

# MÓDULO I. Administración de GNU/Linux

## Sesión 1. Configuración del entorno de prácticas

2025/07/01

### Índice

1. Introducción y Objetivos Principales . . . . .	1
2. ¿Qué es un Contenedor? . . . . .	3
3. ¿Por qué usar Contenedores en nuestras Prácticas? . . . . .	3
4. ¿Qué es Docker y Podman? . . . . .	4
5. Instalación de Podman en Ubuntu . . . . .	4
6. ¿Qué es un Dockerfile? . . . . .	5
7. Preparar nuestro Sistema para las prácticas . . . . .	8
8. Administración de Usuarios y Grupos en Linux . . . . .	10
8.1. Usuario Administrador del Sistema en Linux: root . . . . .	10
8.2. Gestión de Usuarios . . . . .	12
8.3. Gestión de Grupos . . . . .	17
8.4. Usuarios y Grupos Especiales . . . . .	18
9. Organización del Sistema de Archivos y Gestión Básica de Archivos . . . . .	19
La Organización del Sistema de Archivos en UNIX/Linux . . . . .	19
9.1. Organización común en sistemas de archivos tipo Linux. Filesystem Hierarchy Standard (FHS) . . . . .	21
9.2. Órdenes Básicas para Gestión del Sistema de Archivos. . . . .	22
9.2.1. Acceso a Información del Sistema de Archivos y Puntos de Montaje . . . . .	23

### 1. Introducción y Objetivos Principales

Verás que esta guía se habla de “Linux” constantemente. Aunque la mayoría de la gente piensa en un sistema operativo completo, técnicamente, ‘Linux’ es solo el *núcleo* (kernel) del sistema operativo. Este núcleo es la parte central que gestiona los recursos del hardware y permite que el software se comuniquen con él. Para tener un sistema operativo funcional, necesitas muchas otras herramientas y utilidades: un

shell (intérprete de órdenes), compiladores, editores de texto, bibliotecas o librerías, etc. Aquí es donde entra el proyecto GNU. El proyecto GNU (un acrónimo recursivo para “GNU’s Not Unix”) ha desarrollado una vasta colección de software libre que proporciona la mayoría de estas herramientas esenciales para construir un sistema operativo completo. Por lo tanto, lo que comúnmente usamos y conocemos como ‘Linux’ es en realidad una combinación del núcleo Linux con las herramientas del proyecto GNU. De ahí que el nombre más preciso sea GNU/Linux. Las diferentes “versiones” o *flavors* (sabores) de este sistema (como Ubuntu, Fedora, Debian) se llaman *distribuciones GNU/Linux*.

Ahora bien, ¿qué pasa si usas Windows como sistema operativo principal? Microsoft ha desarrollado una característica llamada **Windows Subsystem for Linux (WSL)**. Esto te permite ejecutar un entorno GNU/Linux completo directamente dentro de Windows, sin necesidad de una máquina virtual pesada o de instalar un sistema operativo dual. WSL te da acceso a la línea de órdenes, herramientas y aplicaciones de Linux de forma casi nativa, integrándose muy bien con tu sistema Windows. Para nuestras prácticas, tendrás varias opciones, todas ellas en principio válidas para trabajar con GNU/Linux:

1. **Windows con WSL:** Si ya utilizas Windows, esta es la opción a priori más sencilla y rápida para empezar, ya que te permite tener un entorno Linux funcional sin salir de tu sistema. El problema es que lidiarás con las modificaciones que Microsoft decida hacer, y no vas a encontrar un tutorial de cómo configurarlo en esta guía.
2. **Máquina Virtual (VirtualBox con Ubuntu):** Puedes instalar una máquina virtual (usando software gratuito como VirtualBox) y dentro de ella instalar una distribución GNU/Linux como Ubuntu. Esto te proporciona un entorno Linux completamente aislado de tu sistema principal. **Esta es la opción recomendada en la asignatura.**
3. **Instalación Nativa de GNU/Linux:** Si lo deseas y tienes los suficientes conocimientos adquiridos, puedes instalar una distribución GNU/Linux directamente en tu ordenador como sistema operativo principal o junto a Windows (dual-boot). Ten cuidado, porque si te equivocas puedes perder los datos.

Tu profesor o profesora de prácticas te orientará en cuál es la opción más recomendable, no tengas miedo en preguntarle. A partir de ahora, y para simplificar, usaremos los términos GNU/Linux y Linux indistintamente en los materiales del curso, refiriéndonos siempre al sistema operativo completo.

Esta sesión está diseñada para familiarizarte con el entorno de trabajo que utilizaremos para la mayoría de las prácticas del Módulo I de la asignatura de Sistemas Operativos. Dada la naturaleza de la administración de sistemas operativos y la necesidad de altos privilegios (como root), el uso de contenedores nos proporciona un entorno aislado, consistente y seguro para experimentar sin afectar al sistema operativo anfitrión.

Una vez que te familiarices con el entorno de trabajo en el contenedor, abordarás la “Administración de usuarios y grupos en Linux” (Sección 8). Aquí comprenderás conceptos como usuario, cuenta de usuario, tipos de usuarios, grupo de usuarios y tipos de grupos, utilizando las herramientas básicas para su gestión.

Finalmente, la sesión se centrará en la “Organización del Sistema de Archivos (SA) y gestión básica de archivos” (Sección 9). Esto te permitirá enlazar conocimientos previos de Fundamentos del Software y extenderlos con la comprensión del Filesystem Hierarchy Standard (FHS) y la utilización de programas para la gestión de archivos y sistemas de archivos.

### Objetivos principales:

- Comprender el concepto de contenedor y sus ventajas en el desarrollo y la administración de sistemas.
- Conocer y saber usar Podman como herramienta para gestionar entornos de desarrollo aislados.
- Aprender a construir y gestionar imágenes de contenedores mediante Dockerfiles.
- Configurar tu sistema para utilizar los scripts proporcionados para la gestión de entornos de prácticas con Podman.
- Conocer los tipos de usuarios de un sistema operativo Linux y sus funciones.
- Saber crear cuentas de usuarios y grupos.
- Conocer la organización del estándar FHS.
- Saber utilizar las órdenes básicas para gestionar un sistema de archivos Linux.

## 2. ¿Qué es un Contenedor?

Un **contenedor** es una **unidad de software** que **empaqueta** el **código de una aplicación** y todas sus **dependencias necesarias** (bibliotecas, configuraciones, archivos, etc.) para que se ejecute de manera rápida y confiable en cualquier entorno informático. Se pueden considerar entornos ligeros y portátiles, ideales para el desarrollo y despliegue de aplicaciones, una habilidad crucial en múltiples asignaturas a lo largo de tu grado.

A **diferencia de las máquinas virtuales (VMs)**, que incluyen un sistema operativo invitado completo, los **contenedores** comparten el **mismo sistema operativo subyacente** (el kernel del sistema anfitrión). Esto significa que, si estás **usando un sistema GNU/Linux**, **no podrías ejecutar directamente un contenedor de MS Windows**. Sin embargo, esta característica los hace significativamente más eficientes en términos de consumo de recursos y tiempo de arranque que las máquinas virtuales.

## 3. ¿Por qué usar Contenedores en nuestras Prácticas?

El uso de contenedores en nuestras prácticas de sistemas operativos ofrece múltiples **ventajas clave**, especialmente al realizar el despliegue y la administración de aplicaciones y servicios:

- **Aislamiento:** Los contenedores permiten ejecutar aplicaciones en entornos totalmente aislados, previniendo conflictos con otras aplicaciones o con el sistema anfitrión. Cada contenedor posee su propio sistema de archivos, red y espacio de procesos, asegurando que los cambios realizados en un contenedor no afecten a otros ni al sistema base. Esto es particularmente útil cuando diferentes programas requieren versiones distintas de una misma biblioteca.

