Tema 1

Configuración y gestión básica

Administración de Sistemas Operativos

Índice

[Esquema 3](#_Toc169025194)

[Material de estudio 4](#_Toc169025195)

[1.1. Introducción y objetivos 4](#_Toc169025196)

[1.2. El sistema operativo GNU/Linux. Características. 5](#_Toc169025197)

[1.3. Distribuciones GNU/Linux. 6](#_Toc169025198)

[1.4. Jerarquía de directorios. Utilidades. 9](#_Toc169025199)

[1.4. Referencias bibliográficas 27](#_Toc169025200)

[A fondo 28](#_Toc169025201)

[Entrenamientos 29](#_Toc169025202)

[Test 34](#_Toc169025203)

Esquema

Tabla

Descripción generada automáticamente

Material de estudio

1.1. Introducción y objetivos

Debido a su flexibilidad, seguridad y amplia gama de distribuciones, el sistema operativo GNU/Linux se ha convertido en una opción popular y robusta tanto para usuarios domésticos como para entornos comerciales. Los usuarios pueden personalizar y ajustar el entorno con este sistema operativo de código abierto, que ofrece una alternativa a otros sistemas propietarios. Este tema abordará los elementos básicos de GNU/Linux, como sus características distintivas, las distintas distribuciones disponibles, la estructura y gestión del sistema de archivos, la gestión de usuarios y la configuración de redes.

Los objetivos que se pretenden conseguir al final de este tema son:

* **Comprender las características del sistema operativo GNU/Linux.** Definir las características principales que distinguen a GNU/Linux de otros sistemas operativos. Analizar sus ventajas e inconvenientes.
* **Conocer las distribuciones GNU/Linux**. Explorar las distintas distribuciones de este sistema operativo, sus particularidades y funcionalidades.
* **Familiarizarse con la jerarquía de directorios y utilidades.** Aprender la jerarquía de directorios y archivos, su navegación y gestión.
* **Dominar el uso del intérprete de comandos (Shell) y los comandos básicos.** Conocer la importancia de la Shell en Linux y los comandos básicos de manejo del sistema.
* **Configurar redes en GNU/Linux.** Comprender, configurar y gestionar las redes desde este sistema operativo.
* **Administrar permisos y usuarios.** Conocer la gestión de permisos y aprender a administrar usuarios y grupos.

1.2. El sistema operativo GNU/Linux. Características.

GNU/Linux es un sistema operativo conocido normalmente como Linux, de código abierto, multitarea, multiusuario y multiplataforma. Está compuesto por herramientas y comandos desarrollados bajo el Proyecto GNU (GNU's Not Unix («GNU No es Unix»)).

Las características principales de este sistema operativo son;

* **Código abierto**
* Este sistema operativo es desarrollado y mantenido por una amplia comunidad de desarrolladores y usuarios.
* Licencia GPL: Su código fuente es público y cualquier programador puede modificarlo y distribuirlo. Por tanto, existe gran cantidad de software disponible para este sistema operativo.
* **Multiplataforma**
* Puede instalarse y ejecutarse en multitud de dispositivos.
* **Multitarea y multiusuario**
* Permite ejecutar en paralelo múltiples programas y procesos
* Permite que varios usuarios puedan trabajar de forma concurrente a un sistema y acceder a los recursos .
* **Portable**
* El núcleo de Linux puede ser compilado y ejecutado en múltiples plataformas de CPU.
* **Muy estable**
* Linux es reconocido como uno de los sistemas operativos más estables y robustos que existen, es por ello que es utilizado en la mayoría de los servidores de producción y en sistemas de importancia crítica.
* **Seguro**
* Tiene un control estricto de permisos y usuarios .
* Es difícil de atacar por su arquitectura .
* **Interfaces**
* Puede ser utilizado mediante interfaces gráficas de usuario (GUIs) con funcionalidades distintas.
* También puede ser utilizado por interfaz de línea de comandos (CLI) que proporciona una herramienta muy potente principalmente para tareas de administración del sistema.
* **Distribuciones**
* Las diversas distribuciones que tiene este sistema operativo (Ubuntu, Debian, Fedora…) permite elegir aquella que más se adapta a las necesidades del proyecto.
* **Actualizaciones**
* Las actualizaciones se producen de forma regular lo que permite tener un sistema operativo mejorado y optimizado.
* **Desarrollo de software**
* Proporciona de serie una amplia variedad de herramientas para el desarrollo y compilación de código.
* Tiene soporte para multitud de lenguajes de scripting para la realización de tareas automáticas.
* **Configuración de redes**
* Contiene herramientas avanzadas para la configuración y administración de redes, así como robustez para soportar servicios en red.

1.3. Distribuciones GNU/Linux.

Las distribuciones GNU/Linux, comúnmente denominadas "**distros**", son versiones del sistema operativo GNU/Linux, cada una personalizada y optimizada para distintos propósitos y preferencias. A continuación, se presentan algunas de las distribuciones más destacadas junto con sus características principales (NBX Soluciones, 2024):

