Trabajo Final
Sistemas Operativos

Memoria técnica

Tecnicatura de Desarrollo de Software
Profesor a cargo: Lucas Treser
Integrantes: Roque Tatiana y Ariel Soria
Fecha de entrega: 01/11
Instituto Superior Idra

Índice

Introducción

Este proyecto tiene como objetivo automatizar tareas comunes en sistemas operativos utilizando scripts en Bash. A través de la implementación de un menú interactivo, se permite al usuario seleccionar y ejecutar diferentes funciones: generación de informes sobre el uso de recursos del sistema, eliminación de archivos temporales y automatización de respaldos de directorios. Esta automatización busca facilitar la gestión de recursos y mejorar la eficiencia en el manejo de archivos.

El proyecto consiste en tres scripts principales que realizan las siguientes tareas:

1. **Generación de Informes**: Un script que recopila y guarda información sobre el uso de CPU, memoria y espacio en disco en un archivo log.
2. **Eliminación de Archivos Temporales**: Un script que verifica la existencia de una carpeta ficticia y elimina archivos temporales creados específicamente para este proyecto, demostrando el manejo de archivos en el sistema.
3. **Respaldo de Directorios**: Un script que permite realizar copias de seguridad automáticas de un directorio específico, gestionando los respaldos en una carpeta designada.

Requisitos Técnicos

Herramientas Necesarias

- **Sistema Operativo**: Ubuntu (o cualquier distribución de Linux compatible con Bash).
- **Editor de Texto**: nano, vim, o cualquier otro editor.
- **Git**: Para el control de versiones y la colaboración en el proyecto.

Dependencias

Los siguientes comandos y utilidades son utilizados en los scripts y deben estar disponibles en el sistema:

- `top`: Para monitorear el uso de la CPU.
- `free`: Para obtener información sobre el uso de memoria.
- `df`: Para verificar el espacio en disco.

En el caso de no contar con ellos puedes instalarlos utilizando estos comandos en Ubuntu:

```
sudo apt update
```

```
sudo apt install procps coreutils
```

Estructura del Proyecto

El proyecto se organiza de la siguiente manera:

- `README.md`: proporciona una descripción del proyecto, instrucciones de uso y detalles sobre cómo colaborar.
- `LICENSE`: especifica los términos de la licencia bajo la cual se distribuye el proyecto.
- `temp/`: Carpeta de archivos temporales que serán eliminados.
    - `archivo1.txt`
    - `archivo2.txt`
- `ejemploRespaldo/`: Carpeta que contiene los archivos y directorios de ejemplo que se respaldarán.
    - `directorio/`
        - `texto.txt`
- `Menu.sh`: Script principal que presenta el menú interactivo.
- `scripts/`: Carpeta que contiene los scripts para cada tarea automatizada:
    - `generar_informe.sh`: Script para la generación de informes.
    - `eliminar_temporales.sh`: Script para la eliminación de archivos temporales.
    - `respaldo_directorio.sh`: Script para la automatización del respaldo.

Explicación Detallada del Código

Menu.sh

El script principal presenta al usuario un menú interactivo con cuatro opciones. Utiliza códigos de color para mejorar la experiencia del usuario. Se definen los colores al inicio guardandolos en variables y así poder acceder a ellos más fácilmente, estos sirven para personalizar la salida en la terminal.

Se utilizan `echo` para mostrar las opciones y read para capturar la selección del usuario. Manejando las opciones con un `case`.

Dependiendo de la opción elegida, se ejecuta el script correspondiente utilizando el comando `./scripts/nombre_del_script.sh`.

generar_informe.sh

Este script recopila datos sobre el uso de CPU, memoria y espacio en disco, y los guarda en un archivo log.

Las variables de color (`GREEN`, `BLUE`, `RESET`) mejoran la presentación del informe, diferenciando secciones y facilitando la lectura visual. `clear` limpia la pantalla al inicio para que el usuario pueda ver el informe sin distracciones.

- `mkdir -p $LOG_DIR` asegura que el directorio `Log_Files` exista, creando uno nuevo si es necesario.
- Se crea un archivo con un nombre que incluye la fecha y hora actuales, lo que facilita la organización de informes.
- Recopilación de Datos:
    - `top -b -n1 | grep "Cpu(s)"`: Captura el uso de CPU en modo batch.
    - `free -h`: Muestra el uso de memoria de manera legible.
    - `df -h`: Proporciona información sobre el espacio disponible en disco.
- Salida en Archivo: Los resultados se redirigen a `LOG_FILE`, que se crea cada vez que se ejecuta el script.

eliminar_temporales.sh

Este script verifica la existencia de una carpeta y elimina todos los archivos que contiene.

`TEMP_DIR` define el directorio donde se encuentran los archivos temporales. En este caso, se utiliza una carpeta ficticia (`temp`), simulando un entorno controlado de archivos temporales.

