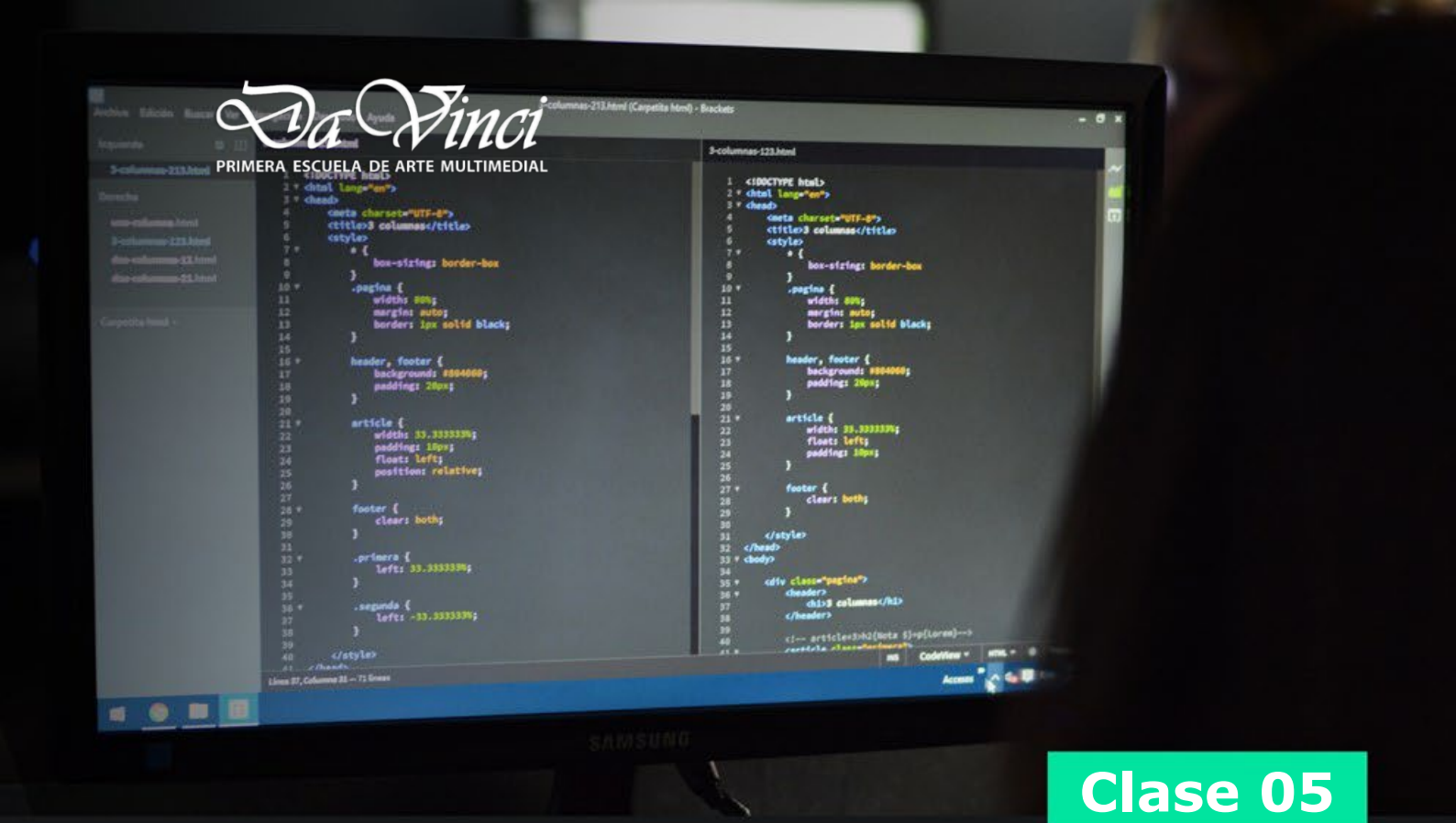


Clase 05

Diseño y Programación Web

Materia:

Sistemas Operativos

Docente contenidista: CARLASSARA, Fabrizio

Revisión: Coordinación

Contenido

Clase 5

¡Te damos la bienvenida a la materia **Sistemas Operativos**!

En esta clase vamos a ver los siguientes temas:

- Concepto de propiedad y permisos.
- Cómo crear y administrar usuarios.
- Cómo crear y administrar grupos.
- Cómo ver y modificar permisos en archivos y directorios.

