TRABAJO INTEGRADOR

TECNICATURA SUPERIOR EN SOFTWARE

MATERIA: SISTEMAS OPERATIVOS.

DOCENTE: LUCAS MARTIN TRESER.

ALUMNOS: BRIAN COUREL, MARCO ALEJANDRO GOMEZ, TOMAS LLERA

FECHA DE ENTREGA: 4/11/2024

NOMBRE INST.: INSTITUTO DE EDUCACION SUPERIOR IDRA.

INDICE:

1) Introducción del proyecto

Este proyecto tiene como objetivo crear un programa que permita a los usuarios gestionar de manera más práctica y sencilla ciertas funcionalidades del sistema operativo Linux, que podrían ser complejas de realizar manualmente.

Las principales tareas automatizadas son:

- **Generar informes sobre el uso de CPU, memoria y disco del sistema.**
- **Limpiar archivos temporales y caché de los navegadores más usados.**
- **Verificar y actualizar paquetes instalados en el sistema.**

2) Requisitos Técnicos

- Sistema operativo: Linux.

El script ha sido diseñado específicamente para ser ejecutado en sistemas operativos basados en Linux. Se recomienda utilizar una distribución popular como Ubuntu, Debian, Fedora o CentOS, ya que estas proporcionan un entorno robusto y compatible con las herramientas necesarias para el funcionamiento del script.

- Terminal: Bash.

El script requiere el uso de una terminal de comandos, específicamente **Bash**, que es la shell por defecto en muchas distribuciones de Linux. Bash permite la ejecución de comandos y scripts de shell, lo que es fundamental para la automatización de tareas en el sistema.

- Permisos de ejecución.

Antes de ejecutar el script, es necesario otorgarle permisos de ejecución. Esto se logra mediante el comando:

```
bash

chmod +x nombre_del_archivo.sh
```

Este paso es esencial, ya que, sin los permisos adecuados, el sistema no permitirá la ejecución del archivo. Asegúrese de ejecutar este comando en la misma carpeta donde se encuentra el script.

- Herramientas y Dependencias.

El script no depende de bibliotecas externas o software adicional, lo que lo hace ligero y fácil de implementar. Sin embargo, se asume que el usuario tiene acceso a los siguientes comandos básicos de Linux:

- **`echo`**: Utilizado para mostrar mensajes y datos en la terminal, así como para redirigir salidas a archivos.
- **`rm`**: Comando para eliminar archivos y directorios, utilizado en la limpieza de caché.
- **`apt`**: Herramienta de gestión de paquetes para sistemas basados en Debian. Se utiliza para verificar y actualizar los paquetes instalados en el sistema.

Asegúrese de que su sistema esté actualizado y de que tenga los permisos necesarios para ejecutar comandos que modifiquen el sistema, como la limpieza de archivos y la actualización de paquetes.

3) Desarrollo

- **Estructura del Código**

El script se compone de varias funciones que gestionan las diferentes tareas. A continuación, se describen algunas funciones clave:

1) Generar Informe de Uso de Recursos

```bash
agregar_archivo_al_log(){
    echo "$1" >> $archivo_log
    echo "" >> $archivo_log
}
```

Esta función genera un informe que se guarda en un archivo log especificado.

2) Limpieza de Archivos Temporales

```
limpiar_navegadores(){
    navegador=$1
    ruta_cache=$2
    echo "Limpiando navegador $navegador"
    if [ -d "$ruta_cache" ]; then
        rm -rf "$ruta_cache"
        echo "Caché del navegador $navegador eliminado."
    else
        echo "No se encontraron archivos caché del navegador $navegador."
    fi
}
```

Esta función se encarga de eliminar los archivos de caché de un navegador especificado.

3) Verificación y Actualización de Paquetes

```
if apt list --upgradable 2>/dev/null | grep -q "upgradable"; then
    # Código para actualizar paquetes
fi
```

Se verifica si hay paquetes actualizables y, en caso afirmativo, se procede a su actualización.

- **Instrucciones de Uso**

1) Acceder a la carpeta del programa:

```
cd nombre_del_programa
```

2) Solicitar permisos de ejecución:

```
chmod +x nombre_del_archivo.sh
```

3) Ejecutar el programa:

```
./nombre_del_programa.sh
```

4) Seleccionar una opción del menú que aparece.

4) Pruebas y validación

Para garantizar que todas las funcionalidades del script operen correctamente, se realizaron pruebas exhaustivas de cada opción del menú. A continuación se describe el procedimiento de prueba para cada funcionalidad, así como los criterios de éxito y los resultados esperados.

- **Generar Informe**: Se ejecuta la opción correspondiente y se verifica que el archivo de log se genere correctamente con la información esperada.