Función `verificar_carpeta`

Se define la función `verificar_carpeta` para comprobar si el directorio temporal existe. Si no existe, el script termina con un mensaje de error.

Función `eliminar_temporales`

Verifica primero si la carpeta `temp` contiene archivos utilizando `ls -A "$TEMP_DIR"`. Si encuentra archivos, utiliza `rm -f "$TEMP_DIR"/*` para eliminarlos, y luego muestra un mensaje confirmando la eliminación exitosa. Si no hay archivos, informa que la carpeta está vacía.

El `sleep 1` añade una pausa breve para hacer más visible el mensaje inicial.

Función `main`

Es la función que organiza y ejecuta el flujo del script, limpiando la pantalla e iniciando los pasos de verificación y eliminación. El llamado a `main` al final del script asegura que el código se ejecute de manera secuencial y ordenada.

respaldo_directorio.sh

Este script permite realizar copias de seguridad automáticas de un directorio especificado.

El menú permite al usuario seleccionar entre iniciar el respaldo, eliminar respaldos antiguos o regresar al menú principal, con opciones claras y diferenciadas por color. Se define una paleta de colores (`GREEN`, `BLUE`, `RESET`) para los mensajes en la terminal, con el objetivo de mejorar la interfaz visual.

El script verifica primero si existen los directorios de origen (`DIRECTORIO_ORIGEN`) y de respaldo (`DIRECTORIO_RESPALDOS`). Si el directorio de origen no existe, muestra un mensaje de error y finaliza la ejecución. Si el de respaldo no existe, lo crea automáticamente con `mkdir -p`, y si ocurre un error al crearlo, el script se detiene.

Función `iniciar_respaldo`

El bucle `while true` permite que los respaldos se realicen continuamente hasta que el usuario decida detenerlo.

`tar -czf` comprime el directorio origen en un archivo `tar.gz` con un nombre que incluye un `TIMESTAMP` y un contador. Un temporizador de 15 segundos informa al usuario sobre el tiempo restante hasta el próximo respaldo. Esto mejora la visibilidad y permite un monitoreo sencillo. El usuario puede interrumpir el ciclo presionando "q", lo cual provoca que el script realice un último respaldo y termine.

Función `borrar_respaldos`

La función usa `ls -1t` para listar los respaldos en orden cronológico, preservando sólo el más reciente. Con `tail -n +2`, se seleccionan todos los respaldos menos el primero (más reciente), que son eliminados mediante `xargs rm -f`. Esto permite al usuario gestionar el espacio de almacenamiento eliminando respaldos antiguos de manera automática.

Pruebas y Validación

Cómo Probar las Funcionalidades

`generar_informe.sh`

Para probar el script `generar_informe.sh`, se debe ejecutar en la terminal de la siguiente manera:

```
bash scripts/generar_informe.sh
```

Después de la ejecución, se puede verificar el contenido del archivo generado en el directorio `Log_Files`. Se espera que el archivo contenga información sobre el uso de CPU, memoria y espacio en disco en el momento de la ejecución.

```
root@DESKTOP-FFJIJF0:~# bash scripts/generar_informe.sh
Informe generado en Log_Files/informe_sistema_20241021_232049.log
```

```
------------- Informe de uso del sistema ------------
Fecha: Mon Oct 21 23:20:49 UTC 2024
Uso de CPU:
%Cpu(s):  1.0 us,  0.0 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
-----------------------------------------------------
Uso de Memoria:
               total        used        free      shared  buff/cache   available
Mem:            15Gi       9.1Gi       6.8Gi        17Mi       223Mi       6.8Gi
Swap:           27Gi        79Mi        27Gi
-----------------------------------------------------
Espacio en Disco:
Filesystem      Size  Used Avail Use% Mounted on
rootfs          223G  155G   69G  70% /
none            223G  155G   69G  70% /dev
none            223G  155G   69G  70% /run
none            223G  155G   69G  70% /run/lock
none            223G  155G   69G  70% /run/shm
none            223G  155G   69G  70% /run/user
tmpfs           223G  155G   69G  70% /sys/fs/cgroup
C:\             223G  155G   69G  70% /mnt/c
D:\             466G   67G  400G  15% /mnt/d
```

`eliminar_temporales.sh`

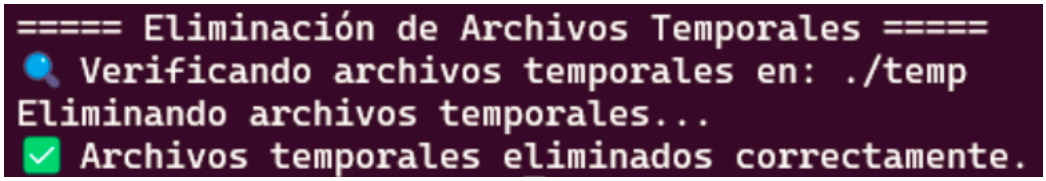
Desde el terminal, ejecuta el script `eliminar_temporales.sh` con el siguiente comando:

```
bash scripts/eliminar_temporales.sh
```