Usuarios, grupos y permisos

Cuando un usuario se loguea en Linux, adquiere ciertos permisos a todos los recursos del sistema operativo de acuerdo a su usuario y grupo al que pertenece. Por eso, es importante tener en cuenta estos conceptos y entender cómo funcionan la idea de propiedad y permisos para poder operar Linux de una forma segura y proteger los recursos de un uso inapropiado.

Propiedad de archivos y directorios

Cuando se crea un archivo o directorio, el usuario que lo hace y el grupo principal al que pertenece reclama propiedad sobre él. Esto hace que sólo dos usuarios en todo el sistema puedan modificar ese archivo: el propio dueño y el usuario root.

Pongamos el caso siguiente, logueados con el usuario root, fueron creados un archivo y un directorio llamados clase_05.txt y clase_05. Cuando escribimos el comando ls -l obtenemos un detalle de los archivos y directorios que hay:

```
[root@localhost ~]# ls -l
total 8
-rw-------. 1 root root 701 mar  5 21:10 anaconda-ks.cfg
drwxr-xr-x. 2 root root   6 mar 20 12:57 clase_05
-rw-r--r--. 1 root root  17 mar 20 12:57 clase_05.txt
[root@localhost ~]#
```

En clases anteriores hemos discutido el significado de cada una de las columnas de esta salida tipo tabla. A lo largo de esta clase, vamos a enfocarnos mucho en la primera, tercera y cuarta columnas. La primera nos da información sobre permisos que vamos a discutir más adelante el significado particular de cada una de esas letras. La tercera y cuarta columna nos dan información del usuario y grupo que tiene propiedad del archivo o directorio respectivamente que, en este caso, tanto usuario como grupo coincide con el root.

Administrar usuarios y grupos

Podemos deducir de lo anterior que sólo el usuario root va a ser capaz de modificar este archivo. Para poder corroborar y entender mejor el concepto de propiedad en Linux, vamos a necesitar jugar un poco con los usuarios y grupos, por lo que necesitamos entender cómo administrarlos y los comandos que nos van a ayudar a hacerlo.

Crear usuarios

Vamos a crear un usuario adicional llamado *alumno*, para eso, vamos a recurrir al comando **useradd**. La sintaxis para este comando es **useradd [opciones] [nombre de usuario]** donde tendremos varias opciones para tratar:

- **-d:** define un directorio de inicio para el usuario.
- **-e:** configura una fecha en la que el usuario va a expirar. La fecha debe ponerse en el formato YYYY-MM-DD.
- **-f:** define cuántos días deben pasar para que se caduque la contraseña y se desactive el usuario.
- **-g:** define el grupo principal al que el usuario pertenece.
- **-m:** crea el directorio personal del usuario si todavía no lo tiene.

Si ahora corremos el comando *useradd -m alumno* y vamos a tener un nuevo usuario con directorio creado que podemos verificar con el comando *ls /home*.

Otro lugar donde verificar los datos del usuario es un archivo especial llamado passwd ubicado en /etc. Si corremos el comando *cat /etc/passwd* podemos obtener algo como esto:

Probablemente hay demasiada información confusa en este archivo, pero tratemos de descifrar un poco lo que dice. Esencialmente cada fila contiene información de un usuario que existe en el sistema operativo y, de hecho, podemos ver que la última fila es el usuario que creamos y el resto son usuarios propios del sistema. La información de cada fila contiene información importante del usuario separada en campos a través de ":". Los campos se encuentran dispuestos de la siguiente manera: name:password:UID:GID:GECOS:home directory:shell. De estos podemos definir:

- **name:** nombre de usuario.
- **password:** contraseña, aunque notarán que en todos los casos figura una x, debido a que ya no se guarda la contraseña en este archivo, sino que se guarda en /etc/shadow de forma encriptada por seguridad.
- **UID:** corresponde al identificador de usuario. Los números menores a 1000 suelen ser para usuarios usados por servicios. El usuario root siempre tiene el identificador 0.
- **GID:** es el identificador del grupo primario del usuario. Si bien un usuario puede pertenecer a muchos grupos, solo uno es el primario. Cuando un usuario crea un archivo o directorio, el grupo primario se convierte en dueño de él.
- **GECOS:** texto opcional descriptivo del usuario. Las siglas vienen de General Electric Comprehensive Operating System.
- **home directory:** ruta absoluta al directorio de inicio del usuario.
- **shell:** shell que usa el usuario.

