Crear y administrar usuarios en Linux

1) Conceptos clave

- **Usuarios** → cada persona/proceso en el sistema necesita una cuenta.
- **Grupos** → conjunto de usuarios, útil para permisos compartidos.
- **Permisos** → controlan lectura/escritura/ejecución de archivos.
- **Propietario y grupo** → cada archivo pertenece a un usuario y a un grupo.
- **sudo** → permite a un usuario ejecutar comandos como root (administrador).

El comando principal es **useradd**, pero se recomienda usar **adduser** en sistemas basados en Debian/Ubuntu, ya que es una *envoltura* que realiza varias tareas automáticamente.

sudo adduser juan

su - juan

2) Crear usuarios y grupos

Crear usuario básico

crear usuario 'ana'

sudo useradd -m ana

-m crea automáticamente su directorio /home/ana

cat /etc/passwd

Asignar contraseña

sudo passwd ana

Crear usuario con comentarios, shell y grupo inicial

sudo useradd -m -c "Ana López" -s /bin/bash -g usuarios ana

La diferencia entre las dos formas de crear el usuario está en **qué valores por defecto usa useradd** cuando no le indicas opciones adicionales.

Comportamiento de useradd en Ubuntu

- Cuando ejecutas:

sudo useradd -m ana

lo único que le estás diciendo es:

- **-m** → crea el directorio /home/ana.
- El resto (shell, grupo, comentario, etc.) se toman de la configuración por defecto en /etc/default/useradd y /etc/login.defs.

En Ubuntu, el **shell por defecto** de useradd suele ser /bin/sh (que en realidad apunta a dash), **no /bin/bash**.

Por eso, si luego haces:

getent passwd ana

verás algo como:

ana:x:1001:1001::/home/ana:/bin/sh

su - ana

Cómo darle bash desde el principio

Si quieres que el usuario tenga **bash** como shell de login, debes especificarlo con -s:

sudo useradd -m -s /bin/bash ana

Si ya lo tenemos creado

Creaste el usuario **ana** correctamente con useradd -m ana.

Para asignarle **Bash** como su *shell* predeterminado, tienes dos métodos principales, siendo el primero el más directo y común:

1. Usar el comando usermod

El comando usermod te permite modificar los atributos de un usuario existente. Usarás la opción -s (shell) para cambiar el intérprete de comandos.

Asegúrate de ejecutar este comando con permisos de superusuario (sudo):

sudo usermod -s /bin/bash ana

Explicación:

- sudo: Ejecuta el comando con permisos de superusuario.
- usermod: El comando para modificar un usuario.
- -s /bin/bash: Establece el *shell* de inicio de sesión del usuario **ana** a /bin/bash.
- ana: El nombre del usuario a modificar.

2. Editar el archivo /etc/passwd directamente (Avanzado)

Puedes editar manualmente la línea de usuario en el archivo /etc/passwd, donde se almacena la información de todos los usuarios.

¡Advertencia! Cualquier error de sintaxis en este archivo puede impedir que los usuarios inicien sesión. Si optas por esta vía, usa siempre vipw para editarlo de forma segura.

1. Ejecuta vipw (que bloquea el archivo para evitar que otros programas escriban en él):

sudo vipw

2. Busca la línea del usuario **ana**. Debería verse similar a esto (la última parte es probablemente /bin/sh o /bin/false):

ana:x:1001:1001:,,,/home/ana:/bin/sh

3. Cambia la última parte para que diga /bin/bash:

ana:x:1001:1001:,,,/home/ana:/bin/bash

4. Guarda y sal del editor.

El segundo comando

sudo useradd -m -c "Ana López" -s /bin/bash -g usuarios ana

- -c "Ana López" → añade un comentario (aparece en /etc/passwd, normalmente usado para el nombre completo).
- -s /bin/bash → fuerza que el shell de login sea bash.
- -g usuarios → pone como grupo primario el grupo usuarios (debe existir previamente).

