Proyecto 1

Jaime Mendez, Carolay Rodriguez y Laura Baez

Marzo 2025

1 Introduction

En el documento se van a presentar una serie de imágenes que muestran diferentes situaciones al trabajar con scripts en un sistema Linux. Desde la ejecución de copias de seguridad hasta el monitoreo de recursos y el análisis de logs, cada imagen muestra las tareas comunes que pueden surgir en la administración del sistema.

A lo largo de estas capturas, se pueden ver errores que ocurren con mucha cotidianidad, como problemas de permisos al ejecutar un script, así como la manera en que se pueden solucionar. También se observa cómo utilizar herramientas como sudo para acceder a funciones avanzadas del sistema y cómo interpretar los registros de actividad. Por ultimo este recorrido visual ayuda a entender mejor cómo funcionan estos procesos y cómo aprovecharlos para el mantenimiento y la seguridad del sistema.

2 Contenido del repositorio

En la imagen se muestra la configuración del archivo de tareas programadas (crontab) en un sistema Linux utilizando el editor de texto Nano. En este archivo, se han definido varias tareas automatizadas que se ejecutan en intervalos específicos. Algunas tareas incluyen la ejecución del script resource-monitor.sh cada 30 minutos para registrar el uso de recursos, limpiar-memoria.sh todos los días a la medianoche para liberar memoria, y ping-test.sh cada hora para registrar la conectividad de red. Además, tambien se han programado otros scripts como backup-manager.sh para gestionar copias de seguridad y log-analyzer.sh para analizar registros del sistema. Cada tarea redirige su salida a un archivo de log correspondiente para su posterior revisión.

El repositorio contiene los siguientes archivos:

- backup-manager.sh- Script para realizar copias de seguridad.
- limpiar-memoria.sh- Guión para liberar memoria.
- log-analyzer.sh- Script para analizar archivos de registros.

- ping-test.sh- Guión para realizar pruebas de conectividad.
- resource-monitor.sh- Guión para monitoreo de recursos del sistema.
- README.md- Documentación del proyecto.
- LICENSE- Licencia del proyecto.
- resultado.txt- Archivo de salida de algunos guiones.

Tarea-1.pdf- Documento relacionado con el primer corte.

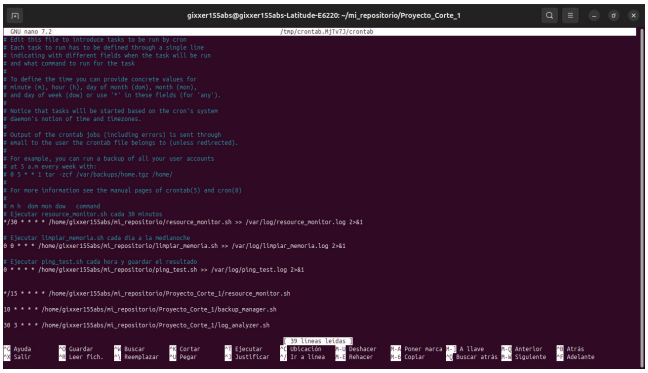

Figure 1: Contenido del repositorio

3 Instalación

Para utilizar estos scripts, sigue los siguientes pasos:

- Clonar el repositorio:

  git clone ¡URL-DEL-REPOSITORIO¿
  cd mi-repositorio/Tarea-1

- Dar permisos de ejecución a los scripts:

  chmod +x *.sh

4 Uso de los scripts

- backup$_manager.sh$
  Realice una copia de seguridad de los archivos especificados en un directorio determinado. Ejecutar con:
  ./backup$_manager.sh$

- limpiar$_m$*emoria.sh*
  Limpie la memoria caché y la swap para optimizar el rendimiento del sistema.
  Ejecutar con:
  ./limpiar$_m$*emoria.sh*
- log$_a$*nalyzer.sh*
  Analiza los logs del sistema en busca de errores y advertencias. Ejecutar con:
  ./log$_a$*nalyzer.sh*
- ping$_t$*est.sh*
  Realice pruebas de conectividad hacia una lista de direcciones IP o dominios.
  Ejecutar con:
  ./ping$_t$*est.sh*
- resource$_m$*onitor.sh*
  Monitorea el uso de CPU, memoria y disco en el sistema. Ejecutar con:
  ./resource$_m$*onitor.sh*

5 Automatización

Para automatizar la ejecución de estos scripts, se puede usar cronen Linux:

- Editar el crontab con:

  crontab -e

- Agregar las tareas programadas, por ejemplo:

  0 * * * * /home/user/mi-repositorio/Tarea-1/resource-monitor.sh ¿¿ /home/user/logs/monitor.log
  Esto ejecutará resource-monitor.shcada hora y guardará la salida en un registro.