Modificar usuarios

Una vez que tengamos los usuarios creados, podemos modificarlos a través del comando **usermod**. Este comando hace los cambios necesarios en los archivos que guardan configuración de usuarios como el /etc/passwd. La sintaxis para usarlo es **usermod [opciones] [usuario]** y algunas de las opciones disponibles son:

- **-c:** nos permite agregar un comentario en el parámetro GECOS del usuario.
- **-d:** nos permite redefinir el directorio de inicio del usuario.
- **-e:** permite configurar la fecha de vencimiento del usuario en el formato AAAA-MM-DD.
- **-g:** nos deja elegir el nuevo grupo primario del usuario.
- **-l:** nos permite renombrar el nombre del usuario.
- **-L:** bloquea un usuario.
- **-U:** desbloquea un usuario.
- **-u:** nos deja redefinir el nuevo número de identificación del usuario.

Configurar contraseñas

Una forma de asignar o cambiar contraseñas es con el comando **passwd**. La sintaxis para usar este comando es **passwd [opciones] [usuario]**. En cuanto a opciones, podemos usar las siguientes:

- **-d:** para borrar la contraseña del usuario elegido.
- **-e:** hace que la contraseña se venza y que en el próximo inicio de sesión se tenga que usar una nueva contraseña.
- **-i:** esta opción nos deja especificar la cantidad de días a esperar para eliminar un usuario luego de que la contraseña se haya vencido.
- **-l:** bloquea el logueo con contraseña para el usuario.
- **-u:** desbloquea el logueo con contraseña.
- **-n:** nos deja establecer el mínimo de días que hay que esperar para cambiar la contraseña.
- **-w:** nos deja establecer la cantidad de días previos a que una contraseña se venza para dar aviso.
- **-x:** establece cuantos días como máximo se puede dejar pasar antes de obligar a cambiar una contraseña.
- **-S:** muestra los parámetros del usuario que podemos encontrar en el archivo /etc/shadow.

Si establecemos una contraseña para alumno con el comando *passwd alumno* veremos que nos va a pedir que la escribamos y confirmemos. Vamos a poder confirmar verificando el archivo /etc/shadow con el comando *cat /etc/shadow* que efectivamente una contraseña fue seteada:

Vamos a ver en este archivo, de forma similar que con /etc/passwd, que tenemos una línea por usuario y cada campo separado por ":" otra vez. En este caso, se disponen como: name:password:last change:min:max:warn:disable1:disable2.

Podemos ver que los primeros dos campos son similares a los de /etc/passwd con la diferencia de que ahora la contraseña aparece pero encriptada. En el resto de los campos tenemos:

- **last change:** la fecha del último cambio de contraseña medido en días que pasaron desde el 1ro de Enero de 1970. Por ejemplo, el número 10957 representa el 1ro de Enero del 2000 porque hay 10957 días de diferencia entre ambas fechas.
- **min:** representa la cantidad de días que se deben esperar antes de cambiar la contraseña.
- **max:** representa la cantidad de días a los que se debe cambiar la contraseña.
- **warn:** representa la cantidad de días previos a que la contraseña expire para avisar al usuario de esto.
- **disable1:** representa la cantidad de días posteriores a que la contraseña se venza durante los cuales el usuario todavía tiene permitido acceder al sistema.
- **disable2:** representa la cantidad de días. medidos desde el 1ro de Enero de 1970, que deben pasar para que el usuario sea deshabilitado del sistema.

