Bitácora Semanal 2: Administración de Sistemas Unix

Equipo Alpine White

[2026-02-09 Mon]

Contents

1 Información General

- **Semana:** Del 9 de Febrero 2026 al 13 de Febrero 2026
- **Equipo:**
  - Arreguín Salgado Gael Emiliano
  - Gramer Muñoz Omar Fernando
  - López Pérez Mariana
  - Nieto Gallegos Isaac Julián

Durante la semana vimos una variedad de comandos que nos permiten ver el potencial de configurabilidad que existe dentro de Linux. Particularmente, estuvimos trabajando en permisos de archivos, montaje de sistemas de archivos y vamos comenzando a prepararnos para la virtualización.

2 Distro Watch

Distro Watch es un sitio web que monitorea y cataloga distribuciones GNU/Linux y sistemas Unix like. Solo es un portal informativo que mantiene bases de datos sobre versiones.

2.1 Page Hit Ranking

- El ranking mostrado en su página principal se basa en el número de visitas a la página informativa de cada distribución.
- No mide instalaciones reales, cuota de mercado ni popularidad efectiva.
- Es un indicador aproximado de interes relativo dentro de la comunidad que consulta el sitio.

3 POSIX y modelo de ejecución en sistemas Unix y GNU/Linux

POSIX (Portable Operating System Interface) es un estándar definido por IEEE que especifica cómo los programas interactúan con el sistema operativo. No es el sistema operativo en sí, sino una especificación que define las interfaces que deben existir. Gracias a POSIX, un programa puede ejecutarse en distintos sistemas tipo Unix, como Linux, BSD o macOS sin modificaciones significativas, siempre que use únicamente las interfaces estándar.

En este modelo, el concepto central es el proceso. Un proceso es un ejemplar de un programa en ejecución gestionado por el kernel. Cada proceso posee su propio espacio de memoria virtual, su propio contexto de ejecución y un identificador único llamado PID (Process ID). El kernel es responsable de crear, planificar y destruir procesos. La creación de procesos se realiza mediante la llamada al sistema `fork()`, que crea una copia del proceso actual. Posteriormente, el proceso puede reemplazar su imagen de memoria usando `exec()` para ejecutar otro programa. Este modelo permite construir sistemas complejos mediante la composición de procesos simples.

La interacción con recursos del sistema se realiza mediante una abstracción fundamental: el file descriptor. Un file descriptor es un entero que representa un recurso abierto, como un archivo, un dispositivo o un canal de comunicación. Por convención, existen tres descriptores estándar:

- 0 corresponde a la entrada estándar (stdin)
- 1 corresponde a la salida estándar (stdout)
- 2 corresponde al error estándar (stderr)

Esta abstracción permite que el sistema trate distintos tipos de recursos de forma uniforme, simplificando el diseño y la implementación de programas.

La comunicación entre procesos se realiza mediante mecanismos de IPC (Inter-Process Communication). Un pipe es un canal unidireccional gestionado por el kernel que conecta la salida de un proceso con la entrada de otro.

Internamente, el kernel crea un buffer temporal y coordina la transferencia de datos entre ambos procesos mediante file descriptors.

Las señales son notificaciones asíncronas enviadas por el kernel o por otros procesos para indicar que ha ocurrido un evento. Por ejemplo, cuando el usuario presiona Ctrl+C, el sistema envía la señal `SIGINT` al proceso activo. Las señales permiten controlar la ejecución de procesos, terminar programas, manejar errores o reaccionar a eventos del sistema. Algunas señales comunes incluyen:

- `SIGINT`: interrupción desde el teclado
- `SIGTERM`: solicitud de terminación
- `SIGKILL`: terminación forzada por el kernel
- `SIGCHLD`: notificación de que un proceso hijo ha terminado

Este modelo, definido por POSIX y adoptado por Unix y GNU/Linux, se basa en principios de aislamiento, simplicidad y composición. Los procesos son entidades independientes que se comunican explícitamente, el kernel controla el acceso a los recursos, y la abstracción de file descriptors permite unificar la interacción con el sistema. Esta arquitectura ha demostrado ser extremadamente robusta y es la base de prácticamente todos los sistemas operativos tipo Unix modernos.