6 Solución de problemas

- Si no se ejecuta un script, verifique los permisos con:

  ls -l script.sh
  Y asegúrese de que tenga permisos de ejecución (chmod +x script.sh).

- Si git statusno se muestran cambios al agregar archivos, use:

  git add -A

- Y luego hacer commit y push:

  git commit -m "Actualización de scripts"
  git push origin main

7 Comando crontab -l

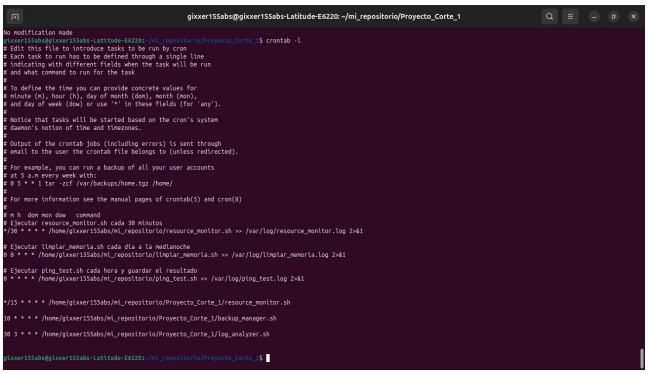

Figure 2: Comando crontab -l

La imagen muestra la ejecución del comando crontab -l, el cual permite listar las tareas programadas en cron para el usuario actual. Se observa que hay varias tareas automatizadas configuradas, incluyendo la ejecución de resource-monitor.sh cada 30 minutos, limpiar-memoria.sh diariamente a la medianoche y ping-test.sh cada hora, redirigiendo sus salidas a archivos de log en /var/log/. Tambien, hay otras tareas programadas, como la ejecución de backup-manager.sh y log-analyzer.sh. Estas tareas permiten el monitoreo y mantenimiento del sistema de forma automática sin intervención manual.

8 monitor de recursos

```bash
#!/bin/bash
echo "=== MONITOR DE RECURSOS ==="
echo "CPU: $(top -bn1 | grep load | awk '{printf "%.2f%%\n", $(NF-2)}')"
echo "Memoria libre: $(free -m | awk '/Mem:/ {print $4 " MB"}')"
echo "Uso de disco: $(df -h / | awk '/\// {print $5}')"
```

Figure 3: monitor de recursos

El script mostrado en la imagen es un monitor de recursos del sistema escrito en Bash. Su función es mostrar información clave sobre el rendimiento del sistema, incluyendo el uso de la CPU, la memoria libre y el uso del disco.

- echo "=== MONITOR DE RECURSOS ==="

Se imprime un encabezado para indicar que es un monitor de recursos.

- echo "CPU: signo pesos (top -bn1 — grep load — awk 'printf "porciento .2fporciento porciento", signo pesos(NF-2)')"

Se usa el comando top en modo batch (-bn1) para obtener la carga del sistema.

- echo "Memoria libre: signo pesos (free -m — awk '/Mem:/ print signo pesos 4 " MB"')"

Se usa free -m para tener el uso de memoria en megabytes.

- echo "Uso de disco: signo pesos(df -h / — awk '// print signo pesos 5')"

Se usa df -h / para tener la información sobre el uso del disco en el sistema de archivos raíz.

Este script es útil para tener rapido el estado del sistema y puede ser ejecutado manualmente o programado en cron para ser monitoreado automático.

9 Copia de seguridad

```
#!/bin/bash
tar -czf /backups/backup_$(date +%Y-%m-%d).tar.gz /ruta/del/directorio
```

Figure 4: Copia de seguridad

- tar -czf /backups/backup-signo pesos (date +porciento Y-porciento m-porciento d).tar.gz /ruta/del/directorio
- tar -czf:
- c: Se crea un archivo comprimido.
- z: Se usa compresión gzip.
- f: Especifica el nombre del archivo de salida.
- /backups/backup-signo pesos (date +porciento Y-porciento m-porciento d).tar.gz:
- Guarda el archivo en /backups/.
- signo pesos (date +porciento Y-porciento m-porciento d): Se grega la fecha en formato AÑO-MES-DÍA al nombre del archivo.
- /ruta/del/directorio: Directorio que se comprimirá.