- **Portabilidad:** Una vez creada, puedes mover un contenedor entre diferentes entornos (tu máquina local, el laboratorio, un servidor de pruebas, un entorno de producción) sin problemas. Al incluir todas las dependencias necesarias, el contenedor garantiza que la aplicación se ejecutará exactamente de la misma manera, sin importar dónde se despliegue.
- **Consistencia:** Garantizan que la aplicación se ejecute de forma idéntica en cualquier entorno. Esto es invaluable en equipos de desarrollo o en un entorno educativo donde diferentes estudiantes o profesores pueden estar trabajando en sistemas operativos o configuraciones ligeramente distintas. Con contenedores, todos operan en un entorno idéntico y predecible.
- **Escalabilidad:** Facilitan enormemente la escalabilidad de las aplicaciones. Puedes desplegar múltiples instancias de un mismo contenedor para manejar picos de carga de trabajo y, de igual manera, reducirlas cuando la demanda disminuya.
- **Eficiencia de Recursos:** Al compartir el kernel del sistema operativo subyacente, los contenedores son mucho más ligeros y utilizan significativamente menos recursos (CPU, RAM, disco) que las máquinas virtuales. Esto permite ejecutar más contenedores en el mismo hardware, optimizando el uso de los recursos disponibles.

## 4. ¿Qué es Docker y Podman?

**Docker** Docker es la plataforma más popular y ampliamente adoptada para el desarrollo, envío y ejecución de aplicaciones en contenedores. Facilita la creación de contenedores utilizando “imágenes” predefinidas, que son plantillas completas que contienen todo lo necesario para ejecutar una aplicación, incluyendo el código, las bibliotecas, las configuraciones y la versión del sistema operativo base. Docker también ha impulsado un vasto ecosistema, con Docker Hub como un repositorio central en línea para compartir y descargar imágenes de contenedores.

**Podman** Podman es una alternativa moderna a Docker que también permite la gestión completa de contenedores. Su principal diferenciador es que **no requiere un demonio en ejecución** (un proceso de fondo que gestiona los contenedores, como `dockerd`). Esto significa que cada contenedor en Podman se ejecuta como un proceso hijo directo del usuario que lo lanza. Además, una de sus características más destacadas es su capacidad para **ejecutar contenedores sin permisos de superusuario (rootless)**, lo que incrementa significativamente la seguridad.

Para nuestras prácticas, utilizaremos **Podman** debido a sus ventajas en seguridad y su compatibilidad total con las imágenes y órdenes de Docker. De hecho, su compatibilidad es tan alta que muchos usuarios suelen crear un alias para trabajar de forma transparente:

```
alias docker=podman
```

## 5. Instalación de Podman en Ubuntu

Para instalar Podman en un sistema Ubuntu, sigue los siguientes pasos detallados. Asegúrate de tener conexión a Internet.

1. **Actualiza los paquetes del sistema:** Es una buena práctica asegurarse de que todos los paquetes existentes estén actualizados.

```
sudo apt update  
sudo apt upgrade -y
```

2. **Instala el paquete de Podman:** Procedemos a instalar el paquete principal de Podman.

```
sudo apt install -y podman
```

3. **Verifica la instalación:** Comprueba que Podman se ha instalado correctamente y visualiza su versión.

```
podman --version
```

4. **Crea el archivo de configuración de registros:** Podman necesita saber de qué registros de imágenes puede descargar imágenes. Vamos a configurar los registros más comunes.

Primero, crea el directorio `containers` si no existe:

```
mkdir -p "$HOME/.config/containers"
```

Luego, crea y edita el archivo `registries.conf`, puedes usar cualquier editor del sistema como `nano`, `emacs`, `vi`, `gedit`, etc., o cualquier otro que quieras. En este ejemplo usamos `nano`:

```
nano "$HOME/.config/containers/registries.conf"
```

Dentro de este archivo, introduce la siguiente línea. Esta lista de destinos permite a Podman buscar y descargar imágenes de los repositorios indicados. Más adelante, al descargar imágenes preguntará cuál de estos servidores usar:

```
unqualified-search-registries = ['docker.io', 'nvcr.io', 'quay.io']
```

Guarda y cierra el archivo.

¡Ahora ya tienes Podman instalado y configurado! Podemos probarlo ejecutando una orden básica en un contenedor de Ubuntu. Por ejemplo, la siguiente orden ejecutará `bash` de forma interactiva (`-it`) en un contenedor `ubuntu:20.04`, y eliminará el contenedor automáticamente cuando el proceso termine (`--rm`):

```
podman run -it --rm ubuntu:20.04 bash
```

Observarás que, aunque tengas ya un entorno de GNU/Linux o WSL, aún no tendrás posiblemente muchas utilidades instaladas. Para automatizar la creación de entornos personalizados, utilizaremos Dockerfiles.

## 6. ¿Qué es un Dockerfile?

Un **Dockerfile** es un archivo de texto simple que contiene una serie de instrucciones secuenciales. Estas instrucciones son utilizadas por Podman (o Docker) para construir una imagen de contenedor. Cada

instrucción define una capa de la imagen, lo que permite una construcción eficiente y reutilizable. El Dockerfile especifica cómo se debe configurar el entorno dentro del contenedor para que la aplicación se ejecute correctamente.

### Ejemplo de fichero Dockerfile:

```
# Usa una imagen base de Ubuntu
```

```
FROM ubuntu:20.04
```

```
# Actualiza los paquetes y instala curl
```

```
RUN apt-get update && apt-get install -y curl
```

```
# Realiza la instalación de los paquetes extras
```

```
RUN apt-get install -y bash fdisk cron at gcc make nano vim emacs adduser man-db
```

```
# Establece el directorio de trabajo predeterminado dentro del contenedor
```

```
WORKDIR /home/
```

```
# Copia uno de nuestros archivos al contenedor, esto copia 'mi_archivo.sh'
```

```
# del directorio local al WORKDIR del contenedor
```

```
COPY ./mi_archivo.sh .
```

```
# Ejecuta la aplicación cada vez que se lance el contenedor por defecto.
```

```
# Si no se especifica una orden al ejecutar el contenedor, se ejecutará 'whoami'
```

```
CMD ["whoami"]
```

### Explicación de las instrucciones clave en un Dockerfile:

- **FROM ubuntu:20.04:** Indica la imagen base sobre la que se construirá la nueva imagen. En este caso, se descarga la imagen oficial de Ubuntu versión 20.04 desde Docker Hub (gracias a la configuración en `registries.conf`).
- **RUN apt-get update && apt-get install -y curl:** Ejecuta órdenes dentro de la imagen durante el proceso de construcción. Se utiliza para instalar software, actualizar paquetes, etc. Cada instrucción RUN crea una nueva capa en la imagen. Se ha añadido `man-db` para que la orden `man` esté disponible.
- **WORKDIR /home/:** Establece el directorio de trabajo predeterminado para las instrucciones RUN, CMD, ENTRYPOINT, COPY, y ADD que le sigan. Si no existe, se crea.
- **COPY ./mi\_archivo.sh .:** Copia archivos o directorios de la máquina local (desde donde se ejecuta la orden `podman build`) al sistema de archivos del contenedor. El primer argumento es la fuente en el host, el segundo es el destino en el contenedor (relativo a WORKDIR si no es una ruta absoluta).
- **CMD ["whoami"]:** Define la orden por defecto que se ejecutará cuando se inicie el contenedor sin

especificar una orden explícita. Si se pasa una orden al `podman run`, esta reemplazará a la orden CMD. Todo esto es muy útil para crear contenedores que ofrecen servicios. Si te animas, con los conocimientos previos que tienes, más los que adquieras al finalizar la sesión, sabrás suficientes órdenes como para crear tus propios contenedores personalizados.

### Construyendo tu imagen:

Para construir una imagen a partir de un Dockerfile (por ejemplo, llamada `mi-imagen`), navega al directorio donde se encuentra tu Dockerfile y ejecuta:

```
podman build -t mi-imagen .
```

- `-t mi-imagen`: Asigna el nombre `mi-imagen` a la imagen resultante.
- `.`: Indica que el Dockerfile se encuentra en el directorio actual.

