¿Qué es la terminal?

- **GLOSARIO**
- Terminal: Es una interfaz gráfica que simula o ejecuta la línea de comandos o *Shell*.
- Línea de comandos (Shell): Es un software que lee diversos comandos y los interpreta para ejecutarlos dentro del S.O.
- Comando: Es un código que se ejecuta desde la terminal y que la Shell puede pasar al sistema operativo.

★ IDEAS PRINCIPALES

★ La terminal es sin duda más rápida y eficiente que realizar los procesos por medio de la interfaz gráfica.

❑ PREGUNTAS

La terminal nos facilita por medio de sus comandos la realización de procesos eficaces en la computadora.

De forma rápida se pueden hacer copias de seguridad, mover archivos y automatizar ciertos procesos.

La terminal es más rápida y confiable que una interfaz gráfica.

La consola de comandos nos permite interactuar de forma muy cercana con el S.O. por eso se debe manejar con precaución.

Existen distintas Shells como por ejemplo:

- BashShell
- Z Shell
- C Shell
- PowerShell

Estas 2 primeras son las más utilizadas.

El curso se desarrollará en las Bash Shell usadas por Linux; este S.O. puede encontrarse en despliegue de desarrollos y en servidores.

RESUMEN

La terminal es una interfaz que nos permite realizar algunos procesos de forma eficiente y fiable por medio de comandos que entiende la Shell y que entonces pasan al S.O. para ejecutarse como tal. La Shell más importante es la Bash Shell usada por los sistemas Linux.

Introducción a la Terminal y Línea de Comandos

Aprendiendo a caminar en la terminal

- **GLOSARIO**
- Rutas absolutas: Muestra la ubicación de un archivo desde la raíz.
- Rutas relativas: Muestran la ubicación de un archivo desde otro archivo.

★ IDEAS PRINCIPALES

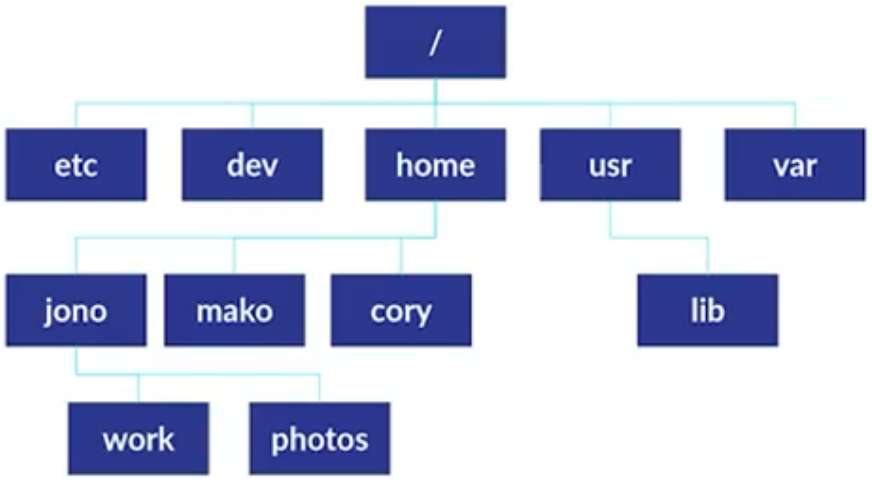
★ Usando las flechas (↑↓) del teclado se pueden recuperar los comandos anteriores.

★ Usando la tecla Tab y el inicio de un nombre de archivo, la terminal lo completa.

❑ PREGUNTAS

La línea de comandos ejecuta los comandos sobre archivos y rutas, por eso hay que tener en cuenta el sistema de archivos del S.O.

En el curso se usará el sistema de archivos de Linux:



La terminal muestra inicialmente el nombre de usuario y el nombre del equipo (o hostname) donde se ejecuta, así como la ruta sobre la cual corre.

Comandos útiles:

- *ls* - Lista los archivos y directorios que están en una carpeta.
  - *ls -l* - Lista los archivos con su fecha y peso en bytes.
  - *ls -lh* - Lista los archivos con su peso en megas, gigas y bytes.
- *cd <ruta o nombre de la carpeta>* - Nos mueve a otro directorio con su ruta, o con el nombre cuando la carpeta está en el directorio actual.
  - *cd* - Lleva a la carpeta “home”.
- *clear* (ctrl + L) - Limpia la terminal.
- *pwd* - Indica la ruta en la que corre la terminal.
- *file <nombre de archivo>* - Indica la codificación, la extensión y demás datos sobre un archivo.

Los comandos pueden tener opciones y argumentos, para esto se coloca un guión (-) seguido de la opción y del argumento.

En las rutas se usa el operador “.” para hacer referencia al directorio actual, y el operador “..” para referenciar al directorio padre.