Cuando creas un usuario con:

sudo useradd -m -s /bin/bash ana

En este paso **no se le asigna ninguna contraseña automáticamente**.

El usuario ana queda creado, con su home /home/ana y con bash como shell, pero **no puede iniciar sesión todavía** porque no tiene contraseña definida.

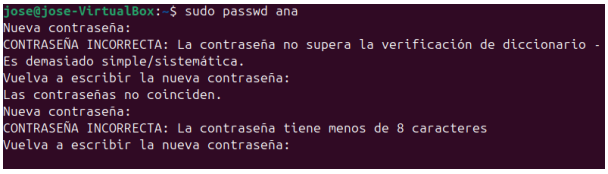
Qué hacer después

Para darle una contraseña debes ejecutar:

sudo passwd ana

Ahí te pedirá que escribas y confirmes la nueva contraseña.

A partir de ese momento, ana ya podrá iniciar sesión en la máquina.



PassAnita091

Resumen

- useradd crea la cuenta, pero no asigna contraseña.
- Hasta que no uses passwd, el usuario no puede loguearse.

Si quieres crear el usuario y asignarle contraseña en un solo paso, puedes hacerlo así:

sudo useradd -m -s /bin/bash ana

echo "ana:contraseña123" | sudo chpasswd

Crear grupo

sudo groupadd marketing

cat /etc/group

Añadir usuario a un grupo

sudo usermod -aG marketing ana

La opción -aG es importante → "append to group". Si solo usas -G, quitas al usuario de otros grupos.

Ejercicio 1

1. Crea un usuario prueba1.
2. Crea un grupo curso.
3. Añade prueba1 al grupo curso.
4. Comprueba con:

id prueba1

groups prueba1

Vamos a **crear un archivo en /home/ana pero que el propio usuario ana no pueda leerlo**. Eso se hace jugando con los **permisos y propietario** del archivo.

1. Estás en /root (lo confirmas con pwd):

pwd

→ /root

sudo -i

2. Creas el archivo como root (con sudo):

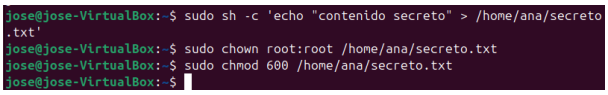
sudo sh -c 'echo "contenido secreto" > /home/ana/secreto.txt'

3. Cambias el propietario para que **no sea de ana**, por ejemplo root:

sudo chown root:root /home/ana/secreto.txt

4. Ajustas permisos para que **solo root pueda leerlo**:

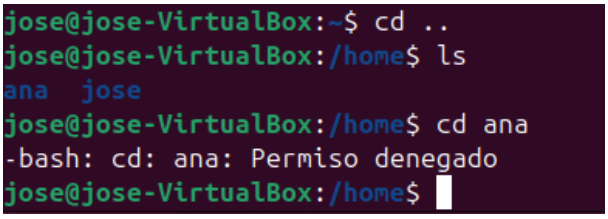
sudo chmod 600 /home/ana/secreto.txt



su - jose

Si intentamos acceder al directorio de ana nos dará un error:

-bash: cd: ana: Permiso denegado



significa que el usuario jose **no tiene permisos para entrar en el directorio /home/ana**.

¿Qué pasa?

- Cuando creamos a ana con useradd -m, su directorio /home/ana se generó con permisos por defecto:

drwx------ 2 ana ana 4096 sep 30 11:00 /home/ana

Eso quiere decir:

- drwx------ → solo el propietario (ana) puede leer, escribir y entrar.
- Nadie más (ni siquiera otros usuarios normales) puede acceder.
- Solo root puede entrar porque tiene privilegios totales.