### Listando tus imágenes:

Una vez construida la imagen, puedes verla listada con:

```
podman image ls
```

**¡Cuidado!** No confundas `podman image ls` (para listar imágenes) con `podman images ls` (que es otra cosa).

### Ejecutando tu imagen:

Ahora puedes ejecutar tu imagen personalizada:

```
podman run -it --rm mi-imagen
```

Observa que en esta ocasión no le hemos indicado ninguna orden explícita a ejecutar. Por defecto, Podman ejecutará la orden que viene especificada con CMD en el Dockerfile (`whoami` en este ejemplo). Esto es extremadamente útil para lanzar servicios como bases de datos, servidores web, proveedores de API, etc. En esta asignatura no veremos este aspecto, pero más adelante en la carrera sí que te será útil este conocimiento.

### Borrando una imagen:

Las imágenes ocupan espacio en disco. Para borrar una imagen que ya no necesitas:

```
podman rmi mi-imagen
```

Es importante que no haya ningún contenedor ejecutándose a partir de esa imagen para poder borrarla.

### Listando y matando procesos de contenedores:

Para ver los contenedores en ejecución (y sus IDs):

```
podman ps
```

La primera columna (CONTAINER ID) te mostrará el identificador único del proceso. Para detener un contenedor específico, puedes usar su ID:

```
podman container kill <ID_del_contenedor>
```

Por ejemplo:

```
podman container kill d75011cb1ccb
```

Donde d75011cb1ccb sería el ID que apareció en la instrucción `podman ps`.

## 7. Preparar nuestro Sistema para las prácticas

Para facilitar la gestión de los entornos de prácticas, hemos preparado una serie de scripts que automatizan las tareas más comunes.

1. **Descarga los scripts:** Puedes descargar los scripts directamente desde su repositorio de GitHub. Para ello, primero instala `git` si no lo tienes:

```
sudo apt-get update
sudo apt-get -y install git
```

Luego, clona el repositorio y asigna permisos de ejecución:

```
git clone https://github.com/german-arroyo-moreno/scripts_podman_so.git

chmod +x scripts_podman_so/*.sh -R
```

Esto descargará los scripts en un directorio llamado `scripts_podman_so` en tu directorio actual.

2. **Ejecutar el script de instalación (`instalacion.sh`):** Este script es para la configuración inicial. Creará los ficheros necesarios para la lista de servidores, instalará el paquete Podman (si no lo hiciste manualmente antes), y en general, realizará la primera configuración.

```
./scripts_podman_so/instalacion.sh
```

Este script no requiere de argumentos adicionales, y creará una imagen base llamada `ubuntu-so` que usaremos en las prácticas.

**Importante:** Este script requerirá permisos de administrador (`sudo`) para ejecutar algunas órdenes de instalación. Se te pedirá tu clave de root del ordenador donde estés realizando la instalación. Este proceso solo debe realizarse una vez (a menos que desinstales la imagen por defecto).

3. **Ejecutar el contenedor de prácticas (`ejecutar.sh`):** Cada vez que quieras comenzar una sesión de prácticas, deberás ejecutar este script.

```
./scripts_podman_so/ejecutar.sh [nombre_de_la_imagen]
```



- [nombre\_de\_la\_imagen] (opcional): Si deseas utilizar una imagen diferente a la predeterminada (ubuntu-so), puedes proporcionar su nombre como argumento.

Por ejemplo, para ejecutar el entorno de prácticas predeterminado:

```
./scripts_podman_so/ejecutar.sh ubuntu-so
```

Esta orden es importante, ya que posiblemente te permita acceder a las imágenes de examen más adelante. También podrías usarlo para lanzar un contenedor basado en una de tus propias imágenes (por ejemplo, mi-imagen creada anteriormente):

```
./scripts_podman_so/ejecutar.sh mi-imagen
```

**Comprensión del ejecutar.sh:** Si examinas el contenido de este script, verás una línea similar a esta:

```
podman run -it --rm --name "$imagen" \
    -v "$($pwd)":/mnt/disco_local \
    -w /mnt/disco_local "$imagen" bash
```

- -v "\$(\$pwd)":/mnt/disco\_local: Este es un parámetro clave que permite **montar un volumen**. Esto significa que el directorio donde te encuentres en tu máquina local (\$(pwd)) se montará y será accesible dentro del contenedor en la ruta /mnt/disco\_local. Todos los cambios que realices dentro de /mnt/disco\_local en el contenedor serán persistentes en tu máquina local, el resto se borrarán al matar o cerrar el proceso.
- -w /mnt/disco\_local: Establece el directorio de trabajo predeterminado dentro del contenedor a /mnt/disco\_local, lo que te coloca directamente en tu espacio de trabajo local al iniciar el contenedor.
- --rm: Elimina el contenedor automáticamente al salir.

**Persistencia de datos:** Recuerda que, una vez que salgas de este bash (usando exit, Ctrl-D o Ctrl-C), todos los cambios que *no* se hayan realizado en el volumen montado (/mnt/disco\_local) se perderán. Esto incluye la instalación de paquetes, la modificación de archivos fuera de /mnt/disco\_local, etc.

4. **Guardar los cambios en un contenedor (guardar-contenedor.sh):** Si realizas cambios significativos dentro de un contenedor (instalación de paquetes, configuración de servicios) y deseas que estos cambios persistan para futuras sesiones, puedes crear una nueva imagen a partir del contenedor en ejecución.

```
./scripts_podman_so/guardar-contenedor.sh <id_contenedor> <nueva_imagen>
```

- <id\_contenedor>: Debes proporcionar el ID del contenedor que está en ejecución. Puedes obtenerlo con `podman ps`. También, si el prompt de tu contenedor es:

```
root@ce81789e7b08:/mnt/disco_local
```

el ID es la cadena que representa la máquina, en este caso: ce81789e7b08.

- `<nueva_imagen>`: Es el nombre que deseas darle a la nueva imagen que se creará a partir de los cambios de tu contenedor.

Fíjate que cuando las líneas son muy largas, es común fraccionarlas escapando el retorno de línea con `\` al final de línea (es muy importante que la línea no termine con espacios en blanco).

#### Ejemplo de uso:

```
./scripts_podman_so/guardar-contenedor.sh ce81789e7b08 \
mi_nueva_imagen_con_cambios
```

Después de esto, podrás iniciar un nuevo contenedor basado en `mi_nueva_imagen_con_cambios` y todos tus cambios previos estarán presentes.

5. **Desinstalar imágenes y contenedores (`desinstalacion.sh`):** Si deseas liberar espacio en disco o limpiar tu entorno, puedes usar este script para eliminar una imagen y todos los contenedores basados en ella.

```
./scripts_podman_so/desinstalacion.sh <nombre_de_la_imagen>
```

- `<nombre_de_la_imagen>`: El nombre de la imagen que deseas eliminar (ej. `ubuntu-so`, `mi_nueva_imagen_con_cambios`).

#### Ejemplo de uso:

```
./scripts_podman_so/desinstalacion.sh ubuntu-so
```

Con estas herramientas, tendrás un control robusto y flexible sobre tu entorno de prácticas, permitiéndote experimentar con la administración de sistemas Linux de forma segura y eficiente.

## 8. Administración de Usuarios y Grupos en Linux

Una vez que hayas ejecutado el script `ejecutar.sh` y te encuentres dentro del contenedor, notarás que el prompt indica que estás trabajando como `root` (por ejemplo, `root@<ID_contenedor>:/mnt/disco_local#`). Esto te proporciona los privilegios necesarios para realizar las tareas de administración de sistemas que veremos a continuación.

### 8.1. Usuario Administrador del Sistema en Linux: `root`

El usuario administrador del sistema, también conocido como superusuario, es el usuario que posee todos los privilegios sobre cualquier archivo, instrucción u orden del sistema. En Linux, y en cualquier sistema UNIX, este usuario se identifica con el nombre de usuario `root`, pertenece al grupo `root` y su directorio home es `/root`.

El administrador del sistema debe tener amplios conocimientos de todo el sistema (hardware, software, datos, usuarios, etc.), una buena capacidad para tomar decisiones, ser eficaz y responsable, ya que trabaja con datos muy importantes.