4 Comandos de Administración en Linux

Definiciones de Comandos de Administración en Linux

Comando
dmesg \
Muestra el registro del kernel (mensajes del sistema), incluyendo detección de hardware, particiones, errores de
fdisk -l /dev/sdf
p (en fdisk)
n (en fdisk)
gpart
smart
efibootmgr
df -l
df -lh
mount /dev/sda3 /mnt
umount /mnt
mount -o subvol=home /dev/sda3 /mnt
mount –bind /dev /mnt/dev
mount –bind /sys /mnt/sys
less /etc/fstab
less /etc/mtab
cat /etc/os-release
chroot /mnt bash

5 Montaje de Particiones, EFI y Administración del Sistema

---

6 1. Identificación y Montaje de Particiones

En clase analizamos cómo identificar las particiones del sistema y montarlas manualmente utilizando el directorio /mnt como punto temporal de montaje.

El directorio /mnt está definido por el FHS (Filesystem Hierarchy Standard) como un punto destinado a montajes temporales realizados por el administrador.

El proceso general consistió en:

- Identificar las particiones con herramientas como fdisk o lsblk.
- Reconocer la partición raíz (/), la partición /boot y la partición /boot/efi.
- Montar manualmente las particiones para inspeccionar su contenido.
- Analizar la estructura interna del sistema de archivos.

---

7 2. EFI (EFI System Partition)

La EFI System Partition (ESP) es una partición especial utilizada en sistemas modernos que emplean UEFI (Unified Extensible Firmware Interface) en lugar del BIOS tradicional.

7.1 Características principales:

- Formateada generalmente en FAT32.
- Contiene los cargadores de arranque (.efi).
- Se monta comúnmente en /boot/efi.
- Es requerida en sistemas con tabla de particiones GPT.

7.2 Proceso de arranque en sistemas UEFI:

1. Encendido del equipo.
2. Ejecución del firmware (UEFI).
3. Localización de la partición EFI.
4. Carga del bootloader (ej. GRUB).
5. Selección y carga del kernel.
6. Inicio del sistema operativo.

La EFI permite mayor flexibilidad, soporte para discos mayores a 2TB y mecanismos como Secure Boot.

---

8 3. Dispositivos tipo /dev/sdX y contexto de "sd3"

En Linux, los dispositivos de almacenamiento se representan como block devices dentro del directorio /dev.

Ejemplos:

- /dev/sda → Primer disco detectado
- /dev/sdb → Segundo disco
- /dev/sda1 → Primera partición del disco sda
- /dev/sda3 → Tercera partición del disco sda

Cuando se mencionó "sd3", se hacía referencia probablemente a /dev/sda3, que corresponde a una partición específica del disco principal. Esta partición puede contener:

- La raíz del sistema (/)
- Un subvolumen en Btrfs

- El directorio /home
- Otro sistema de archivos independiente

En sistemas modernos es común que la raíz esté en /dev/sda3 utilizando Btrfs. Ya que las primeras dos particiones suelen justamente dedicarse al arranque, o a otro sistema.

---

9 4. Buenas prácticas: uso de usuario con privilegios sudo

Una recomendación importante fue evitar trabajar directamente como root.

9.1 Justificación técnica:

En sistemas Unix existe separación de privilegios:

- Usuario normal → permisos limitados.
- Root (UID 0) → control total del sistema.

Trabajar siempre como root implica riesgos:

- Eliminación accidental de archivos críticos.
- Modificación indebida de configuraciones.
- Mayor impacto ante errores humanos.

9.2 Buena práctica recomendada:

- Crear un usuario administrador.
- Asignarlo al grupo sudo o wheel.
- Elevar privilegios únicamente cuando sea necesario usando sudo.