Entrar como root

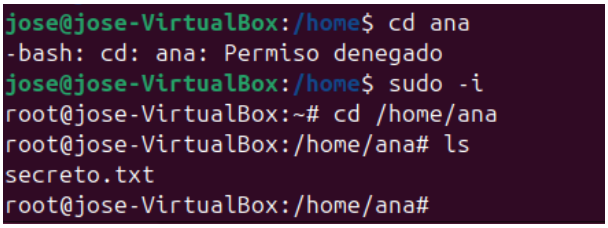
Cambiamos a un usuario con privilegios

su - jose

sudo -i

cd /home/ana

(funciona porque root ignora los permisos).



Dar acceso a jose

Si quieres que jose pueda entrar en /home/ana, cambia los permisos:

sudo chmod 750 /home/ana

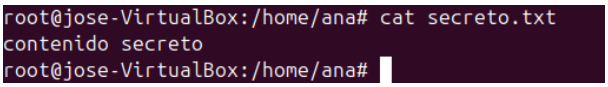
→ ahora el propietario (ana) y los miembros de su grupo podrán entrar.

Si además quieres que jose tenga acceso, añádelo al grupo de ana:

sudo usermod -aG ana jose

Resultado

- El archivo existe en /home/ana/secreto.txt.
- Propietario: root.
- Permisos: rw------- (solo root puede leer y escribir).



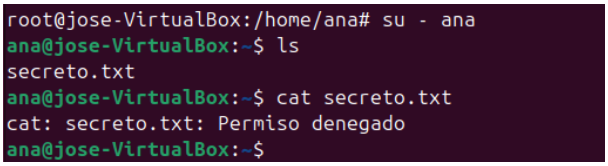
Si ana intenta leerlo:

su - ana

cat /home/ana/secreto.txt

obtendrá:

cat: /home/ana/secreto.txt: Permiso denegado



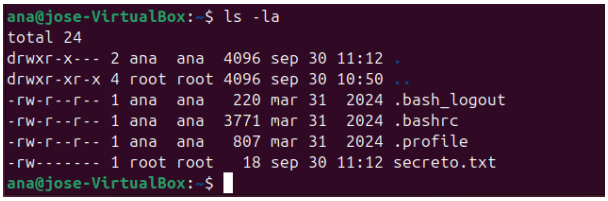
3) Permisos de archivos (chmod)

Los permisos se muestran con ls -l:

-rw-r--r-- 1 ana curso 1234 sep 28 09:00 archivo.txt

Significado:

- r → lectura
- w → escritura
- x → ejecución
- primer bloque (rw-) propietario → usuario
- segundo bloque (r--) grupo
- tercer bloque (r--) otros



Modificar permisos

chmod u+x script.sh # dar permiso de ejecución al propietario

chmod g+w archivo.txt # dar escritura al grupo

chmod o-r archivo.txt # quitar lectura a otros

Usando notación numérica

chmod 755 script.sh

7 = rwx (propietario)

5 = r-x (grupo)

5 = r-x (otros)

Ejercicio 2

1. Crea el archivo ejemplo.sh con nano y pon echo "Hola mundo".
2. Comprueba permisos con ls -l ejemplo.sh.
3. Asigna permisos de ejecución al propietario

chmod u+x ejemplo.sh

4. Ejecútalo con

sudo ./ejemplo.sh

4) Cambiar propietario y grupo (chown, chgrp)

Cambiar propietario

Cambiamos a un usuario con privilegios:

ana@VM:~$ sudo -i

[sudo] password for ana:

***ana is not in the sudoers file*.**

su – jose

Entrar como root

sudo -i

cd /home/ana

sudo chown ana secreto.txt

```
root@jose-VirtualBox:/home/ana# ls
secreto.txt
root@jose-VirtualBox:/home/ana# sudo chown ana secreto.txt
root@jose-VirtualBox:/home/ana#
```

Qué hace

- **chown** = *change owner* → cambia el propietario de un archivo o directorio.
- **ana** = el nuevo propietario (usuario).
- **secreto.txt** = el archivo al que se le cambia el dueño.
- **sudo** = lo ejecutas como administrador, porque normalmente solo root puede cambiar propietarios.