Las tareas asignadas a un administrador del sistema suelen ser:

- Añadir nuevos usuarios (Linux es un sistema operativo multiusuario).
- Controlar el rendimiento del sistema.
- Realizar copias de seguridad (y restaurarlas).
- Añadir/eliminar elementos de hardware (virtualizados en un contenedor).
- Instalar/actualizar/desinstalar software.
- Controlar la seguridad del sistema.
- Controlar el correcto arranque del sistema (del contenedor o host).
- Monitorización del sistema.
- Localizar y resolver problemas del sistema.
- Resolver dudas de los usuarios.
- Etc.

Dentro de un sistema Linux/UNIX, es posible iniciar sesión directamente como root. Sin embargo, si ya has iniciado un shell con otro usuario (por ejemplo, si más adelante creas un usuario no root dentro de tu contenedor y cambias a él), puedes usar la orden `su` (o indistintamente la orden `sudo`, que se explicará más adelante) para cambiar temporalmente al usuario root. Esta orden solicitará la contraseña de root y lanzará un nuevo proceso que ejecutará el mismo programa shell pero con privilegios de root.

```
patricia@container-id:/$ whoami # Pido a la shell el nombre de usuario actual
patricia
```

```
patricia@container-id:/$ sudo bash # Solicitud de cambio a usuario root
Password: # El sistema solicita la contraseña de root
```

^D

```
patricia@container-id:/$ whoami # Pido a la shell el nombre de usuario actual
patricia
```

```
patricia@container-id:/$ su # Solicitud de cambio a usuario root
Password: # El sistema solicita la contraseña de root
```

```
root@container-id:/# whoami # Suele cambiar el prompt del sistema
root
```

**Nota:** Cuando estemos trabajando con un sistema Linux/UNIX, por seguridad, deberíamos hacerlo siempre bajo una cuenta de usuario normal y no usar la cuenta del administrador (root), a no ser que

queramos realizar tareas de administración. Debemos tener en cuenta que el administrador es el usuario con el mayor privilegio.

## 8.2. Gestión de Usuarios

Un **usuario** (user) es una persona que trabaja en el sistema mediante una cuenta de usuario a la que accede a través de una identificación. En Linux, un usuario se caracteriza por:

- **Su nombre de usuario**, también conocido como username.
- **Su identificador de usuario (UID)**, del inglés User IDentifier, que es un número entero que le asigna internamente el sistema y que lo representa (el sistema operativo no trabaja con su nombre sino con su UID). El UID de root es el 0.
- **El grupo o grupos a los que pertenece (GID)**, del inglés Group Identifier. Un usuario tiene asignado un grupo principal (primary group), que es el grupo que aparece especificado en el archivo /etc/passwd, pero puede pertenecer a más de un grupo. Los grupos adicionales a los que puede pertenecer un usuario se denominan grupos suplementarios (supplementary groups). Todos los grupos y los usuarios que pertenecen a cada grupo están especificados en el archivo /etc/group. El GID principal del superusuario es el 0.

Como acabamos de ver, la información relativa a usuarios y grupos es almacenada por el SO en varios archivos de texto plano. A continuación se muestra una tabla que incluye el nombre de estos archivos y una breve descripción de su contenido. Te animamos a visualizar estos archivos por pantalla (por ejemplo, con `cat /etc/passwd`) para familiarizarte con el formato y contenido que almacenan.

**Tabla 1. Archivos que especifican los usuarios, grupos y contraseñas (passwords) del sistema.**

Archivo	Descripción
/etc/passwd	Almacena información de las cuentas de usuarios
/etc/shadow	Guarda los passwords encriptados e información de “envejecimiento” de las cuentas
/etc/group	Definición de los grupos y usuarios miembros

**a) Creación de cuentas de usuario** En un sistema GNU/Linux, la información sobre los usuarios y los grupos a los que pertenecen se almacena en archivos de texto plano específicos. Comprender la estructura de estos archivos es crucial para entender cómo funciona la gestión de usuarios, incluso si en la práctica utilizaremos herramientas automáticas para simplificar la tarea.

Los archivos principales involucrados son:

- /etc/passwd: Contiene la información básica de cada usuario del sistema (nombre de usuario, ID de usuario (UID), ID de grupo principal (GID), directorio de inicio, shell por defecto, etc.).
- /etc/group: Almacena la información sobre los grupos y los usuarios que pertenecen a ellos.

- `/etc/shadow`: Este archivo guarda las contraseñas de los usuarios de forma cifrada. Es un archivo de acceso restringido por motivos de seguridad.

Para añadir un nuevo usuario al sistema, conceptualmente se deben seguir una serie de pasos que implican la manipulación de la información en estos archivos. Si bien **podrías modificarlos manualmente** (lo cual te daría un control total y una comprensión profunda), es un proceso propenso a errores que podría dejar tu sistema inestable si no se hace correctamente. Por ello, en la gestión diaria, se prefieren las **herramientas automáticas**.

A continuación, enumeramos los pasos conceptuales para la creación de un usuario, y luego veremos las herramientas que los automatizan:

1. **Decidir el nombre de usuario (username)** y los grupos a los que va a pertenecer el usuario que utilizará esa cuenta (grupo principal y grupos suplementarios).
2. **Registrar la información en `/etc/passwd` y `/etc/group`**. Por ejemplo, una línea en `/etc/passwd` para un usuario alumno podría verse así (cada campo separado por `:`):

```
alumno:x:1001:1001:Alumno de Prácticas:/home/alumno:/bin/bash
```

El campo `x` indica que la contraseña real está en `/etc/shadow`. Y en `/etc/group`, para un grupo practicas al que pertenece alumno:

```
practicas:x:1002:alumno
```

3. **Establecer la contraseña**: Esto se hace en `/etc/shadow`. Se usa la orden `passwd` dado que viene cifrada.
4. **Establecer los parámetros de envejecimiento** de la cuenta (caducidad de la contraseña, etc.).
5. **Crear el directorio de inicio del nuevo usuario (HOME)**, normalmente en `/home`, y establecer el propietario, grupo y permisos adecuados.
6. **Copiar los archivos de inicialización** del shell (`.bash_profile`, `.bashrc`, etc.) desde un directorio plantilla (generalmente `/etc/skel`) al nuevo directorio HOME.
7. **Establecer otras facilidades** específicas: cuotas de disco, permisos para imprimir, etc.
8. **Ejecutar cualquier tarea de inicialización** propia del sistema.
9. **Probar la nueva cuenta**.

Se recomienda utilizar la orden `man` (Sección 1.3) para consultar el funcionamiento y todas las opciones de las órdenes; por ejemplo:

```
man useradd
```

Las herramientas automáticas para la creación de cuentas de usuario realizan la mayoría de las tareas básicas anteriores de forma segura y eficiente. Son las que recomendamos usar para la gestión diaria:

- **useradd**: La herramienta de **bajo nivel estándar**. Generalmente, `useradd` es la orden base para la creación de usuarios. Se utiliza con múltiples opciones para especificar todos los detalles.

Ejemplo:

```
sudo useradd -m -g users -s /bin/bash alumno
```

- **adduser**: Un script de alto nivel, más interactivo y amigable. En muchas distribuciones (especialmente Debian/Ubuntu), adduser es un script interactivo que llama a useradd con valores por defecto y te guía paso a paso. Es muy recomendable para principiantes por su facilidad de uso.

Ejemplo:

```
sudo adduser alumno
```

Después de crear el usuario con useradd o adduser, deberás asignarle una contraseña utilizando la orden passwd:

```
sudo passwd nombre_de_usuario
```

Si ejecutamos dichas órdenes sin argumentos, nos suelen mostrar la lista de opciones por stderr (salida de error estándar). Esta orden toma los valores por defecto que se le van a asignar al usuario (a su cuenta) a partir de la información especificada en los archivos /etc/default/useradd y /etc/login.defs.

### Actividad 1.2. Valores por omisión para nuevas cuentas

1. Visualiza el contenido de los archivos /etc/default/useradd y /etc/login.defs dentro de tu contenedor. Comprueba cuáles son las opciones por defecto que tendría un usuario que se creara en tu sistema.
2. A continuación, crea una cuenta de usuario y visualiza el contenido de los archivos /etc/passwd, /etc/group, y el directorio /home para comprobar que los nuevos datos se han rellenado conforme a la especificación tomada de /etc/default/useradd y /etc/login.defs.