|  |  |
| --- | --- |
| **Distribución** | **Características principales** |
| UBUNTU | Esta distribución está basada en Debian y es una de las más utilizadas. Su objetivo principal es ser amigable para todos los usuarios ya que hasta el momento Linux se había convertido en un sistema operativo difícil de manejar para usuarios no técnicos. Es muy utilizada en escritorio (versión GNOME) y para servidores. |
| DEBIAN | Es una de las distribuciones más fiables hasta el momento. Muchas de las distribuciones posteriores que han surgido de Linux se originan en Debian porque ofrece una gran estabilidad y calidad. Al crear una distribución basada en Debian, se tiene acceso a todas las actualizaciones de seguridad, rendimiento y compatibilidad que el sistema ya proporciona. Al igual que Ubuntu, su uso principal es en escritorio(GNOME, KDE, entre otros) y sobre todo en servidores. Al ser una de las distribuciones más veteranas existe gran cantidad de información y software disponible. |
| FEDORA | Esta distro está basada en Red Hat Enterpise Linux. El objetivo de este sistema operativo es el usuario final. Su uso es principalmente en escritorio y para el desarrollo. Está enfocada en la incorporación de nuevas tecnologías. Aunque es un sistema operativo de código libre, la empresa Red Hat apoya y colabora en esta distribución y en la mejora de su kernel. |
| CENTOS | (Community Enterprise Operating System) Esta distribución está orienta al entorno empresarial principalmente. Su propósito es ser la alternativa gratuita y libre de Red Hat Enterprise Linux (RHEL) que es una distribución comercial basada en GNU/Linux pero que contiene servicios y características que requieren de pagos. En este contexto CentOS surge para cubrir todas las necesidades de forma totalmente gratuita y garantizar la compatibilidad con RHEL excluyendo los servicios que están bajo suscripción. |
| ARCH LINUX | Su base es independiente de cualquier otra distribución. No tiene interfaz gráfica por lo que su utilización se hace compleja. Está orientado a usuarios avanzados con conocimientos técnicos. Entre sus características destaca Rolling release por lo que está siempre actualizado sin necesidad de reinstalar. Permite realizar configuraciones múltiples para componer distintos contextos y administrar controles estrictos de usuario. Existe mucha documentación sobre esta distribución. |
| SUSE | Suse es una de las distros más extendidas y la más antigua de Linux. Originalmente apareció para el mundo empresarial pero también se desarrolló la versión para el usuario final denominada OpenSuse.  Es un sistema muy estable cuya instalación es sencilla. Proporciona herramientas avanzadas de serie como Yast para la instalación y administración de software.  OpenSuse contiene muchas de las herramientas proporcionadas por Suse Linux Enterprise, como por ejemplo Yast. |
| LINUX MINT | Esta distro está basada en Ubuntu. Tiene una interfaz sencilla muy similar a la de Windows. Su objetivo es la comodidad de uso del usuario final para que pueda utilizar Linux sin necesidad de conocimiento técnico. |
| MANJARO | Su núcleo está basado en Arch Linux. Soporta varios entornos de escritorios como Gnome y Xfce y está orientado al público general debido a su facilidad de uso. Entre sus características desctaca el Rolling release de Arch Linux que le permite estar siempre actualizado a la última versión. |
| KALI LINUX | Esta distro está especialmente orientada a temas de seguridad y pruebas de acceso malintencionado. Incluye herramientas para hacking ético y seguridad. Utilizada comúnmente por profesionales avanzados especialistas en seguridad informática. |

Tabla 1. Distribuciones de GNU/Linux. Fuente: elaboración propia.

Puedes encontrar cómo instalar una distribución de Ubuntu sobre VirtualBox en el apartado “A fondo”

1.4. Jerarquía de directorios. Utilidades.

En GNU/Linux la organización de los directorios sigue una estructura jerárquica. Cada uno de los directorios tiene una función típica que veremos a continuación. Todos los directorios y ficheros cuelgan del directorio raíz (/) que es la base del sistema de archivos.

Dentro de los directorios de GNU/Linux, aquellos que son más importantes son los siguientes (Caballero, 2017):

**/bin (Binaries)** Contiene los comandos esenciales para el correcto funcionamiento del sistema cuando se monta en modo monousuario. Estos comandos son por ejemplo ls, cat, cd, mv, rm, cp, etc. En este directorio se encuentra tambien la Shell bash y otras utilidades importantes para el sistema.

**/sbin (System Binaries)** Similar a /bin pero con comandos específicos para la administración del sistema que sólo pueden ser utilizados por el usuario root.

**/boot** Contiene los archivos necesarios para el arranque del sistema y son manejador por el gestor de arranque. Entre otros se encuentran archivos con la configuración del kernel, configuración y módulos del gestor de arranque, etc.

**/dev (Devices)** Linux identifica los dispositivos como archivos. Este directorio contiene los archivos que representan dispositivos. Por ejemplo /dev/sda representa el primer disco duro del sistema, /dev/tty las terminales, etc. Además, este directorio contiene también ficheros que representan dispositivos virtuales que no tienen identidad física, /dev/random produce números aleatorios y /dev/null descarta la la salida de un comando.

**/etc (Etcetera)** Este directorio contiene los ficheros de configuración de todo el sistema. Estos ficheros son editables y pueden ser modificados por un usuario (depende de permisos) para cambiar la configuración. Contiene también scripts (ejecutables) para iniciar y para el servidor o demonios. Ejemplo /etc/passwd contiene los usuarios del sistema.

**/home** es el directorio específico de cada usuario. En este directorio se guardan los archivos personales de datos y configuración. Cada usuario tiene un directorio /home de la forma /home/user.

**/lib (Libraries)** Contiene las librerías necesarias para los binarios de las librerías /bin y /sbin. Incluye librerías del núcleo kernel.

**/lost+found** Este directorio contiene los archivos dañados que el sistema ha encontrado. En cada arranque del sistema se chequea el sistema de ficheros y aquellos corruptos los deja en este directorio con la finalidad de recuperar la mayor cantidad posible.

**/media** Bajo este directorio se crea un subdirectorio por cada uno de los dispositivos extraíbles que se quieran montar. Por ejemplo, si queremos usar una unidad de USB en este directorio se montará automáticamente una subcarpeta para poder ser utilizado.

**/mnt (mount)** Punto de montaje temporal para montar sistemas de archivos. Es utilizado por los administradores de sistemas para montar dispositivos de forma temporal.

**/opt (Optional)** Contiene subcarpetas con paquetes de software opcionales comúnmente utilizados por software propietario desde fuera del sistema de paquetes del sistema.

**/proc** Directorio donde se encuentra toda la información del sistema desde el kernel a datos de configuración y procesos en ejecución. Por ejemplo /porc/cpuinfo proporciona la información sobre la CPU.

**/root** Es un directorio con la misma funcionalidad que /home pero exclusivo para el usuario root o administrador.

**/run** Contiene toda la información sobre la ejecución del sistema, como sockets, identificadores de proceso y su estado, etc.

**/tmp (Temporary)** Ubicación donde se encuentran los ficheros temporales utilizados por programas y usuarios. Estos ficheros son eliminados en cada reinicio de sistema o pueden ser eliminados en cualquier momento. Si un programa necesita que los datos perduren más allá del reinicio del sistema, aunque sean ficheros temporales, debe utilizar el directorio /var/tmp.