Resultado

- Antes: secreto.txt podía ser propiedad de root o de otro usuario.
- Después: el propietario pasa a ser ana.
- El grupo **no cambia** (a menos que lo indiques.

Ejemplo:

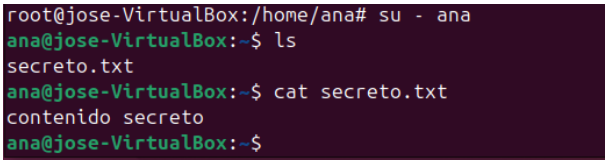
ls -l archivo.txt

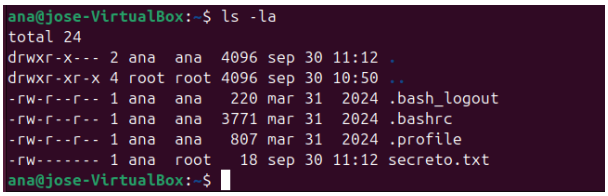
Antes:

-rw-r--r-- 1 root root 42 sep 30 11:20 archivo.txt

Después de sudo chown ana archivo.txt:

-rw-r--r-- 1 ana root 42 sep 30 11:20 archivo.txt





-rw------- 1 ana root 18 sep 30 11:12 secreto.txt

Desglose

- -rw------- → permisos:
  - rw- para el **propietario** (usuario ana).
  - --- para el **grupo** (root).
  - --- para **otros**.
- Propietario: ana.
- Grupo: root.

¿Qué significa en la práctica?

- **ana (propietario)** → puede leer y escribir el archivo.
- **Usuarios del grupo root** → no tienen permisos (aunque root como superusuario puede ignorar esto).
- **Otros usuarios** → tampoco tienen permisos.

Por eso, ahora mismo **ana sí puede leer el archivo**, porque es la propietaria.

El hecho de que el grupo sea root no cambia nada: los permisos efectivos se miran primero en el propietario, y ana tiene rw.

Variantes útiles

- Cambiar **usuario y grupo** a la vez:

sudo chown ana:usuarios archivo.txt

- Cambiar de forma recursiva (para carpetas enteras):

sudo chown -R ana:usuarios /home/ana

Cambiar grupo

sudo chgrp marketing secreto.txt

Cambiar ambos

sudo chown ana:marketing secreto.txt

Ejercicio 3

1. Crea el archivo notas.txt como root (sudo touch notas.txt).
2. Asigna como propietario a ana y grupo curso.
3. Verifica con ls -l notas.txt.

5) Configurar sudoers

El archivo de configuración es /etc/sudoers. **Nunca lo edites directo**, usa:

sudo visudo

Esto evita errores de sintaxis.

Dar acceso sudo a un usuario

En visudo agrega:

User privilege specification

root ALL=(ALL:ALL) ALL

ana ALL=(ALL:ALL) ALL

Dar acceso a un grupo

%curso ALL=(ALL:ALL) ALL

Así todos los usuarios del grupo curso pueden usar sudo.

Ejemplo de restricciones

- Solo permitir reinicio:

ana ALL=(ALL) /sbin/reboot

- Permitir sin pedir contraseña:

ana ALL=(ALL) NOPASSWD: ALL

Ejercicio 4

1. Añade a ana al grupo sudo (Ubuntu/Debian):

sudo usermod -aG sudo ana

2. Conéctate como ana y prueba:

sudo whoami

→ debería responder root.

El comando principal para ver los permisos de archivos y directorios en Linux es **ls** con la opción **-l** (formato largo).