Para modificar los valores asociados a una cuenta, disponemos de las siguientes órdenes:

**Tabla 2. Órdenes para gestión de cuentas de usuario.**

Orden	Descripción
usermod	modifica una cuenta de usuario ya existente
userdel	elimina una cuenta de usuario (por defecto no borra el directorio HOME)
newusers	crea cuentas de usuarios utilizando la información introducida en un archivo de texto, que ha de tener el formato del archivo /etc/passwd
system-config-users	herramienta en modo gráfico (no disponible en nuestro entorno de consola)

En el directorio /etc/skel se guardan unos archivos de configuración del shell, los cuales se copian al directorio HOME asignado cuando se crea una cuenta de usuario. Posteriormente, cada usuario podrá

personalizar su copia. Estos archivos son guiones shell que realizan determinadas tareas como inicializar variables, ejecutar funciones específicas, establecer los alias, etc. Estos archivos dependen del intérprete de órdenes seleccionado y en el caso del bash son:


**Tabla 3. Archivos de configuración para el shell Bash.**

Archivo	Descripción
<code>.bash_profile</code>	se ejecuta al <b>hacer el login</b> (conectarnos al sistema) y en él podremos indicar alias, variables, configuración del entorno, etc. que deseamos iniciar al principio de la sesión.
<code>.bashrc</code>	su contenido se ejecuta <b>cada vez que se ejecuta una shell</b> , tradicionalmente en este archivo se indican los programas o scripts a ejecutar.
<code>.bash_logout</code>	se ejecuta al <b>salir el usuario del sistema</b> y en él podremos indicar acciones, programas, scripts, etc., que deseamos ejecutar al salirnos de la sesión.

### Actividad 1.3. Creación de usuarios

1. Utiliza el manual en línea (`man useradd`) para leer la sintaxis completa de la utilidad para creación de cuentas y crea dos o tres usuarios en tu sistema cambiando alguno de los valores por defecto (por ejemplo, el directorio home con `-d`, o el shell con `-s`).
  - Para navegar por un manual en línea (`man`), una vez abierto, puedes usar las flechas del teclado para desplazarte línea a línea. Para avanzar una página completa, pulsa la tecla **Barra espaciadora** o **Page Down**. Para retroceder una página, usa **Page Up**. Si quieres buscar un término específico dentro del manual (por ejemplo, “home directory”), pulsa la tecla **/** (barra inclinada, la de los directorios), escribe el término y pulsa **Enter**; luego, para ir a la siguiente ocurrencia, pulsa **n**. Para ir al inicio del manual, pulsa **g**, y para ir al final, pulsa **G**. Finalmente, para salir del manual y volver al *prompt* de la shell, pulsa la tecla **q**.
2. Elimina alguno de ellos (`userdel <nombre_usuario>`) y comprueba qué “rastro” ha dejado la cuenta recién eliminada en el sistema (revisa `/etc/passwd`, `/etc/group`, y el directorio `/home`). ¿Cómo borrar el directorio home también?
3. Entra (orden `su`) en el sistema como uno de estos usuarios que has creado y mira qué archivos tiene en su directorio home.
4. Investiga el uso de la orden `sudo`. ¿Cómo permite `sudo` a un usuario normal ejecutar órdenes con privilegios de supervisor sin necesidad de cambiar a root con `su`? (Para que `sudo` funcione, el usuario debe estar configurado en el archivo `/etc/sudoers`).




 **Actividad 1.4. Archivo /etc/passwd** Visualiza el archivo /etc/passwd e indica cuál es el formato de cada línea de dicho archivo (es decir, qué significa cada campo, separados por :) Para ello también puedes consultar el manual en línea (`man 5 passwd` o `info passwd`). ¿Quién es el propietario de este archivo y cuáles son sus permisos?

**b) Cambio de contraseña (password)** Para asignar o cambiar una contraseña a un usuario se puede usar la orden `passwd`. Esta orden también se utiliza para cambiar la contraseña del usuario que la ejecuta si no se le indica ningún argumento. El usuario administrador (`root`) puede cambiar las contraseñas de todos los usuarios del sistema, incluyendo la suya propia. Sin embargo, un usuario normal solamente podrá cambiar la contraseña asociada a su propia cuenta.

A continuación se muestra la orden:

```
passwd <nombre_usuario>
```

Podemos asignar o cambiar el intérprete de órdenes por defecto que usará el usuario cuando se conecte al sistema. En el último campo del archivo /etc/passwd se establece para cada usuario el intérprete de órdenes que se ejecuta por defecto cada vez que entra al sistema. En el archivo /etc/shells se indican los shells permitidos, es decir, los que están instalados en el sistema. Con la orden `chsh` el usuario puede cambiar su shell (el nuevo debe estar entre los permitidos). Si un usuario no tiene asignado ningún intérprete de órdenes, se usará el shell por defecto, representado por el archivo /bin/sh. Si se desea que el usuario no pueda entrar al sistema, se le puede asignar al campo que indica el shell los nombres de archivo /bin/false o /sbin/nologin. También se puede establecer como shell un archivo ejecutable, de forma que cuando el usuario entre al sistema, se ejecutará dicho archivo y al finalizar su ejecución, se acabará la sesión de trabajo del usuario.

 **Actividad 1.5. Archivo /etc/shadow** Visualiza el archivo /etc/shadow desde un usuario distinto al `root` (por ejemplo, el que creaste en la Actividad 1.2). ¿Te da algún problema? ¿Sabes por qué? Intenta averiguarlo investigando los permisos del archivo.

**c) Parámetros de configuración de una cuenta.** Para las cuentas de los usuarios se pueden establecer restricciones de tiempo, también llamadas de envejecimiento (`aging`), respecto a la validez de la cuenta o de la contraseña. Los valores se guardan en el archivo /etc/shadow. La siguiente tabla muestra los valores posibles.

**Tabla 4. Valores para controlar el envejecimiento de contraseñas y cuentas.**

Campo	Descripción
<code>changed</code>	fecha del último cambio de contraseña (días desde 1970-01-01)
<code>minlife</code>	número de días que han de pasar para poder cambiar la contraseña
<code>maxlife</code>	número de días máximo que puede estar con la misma contraseña sin cambiarla



Campo	Descripción
warn	cuántos días antes de que la contraseña expire (maxlife) será informado sobre ello, indicándole que tiene que cambiarla
inactive	número de días después de que la contraseña expire que la cuenta se deshabilitará de forma automática si no ha sido cambiada
expired	fecha en la que la cuenta expira y se deshabilita de forma automática (días desde 1970-01-01)

Los valores los establece el administrador con las órdenes chage o con passwd. Recordemos que el archivo /etc/login.defs tiene los valores por defecto. La siguiente tabla muestra algunas opciones y argumentos útiles para la orden chage.

**Tabla 5. Posibles escenarios de uso de la orden chage.**

Orden	Descripción
-d ult_día	Establece la fecha del último cambio de password (días desde 1970-01-01).
-m min_días	Establece el número de días que han de pasar para poder cambiar la contraseña.
-M max_días	Establece el número de días máximo que puede estar con la misma contraseña sin cambiarla.
-W warn_días	Establece cuántos días antes de que la contraseña expire (maxlife) será avisado de ello, indicándole que tiene que cambiarla.
-I inac_días	Establece el número de días después de que la contraseña expire que la cuenta se deshabilitará de forma automática si la contraseña no ha sido cambiada.
-E exp_días	Establece la fecha en la que la cuenta expira y se deshabilita de forma automática (en formato YYYY-MM-DD o días desde 1970-01-01).

Por ejemplo:

```
chage -m 30 nombre_usuario
```

Establecería un mes antes de que el usuario nombre\_usuario pudiera cambiar su contraseña.

### 8.3. Gestión de Grupos

Un **grupo** es un conjunto de usuarios que comparten recursos o archivos del sistema. Con los grupos se pueden garantizar permisos concretos para un conjunto de usuarios, sin tener que repetirlos cada vez que se desee aplicarlos individualmente.

Un grupo se caracteriza por:

- Nombre del grupo, o groupname.