  - **Procedimiento**:
    - Ejecutar el script y seleccionar la opción "1: Generar un informe de uso de memoria, disco y CPU".
    - Esperar a que el proceso se complete y que se genere el archivo de log.

  - **Resultado**:
    - Verificar que el archivo de log se crea en la ubicación especificada.
    - Abrir el archivo y comprobar que contenga información precisa sobre el uso de memoria, disco y CPU, formateada correctamente.

- **Limpieza de Caché**: Se selecciona un navegador, y se verifica que la ruta de caché se elimine correctamente, observando los mensajes generados.

  - **Procedimiento**:
  - Seleccionar la opción "2: Realizar limpieza de archivos temporales del sistema y caché de los navegadores".
  - Elegir un navegador específico y proporcionar la ruta de caché correspondiente.
  - Ejecutar la limpieza y observar los mensajes generados durante el proceso.

  - **Resultado**:

  - Verificar que los mensajes impresos en la terminal confirmen la eliminación exitosa de los archivos de caché.
  - Comprobar que la ruta de caché ya no contenga los archivos eliminados.

- **Actualización de Paquetes**: Se ejecuta la opción de actualización y se revisa que los paquetes listados como "upgradable" sean efectivamente actualizados.

  - **Procedimiento**:
    - Seleccionar la opción "3: Verificar y actualizar paquetes instalados en el sistema".
    - Observar la salida en la terminal para identificar si hay paquetes "upgradable".
    - Confirmar que los paquetes que aparecen como actualizables se actualicen sin errores.

  - **Resultado**:
    - Los paquetes listados como "upgradable" deben actualizarse correctamente.
    - No deben aparecer mensajes de error durante el proceso de actualización.

- **Capturas de pantalla**

5) Reflexiones finales

- **Dificultades Encontradas**

  Durante el desarrollo del script, se presentaron varios desafíos que requerían atención y resolución:

  - **Gestión de Permisos**: Uno de los primeros obstáculos fue garantizar que el script tuviera los permisos adecuados para ejecutarse en el sistema. Aprender a utilizar el comando `chmod` y asegurarse de que los usuarios comprendieran este paso resultó esencial. Se creó una sección de instrucciones clara para abordar este aspecto.
  - **Verificación de Rutas**: La verificación de rutas para la limpieza de archivos temporales y caché fue otro desafío. Asegurarse de que las rutas proporcionadas por los usuarios fueran correctas y existieran en el sistema

requería una validación cuidadosa. Esto llevó a implementar controles adicionales dentro del script para manejar situaciones en las que las rutas no estuvieran disponibles.

- **Manejo de Errores**: Inicialmente, el script no contemplaba adecuadamente el manejo de errores, lo que podría causar que el programa se detuviera inesperadamente. Esto impulsó la necesidad de incorporar mensajes de error más claros y opciones de recuperación para mejorar la experiencia del usuario.

A pesar de estas dificultades, el enfoque proactivo en la solución de problemas permitió implementar todas las funciones requeridas de manera efectiva, cumpliendo con los objetivos del proyecto.

- **Posibles Mejoras**

A medida que se reflexiona sobre el desarrollo del script, surgen varias áreas en las que se pueden realizar mejoras significativas:

- **Interfaz Gráfica**: Implementar una interfaz gráfica (GUI) podría mejorar considerablemente la experiencia del usuario, haciendo que las interacciones sean más intuitivas y accesibles. Esto permitiría a los usuarios que no están familiarizados con la línea de comandos interactuar con el programa de manera más cómoda.

- **Soporte para Múltiples Navegadores**: Actualmente, el script está limitado a la limpieza de caché de ciertos navegadores. Agregar soporte para otros navegadores populares, así como opciones de limpieza más configurables, brindaría a los usuarios una mayor flexibilidad y utilidad.

- **Registro de Errores**: Incluir un sistema de registro de errores permitiría un mejor manejo de excepciones y facilitaría la depuración en caso de que ocurran problemas. Esto podría incluir la creación de un archivo de log donde se registren los errores y advertencias durante la ejecución del script, lo que ayudaría a los usuarios a identificar y resolver problemas rápidamente.

- **Documentación y Tutoriales**: Proporcionar documentación adicional y tutoriales paso a paso para la instalación y uso del script mejoraría la accesibilidad para usuarios menos experimentados. Esto podría incluir videos, ejemplos prácticos y preguntas frecuentes (FAQ) que aborden problemas comunes.

En resumen, el desarrollo del script fue un proceso enriquecedor que permitió aprender sobre la automatización de tareas en Linux, y aunque se presentaron desafíos, las soluciones implementadas sentaron las bases para futuras mejoras y ampliaciones del proyecto.

6) Link de referencia.

- https://n9.cl/l67z3