Esto sigue el principio de:

Principio de mínimo privilegio (Principle of Least Privilege)

Este principio es ampliamente recomendado en comunidades técnicas, documentación oficial y foros especializados en administración Linux.

---

10 5. Sistema de archivos Btrfs

Se revisó el uso del sistema de archivos Btrfs, el cual es moderno y avanzado, diseñado para ofrecer:

- Copy-on-Write (CoW)
- Snapshots

- Subvolúmenes
- Compresión
- Integridad de datos

---

11 5.1 Opción compress=zstd:1 0 0

La opción:
compress=zstd:1
es una opción de montaje en Btrfs que habilita compresión usando Zstandard.

11.1 Componentes:

- compress → activa compresión.
- zstd → algoritmo Zstandard.
- :1 → nivel de compresión (1 = bajo, más rápido).

Ventajas:

- Reduce uso de espacio en disco.
- Mejora rendimiento en ciertos escenarios (menos I/O).
- Transparente para el usuario.

En /etc/fstab suele verse algo como:
UUID=xxxxx / btrfs defaults,compress=zstd:1 0 0
Los últimos dos valores (0 0) indican:

- Dump (respaldo) → generalmente 0.
- Orden de chequeo con fsck → generalmente 0 en Btrfs.

---

12 5.2 Subvolúmenes (subvol)

Un subvolumen en Btrfs es una división lógica dentro del mismo sistema de archivos.
No es una partición física, sino una estructura interna.
Ejemplo común:

- Subvolumen @ → raíz del sistema
- Subvolumen @home → directorio /home

Ventajas:

- Permite snapshots independientes.

- Separación lógica sin particionar físicamente.
- Facilita recuperación y administración.

Montaje ejemplo:
mount -o subvol=@home /dev/sda3 /mnt

---

13 6. Archivo /etc/fstab

El archivo /etc/fstab define qué sistemas de archivos se montan al inicio.

Incluye:

- UUID
- Punto de montaje
- Tipo de sistema de archivos
- Opciones de montaje
- Flags de dump y fsck

Es clave en administración Unix para garantizar persistencia de configuración.

---

14 7. Conceptos Complementarios Propuestos

A continuación se listan temas estructurales para desarrollo posterior dentro de la materia Administración de Unix.

Esta semana permitió comprender cómo Unix/Linux organiza el almacenamiento desde el nivel físico hasta el nivel lógico, cómo se monta el sistema de archivos y cómo el proceso de arranque interactúa con la EFI y el bootloader.

Asimismo, se reforzaron buenas prácticas administrativas como el uso de usuarios con privilegios controlados y la comprensión del sistema de archivos Btrfs y sus capacidades avanzadas.

Estos conocimientos constituyen una base esencial para la administración segura y eficiente de sistemas Unix.

15 8. Estructura lógica y administración de sistemas de archivos

15.1 ¿Qué es un sistema de archivos?

Un sistema de archivos (Filesystem) es el mecanismo lógico que utiliza el sistema operativo para organizar, almacenar y recuperar datos en un dispositivo de almacenamiento.

15.1.1 Organización lógica de datos

Permite estructurar la información en archivos y directorios jerárquicos, independientemente de cómo estén físicamente almacenados los bloques en el disco.

Ejemplo: Un archivo llamado documento.txt puede estar dividido en múltiples bloques físicos, pero el sistema lo presenta como una unidad lógica.

15.1.2 Relación con particiones

Un sistema de archivos se crea sobre una partición o dispositivo. La partición define el espacio físico; el sistema de archivos define cómo se organiza ese espacio.

Disco → Partición → Sistema de archivos → Archivos y directorios

15.1.3 Abstracción sobre el hardware

El sistema de archivos actúa como una capa de abstracción entre el usuario y el hardware, ocultando detalles como sectores, cilindros y bloques físicos.

—

15.2 Sistemas de archivos comunes en Linux

15.2.1 ext4

Sistema de archivos por defecto en muchas distribuciones Linux. Soporta journaling, archivos grandes y buena estabilidad.