- Identificador del grupo (**GID**, del inglés Group Identifier), que es un número que permite al sistema identificar al grupo (ver sección 8.2).
- Archivo de configuración `/etc/group`. Cada línea de este archivo presenta el siguiente formato: `nombre:x:gid:lista_de_usuarios` (donde `x` indica que la contraseña del grupo se almacena en `/etc/gshadow` si existe).

**Tabla 6. Órdenes relacionadas con la gestión de grupos.**

Orden	Descripción
<code>groupadd grupo</code>	crea un nuevo grupo
<code>groupmod grupo</code>	modifica un grupo existente
<code>groupdel grupo</code>	elimina un grupo
<code>newgrp grupo</code>	<b>cambia de grupo activo</b> (lanza un shell con ese grupo como primario)
<code>gpasswd grupo</code>	asigna una contraseña a un grupo (raramente usado)
<code>gpasswd -a user grupo</code>	<b>añade un usuario a un grupo</b>
<code>groups [usuario]</code>	<b>informa de los grupos a los que pertenece un usuario</b> (o el actual si no se especifica)
<code>id [usuario]</code>	lista el identificador del usuario y los grupos a los que pertenece
<code>grpck</code>	comprueba la consistencia del archivo de grupos

### **Actividad 1.6. Creación de grupos**

1. Crea un par de grupos (`groupadd <nombre_grupo>`) y asignáelos a algunos de los usuarios que has creado en tu sistema (`usermod -aG <grupo> <usuario>`).
2. Utiliza la orden `groups` y `id` para verificar los cambios.
3. ¿Qué información devuelve la orden `id` si estás conectado como `root`? ¿Es diferente si la ejecutas como un usuario normal?

## **8.4. Usuarios y Grupos Especiales**

Los usuarios especiales son aquellos que no están asociados a personas físicas, sino que sirven para la ejecución de servicios del sistema o como propietarios de archivos. A continuación se muestran dos tablas, la primera incluye los nombres de usuario de algunos usuarios especiales comunes en los sistemas UNIX y, la segunda, incluye grupos estándar que se encuentran en la mayoría de los sistemas UNIX.

**Tabla 7. Usuarios especiales del sistema.**

Usuario	Descripción
<code>root</code>	Usuario administrador del sistema

Usuario	Descripción
bin, daemon, lp, sync, shutdown,...	Tradicionalmente usados para poseer archivos o ejecutar servicios
mail, news, ftp,...	Asociados con herramientas o utilidades específicas
postgres, mysql, xfs,...	Creados por herramientas instaladas en el sistema para administrar y ejecutar sus servicios
nobody ó nfsnobody	Usada por NFS y otras utilidades para permisos de acceso anónimo o bajo privilegio

**Tabla 8. Grupos estándar del sistema.**

Grupo	Descripción
root, sys, bin, daemon, adm, lp, disk, mail, ftp, nobody	Algunos de los nombres de grupo preconfigurados por los sistemas UNIX. Los GIDs inferiores a 500 (o 1000 en algunas distribuciones como Ubuntu) están reservados para estos grupos.
tty, dialout, disk, cdrom, audio, video	Nombres de grupo específicos para dispositivos (control de acceso a hardware).
kernel	Grupo propietario de los programas para leer la memoria del kernel (no es un grupo común por defecto).
users	Puede usarse como grupo por defecto para todos los usuarios normales del sistema (común en algunas distribuciones, pero en Ubuntu el grupo principal de un nuevo usuario es un grupo con su mismo nombre).

## 9. Organización del Sistema de Archivos y Gestión Básica de Archivos

### La Organización del Sistema de Archivos en UNIX/Linux

La organización de los archivos en un sistema de archivos de tipo UNIX/Linux se basa en una **estructura jerárquica en forma de árbol**. Un principio fundamental en estos sistemas es la abstracción de “**todo es un archivo**” (everything is a file), lo que unifica el acceso a diferentes recursos del sistema (datos, dispositivos de hardware, mecanismos de comunicación interproceso) a través de la interfaz del sistema de archivos.

Dentro de esta jerarquía, que se inicia en el directorio raíz, podemos encontrar distintos tipos de archivos, cada uno con una función específica:

- **El Directorio Raíz (/):** Es el punto de partida de toda la jerarquía del sistema de archivos. Todos los demás directorios y archivos se encuentran bajo él, y es el punto donde se montan otros sistemas de archivos (particiones, dispositivos de almacenamiento externos, etc.), formando una estructura unificada.

- **Archivos Regulares (o Normales):** Son los tipos de archivos más comunes y se utilizan para almacenar datos de usuario, texto, programas ejecutables, imágenes, documentos, etc. Constituyen la mayor parte de los ficheros en cualquier sistema.
- **Archivos de Directorio:** Son archivos especiales cuya función principal es organizar y soportar la estructura jerárquica del sistema de archivos. Su contenido no son datos directamente legibles, sino una lista de entradas. Cada entrada en un directorio contiene el nombre de un archivo (que puede ser regular, otro directorio, un enlace, etc.) y una referencia a sus **metadatos** (información sobre el archivo, como su tipo, permisos, tamaño, propietario, fechas de modificación, etc.), usualmente a través de un `inode` en sistemas UNIX/Linux.
- **Archivos Especiales de Dispositivo:** Estos archivos permiten que el sistema de archivos actúe como una interfaz uniforme para interactuar con los dispositivos de hardware de entrada/salida (E/S). Esto significa que se pueden utilizar órdenes de archivo estándar (como `cat`, `cp`, `dd`) para leer o escribir directamente en dispositivos de hardware, a través de la abstracción que estos archivos proporcionan. Se ubican típicamente en el directorio `/dev`, siguiendo el estándar FHS (Filesystem Hierarchy Standard). Existen dos categorías principales:
  - **Dispositivos de Bloque:** Representan dispositivos que transfieren datos en bloques de tamaño fijo (ej. discos duros, unidades USB). Ejemplos: `/dev/sda`, `/dev/nvme0`.
  - **Dispositivos de Carácter:** Representan dispositivos que transfieren datos byte a byte (ej. terminales, puertos serie, impresoras, tarjetas de sonido). Ejemplos: `/dev/ttyS0` (un puerto serie), `/dev/lp0` (una impresora), `/dev/null` (un dispositivo especial que descarta toda la información que se le envía y devuelve un EOF inmediatamente si se intenta leer de él), `/dev/random` (fuente de datos aleatorios).
- **Archivos Especiales para Comunicación entre Procesos:** Facilitan la transferencia de información entre procesos que se ejecutan en el mismo sistema. Se representan en el sistema de archivos, permitiendo que los procesos se comuniquen a través de ellos. Los dos tipos más comunes son:
  - **Archivos de tipo FIFO (Named Pipes):** También conocidos como *tuberías con nombre* o *cauces*, son archivos especiales que **permiten la comunicación unidireccional entre procesos**. Los datos se escriben por un extremo y se leen por el otro, funcionando como una cola de datos en memoria, pero con una entrada persistente en el sistema de archivos.
  - **Archivos de tipo Socket (Unix Domain Sockets):** Son archivos especiales que permiten la comunicación bidireccional entre procesos que se ejecutan en la misma máquina. A diferencia de los sockets de red, los sockets de dominio Unix utilizan el sistema de archivos como su punto de dirección, lo que los hace muy eficientes para la comunicación local y les permite ser gestionados con permisos de archivo.
- **Archivos de Enlace (Links):** Permiten que un mismo archivo tenga múltiples nombres y/o ubicaciones dentro de la estructura de directorios, facilitando su referencia desde diferentes puntos. Son muy útiles para compartir archivos, gestionar versiones de software (ej. `aplicacion_v2.0` y

un enlace simbólico `aplicacion_actual` que apunta a ella) o simplificar rutas complejas. Los sistemas UNIX/Linux distinguen dos tipos principales:

- **Enlace Duro (`hard link`):** Crea una nueva entrada de directorio que apunta al mismo `inode` (y, por lo tanto, a los mismos datos en disco) que el archivo original. Esto significa que ambos nombres son igualmente válidos para el mismo conjunto de datos. Solo pueden crearse para archivos regulares y dentro del mismo sistema de archivos. No pueden apuntar a directorios. El archivo original no se elimina del disco hasta que todos sus enlaces duros (incluido el nombre original) hayan sido borrados.
- **Enlace Simbólico (`soft link` o `symlink`):** También conocido como atajo o acceso directo, es un archivo especial que contiene la ruta de acceso (relativa o absoluta) a otro archivo o directorio. A diferencia de los enlaces duros, puede apuntar a archivos o directorios ubicados en sistemas de archivos diferentes e incluso a rutas que no existen en el momento de su creación (lo que resulta en un “enlace roto” si el destino no es válido). Si el archivo o directorio original es eliminado o movido, el enlace simbólico se “rompe” y deja de funcionar.