Crear y agregar a grupos

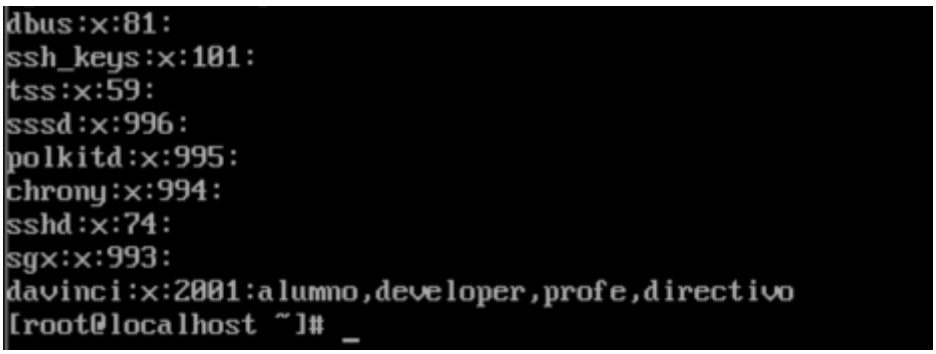
Así como existen usuarios, existen grupos a los que pueden pertenecer estos usuarios y que les puede dar algunos permisos adicionales. La forma de crear un usuario es con el comando **groupadd**. La sintaxis para el comando será **groupadd [opciones] [grupo]**. Como opciones destacadas podemos mencionar **-g que** nos deja especificar un identificador de grupo específico. Por ejemplo, podríamos crear el grupo davinci y asignarle el GID 2001 con el comando *groupadd -g 2001 davinci*.

Una vez que tengamos un grupo creado, podemos agregar usuarios a determinados grupos con el comando usermod, que vimos anteriormente. Si queremos solamente agregarle un grupo al usuario sin cambiar su grupo primario, podemos usar una variante de la opción -g que es -aG (corto para --append --group) lo que hace que el grupo que indiquemos sólo se agrega al usuario. Por ejemplo, si nuestro usuario alumno ya pertenece al grupo alumnos, podemos agregarle a un grupo llamado davinci con el comando *usermod -aG davinci alumno*.

Podemos agregar múltiples usuarios al mismo grupo, por ejemplo, acá se ve en el archivo /etc/passwd que discutimos anteriormente que los usuarios: alumno, developer, profe y directivo todos tienen un GID de 2001 que corresponde al grupo que creamos anteriormente:

Se puede verificar en el archivo /etc/group que el GID del grupo davinci es de 2001 y que coincide con el de estos usuarios. En este archivo vamos a ver la información como en los demás que hemos

visto, donde cada fila es un grupo y las columnas tienen los datos name:password:GID:members. Los campos son similares a los de los archivos que estuvimos analizando, donde el tercer campo, GID, es el identificador de grupo que vimos en el archivo anterior. El último campo contiene los miembros que pertenecen a ese grupo.



Si queremos modificar el grupo, podemos usar el comando **groupmod** que nos permite cambiar el nombre y el GID. La sintaxis es **groupmod [opciones] [grupo]** donde las opciones serán:

- **-g:** nos permite especificar el GID para cambiarle al grupo.
- **-n:** nos permite indicar el nuevo nombre del grupo.

Los cambios que hagamos con este comando se van a ver reflejados en todos los usuarios que pertenezcan a este grupo.

Eliminar usuarios y grupos

Ya hemos visto cómo crear y modificar usuarios y grupos, nos queda analizar cómo eliminarlos.

Si queremos eliminar usuarios, usamos el comando **userdel [opciones] [usuario]** donde como opción podemos usar **-r** que eliminará el directorio del usuario junto con él. Este comando elimina las entradas que mencionen al usuario de los archivos que discutimos anteriormente pero todos aquellos archivos que hayan sido propiedad de este usuario eliminado quedan como propiedad del UID de este usuario. Si eventualmente se crea un usuario con el mismo UID, este usuario va a tener propiedad sobre esos archivos.

Para eliminar grupos, usamos el comando **groupdel [grupo]**. De esta forma, eliminamos el grupo elegido siempre que no sea el grupo primario de algún usuario.

Otros comandos

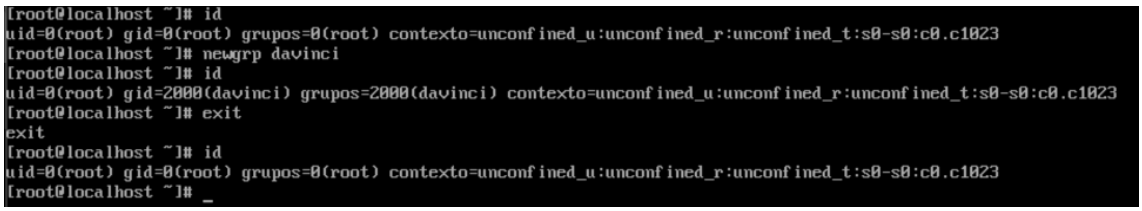
Podemos mencionar algunos comandos adicionales que nos pueden servir cuando estemos trabajando con usuarios y grupos.