El script es útil para programar copias de seguridad automáticas mediante cron.

```
#!/bin/bash
grep -i "error" /var/log/syslog | awk '{print $1, $2, $3, $6}'
```

Figure 5: Bash

10 Bash

Este script en Bash busca y extrae información sobre errores registrados en el archivo de logs del sistema (/var/log/syslog), utilizando grep para filtrar líneas con la palabra "error" sin distinguir mayúsculas o minúsculas, y awk para extraer la fecha, hora y el proceso que generó el mensaje.

11 GNU nano

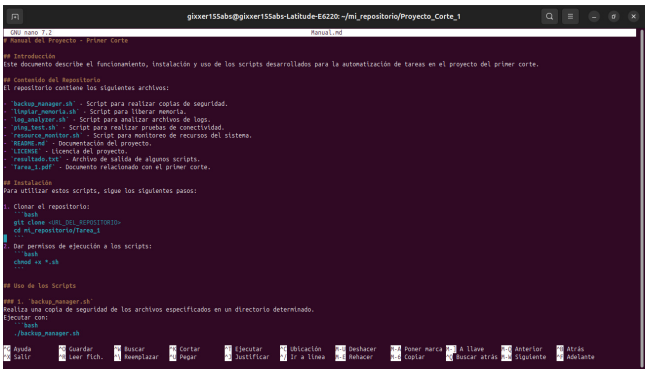

Figure 6: GNU nano

En la imagen se muestra un archivo llamado Manual.md editado en GNU nano, en el cual se documenta la instalación, uso y propósito de varios scripts para la automatización de tareas en un proyecto. En el manual se describen los scripts disponibles, como backup-manager.sh para copias de seguridad, limpiar-memoria.sh para liberar memoria, log-analyzer.sh para analizar logs, ping-test.sh para pruebas de conectividad y resource-monitor.sh para monitoreo de recursos. Tambien se incluyen la instrucciones para clonar el repositorio, otorgar permisos de ejecución y ejecutar cada script, proporcionando una estructura clara y orientada a la documentación técnica.

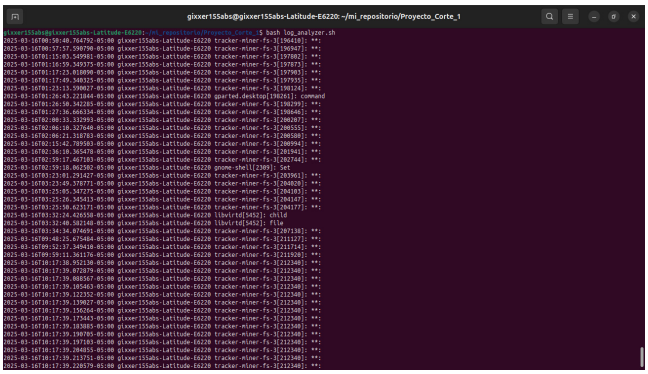

Figure 7: log$_a$*nalyzer.sh*

12 log$_a$*nalyzer.sh*

En la imagen se muestra la ejecución del script log-analyzer.sh en una terminal Linux, donde se analizan los registros del sistema en busca de eventos relevantes. Cada línea de salida incluyen una marca del tiempo, el nombre del host y procesos como tracker-miner-fs-3, gparted.desktop, gnome-shell y libvirtd. Algunos mensajes contienen , lo que podría indicar eventos específicos o errores.

13 backup$_m$*anager.sh*

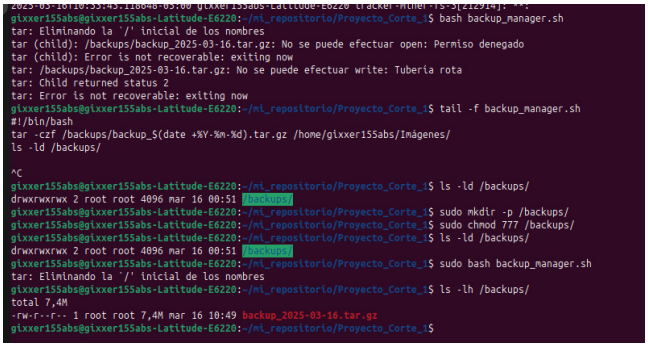

Figure 8: backup$_m$*anager.sh*

En la imagen se muestra la ejecución del script backup$_m$*anager.shylaresoluciónde*problemasrelacionadosco

14 Resource$_m$*onitor.sh*

La imagen muestra la ejecución del script resource$_m$*onitor.sh*, *quemonitoreaelusoderecursosdelsistema*, *incluy*

Figure 9: resource$_m$*onitor.sh*