### 9.1. Organización común en sistemas de archivos tipo Linux. Filesystem Hierarchy Standard (FHS)


El FHS es un estándar que propone una forma sistemática de organizar toda la información que almacena un sistema operativo tipo Linux. La Fundación Linux (Linux Foundation) es la encargada de mantener este estándar. Esta organización integra entre sus miembros a muchas de las empresas líderes en el sector de la informática, tanto hardware como software. Para más información puedes visitar la página: <http://www.linuxfoundation.org/>

En el primer apartado de prácticas ya has podido conocer uno de los directorios importantes recogidos en el estándar FHS, el directorio `/etc`. Básicamente, en este directorio podemos encontrar todos los archivos de configuración del sistema. Obviamente, para un administrador de sistema, conocer y comprender el contenido de este directorio es fundamental para el correcto desempeño de su labor. A continuación se muestra una tabla con algunos de los directorios que es interesante conocer y que recoge este estándar. Para más información puedes visitar la página: [http://en.wikipedia.org/wiki/Filesystem\\_Hierarchy\\_Standard](http://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard)

**Tabla 9. Nombres de directorio y tipo de información que almacenan en el estándar FHS.**

Directorio	Descripción
<code>/bin</code>	Programas de utilidad fundamentales para ser utilizados por cualquier usuario del sistema.
<code>/sbin</code>	Programas de utilidad fundamentales para ser utilizados por el usuario <code>root</code> .
<code>/boot</code>	Archivos fundamentales para el programa Boot Loader (gestión del arranque).
<code>/dev</code>	Todos los archivos especiales de dispositivo (interfaz con el hardware).
<code>/etc</code>	Archivos de configuración del sistema.

Directorio	Descripción
/home	Los directorios de inicio de todos los usuarios que disfrutan de una cuenta en el sistema, excepto el directorio de inicio del root: /root.
/lib	Bibliotecas sin las que no pueden funcionar los programas ubicados en /bin y /sbin.
/media	Este directorio actúa como punto de montaje para dispositivos extraíbles: DVD-ROM, dispositivos USB, etc.
/mnt	Este directorio actúa como punto de montaje para sistemas de archivos montados temporalmente.
/opt	Normalmente aquí se ubican los programas que no forman parte de la distribución instalada en el sistema (software de terceros).
/proc	Sistema de archivos virtual que hace de interfaz con el núcleo y los procesos (contiene información del estado del sistema en tiempo real).
/run	Datos usados en tiempo de ejecución. Suele almacenarse en RAM, y no en el disco real.
/tmp	Archivos temporales que normalmente no se mantienen una vez se apaga el sistema.
/usr	Archivos ejecutables, archivos de código fuente, bibliotecas, documentación y, en general, todos los programas y utilidades de usuario.
/var	Los archivos cuyo contenido se espera que cambie durante el funcionamiento normal del sistema (logs, colas de impresión, caches, etc.).

 **Actividad 1.8. Organización del SA** Un programa que se ejecuta en modo root dentro de tu contenedor, ¿dónde podría guardar la información temporal de forma que ésta se mantuviese entre arranques del contenedor? Ten en cuenta el concepto de volumen montado que hemos visto con `podman run -v`.

## 9.2. Órdenes Básicas para Gestión del Sistema de Archivos.

En la asignatura Fundamentos del Software se utilizaron diversas órdenes para explorar la estructura jerárquica de directorios/archivos: `pwd`, `ls` y `cd`; para crear y borrar directorios y archivos: `mkdir`, `rmdir`, `cat` y `rm`; y para copiar y mover archivos y directorios a distintas ubicaciones dentro de la estructura jerárquica: `cp` y `mv`. También se vieron algunas órdenes para modificar los atributos de los archivos: `chmod` y `touch`, así como para consultar algunos de sus atributos: `file` y `ls`.

En nuestras prácticas, el objetivo principal es el trabajo con el sistema de archivos, es decir, no nos centraremos en órdenes que realizan su funcionalidad sobre archivos individuales, sino sobre todo el sistema de archivos en su conjunto. La funcionalidad que requiere un administrador del sistema en este ámbito incluye, pero no está reducida a:

- Acceso a información del SO relativa a sistemas de archivos.
- Ampliación de la estructura jerárquica de directorios mediante la orden `mount`.

- Instalación y configuración de nuevos dispositivos de almacenamiento y creación de sistemas de archivos sobre estos (tareas que, en el contexto de los contenedores, suelen realizarse al construir la imagen o con volúmenes).
- Comprobación del estado del sistema de archivos.
- Cuotas de disco.


En esta sesión abordaremos exclusivamente algunas órdenes que nos permitirán acceder a la información relativa a qué sistemas de archivos tenemos disponibles actualmente en nuestra estructura de directorios y los atributos con los que fueron montados estos sistemas de archivos.

### 9.2.1. Acceso a Información del Sistema de Archivos y Puntos de Montaje

Para obtener información sobre la configuración y el estado de los sistemas de archivos en un sistema Linux, tradicionalmente se han utilizado archivos de configuración y archivos virtuales ubicados en directorios clave, siguiendo el estándar FHS (Filesystem Hierarchy Standard). Los dos archivos más relevantes en este contexto son `/etc/fstab` y la información accesible a través de `/proc/mounts`, o a veces `/proc/self/mounts`. Aunque es habitual referirse a este fichero por su enlace simbólico: `/etc/mtab`.

Antes de continuar, en este punto es importante entender qué es un disco y qué una partición. Verás que a veces hablamos de un dispositivo `/dev/sda`, pero luego se monta otro archivo distinto `/dev/sda1` en el sistema de archivos. Esto se debe a que:

- **`/dev/sda` (El Disco Físico/Virtual):** Este nombre, como `/dev/sda` (o `/dev/hdb`, `/dev/nvme0n1`, etc., dependiendo del tipo de interfaz y la enumeración del sistema), representa el **dispositivo de almacenamiento físico o virtual completo**. Es el disco duro en sí mismo, una unidad de estado sólido (SSD), o incluso una unidad flash USB. Es la entidad *en bruto* que contiene todo el espacio de almacenamiento disponible, antes de que se le dé una estructura lógica. En sistemas Linux, los dispositivos SCSI, SATA y USB suelen ser nombrados como `sdX` (donde X es una letra que indica el dispositivo: a para el primero, b para el segundo, etc.).
- **`/dev/sda1` (Una Partición del Disco):** Una **partición** es una **división lógica del espacio de almacenamiento de un disco físico**. Es como tomar un gran terreno (el disco) y dividirlo en varias parcelas más pequeñas (las particiones). Cada una de estas particiones (`/dev/sda1`, `/dev/sda2`, `/dev/sda3`, etc.) es un segmento independiente del disco. El número que acompaña al nombre del disco (ej. 1 en `sda1`, 2 en `sda2`) indica la partición específica dentro de ese dispositivo. Normalmente, lo que buscamos es montar una partición que a su vez contiene ficheros y directorios.

 **Actividad 1.9. Información de los Sistemas de Archivos** Los archivos `/etc/fstab` y `/proc/mounts` muestran información sobre los sistemas de archivos que se encuentran montados en el sistema (o que se montarán). ¿Cuál es la diferencia fundamental entre la información que muestra cada uno de ellos? (Pista: uno es una **configuración estática** y el otro un **estado dinámico**).