El comando **id** nos brinda información sobre los IDs de un usuario. La sintaxis es **id [opciones] [usuario]** donde si el usuario se obvia, obtenemos información del usuario actual. Algunas opciones son:

- **-g:** muestra solo id del grupo primario.
- **-G:** muestra solo GIDs.
- **-n:** muestra nombres en vez de IDs.
- **-u:** muestra solo el ID del usuario.

El comando **groups [usuario]** nos muestra todos los grupos a los que pertenece un usuario en particular. Si no indicamos el usuario, nos brinda la información sobre el usuario actual.

Por otro lado, si temporalmente queremos cambiar de grupo para crear o modificar un archivo, podemos usar el comando **newgrp [grupo]**. Vemos abajo un ejemplo del usuario root cambiando del grupo root al grupo davinci temporalmente, por lo que todo lo que root cree durante ese tiempo, pertenece al grupo davinci. Verificamos esto con el comando id que mencionamos anteriormente.



Permisos de archivos y directorios

Ahora que entendemos un poco mejor cómo administrar usuarios y grupos, podemos discutir los distintos permisos que tenemos para archivos y directorios en Linux.

Cuando vimos el comando ls -a al principio de la clase obtuvimos un resultado como este:

```
[root@localhost ~]# ls -l
total 8
-rw-------. 1 root root 701 mar  5 21:10 anaconda-ks.cfg
drwxr-xr-x. 2 root root   6 mar 20 12:57 clase_05
-rw-r--r--. 1 root root  17 mar 20 12:57 clase_05.txt
[root@localhost ~]#
```

Ahora entendemos que estos archivos pertenecen al usuario y grupo root. Sin embargo, nos queda discutir la primera columna que es la de permisos. El primer carácter nos indica que tipo de archivo es, pero luego, tenemos nueve caracteres acomodados de la siguiente forma:

Donde cada letra indica: permisos de lectura (r), escritura (w) y ejecución (x). Estas siglas, si están presentes, nos dicen qué clase de permisos tiene el usuario propietario, el grupo propietario y cualquier otro usuario o grupo del sistema respectivamente. Cuando ese permiso no existe, se reemplaza por un "-".

El significado de lectura, escritura y ejecución tiene significado distinto si es un archivo o directorio. Abajo dejamos una tabla con un resumen de lo que significa cada caso:

Permiso	Definición para archivos	Definición para directorios
r	Permite que el usuario abra y lea el contenido del archivo.	Permite que el usuario liste el contenido de un directorio si también tiene permisos de ejecución.
w	Permite que el usuario abra, lea y edite el contenido del archivo.	Permite que el usuario agregue o quite archivos del y al directorio si también tiene permisos de ejecución.
x	Permite que el usuario ejecute el archivo en memoria si es un programa o script.	Permite que el usuario entre al directorio y trabaje con su contenido.

Estos permisos son importantes de entender. El hecho de que tengamos permisos de lectura en un archivo hace que podamos ejecutar comandos como cat, more, head, tail o less. Si tenemos permisos de lectura pero no de escritura para un archivo, podemos abrir el archivo con un editor de texto como vi o nano, pero no podremos modificarlo y guardar los cambios.

Por otro lado, los permisos de ejecución nos habilitan a poder correr un programa o un script de shell. Noten que normalmente los archivos no tienen permisos de ejecución para evitar que no sean interpretados por el shell erróneamente a menos que explícitamente necesitemos ejecutar ese archivo.

En el caso de los directorios, necesitaremos permisos de lectura para listar el contenido de uno con un comando como ls. Si queremos mover, copiar o eliminar archivos de un directorio con comandos como mv, cp o rm, vamos a necesitar permisos de escritura.

El permiso de ejecución en un directorio asegura que los otros dos permisos se puedan usar. Esto hace que una forma sencilla de negarle a un usuario o grupo permisos sobre un directorio es directamente quitarle permisos de ejecución a ese usuario.

Cambio de permisos

Ahora que medianamente entendemos en qué consisten los permisos de archivos y directorios, nos queda entender cómo cambiarlos.

En primer lugar, tenemos que entender que sólo los usuarios o grupos propietarios y el root pueden cambiar los permisos de un archivo o directorio.

Con el usuario que corresponda, vamos a usar el comando **chmod** para cambiar permisos de archivos y directorios. La sintaxis para usarlo es **chmod [permisos] [archivo]**. En cuanto a la forma de asignar permisos, se puede hacer con una combinación de los campos que mostramos a continuación.