Uso Básico para Ver Permisos (ls -l)

Para ver los permisos de los archivos y directorios en tu ubicación actual, usa:

ls -l

Ejemplo de Salida:

-rw-rw-r-- 1 jose jose 1245 Oct 8 09:30 archivo.txt

drwxr-xr-x 2 jose jose 4096 Sep 20 15:10 mis_documentos/

Parte	Significado
drwxr-xr-x	**Permisos** (los primeros 10 caracteres)
2	Número de enlaces duros
jose	Propietario del archivo/directorio
jose	Grupo propietario
4096	Tamaño en bytes
Sep 20 15:10	Fecha y hora de modificación
mis_documentos/	Nombre del archivo/directorio

Búsqueda de Archivos por Permisos Especiales (find)

Puedes usar el comando **find** para buscar archivos que tengan **permisos especiales** o **UIDs/GIDs especiales**.

El comando **find** te permite buscar archivos por su tipo, nombre, fecha y, crucialmente, por sus permisos.

El número **4000** (por ejemplo) hace referencia al *bit* **SetUID (SUID)**.

Comando para buscar archivos con el *bit* SUID (4000):

Para buscar todos los archivos en el sistema (/) que tengan el *bit* SUID activado:

sudo find / -perm /4000 2>/dev/null

Parte	Significado
find /	Busca archivos, comenzando desde el directorio raíz (/).
-perm /4000	Filtra los resultados para buscar archivos que tengan el permiso especial **SUID (4)**. Esto se utiliza a menudo por razones de seguridad para identificar binarios que se ejecutan con permisos de *root*.
2>/dev/null	Esto redirige el **Error Estándar (2)** a /dev/null. Se utiliza para **ocultar** todos los mensajes de "Permiso denegado" que aparecen cuando find intenta acceder a directorios a los que no tiene acceso como usuario normal, dejando visible solo la lista de archivos encontrados.

Aparte del permiso especial **4000 (SetUID)**, existen otros dos bits de permisos especiales importantes en Linux que se usan para propósitos de seguridad y gestión:

- **2000 (SetGID - Set Group ID)**

El bit **SetGID** se representa con el valor octal **2000**.

Característica	Detalle
Valor Octal	**2000**
En ls -l	Reemplaza la x (ejecución) por una **s** en el campo del **grupo**. Si el grupo no tiene permiso de ejecución, se muestra como una **S** mayúscula.
En Archivos Binarios	Un programa que tiene SetGID activo se ejecuta con los permisos del **grupo propietario** del archivo, en lugar de con los permisos del grupo del usuario que lo ejecuta.
En Directorios (Uso Principal)	Cuando se crea un nuevo archivo o subdirectorio dentro de un directorio con SetGID, ese nuevo objeto **hereda el grupo propietario del directorio padre**, en lugar de tomar el grupo primario del usuario que lo creó. Esto es muy útil para directorios de trabajo compartidos.

2. 1000 (Sticky Bit)

El **Sticky Bit** (Bit Pegajoso) se representa con el valor octal **1000**.

Característica	Detalle
Valor Octal	**1000**
En ls -l	Reemplaza la x (ejecución) por una **t** en el campo de **"otros"**. Si "otros" no tienen permiso de ejecución, se muestra como una **T** mayúscula.
Uso Principal	Se usa casi exclusivamente en **directorios** (nunca en archivos binarios en sistemas modernos). Un directorio con el Sticky Bit activado permite que **cualquier usuario pueda crear archivos** dentro de él, pero **solo el propietario del archivo o el *root*** pueden renombrar, mover o borrar ese archivo.
Ejemplo Famoso	El directorio /tmp (donde todos los usuarios necesitan escribir) tiene el Sticky Bit activado para evitar que los usuarios borren los archivos de otros.

Combinación de Permisos Especiales

Los permisos especiales se suman a los permisos octales estándar (que van del 000 al 777).

Para establecer los tres bits especiales a la vez (algo inusual), usarías:

Un permiso octal completo tiene cuatro dígitos: [bits especiales][propietario][grupo][otros].