**/usr (User)** Contiene las aplicaciones y archivos que son utilizados por el usuario. Son datos de sólo lectura. Los más importantes son:

• /usr/bin: comandos binarios no esenciales.

• /usr/sbin: comandos binarios no esenciales para el administrador del sistema.

• /usr/lib: bibliotecas no esenciales.

• /usr/local: software y datos instalados localmente.

• /usr/share: datos compartidos, como documentación y archivos de configuración.

**/var (Variable)** Los archivos de este directorio contiene datos variables como los registros de log del sistema ( /var/log), cache de datos (/var/cache) o colas de correo (/var/mail)

1.5. Intérprete de comandos. Shell. Comandos básicos.

El intérprete de comandos, conocido como shell, es un programa presente en sistemas operativos Unix y similares, como GNU/Linux, que ofrece una interfaz para que el usuario pueda interactuar con el sistema operativo. Funciona como un intermediario entre el usuario y el núcleo del sistema, permitiendo la ejecución de comandos, la automatización de tareas y la gestión del sistema.

No todas las Shell son iguales, habiendo varios tipos disponibles en GNU/Linux. Cada una de ellas tiene sus propias características y difieren sustancialmente en la sintaxis. Las más comúnmente utilizadas son:

* **Bash (Bourne Again Shell):** Este es el más usado y viene por defecto en

muchas distribuciones de Linux. Es fácil de usar y muy potente. Es compatible con Bourne Shell presente en todas las distribuciones.

* **Sh (Bourne Shell):** Es el primero de los shells creados. Es más simple y básico,

y aunque ha sido reemplazado por Bash en muchos casos, aún se usa en scripts antiguos.

* **Zsh (Z Shell):** Este es como un Bash con mejores características y mejoras de

usabilidad. Tiene muchas características adicionales que lo hacen muy popular entre los usuarios avanzados.

* **Ksh (Korn Shell):** Un shell antiguo que combina características de bash y sh.

Es potente y sigue siendo usado en algunos sistemas.

* **Csh (C Shell):** Tiene una sintaxis similar a C, lo que puede ser familiar para los

programadores de C.

* **Tcsh (Tenex C Shell):** Basado en el C Shell (csh), es conocido por su enfoque

en la programación y algunas características interactivas avanzadas.

* **Fish (Friendly Interactive Shell):** Este es un shell diseñado para ser fácil de

usar y muy amigable, con muchas características útiles y una bonita presentación.

1.5.1. Funciones del Shell.

El Shell ofrece funciones variadas para el funcionamiento y administracion: ejecución de comandos, automatización, scripting, configuración, etc. A continuación, vamos a ver las más utilizadas:

* **Ejecución de Comandos**

Permite a los usuarios ejecutar comandos usando para ello la sintaxis

comando [opciones] [argumentos]

• Ejemplo: ls -l /home/usuario devuelve el listado de ficheros que hay en el directorio /home/usuario

Texto

Descripción generada automáticamente

Figura 1. Ejemplo de utilización del comando ls. Fuente: elaboración propia.

* **Automatización y Scripting**

Facilita la creación de scripts, que consisten en unos archivos de texto con una serie de comandos para automatizar tareas repetitivas.

Los scripts pueden incorporar lógica de programación como bucles, condicionales, funciones, etc.

Aquí tienes un ejemplo de un script sencillo donde #/!bin/bash indica a la Shell que lo que se está escribiendo es para que lo interprete Bash y a continuación muestra un resultado “Hola Usuario”

Imagen que contiene naranja, foto, oscuro, alimentos

Descripción generada automáticamente

Figura 2. Ejemplo de creación de un script básico. Fuente: elaboración propia.

Resultado de la ejecución

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

Figura 3. Resultado de la ejecución del script. Fuente: elaboración propia.

* **Redireccionamiento de entrada y salida**

Mediante comandos y programas se puede volcar una salida a un destino o se puede coger una entrada de un origen.

• > para redirigir la salida a un archivo.

• < para redirigir la entrada desde un archivo.

• | (pipe) para canalizar la salida de un comando como entrada de otro.

• Ejemplo: ls -l > listado.txt (redirige la salida del comando ls a listado.txt por lo que en este fichero se guardará el listado de ficheros del directorio en el que nos encontremos).

* **Gestión de Procesos**

Permite la gestión de procesos mediante comandos que lanzan, paran o eliminan proceso en el sistema.

Ejemplo de comandos para este fin son

• ps: Muestra una lista de procesos activos en el sistema.

Ejemplo: ps aux (muestra todos los procesos en ejecución).

• kill : Finaliza un proceso identificado con su pid

• bg: Reanuda un proceso detenido (suspendido)

• fg: Trae un proceso de segundo plano o detenido al primer plano.

• jobs: Lista de trabajos en curso en la sesión actual

* **Variables y Alias**

Permite el uso de variables para almacenar datos y configuraciones y alias para crear atajos de comandos largos o complejos.

* **Historial de Comandos**

Mantiene un registro de los comandos ejecutados, permitiendo reutilizarlos fácilmente. Se utilizan las flechas para poder moverse por el historial. El comando history muestra el historial de comandos.

1.5.2. Características avanzadas de Bash.

Como ya se ha comentado, Bash es uno de los Shell más utilizados por su amplia trayectoria y su potencia de uso. Esta Shell, además, ofrece características propias de gran utilidad:

* **Autocompletado**: Al presionar la tecla Tab, Bash puede completar automáticamente nombres de archivos, directorios y comandos.
* **Historial de comandos:** Permite navegar por los comandos previamente ejecutados utilizando las teclas de flecha.
* **Alias**: Facilita la definición de alias para simplificar la ejecución de comandos largos.
* **Scripts de inicio**: Archivos como .bashrc, .bash\_profile y .bash\_logout permiten configurar el entorno del shell y ejecutar comandos automáticamente al inicio y cierre de la sesión.
* **Funciones**: Permiten la definición de funciones dentro del shell para agrupar y reutilizar comandos.

1.5.3. Ejemplo de un Script Bash.

Para poner en práctica todo lo aprendido vamos a crear un script denominado backupDir.sh que va a guardar todo el contenido de la carpeta /home/usuario en un directorio de destino que se llama /backups y lo va a guardar con nombre /backups/backup-fecha.tar.gz donde ‘fecha’ es la fecha del sistema.