15.2.2 xfs

Optimizado para alto rendimiento y manejo de grandes volúmenes de datos. Muy usado en servidores.

15.2.3 btrfs

Sistema moderno con características avanzadas como snapshots, compresión y subvolúmenes.

15.2.4 vfat

Compatible con sistemas Windows. Usado comúnmente en memorias USB.

15.2.5 ntfs

Sistema de archivos de Windows. Linux puede leerlo y escribirlo mediante controladores como ntfs-3g.

—

15.3 Estructura interna básica de un archivo

15.3.1 Superbloque

Contiene información global del sistema de archivos:

- Tamaño total
- Número de inodos
- Tamaño de bloque
- Estado del sistema

Es esencial para montar el sistema.

15.3.2 Inodos

Estructuras que almacenan metadatos del archivo:

- Permisos
- Propietario
- Tamaño
- Punteros a bloques de datos

Importante: el nombre del archivo NO está en el inodo, sino en el directorio.

15.3.3 Bloques de datos

Son las unidades físicas donde se almacena el contenido real del archivo.

15.3.4 Journaling

Mecanismo que registra cambios antes de aplicarlos. Permite recuperación ante fallos inesperados (apagones).
Ejemplo: ext4 usa journaling para evitar corrupción tras un corte eléctrico.

15.3.5 Metadatos

Información sobre el archivo, no su contenido. Ejemplo:

- Fecha de creación
- Permisos
- Tamaño
- UID y GID

—

15.4 Creación y formateo de particiones

Formatear significa crear un sistema de archivos dentro de una partición.

15.4.1 mkfs

Comando general para crear sistemas de archivos.
Ejemplo:

```
mkfs -t ext4 /dev/sdb1
```

15.4.2 mkfs.ext4

Forma específica para crear ext4.

```
mkfs.ext4 /dev/sdb1
```

15.4.3 mkfs.xfs

Para crear sistema de archivos XFS.

```
mkfs.xfs /dev/sdb1
```

15.4.4 Verificación con fsck

Herramienta para revisar y reparar sistemas de archivos.

```
fsck /dev/sdb1
```

```
fsck from util-linux 2.41.3
```

—

15.5 Montaje

Montar significa asociar un sistema de archivos a un directorio del sistema.

15.5.1 mount

Permite montar manualmente.

```
mount /dev/sdb1 /mnt
```

15.5.2 umount

Desmonta el sistema.

```
umount /mnt
```

15.5.3 Puntos de montaje (/mnt, /media)

- /mnt → montajes temporales manuales.
- /media → dispositivos removibles (USB).

15.5.4 Montaje temporal vs permanente

Temporal: Se pierde al reiniciar.
Permanente: Se configura en /etc/fstab para montarse automáticamente.
—

15.6 Archivo /etc/fstab

Archivo de configuración que define qué sistemas de archivos se montan al inicio.
Formato general: dispositivo $\text{punto}_{\text{montaje}}$ tipo opciones dump pass

15.6.1 Montaje automático al inicio

Permite que las particiones se monten automáticamente durante el arranque.

15.6.2 UUID vs nombre de dispositivo

- /dev/sda1 puede cambiar.
- UUID es único y estable.

Ejemplo:

```
UUID=xxxx-xxxx /home ext4 defaults 0 2
```

15.6.3 Opciones comunes

- defaults → configuración estándar
- noatime → no actualiza fecha de acceso
- ro → solo lectura
- rw → lectura y escritura

---

16 9. Organización del sistema, permisos y almacenamiento avanzado

16.1 FHS (Filesystem Hierarchy Standard)

Estándar que define cómo se organizan los directorios en sistemas Unix/Linux.
Garantiza consistencia entre distribuciones.
—

16.2 Estructura de directorios

16.2.1 /

Raíz del sistema. Todo parte desde aquí.

16.2.2 /home

Directorios personales de los usuarios.