Categoría	Operación	Permiso
u (usuario)	+ (agrega permiso)	r (lectura)
g (grupo)	- (quita permiso)	w (escritura)
o (otros)	= (hace que el permiso sea el indicado)	x (ejecución)
a (todos)		

Algunos ejemplos de esto comando podrían ser:

- **chmod u+w file** que le agregaría permisos de escritura al archivo file para el usuario propietario.
- **chmod g+r-w,o=rwx file** que le agregaría permisos de lectura y quitaría de escritura a los miembros del grupo propietario mientras que le asigna a todos los demás permisos de lectura, escritura y ejecución.
- **chmod a+w file** que le agregaría a todos permisos de escritura sobre el archivo file.
- **chmod +x file** que le agregaría permisos de ejecución a todos al omitir la categoría.

Como alternativa, podemos configurar los permisos de forma numérica, sabiendo que r = 4, w = 2 y x = 1. Esto nos brinda las siguientes posibilidades:

Modo (solo una sección)	Número correspondiente
rwx	4 + 2 + 1 = 7
rw-	4 + 2 = 6
r-x	4 + 1 = 5
r--	4
-wx	2 + 1 = 3
-w-	2
--x	1
---	0

Esto implicaría que podemos setear permisos con el mismo comando con números de tres dígitos donde cada dígito corresponde a los permisos para usuario, grupo y otros respectivamente. Por ejemplo:

- **chmod 540 file** que asigna los permisos r-xr-----.
- **chmod 644 file** que asigna los permisos rw-r--r--.
- **chmod -R 755 dir** que asigna al directorio y todo su contenido recursivamente (-R) los permisos rwxr-xr-x.

Cambio de dueño

Por último, analizaremos el caso de si uno quisiera cambiar la propiedad sobre un archivo. Hemos visto que existen dos tipos de dueño sobre un archivo: el del usuario y el de grupo. Si llegáramos a tener la necesidad de cambiar esa propiedad, podemos usar el comando **chown**. La sintaxis para usarlo es **chown [opciones] [usuario]:[grupo] [archivo]**. Como opciones, podemos destacar el uso de **-R** que nos permitiría cambiar recursivamente la propiedad de todos los archivos que existan dentro del directorio al que nos estamos refiriendo. Algunos ejemplos que podemos mencionar son:

- **chown user file** que cambia la propiedad del archivo file a user pero mantiene el grupo.
- **chown :group file** que cambia la propiedad grupal del archivo file a grupo pero mantiene el usuario.
- **chown user:group file** los dos casos anteriores en un solo comando.
- **chown -R user:group dir** cambia recursivamente la propiedad de dir y todo lo que contenga al usuario user y el grupo group.

Este comando solamente puede ser usado por el usuario o grupo que tiene propiedad sobre ese archivo o el usuario root, no puede cambiarse con un usuario que no es dueño de este archivo o directorio.

Hemos llegado así al final de esta clase en la que vimos:

- Concepto de propiedad y permisos.
- Cómo crear y administrar usuarios.
- Cómo crear y administrar grupos.
- Cómo ver y modificar permisos en archivos y directorios.

El entender cómo se administran los permisos y la propiedad sobre archivos y directorios nos puede ayudar a entender mejor cómo hacer que nuestro sistema sea menos vulnerable y proteger información o programas sensibles.

La próxima clase veremos algunos otros comandos más avanzados de Linux.

Te esperamos en la **clase en vivo** de esta semana.
No olvides realizar el **desafío semanal**.

¡Hasta la próxima clase!

Bibliografía

Eckert, J. W. (2020). Linux+ and LPIC-1: Guide to Linux Certification. Cengage.

Para ampliar la información

IONOS. (s.f.). Linux useradd: comando para crear usuarios. En Digital Guide de IONOS. Recuperado de https://www.ionos.es/digitalguide/servidores/configuracion/comando-useradd-de-linux/

Red Hat. (s.f.). How to create, delete and modify groups in Linux. En Red Hat. Recuperado de https://www.redhat.com/sysadmin/linux-groups

SoftZone. (s.f.). Qué son y cómo se cambian los permisos en Linux. En SoftZone. Recuperado de https://www.softzone.es/linux/tutoriales/permisos-archivos-directorios-linux/