El código correspondiente a este script es el siguiente:

#!/bin/bash

# Definir variables

backup\_dir="/backups"

source\_dir="/home/usuario"

date=$(date +%Y-%m-%d)

backup\_file="$backup\_dir/backup-$date.tar.gz"

# Crear el directorio de backup si no existe

mkdir -p $backup\_dir

# Crear el archivo de backup

tar -czf $backup\_file $source\_dir

# Mostrar mensaje de éxito

echo "Backup de $source\_dir completado en $backup\_file"

Una vez creado el script podemos ejecutarlo indicando en la consola de comandos podemos hacerlo con el comando ./backupDir.sh.

1.5.4. Comandos.

Los comandos de GNU/Linux son sentencias que los usuarios ejecutan en una línea de comandos llamada CLI. Estos comandos permiten realizar múltiples tareas sobre el sistema. A continuación, veremos los comandos más comunes y su funcionalidad.

**Comandos Básicos**

**ls**(List): Lista archivos y directorios.

ls -l muestra detalles en formato largo.

ls -a muestra archivos ocultos.

**cd** (Change Directory): Cambia el directorio actual.

cd /ruta/al/directorio te mueve al directorio especificado.

cd .. sube un nivel en el árbol de directorios.

**pwd** (Print Working Directory) Muestra la ruta completa del directorio en el que se encuentra el usuario.

**cp** (Copy) Copia archivos o directorios.

cp archivo\_origen archivo\_destino copia un archivo.

cp -r directorio\_origen directorio\_destino copia un directorio de forma recursiva.

**mv**(Move) : Mueve o renombra archivos y directorios.

mv archivo\_origen archivo\_destino mueve o renombra un archivo.

mv directorio\_origen directorio\_destino mueve o renombra un directorio.

**mkdir** (Make Directory) : Crea un nuevo directorio.

mkdir nombre\_directorio crea un directorio con el nombre especificado.

rmdir: Elimina un directorio vacío.

**rmdir** (Remove Directory): elimina el directorio especificado si está vacío.

rm nombre\_archivo elimina el archivo especificado.

rm -r nombre\_directorio elimina el directorio y su contenido de forma recursiva.

**Comandos de información del sistema**

**uname**: Muestra información del sistema. Indica el sistema operativo que está ejecutándose

**top**: Muestra los procesos en ejecución y el uso del sistema en tiempo real.

**df** (Disk Free): Muestra un resumen del espacio en disco de todos los sistemas de archivos montados.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Figura 4. Resultado de la ejecución del comando df. Fuente: elaboración propia.

**Free**: Muestra la información sobre el uso de la memoria del sistema

**Comandos de manipulación de archivos**

**echo**: Muestra un mensaje en la pantalla. Por ejemplo

echo "Hola Mundo" muestra "Hola Mundo".

**cat**: Muestra el contenido de un archivo.

Texto

Descripción generada automáticamente

Figura 4. Resultado de la ejecución del comando cat. Fuente: elaboración propia

**less**: Muestra el contenido de un archivo página por página. Es muy útil para archivos muy grandes que de otra forma serían imposibles de visualizar.

**head**: Muestra las primeras líneas de un archivo

**tail**: Muestra las últimas líneas de un archivo

**touch**: Se usa principalmente para crear archivos vacíos o actualizar las marcas de tiempo (timestamps) de archivos existentes.

El **comando vi** abre un editor de texto para crear fichero. Sus opciones principales son:

|  |  |
| --- | --- |
| **Comando** | **Descripción** |
| vi nombreFichero | abre un editor para el archivo indicado |
| vi -r nombrFichero | abre el editor para el archivo, pero en modo recuperación después de un fallo del sistema |

Tabla 2. Comando vi. Fuente: elaboración propia.

Una vez dentro del editor se pueden realizar distintas operaciones, algunos ejemplos son:

|  |  |
| --- | --- |
| **Opción** | **Descripción** |
| : w | guarda el fichero, si se indica un nombre de fichero a continuación implica guardar en el fichero indicado |
| :q | sale del editor. Si el fichero ha sido modificado y no queremos guardar cambios hay que forzar la salida con :q! |
| :wq | sale del editor guardando cambios. Equivale a :x |

Tabla 3. Opciones disponibles dentro de vi. Fuente: elaboración propia.

**Comandos de búsqueda y filtros**

**grep** "patrón" archivo busca el patrón especificado dentro del archivo.



Figura 5. Resultado de la ejecución del comando grep. Fuente: elaboración propia

**find** /ruta -name "nombre\_archivo" busca un archivo por nombre en la ruta especificada.

**sort** ordena las líneas de un archivo en base a unas opciones proporcionadas

1.6. Configuración de red.

La configuración de una red en GNU/Linux depende de la distribución en la que estemos trabajando y la red que se esté configurando. Existe métodos comunes a todas las distribuciones, pero determinados aspectos difieren entre ellas como por ejemplo los archivos de configuración.

A continuación, me muestra una tabla donde se indica, por cada distribución, el fichero de configuración que es necesario utilizar:

Distribuciones Debian/Ubuntu:

• Fichero de configuración de red : /etc/network/interfaces

• Fichero de configuración de DNS: /etc/resolv.conf

Distribuciones Red Hat/CentOS/Fedora:

• Fichero de configuración de red : /etc/sysconfig/network-scrips (directorio)

• Fichero de configuración de DNS: /etc/resolv.conf

Distribuciones Suse, OpenSuse:

• Fichero de configuración de red : /etc/sysconfig/network

• Fichero de configuración de DNS: /etc/resolv.conf

Para empezar a realizar la configuración debemos conocer nuestro nombre de servidor, lo cual está almacenado en el fichero /etc/hostname. Este nombre viene por defecto con la instalación de la distribución, pero puede ser modificado usando el comando hostnamectl con permisos de administrador.

Como hemos comprobado cada una de las distribuciones tiene sus propios ficheros de configuración de red, aunque es común el fichero de configuración de nombres de DNS /etc/resolv.conf. En este fichero se encuentran los nombres de dominio y su tracción a IPS numéricas que lo que realmente es comprendido por el sistema.