16.2.3 /etc

Archivos de configuración del sistema.

16.2.4 /var

Archivos variables:

- logs
- bases de datos
- spool

16.2.5 /usr

Programas y bibliotecas del sistema.

16.2.6 /boot

Archivos necesarios para el arranque:

- kernel
- initramfs
- bootloader

16.2.7 /dev

Archivos especiales que representan dispositivos.
Ejemplo: /dev/sda → disco /dev/null → dispositivo nulo

16.2.8 /proc

Sistema virtual que expone información del kernel y procesos en tiempo real.

16.2.9 /sys

Sistema virtual que expone información del hardware y dispositivos.
—

16.3 Permisos y Propiedad

Cada archivo tiene permisos para:

- Usuario (owner)
- Grupo
- Otros

16.3.1 Lectura (r)

Permite ver contenido.

16.3.2 Escritura (w)

Permite modificar.

16.3.3 Ejecución (x)

Permite ejecutar como programa.

16.3.4 Ver permisos

```
ls -la
```

total	464							
drwxr-xr-x	1	mrtaichi	mrtaichi	158	Feb	18	23:07	.
drwxr-xr-x	1	mrtaichi	mrtaichi	186	Feb	18	21:37	..
-rw-r–r–	1	mrtaichi	mrtaichi	23480	Feb	18	21:53	bitacora1.org
-rw-r–r–	1	mrtaichi	mrtaichi	250680	Feb	18	21:54	bitacora1.pdf
-rw-r–r–	1	mrtaichi	mrtaichi	30106	Feb	18	21:54	bitacora1.tex
-rw-r–r–	1	mrtaichi	mrtaichi	20686	Feb	18	23:07	bitacora2.org
drwxr-xr-x	1	mrtaichi	mrtaichi	166	Feb	18	23:07	.git
-rw-r–r–	1	mrtaichi	mrtaichi	25	Feb	18	21:54	test.txt
-rw-r–r–	1	mrtaichi	mrtaichi	133687	Feb	7	12:15	ventoy$_{\text{boot.jpg}}$

Ejemplo de salida: -rwxr-xr– 1 user group 1024 archivo.sh
—

16.4 Conceptos de Usuarios y grupos

16.4.1 Propietario

Usuario dueño del archivo.

16.4.2 Grupo

Conjunto de usuarios con permisos compartidos.

16.4.3 Otros

Todos los demás usuarios.

16.4.4 Comandos relacionados

1. chmod Modifica permisos.

   ```
   chmod 755 archivo.sh
   ```

2. chown Cambia propietario.

   ```
   chown user archivo.txt
   ```

3. chgrp Cambia grupo.

   ```
   chgrp developers archivo.txt
   ```

   —

16.5 Permisos especiales

16.5.1 SUID

Permite ejecutar un archivo con privilegios del propietario.
Ejemplo típico: passwd

16.5.2 SGID

Similar al SUID pero aplicado al grupo. En directorios, los archivos heredan el grupo.

16.5.3 Sticky bit

En directorios compartidos, solo el propietario puede eliminar sus propios archivos.
Ejemplo: /tmp
—

16.6 Gestión Avanzada de Almacenamiento

16.6.1 LVM (Logical Volume Manager)

Permite administrar almacenamiento de forma flexible.

1. Volúmenes físicos (PV) Discos o particiones inicializadas para LVM.
2. Grupos de volúmenes (VG) Agrupación de múltiples PV.
3. Volúmenes lógicos (LV) Particiones virtuales creadas dentro de un VG.
   Ventaja: Permite redimensionar sin reorganizar físicamente el disco.

16.7 Sistemas de archivos en red

Permiten compartir almacenamiento entre máquinas.

16.7.1 NFS

Network File System. Común en entornos Linux/Unix.
Permite montar un directorio remoto como si fuera local.

16.7.2 SMB

Server Message Block. Protocolo usado por Windows. En Linux se implementa mediante Samba.