Ejemplo Práctico:

Para aplicar permisos de lectura/escritura/ejecución (777) y el Sticky Bit (1000) a un directorio:

chmod 1777 /mi/directorio/compartido

- **1**777: Sticky Bit activado.
- El resultado en ls -l se vería así: drwxrwxrwt.

6) Ejercicio integrador

Escenario: la empresa necesita un usuario nuevo juan en el área de marketing. Este usuario debe:

- Tener su directorio /home/juan.
- Formar parte del grupo marketing.
- Poder usar sudo para instalar programas, pero **solo** apt.

Pasos esperados del alumno:

1. Crear usuario juan.
2. Crear grupo marketing si no existe y añadir a juan.
3. Darle permisos de sudo específicos en visudo:

juan ALL=(ALL) NOPASSWD: /usr/bin/apt, /usr/bin/apt-get

- juan → el usuario al que le das permisos.
- ALL=(ALL) → puede ejecutar como cualquier usuario (root incluido).
- NOPASSWD: → opcional, evita que pida contraseña cada vez (puedes quitarlo si quieres que la pida).
- /usr/bin/apt, /usr/bin/apt-get → **únicos comandos permitidos** con sudo.

Guarda y cierra.

4. Verificar iniciando sesión como juan y ejecutando sudo apt update.

7) Cheatsheet rápido

usuarios y grupos

sudo useradd -m usuario

sudo passwd usuario

sudo groupadd grupo

sudo usermod -aG grupo usuario

id usuario

groups usuario

permisos

ls -l

chmod 755 archivo

chmod u+x archivo

chown usuario:grupo archivo

sudo

sudo visudo

añadir usuario o grupo con permisos

usuario ALL=(ALL:ALL) ALL

%grupo ALL=(ALL:ALL) ALL

drwxrwxrwt

La **t** al final de los permisos en un directorio como /tmp (drwxrwxrwt) significa que el **Sticky Bit** (Bit Pegajoso) está activado.

Significado del Sticky Bit (t) en Directorios

El Sticky Bit es un **permiso especial** que se aplica a directorios y tiene un propósito de seguridad crucial en entornos multiusuario, especialmente en directorios de acceso público como /tmp y /var/tmp.

Función Específica

Cuando el Sticky Bit está activo en un directorio, **solo el propietario del archivo o el usuario root pueden renombrar o eliminar los archivos** dentro de ese directorio, incluso si otros usuarios tienen permisos de escritura (w) sobre el directorio.

Contexto de /tmp

El directorio /tmp se utiliza para archivos temporales que pueden ser creados por cualquier usuario. Si no tuviera el Sticky Bit activado, cualquier usuario podría:

1. Crear un archivo en /tmp.
2. Eliminar o renombrar los archivos creados por *otros* usuarios, simplemente porque tienen el permiso de escritura (w) en el directorio.

Al tener la **t** (Sticky Bit) activada, se garantiza que:

- Cualquier usuario puede crear un archivo.
- **Solo el creador** (propietario) o root pueden borrar ese archivo, protegiendo así los archivos temporales de un usuario de ser manipulados por otro.

La diferencia entre t y T

- **t (minúscula):** Indica que el Sticky Bit está activado **Y** la terna de permisos para "otros" **tiene** el permiso de ejecución (x) activo. (Como en el ejemplo: drwxrwxrwt).
- **T (mayúscula):** Indica que el Sticky Bit está activado **PERO** la terna de permisos para "otros" **NO tiene** el permiso de ejecución (x) activo.

Permiso	Significado
d	Es un **directorio**.
rwx	Permisos del **propietario** (root).
rwx	Permisos del **grupo** (root).
rwt	Permisos para **otros** (el resto de usuarios).
t	Indica que el **Sticky Bit** está activo y previene que los archivos sean borrados por usuarios que no sean sus propietarios, aunque tengan permiso de escritura (w).