Nuestro sistema operativo va a ir a buscar al fichero /etc/hosts (presente en todos los sistemas operativos, no solo Linux) en primer lugar estas traducciones. Si la información que necesita se encuentra en este fichero lo lee de él y sigue su curso, en caso contrario busca en el fichero DNS para saber dónde debe dirigirse.

GNU/Linux ofrece algunos comandos para la gestión y configuración de las redes. A continuación, se describen aquellos más comúnmente utilizados:

**Ifconfig** es un comando que configura los interfaces de red. En ocasiones puede suceder que nuestra distribución no venga con este comando preinstalado , lo que provocará error de comando no encontrado, por lo que deberemos instalarlo, para ello sobre la consola escribir **sudo apt install net-tools**

Una vez instalado podemos ejecutar el comando sin parámetros cuyo resultado será el informe de configuración de los interfaces de red.

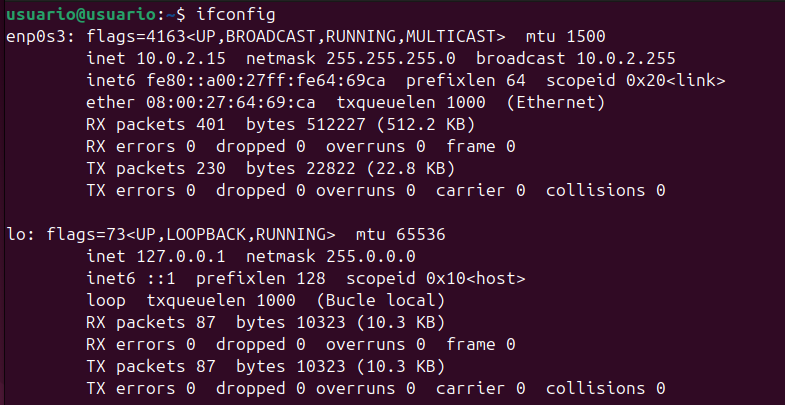


Figura 6. Resultado de la ejecución del comando ifconfig. Fuente: elaboración propia

Si queremos establecer la dirección IP del interfaz de red **usaremos ifconfig eth0 *ipcompleta***, por ejemplo:

ifconfig eth0 192.1.100.22.

Una vez establecida la dirección IP podemos levantar la interfaz con **ifconfig eth0 up** y desactivarla con **ifconfig eth0 down.**

Otro comando para la configuración de red es **ip.** Este comando es la versión más completa de ifconfig y permite gestionar la configuración de redes, rutas y túneles.

Si queremos mostrar las interfaces de red usaremos el comando **ip addr.**

Si lo queremos establecer una dirección IP tendremos que usar el comando **ip address add *ipcompleta* dev eth0**. Una vez establecido lo activaremos con **ip link set eth0 up** y lo desactivaremos con **ip linst set eh0 down.**

Ip permite también añadir una ruta por defecto, esto es, indicar una ruta cuando el router no es capaz de encontrar el camino en las tablas de rutas. Para ello se utiliza **ip route add default via ipcompleta dev eth0**.

Finalmente, si lo que queremos es consultar la tabla de rutas que está dada de alta en el momento actual en el sistema usaremos **ip route show.**

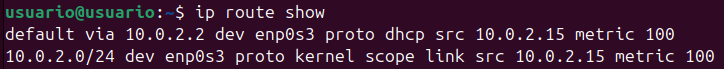


Figura 7. Resultado del comando ip route show. Fuente: elaboración propia

En cuanto a las tablas de enrutamiento ip del sistema, Linux tiene el comando **route** que nos permite gestionar dichas tablas. Para mostrar la tabla de rutas usaremos el comando sin parámetro

Texto

Descripción generada automáticamente

Figura 8. Resultado del comando route. Fuente: elaboración propia

Podemos añadir una ruta por defecto con **route add default gw *ipcompleta* eth0** o añadir una ruta estática **con route -p add -net ipcompleta /24 -gateway ipcompleta /24.**

Cuando hemos finalizado de configurar todo lo relacionado con la red, el comando **ping**, nos permitirá comprobar con certeza la conectividad. Para ello envía paquetes ICMP a través de la red que son solicitudes de eco hacia una dirección IP. Si la dirección IP está levantada responderá al ping con un ICMP Echo Reply.

**netstat** es un comando que nos permite conocer las conexiones de red (tanto entrantes como salientes), las tablas de enrutamiento y una serie de estadísticas de las interfaces de red. Este comando no es exclusivo, al igual que ip, de sistemas operativos Linux, sino que también se encuentra en Sistemas operativos Windows.

1.7. Permisos. Administración de usuarios.

La gestión de permisos y usuarios en Linux es un componente crítico para garantizar la seguridad y la integridad del sistema. Cada archivo y directorio en Linux tiene permisos que especifican quién puede leer, escribir o ejecutar el recurso. Estos permisos se clasifican en tres categorías: **usuario (propietario), grupo y otros**. Los comandos como **chmod**, **chown** y **chgrp** se pueden usar para gestionarlos. Además, la gestión de cuentas de usuario incluye la creación, modificación y eliminación de cuentas de usuario, así como la asignación de cuentas de usuario a grupos para que sea más fácil administrar permisos.

El uso de comandos como **useradd**, **usermod** y **userdel** es esencial para este proceso. La configuración adecuada de permisos y usuarios permite el control del acceso a los recursos del sistema, lo que protege la información y permite un entorno operativo seguro y eficiente.

A continuación, vamos a analizar los comandos mencionados, y otros adicionales, y sus principales características.

