TUTORIAL DE LINUX 2

1. Un usuario en Linux es una entidad que puede interactuar con el sistema operativo. Puede representar a una persona física, un servicio o un proceso que necesita acceder a recursos del sistema, como archivos, programas y configuraciones. Cada usuario tiene un perfil único identificado por un nombre de usuario y un identificador (UID), además de pertenecer a uno o más grupos.

2. Diferencia entre un usuario normal y el usuario *root*:

3. Usuario normal: Tiene permisos limitados y puede realizar operaciones básicas, como crear, modificar o eliminar sus propios archivos, ejecutar programas y acceder a recursos asignados. Este tipo de usuario no tiene acceso directo a configuraciones críticas o archivos sensibles del sistema.

4. Usuario *root*: Es el superusuario y tiene privilegios completos en el sistema. Puede realizar cualquier acción, como modificar configuraciones del sistema, instalar o desinstalar software, acceder a todos los archivos y ejecutar comandos sin restricciones. Debido a esto, el acceso como *root* debe ser utilizado con precaución para evitar daños accidentales o comprometer la seguridad.

5. Por qué es necesario usar usuarios:

6. Seguridad: Separar las acciones y permisos de cada usuario ayuda a proteger los archivos y recursos del sistema. Un usuario no autorizado no puede modificar o acceder a archivos críticos.

7. Organización: En un entorno con múltiples usuarios, gestionar perfiles y permisos individuales asegura un acceso controlado y facilita la administración.

8. Colaboración: La creación de usuarios y grupos permite trabajar en conjunto compartiendo archivos y proyectos, mientras se mantiene la seguridad y privacidad.

9. Prevención de errores: Limitar las acciones de usuarios normales reduce el riesgo de ejecutar comandos peligrosos que podrían afectar al sistema.

TUTORIAL DE LINUX 2

10. Comandos para usuarios

Estos comandos están orientados a facilitar la navegación y administración para los usuarios:

- ls: Lista los archivos y directorios en tu ubicación actual.
    - Ejemplo: ls -l muestra una lista detallada con permisos y fechas.
- cd: Cambia el directorio de trabajo.
    - Ejemplo: cd /home/user te lleva al directorio indicado.
- pwd: Muestra la ruta completa del directorio actual en el que estás.
    - Ejemplo: Si estás en /home/user/docs, pwd te dará esa salida.
- man: Abre el manual de ayuda para cualquier comando.
    - Ejemplo: man ls muestra toda la información sobre el comando ls.
- exit: Cierra la terminal o sesión actual.

11. Comandos para archivos

Se utilizan para manejar, copiar, mover o modificar archivos y directorios:

- mkdir: Crea un nuevo directorio. ○ Ejemplo: mkdir nuevo_directorio crea un directorio con ese nombre.
- rm: Elimina archivos o directorios.
    - Ejemplo: rm archivo.txt elimina un archivo, y rm -r carpeta elimina un directorio y su contenido. • cp: Copia archivos o directorios.
    - Ejemplo: cp archivo.txt /ruta/destino/ copia el archivo a otra ubicación.
- mv: Mueve o renombra archivos.
    - Ejemplo: mv archivo.txt nuevo_nombre.txt cambia el nombre de un archivo.
- cat: Muestra el contenido de un archivo en la terminal.
    - Ejemplo: cat archivo.txt imprime su contenido.
- find: Busca archivos en el sistema.
    - Ejemplo: find / -name archivo.txt busca el archivo por nombre.

TUTORIAL DE LINUX 2

12. Comandos de configuración

Ayudan a cambiar permisos, propiedades o gestionar configuraciones del sistema:

- chmod: Cambia los permisos de un archivo o directorio. ○ Ejemplo: chmod 755 archivo otorga permisos específicos al archivo.
- chown: Cambia el propietario de un archivo o directorio.
    - Ejemplo: chown usuario:grupo archivo asigna nuevo propietario y grupo.
- top: Muestra los procesos en ejecución en tiempo real. ○ Ejemplo: top es útil para monitorear el uso de recursos.
- df: Muestra el espacio libre en sistemas de archivos.
    - Ejemplo: df -h muestra los valores en un formato legible (GB, MB).
- du: Muestra el espacio que ocupan los archivos o directorios.
    - Ejemplo: du -sh carpeta da un resumen del espacio utilizado por la carpeta.
- tar: Comprime o descomprime archivos.
    - Ejemplo: tar -czvf archivo.tar.gz carpeta crea un archivo comprimido.