**El archivo `/etc/fstab`** El archivo `/etc/fstab` (del inglés *file system table*) es un archivo de configuración **estático** que contiene una lista de todos los sistemas de archivos disponibles en el sistema y cómo deben ser montados. El sistema operativo lo utiliza, por ejemplo, durante el proceso de arranque para montar automáticamente las particiones y dispositivos necesarios.

Cada línea en `/etc/fstab` define un sistema de archivos y consta de seis campos, separados por espacios o tabuladores:

1. **`fs_spec` (Dispositivo/UUID/LABEL):** Especifica el sistema de archivos a montar. **En sistemas Linux modernos, la práctica recomendada es utilizar el UUID (Universally Unique Identifier) o el LABEL (etiqueta de volumen) en lugar de los nombres de dispositivo tradicionales (como `/dev/sda1`).**
  - **UUIDs:** Ofrecen una mayor robustez y persistencia. Los nombres de los dispositivos pueden cambiar (ej. `/dev/sda1` podría convertirse en `/dev/sdb1` si se añade otro disco), pero el UUID de una partición es único y constante. Puedes obtener los UUIDs de tus particiones usando órdenes como `blkid` o `lsblk -f`.
    - Ejemplo: `UUID=a1b2c3d4-e5f6-7890-1234-567890abcdef`
  - **LABELs:** Son etiquetas legibles por humanos que se asignan a los sistemas de archivos. Son útiles para una mejor identificación y también ofrecen persistencia frente a cambios en los nombres de los dispositivos, aunque no garantizan la unicidad global como los UUIDs.
    - Ejemplo: `LABEL=Datos_Linux`
  - Si se sigue usando un nombre de dispositivo (ej. `/dev/sda1`), el sistema es más vulnerable a errores de montaje si el orden de los dispositivos cambia.
2. **`fs_file` (Punto de Montaje):** El directorio donde se montará el sistema de archivos (ej. `/`, `/home`, `/var`, `/mnt/datos`).
3. **`fs_vfstype` (Tipo de Sistema de Archivos):** El tipo de sistema de archivos (ej. `ext4`, `xfs`, `ntfs`, `vfat`, `swap`).
4. **`fs_mntops` (Opciones de Montaje):** Una lista de opciones separadas por comas que controlan cómo se monta el sistema de archivos. Algunas de las opciones más comunes son:
  - **`defaults`:** Combina un conjunto de opciones predeterminadas (comúnmente `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, `async`).
  - **`rw` / `ro`:** Monta el sistema de archivos en modo lectura/escritura (`rw`) o solo lectura (`ro`).
  - **`auto` / `noauto`:** Indica si el sistema de archivos debe montarse automáticamente al arrancar el sistema (`auto`) o si debe ser montado manualmente (`noauto`). Si es `noauto`, no se montará ni siquiera con la orden `mount -a`; solo se montará de forma explícita con la orden `mount [punto de montaje]`.
  - **`exec` / `noexec`:** Permite (`exec`) o deniega (`noexec`) la ejecución de binarios en ese sistema de archivos. Útil por seguridad en particiones que solo contienen datos.



- **suid / nosuid**: Controla si los bits SUID/SGID se respetan (suid) o se ignoran (nosuid). Por seguridad, a menudo se usa nosuid en particiones montadas de dispositivos externos.
  - **dev / nodev**: Permite (dev) o deniega (nodev) la interpretación de archivos especiales de dispositivo (/dev) en ese sistema de archivos. Similar a nosuid, útil por seguridad.
  - **user / users / owner**: Permite a un usuario normal (user), a cualquier usuario (users), o solo al propietario del dispositivo (owner) montar y desmontar el sistema de archivos.
  - **nofail**: Impide que el sistema de arranque se detenga si este sistema de archivos no se puede montar (por ejemplo, si un disco externo no está conectado).
  - **noatime / relatime**: Opciones de rendimiento. noatime evita actualizar la hora de último acceso a un archivo cada vez que se lee (lo que reduce escrituras y mejora el rendimiento, especialmente en SSDs). relatime es un compromiso más moderno que actualiza el atime solo si el mtime (modificación) o ctime (cambio de metadatos) también ha cambiado, o si el atime es anterior al mtime en más de 24 horas.
  - **usrquota / grpquota**: Indica que se habilitan las cuotas para usuarios o grupos en ese sistema de archivos.
  - **uid=<num> / gid=<num> / umask=<mask>**: Opciones para sistemas de archivos que no soportan permisos de Unix de forma nativa (ej. vfat, ntfs). Permiten especificar el propietario de los archivos (uid), el grupo propietario (gid) y la máscara de permisos por defecto (umask) para los archivos recién creados.
5. **fs\_freq (Opción dump)**: Un número (0 o 1) que indica si el sistema de archivos debe ser volcado por la utilidad dump. 0 significa no volcar.
  6. **fs\_passno (Orden de verificación fsck)**: Un número que determina el orden en que fsck (el programa de verificación de la consistencia del sistema de archivos) comprueba los sistemas de archivos durante el arranque. 0 significa no verificar, 1 se usa para el sistema de archivos raíz, y 2 para los demás sistemas de archivos.

**El archivo virtual /proc/mounts (y /etc/mtab)** Mientras que /etc/fstab describe los sistemas de archivos *deseados* para el montaje, el archivo virtual /proc/mounts proporciona una vista **dinámica y en tiempo real** de todos los sistemas de archivos que están **actualmente montados** en el sistema. Es un archivo *pseudo-filesystem* o *sistema de archivos virtual*, lo que significa que no se almacena en el disco, sino que el kernel lo genera “al vuelo” cada vez que se accede a él, reflejando el estado actual de los montajes.


Históricamente, /etc/mtab era un archivo real que se mantenía sincronizado con los sistemas de archivos montados. Sin embargo, en distribuciones Linux modernas (especialmente aquellas que utilizan systemd), /etc/mtab suele ser un **enlace simbólico a /proc/mounts**. Esto asegura que cualquier programa que lea /etc/mtab obtenga siempre la información más actualizada directamente del kernel, eliminando la necesidad de mantener un archivo separado sincronizado.

**Diferencias clave entre /etc/fstab y /proc/mounts (o /etc/mtab symlink):**

- /etc/fstab: **Configuración estática**. Contiene lo que *debería* estar montado o cómo se *debe*

montar.


- `/proc/mounts` (y `/etc/mtab`): **Estado dinámico**. Muestra lo que *está* realmente montado en este preciso momento. Puede incluir montajes temporales (como dispositivos USB recién conectados) o montajes específicos del sistema que no están en `/etc/fstab` (ej. `/sys`, `/run`, montajes de `snap` o `flatpak`).

 **Actividad 1.10. Información de los SAs** Edita el archivo `/etc/fstab` del sistema de archivos que estás utilizando en modo `root` (dentro del contenedor) y anota y describe la información que tiene registrada. Si no conoces alguna opción, puedes consultar el manual en línea: `man fstab`. Ten en cuenta que, al estar en un contenedor, `fstab` puede no reflejar el sistema de archivos del *host* real, sino el del entorno del contenedor.

Otro directorio del FHS que nos va a ser muy útil para obtener información es el `/proc`, el cual soporta el sistema de archivos virtual `procfs`. Este directorio contiene archivos de texto que permiten acceder a información de estado del sistema. En este apartado solamente utilizaremos los archivos que tienen información relativa a los sistemas de archivos, pero es conveniente familiarizarse con él porque se utilizará en varios apartados del módulo de administración del SO Linux.

**Tabla 10. Archivos con información de sistema de archivos que proporciona `/proc`.**

Archivo	Descripción
<code>/proc/filesystems</code>	Enumera, uno por línea, todos los tipos de sistemas de archivos disponibles en el kernel.
<code>/proc/mounts</code>	Sistemas de archivos montados actualmente, incluyendo los que se hayan montado manual o automáticamente tras el arranque del sistema (similar a <code>mount</code> sin argumentos).

 **Actividad 1.11. Archivos de información para los SAs** Compara la información que contienen los cuatro archivos de texto que se han presentado en este apartado (`/etc/fstab`, `/etc/mtab`, `/proc/filesystems` y `/proc/mounts`). Describe en un párrafo para qué te sirve la información que registra cada archivo y cuáles son las diferencias clave entre ellos en el contexto de tu contenedor.