1.7.1. Comandos para la gestión de permisos de usuario

En Linux, la mayoría de las tareas de gestión de usuarios se realizan mediante el uso de comandos específicos que permiten la creación, modificación y eliminación de cuentas de usuario, así como la gestión de grupos y permisos. A continuación, se enumeran algunos de los comandos de gestión de usuarios más cruciales para Linux:

* **add user**: Crea un usuario nuevo al sistema. Permite opciones adicionales, como establecer el directorio principal **sudo useradd -m -d /home/username username**
* **usermod:** Cambia las características de una cuenta de usuario que ya existe.Si queremos agregar un usuario a un grupo existente: **sudo usermod -aG groupname username**
* **userdel**. Borra un usuario. **userdel username** borra el usuario con nombre username. Este comando sólo elimina el usuario por lo que si queremos eliminar también el directorio home del usuario usarmos **sudo userdel -r username**
* **passwd:** Modifica la contraseña del usuario. Este comando usando sin más opciones modifica la propia contraseña. Si se quiere cambiar la contraseña de otro usuario (debe hacerse con permisos de root) se usará **sudo passwd username**
* **usermod:** Modifica la información de un usuario. Ejemplo de uso: **sudo usermod -aG grupo1 username**, este comando agregra al usuario con nombre username al grupo1 que debe estar previamente creado.
* **id**: Muestra el UID (User ID), GID (Group ID) y los grupos a los que pertenece un usuario.
* **sudo**: Es uno de los comandos más importantes. Permite ejecutar comandos que requieren privilegios de superusuario para su ejecución. Se utiliza poniendo el comando sudo delante del comando principal que se quiere ejecutar.

1.7.2. Comandos para la gestión de permisos de grupos

La gestión de permisos de grupo en Linux es esencial para mantener la seguridad y la eficacia de un sistema multiusuario. Es fundamental establecer controles claros sobre quién puede acceder y modificar recursos en un entorno donde varios usuarios interactúan con diferentes archivos y recursos.

Algunos comandos para la gestión de permisos de grupo son:

* **groupadd**: Añade un nuevo grupo al sistema. Ejemplo: **sudo groupadd groupname** añade el grupo con nombre groupname al sistema.
* **groupdel**: Elimina un grupo del sistema. **sudo groupdel groupname**
* **gpasswd**: Añade o elimina usuarios de un grupo. El comando sudo **gpasswd –a username groupname** añade un usuario al grupo groupname y el comando **sudo gpasswd -d username groupname** lo elimina.

1.7.3. Permisos de archivos y directorios

Los usuarios y grupos que pueden leer, escribir o ejecutar un archivo o directorio se rigen por los permisos de GNU/Linux. Los tipos de permisos permitidos son **lectura (r)** que permite ver el contenido del archivo o listar el contenido del directorio; **escritura (w)** que permite modificar el contenido del archivo o añadir/eliminar archivos en el directorio y **ejecución (x)** que permite ejecutar el archivo (si es un programa/script) o acceder al directorio.

Cada archivo y directorio tiene un grupo asociado y un propietario. Los permisos se definen para el propietario, el grupo y los demás ( el resto de posibles usuarios).

En Linux, se pueden cambiar los permisos de acceso de archivos y directorios mediante el comando **chmod**. Este comando permite administrar la seguridad y el acceso a los recursos del sistema modificando quién puede leer, escribir o ejecutar un archivo o directorio.

Los usos más comunes de chmod incluyen asignación de permisos mediante notación simbólica. Partiendo de los permisos comentados, un ejemplo de uso sería añadir permisos de ejecución (x) al usuario (u) para el archivo script.sh.

Texto

Descripción generada automáticamente

Figura 9. Ejecución del comando chmod y visualización del resultado. Fuente: elaboración propia

De esta forma podemos dar cualquier tipo de permiso (r,w,x) a los posibles miembros (u (usuario), g(grupo), o(resto) ), basta con indicar en primer lugar el miembro al que se le da el permiso para a continuación indicar qué permiso. Por ejemplo, **chmod ugo+rwx fichero.sh** le da todos los permisos a todos los usuarios, grupos y otros.

Otra manera de utilizar chmod es mediante la notación octal. En este caso en vez de indicar con letras los permisos, se realiza indicando por cada miembro, que combinación de permisos se le asigna usando la siguiente notación.

|  |  |
| --- | --- |
| **Dígito** | **Descripción** |
| 0 | Ningún permiso |
| 1 | --x. Ejecución de archivos o acceso a directorios |
| 2 | -w-. Permiso de escritura |
| 3 | -wx Permisos de escritura y ejecución de archivos o acceso a directorios |
| 4 | r--. Sólo permiso de lectura |
| 5 | r-x. Permisos de lectura y ejecución de archivos o acceso a directorios |
| 6 | rw-. Permisos de lectura y escritura de archivos |
| 7 | Todos los permisos tanto para archivos como directorios |

Tabla 4. Permisos en Linux en base octal. Fuente: elaboración propia.

Descubre más sobre cómo funciona el comando **chmod** en el apartado “A fondo”

En GNU/Linux, se puede usar el comando **chown** para cambiar el propietario y, opcionalmente, el grupo de un archivo o directorio.

Para cambiar el propietario de un archivo.txt a un usuario conocido como nuevo\_usuario usaremos **chown nuevo\_usuario archivo.txt.**

Adicionalmente también permite cambiar tanto el propietario como el grupo de un archivo usando el comando **chown nuevo\_usuario:nuevo\_grupo archivo.txt.**

1.4. Referencias bibliográficas

Caballero, A. J., Administración de Sistemas Operativos. Ra-Ma, 2017.

García Carrasco, J. (2016). Administración de Sistemas Operativos. Editorial Síntesis.

NBX Soluciones ( 18 de enero de 2024) *Las distribuciones Linux que marcarán 2024.* https://www.linkedin.com/pulse/las-distribuciones-linux-que-marcar%C3%A1n-2024-nbx-soluciones-bhadc/

A fondo

Comando Chmod: Cómo cambiar permisos de archivo en Linux.

Rosa, D. (2024). *Comando Chmod: Cómo cambiar permisos de archivo en Linux.* [FreeCodeCamp.](https://www.redalyc.org/articulo.oa?id=337630234004) https://www.freecodecamp.org/espanol/news/comando-chmod-como-cambiar-permisos-de-archivo-en-linux/

Esta página web explica al detalle los permisos que pueden existir, cómo leerlos y los pasos a seguir para su modificación.

Manual de instalación de Ubuntu sobre VirutalBox

ProgrammingKnowledge. (2024). *How to install Ubuntu 24,04 TLS on VirtualBox in Windows 11* [Vídeo]. Youtube. https://youtu.be/DhVjgI57Ino?si=csk63pm3nlYZa2jW

En este vídeo te muestra cómo instalar la herramienta VirtualBox como contenedor de máquinas virtuales y cómo descargar y añadir un sistema operativo Ubuntu.