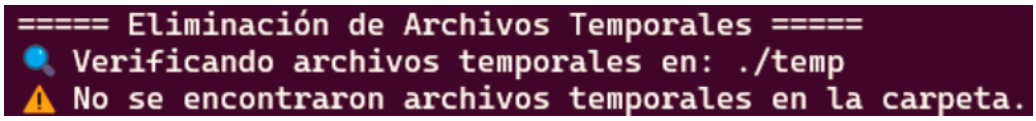
Durante la ejecución, el script debería verificar la existencia del directorio `temp/` y proceder a eliminar su contenido.

En la consola, aparecerán mensajes indicando si se encontraron y eliminaron los archivos temporales, o si el directorio estaba vacío.

Mensaje si el script terminó exitosamente, borrando todos los archivos que se contenían en `temp/` :



Mensaje que se mostraria si el script no encontrase archivos en la carpeta:



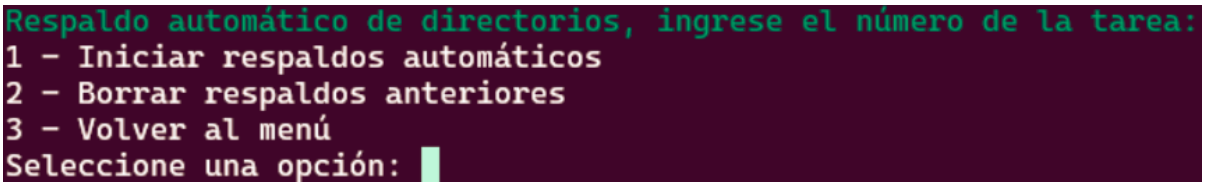
Mensaje si no se hubiera encontrado la carpeta:

respaldo_directorio.sh

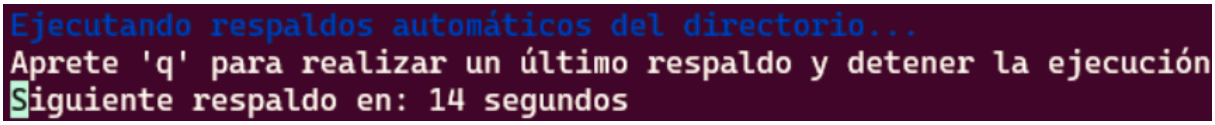
Desde la terminal, ejecuta el script de respaldo:

```
bash scripts/respaldo_directorio.sh
```

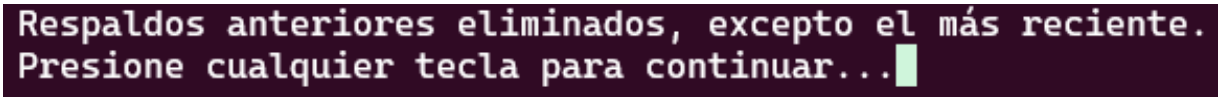
Selecciona la opción 1 para iniciar los respaldos automáticos. El script iniciará el proceso de respaldo, comprimiendo el directorio y mostrando un temporizador para el siguiente respaldo.



Para finalizar, presione ‘q’ y espera a que el último respaldo termine.



Para borrar los respaldos anteriores ejecute nuevamente el script y seleccione la opción 2 para borrar respaldos anteriores, confirmando que solo el respaldo más reciente permanece en el directorio.



Reflexiones Finales

Dificultades Encontradas

Durante el desarrollo de los script, se presentaron algunos desafíos. Inicialmente, tuvimos dificultades con la instalación y habilitación de Ubuntu en nuestro entorno de trabajo. Configurar correctamente el sistema operativo, entender la estructura de directorios de Linux y familiarizarnos con comandos básicos fue un proceso que nos requirió tiempo y práctica.

Además, tuvimos que aprender el uso práctico de Git y GitHub, herramientas con las que no habíamos trabajado previamente. El manejo de versiones y la resolución de conflictos de merge(merge conflicts) fueron temas que costaron entender. Tuvimos que investigar sobre el uso de ramas, el flujo de trabajo de commits, y sobre cómo sincronizar correctamente los cambios entre el repositorio local y el remoto.

Otra dificultad fue el diseño y estructuración de los scripts. Definir rutas relativas en lugar de absolutas, crear menús interactivos y asegurar que los scripts fueran compatibles en distintos entornos Linux fue un proceso de prueba y error.

Posibles Mejoras

Se podría mejorar el menú para que permita al usuario seleccionar diferentes configuraciones, como el intervalo de tiempo para generar informes de uso, la eliminación automática de archivos de informe antiguos. Además, que el usuario pueda agregar la ubicación del respaldo. De esta forma, el script sería más flexible y adaptable a diferentes necesidades.