13. Grupos

- Los grupos en Linux sirven para organizar usuarios y gestionar permisos de manera colectiva. Al asignar usuarios a un grupo, puedes controlar su acceso a archivos y recursos sin configurar permisos individualmente para cada usuario.

Comandos para gestionar grupos

1. groupadd: Se utiliza para crear un nuevo grupo en el sistema. *Ejemplo*: sudo groupadd desarrolladores creará un grupo llamado "desarrolladores".
2. groupdel: Elimina un grupo existente del sistema. *Ejemplo*: sudo groupdel desarrolladores eliminará el grupo "desarrolladores".

Permisos y privilegios (RWX)

- **R (Read)**: Permite leer el contenido del archivo o listar un directorio.
- **W (Write)**: Permite modificar el contenido de un archivo o cambiar los archivos en un directorio.
- **X (Execute)**: Permite ejecutar un archivo (como un programa) o entrar a un directorio.

Cada archivo/directorio tiene tres niveles de permisos:

TUTORIAL DE LINUX 2

1. **Propietario (u)**: Usuario que posee el archivo.
2. **Grupo (g)**: Grupo asociado al archivo.
3. **Otros (o)**: Cualquier otro usuario del sistema.

14. Comandos de gestión de permisos y propiedad

1. chmod: Cambia los permisos de un archivo o directorio. *Ejemplo*: chmod u+rwx archivo.txt otorga todos los permisos al propietario.
2. chown: Cambia el propietario de un archivo o directorio. *Ejemplo*: sudo chown usuario archivo.txt asigna "usuario" como propietario.
3. chgrp: Cambia el grupo asociado a un archivo o directorio. *Ejemplo*: sudo chgrp grupo archivo.txt asigna "grupo" al archivo.

15. Uso de sudo

- sudo permite a usuarios ejecutar comandos con permisos de superusuario de manera temporal. Esto es especialmente útil para realizar tareas administrativas sin necesidad de iniciar sesión como *root*, lo que mejora la seguridad. *Ejemplo*: sudo apt update actualiza los paquetes del sistema con privilegios elevados.

CONFIGURACION AVANZADA

La configuración avanzada en Linux incluye una variedad de herramientas y comandos que permiten gestionar y personalizar el sistema para maximizar su funcionalidad y seguridad. Uno de estos comandos es chage, que se utiliza para administrar las políticas de expiración de contraseñas de los usuarios.

16. Uso del comando chage

El comando chage permite configurar y visualizar información sobre la expiración de las contraseñas de los usuarios en Linux. Es útil para garantizar que los usuarios cambien sus contraseñas regularmente, mejorando la seguridad del sistema.

Opciones comunes de chage

1. -l: Muestra la información sobre la expiración de la contraseña de un usuario. ***Ejemplo*: sudo chage -l usuario muestra la fecha de caducidad de la contraseña del usuario, el tiempo entre cambios, y más.**
2. -E: Establece una fecha en la que el acceso del usuario al sistema caduca. ***Ejemplo*: sudo chage -E 2025-12-31 usuario configura que el usuario no pueda acceder al sistema después del 31 de diciembre de 2025.**
3. -M: Especifica el número máximo de días que una contraseña es válida antes de que deba cambiarse**. *Ejemplo*: sudo chage -M 90 usuario obliga al usuario a cambiar su contraseña cada 90 días.**

TUTORIAL DE LINUX 2

4. -m: Define el número mínimo de días antes de poder cambiar la contraseña. ***Ejemplo*: sudo chage -m 7 usuario establece que el usuario solo puede cambiar su contraseña después de 7 días.**
5. -I: Establece el número de días de inactividad antes de desactivar la cuenta. ***Ejemplo*: sudo chage -I 30 usuario desactiva la cuenta si el usuario no inicia sesión en 30 días.**
6. -W: Configura el número de días de advertencia antes de que la contraseña caduque. ***Ejemplo*: sudo chage -W 7 usuario muestra un aviso 7 días antes de que la contraseña expire.**