Entrenamientos

Entrenamiento 1

* Crea un nuevo fichero bajo tu ruta actual y dale todos los permisos al usuario y permisos de lectura y escritura, pero no de ejecución al grupo y resto
* Desarrollo paso a paso
  + El comando touch crea un nuevo archivo bajo tu ruta actual
  + Chmod cambia todos los permisos al usuario, permisos de lectura y escritura al grupo, y ningún permiso al resto
* Solución

touch nuevo\_archivo.txt # Crea un nuevo archivo bajo tu ruta actual

chmod u+rwx,g+rw,o-rwx nuevo\_archivo.txt

Entrenamiento 2

* Crear un directorio llamado mi\_directorio en tu directorio home. A continuación, crea un archivo de texto llamado archivo.txt dentro de mi\_directorio y ecribe la frase "Hola, mundo!" en archivo.txt y muestra el contenido de archivo.txt en la terminal.
* Desarrollo paso a paso
* Navegamos al directorio home usando el comando cd
* Para crear un directorio utilizaremos el comando mkdir de la forma mkdir mi\_directorio dentro de la ruta actual.
* Para crear el archivo y añadirle el texto podemos hacerlo de dos formas:
  + Usando el comando vi y en el editor escribir el texto
  + Usando la redirección de la forma echo “Hola Mundo” > /mi\_directorio/archivo.txt.
* Para mostra el contenido del archivo podemos usar el comando cat
* Solución

cd ~

mkdir mi\_directorio

cd mi\_directorio

echo "Hola, mundo!" > archivo.txt

cat archivo.txt

Entrenamiento 3

* Crea un directorio en tu home llamado proyecto\_simple. Dentro de este directorio, hay varios archivos de texto. Necesitas realizar las siguientes tareas:
  + Crear un subdirectorio llamado backup.
  + Mover todos los archivos de texto (\*.txt) desde proyecto\_simple al subdirectorio backup.
  + Listar los archivos en el subdirectorio backup y guardar la lista en un archivo llamado lista\_backup.txt.
* Paso a paso
  + Cd navega al directorio home
  + Estando en el directorio home creamos un nuevo directorio con mkdir
  + Con el comando cd navegamos al nuevo directorio
  + Dentro de este nuevo directorio volvemos a ejecutar mkdir para crear un subdirectorio
  + El comando mv mueve todos los archivos de texto (\*.txt) al subdirectorio backup
  + Ls -l Lista los archivos que hay dentro del directorio backup y redirige la salida a lista\_backup.txt
* Solución

cd ~

mkdir proyecto\_simple

cd proyecto\_simple

mkdir backup

mv \*.txt backup/

ls backup/ > lista\_backup.txt

Entrenamiento 4

* Desarrolla un script que cree un archivo con el resultado de listar los ficheros y sus características del directorio actual y le de permisos de escritura y lectura, pero no de ejecución al usuario actual. A continuación, lanza el script.
* Desarrollo paso a paso
  + Abrimos el editor vi y añadimos como primera línea #!/bin/bash y escribimos el resto de comandos.
  + Ls -l lista el contenido del directorio actual y la salida la guarda en fichero
  + Con chmod se dan permisos de lectura(r) y escritura(w) al usuario (u) actual para ese archivo
  + Volvemos a usar chmod pero con permisos para grupo (g) y otros(o) a los que solo le damos permiso de lectura (r)
  + Cerramos el editor con :w para guardar y :q para salir
  + Para la ejecución del script demos darnos permisos de ejecución (x) y a continuación ejecutarlo anteponiendo un ./ antes del nombre del script
* Solución

#!/bin/bash

ls -l > lista\_directorio.txt

chmod u+rw lista\_directorio.txt

chmod go+r lista\_directorio.txt

#Para la ejecución

chmod +x scriptListado.sh

./scriptListado.sh

Entrenamiento 5

* Tienes un directorio de proyecto ubicado en /home/tuusuario/proyecto. Dentro de este directorio, hay varios subdirectorios y archivos. Necesitas crear un script que realice las siguientes tareas:
  + Crear un respaldo de todos los archivos y subdirectorios de /home/tuusuario/proyecto en un nuevo directorio llamado respaldo\_proyecto ubicado en /home/tuusuario/respaldo.
  + Cambiar los permisos del directorio respaldo\_proyecto para que el propietario tenga todos los permisos, el grupo tenga permisos de lectura y ejecución, y otros tengan solo permisos de lectura.
  + Registrar la fecha y hora en que se realizó el respaldo en un archivo llamado log\_respaldo.txt dentro del directorio respaldo\_proyecto.
* Desarrollo paso a paso
  + Abrimos el editor vi y añadimos como primera línea #!/bin/bash y escribimos el resto de los comandos.
  + Como vamos a utilizar los nombres de directorios en varias ocasiones vamos a usar variables donde guardarlas.
  + Creamos el directorio de backup con mkdir en la ruta actual
  + Copiamos de forma recursiva ( cp -r) el contenido del directorio actual al de backup
  + Usando chmod cambiamos los permisos del directorio de respaldo
  + Usando la función $(date) recogemos la fecha y hora y componemos una frase cuya salida se guarda en el fichero de log
* Solución

#!/bin/bash

directorio\_proyecto="/home/tuusuario/proyecto"

directorio\_backup ="/home/tuusuario/respaldo/respaldo\_proyecto"

archivo\_log="$directorio\_respaldo/log\_respaldo.txt"

mkdir "$directorio\_backup"

cp -r "$directorio\_proyecto" "$directorio\_backup"

chmod -R 750 "$directorio\_backup"

echo "Fecha y hora del respaldo: $(date)" > "$archivo\_log"

Test

1. ¿Cuál de las siguientes es una característica principal de GNU/Linux?:

A. Es un software de pago.

B. Es un sistema operativo propietario

\_ C. Es un software libre y de código abierto

D. No permite personalización

GNU/Linux es un software libre y de código abierto en la que participa la comunidad de desarrolladores y que por tanto pueden modificar, compilar y distribuir el código.

1. ¿Cuál de las siguientes es una distribución de GNU/Linux?:

\_ A. Ubuntu.

