario pueda ajustar parámetros (como la ruta de respaldo y el límite de backups) sin modificar el código fuente.
    - **Más Opciones en la Gestión de Usuarios**: Ampliar las funcionalidades para permitir la eliminación o modificación de usuarios.
    - **Compatibilidad Mejorada**: Añadir más validaciones para asegurar que el script funcione en diferentes distribuciones de Linux.

6. Estructura del Repositorio (Entrega)

En el repositorio de GitHub, asegúrate de incluir los siguientes archivos:

- ***script.sh***: El código principal del script en Bash.
- ***README.md***: Una breve descripción del proyecto, instrucciones de uso, ejemplos y una guía para colaboradores.
- ***DOCUMENTACION.pdf***: Esta documentación técnica en un formato PDF, detallando cada uno de los puntos mencionados.
- **Opcional**: Archivos de log o ejemplos de salida para facilitar la comprensión del funcionamiento del script.