B. Windows 10.

C. MacOS.

D. iOS.

Ubuntu es una distro de GNU/Linux, una de las más populares de este sistema operativo.

1. ¿Cuál es la característica principal de una distribución "rolling release"?

\_A. Actualizaciones continuas sin necesidad de reinstalar.

B. Foco exclusivo en servidores.

C. Compatibilidad únicamente con hardware antiguo.

D. Lanzamientos periódicos con versiones estables.

Las versiones "rolling release" se actualizan constantemente y no se requieren nuevas instalaciones para obtenerlas.

1. ¿Cuál es el directorio raíz en la jerarquía de directorios de GNU/Linux?:

A. /home

\_ B. /.

C. /root

D. /usr

En GNU/Linux, la jerarquía de directorios comienza en el directorio raíz, que se representa con "/".

1. ¿En qué directorio se encuentran los archivos ejecutables del sistema?:

A. /var.

B. /tmp.

C. /etc

\_ D. /bin.

En el directorio /bin es donde se encuentran los ejecutables necesarios para el sistema.

1. ¿Dónde se almacenan habitualmente los archivos de configuración del sistema en GNU/Linux?:

A. /usr.

\_B. /etc.

C. /lib.

D. /opt.

Los archivos de configuración del sistema y de las aplicaciones se guardan en el directorio /etc.

1. ¿ Cuál es el intérprete de comandos más comúnmente usado en GNU/Linux?

A. cmd.exe

B. PowerShell.

\_ C. Bash.

D. sh

Muchas distribuciones de GNU/Linux utilizan Bash (Bourne Again Shell) como intérprete de comandos predeterminado.

1. ¿Qué comando se utiliza para listar los archivos y directorios en GNU/Linux?

A. dir

\_B. ls.

C. list.

D. show

El comando ls muestra los archivos y directorios del directorio actual.

1. ¿Qué comando permite cambiar de directorio en una terminal de GNU/Linux?

\_A. cd

B. mv

C. cp

D. rm

El comando cd (change directory) se utiliza para cambiar el directorio actual. El uso de cd .. sube un nivel en el árbol de directorios.

1. ¿Qué comando se utiliza para copiar archivos en GNU/Linux?

A. copy

\_B. cp

C. mv

D. paste

El comando cp copia archivos y directorio, usando la opción -r se copia un directorio origen a un directorio destino de forma recursiva.

1. ¿Para qué sirve el comando 'chmod' en GNU/Linux?

\_A. Cambiar permisos de archivos y directorios

B. Mover archivos

C. Borrar archivos

D. Cambiar el nombre de archivos

Con el comando chmod y sus distintas notaciones (octal o simbólica) se pueden modificar los permisos de archivos y directorios.

1. ¿Qué comando se usa para ver el contenido de un archivo en la terminal?

A. read

B. see

C. open

\_D. cat

El comando cat permite mostrar el contenido del fichero en una terminal.

1. ¿Qué archivo contiene la configuración de red estática en Debian/Ubuntu?

\_A. /etc/network/interfaces

B. /etc/netplan/config

C. /etc/sysconfig/network

D. /etc/hosts

La configuración de la red estática en Debian y Ubuntu se encuentra en el archivo /etc/network/interfaces.

1. ¿Qué comando se utiliza para ver la configuración de red actual en GNU/Linux?

A. netstat

B. ipconfig

\_C. ifconfig

D. nslookup

Ifconfig muestra la configuración de la red. Usado sin parámetros muestra un informe de configuración de los interfaces de red.

1. ¿Cuál es la finalidad del comando 'ping'?

A. Configurar interfaces de red

B. Ver tablas de enrutamiento

\_C. Comprobar la conectividad de red

D. Mostrar la caché de DNS

El comando ping envía paquetes ICMP a través de la red que son solicitudes de eco hacia una dirección IP. Si la dirección IP está activa responderá a las solicitudes.

1. ¿Qué comando se usa para agregar un nuevo usuario en GNU/Linux?

A. adduser

B. newuser

C. createuser

\_D. useradd

Useradd crea un nuevo usuario en el sistema de usuarios de Linux. Permite opciones adicionales, como establecer el directorio principal

1. ¿Qué comando se usa para cambiar la contraseña de un usuario en GNU/Linux?

A. change

\_B. passwd

C. usermod

D. password

Modifica la contraseña del usuario. Este comando usado sin más opciones modifica la propia contraseña.

1. ¿Qué permiso de archivo permite ejecutar un archivo en GNU/Linux?

A. r

B. w

\_C. x

D. e

Los permisos en GNU /Linux son lectura (r), escritura (w) y ejecución (x)

1. ¿Qué representa el número '7' en los permisos de archivos de GNU/Linux?

A. Lectura y ejecución

B. Escritura y ejecución

\_C. Lectura, escritura y ejecución

D. Solo lectura

El número '7' en los permisos de archivo representa la combinación de permisos de lectura (r), escritura (w) y ejecución (x): 4 para lectura, 2 escritura y 1 para ejecución.

1. ¿Qué comando se utiliza para cambiar el propietario de un archivo en GNU/Linux?

A. chmod

\_B. chown

C. chgrp

D. chmod 777

Usaremos chown para cambiar el propietario y, opcionalmente, el grupo de un archivo o directorio.

1. ¿Cuál es la finalidad del directorio /dev en GNU/Linux?

A. Almacenar los archivos de configuración del sistema

\_B. Contener los archivos de dispositivos (device files)

C. Guardar los archivos de usuario

D. Almacenar los binarios esenciales del sistema

Los archivos de dispositivos, que son interfaces para interactuar con los dispositivos hardware del sistema, como discos duros, dispositivos de entrada/salida y terminales, se encuentran en el directorio /dev.

1. ¿Cuál es el objetivo de incluir #!/bin/bash al principio de un archivo en GNU/Linux?

A. Indicar que el archivo está en el directorio /bin/bash

\_B. Especificar que el archivo debe ejecutarse con el intérprete de comandos Bash

C. Convertir el archivo en un binario ejecutable

D. Habilitar la ejecución de comandos como root automáticamente

Incluir #!/bin/bash indica al sistema que el intérprete de comandos Bash, la shell de Unix muy común en sistemas GNU/Linux, debe interpretarlo y ejecutarlo.