RESUMEN

Los comandos en la Shell se ejecutan sobre el sistema de archivos de Linux; por eso es útil conocer los archivos en una carpeta (con *ls*), cambiar entre carpetas (con *cd*), ver las rutas (con *pwd*) y obtener la información sobre un archivo (con *file*).

Introducción a la Terminal y Línea de Comandos

Manipulando archivos y directorios

GLOSARIO

- Archivos ocultos: Son propios del sistema operativo y no se muestran de forma normal; tienen un . antes del nombre.

★ IDEAS PRINCIPALES

★ Los directorios no deberían nombrarse con comillas, en su defecto se nombraran encerrados en comillas.

❑ PREGUNTAS

Desde la terminal se pueden crear, eliminar, renombrar y mover archivos como carpetas de forma más rápida (y efectiva) que con la interfaz gráfica.

Comandos útiles:

- *ls*
  - *ls -la* - Sirve para poder visualizar todos los archivos incluyendo los ocultos.
  - *ls -lS* - Ordena los archivos de acuerdo a su peso.
  - *ls -lSh* - Ordena los archivos por su peso usando mega y kilobytes.
  - *ls -lr* - Lista los archivos al inverso de la clasificación normal.
  - *ls <directorio>* - Lista los archivos dentro de una carpeta.
  - *ls -d* - Es una bandera que solo busca los directorios.
- *tree* - Muestra los archivos debajo del directorio actual como un árbol.
  - *tree -L <niveles>* - Permite ver un número estricto de niveles en el árbol.
- *mkdir <directorio>* - Crea una carpeta dentro del directorio actual.
- *touch <archivo>* - Crea un archivo.
  - *touch <archivo.extensión>* - Crea un archivo con extensión específica.
- *cp <archivo> <ruta de la copia>/<nombre de la copia>* - Copia un archivo en una ruta en específica y le da un nombre.
- *mv <archivo> <ruta a mover>* - Mueve el archivo a una ruta dada.
  - *mv <archivo> <nuevo nombre>* - Renombra un archivo o directorio usando las propiedades del comando *mv*.
- *rm <archivo>* - Borra definitivamente un archivo.
  - *rm -i <archivo>* - Verifica si realmente se quiere borrar un archivo.
  - *rm -r <directorio>* - Borra un directorio usando la recursividad.
  - *rm -ri <directorio>* - Borra un directorio verificando archivo por archivo.

Se pueden crear varios archivos y directorios en un solo comando dejando un espacio entre sus nombres.

RESUMEN

La terminal es muy útil para ver directorios y archivos, con el comando *ls* y con el comando *tree*, así como para manipularlos ya sea creando archivos (*touch*) o carpetas (*mkdir*), moviendolos o renombrandolos (*mv*) y hasta copiandolos (*cp*); la terminal tiene la capacidad de borrar archivos y carpetas definitivamente (*rm*), así se debe tener cuidado.

Explorando el contenido de nuestros archivos

- **GLOSARIO**

★ IDEAS PRINCIPALES

❑ PREGUNTAS

❑ ¿Cómo matar un proceso en la terminal? Se utiliza la combinación de teclas Ctrl + C

La terminal nos permite visualizar los archivos de forma rápida y cómoda.

Es muy útil para trabajar con archivos, carpetas y directorios de una forma rápida y sencilla.

Comandos útiles:

- *head <archivo>* - Muestra las primeras 10 líneas de un archivo de texto.
  - *head <archivo> -n <# de líneas>* - Muestra el número de líneas indicado en un archivo de texto.
- *tail <archivo>* - Muestra las últimas 10 líneas de un archivo de texto.
  - *tail <archivo> -n <# de líneas>* - Muestra el número de líneas indicado en un archivo de texto.
- *less <archivo>* - Muestra un archivo de texto recorriendolo con ↑↓, y buscar frases dentro de él usando el slash /, se utiliza la q para salir.
- *xdg-open <archivo>* - Abre un archivo usando la aplicación predeterminada para su extensión (en Linux de forma nativa).

Al abrir un archivo con un editor externo se creará un proceso interno en la terminal que no permitirá escribir.

Comandos para entorno WSL:

- *wslview <archivo.extensión>* - Abre un archivo en el editor o app configurado para su extensión.
- *wslview <directorio>* - Abre el explorador de archivos de Windows dentro de la carpeta seleccionada.
- *explorer.exe .* - Ejecuta el explorador de archivos de Windows en la ubicación actual de la terminal.

RESUMEN

La terminal es muy útil para explorar y ver archivos de forma rápida y concisa sea viendo solo sus primeras líneas (con *head*), sus últimas líneas (con *tail*) o todo el archivo pudiendo recorrerlo y realizar búsquedas (con *less*). También en la terminal se pueden abrir archivos desde app's y hasta el explorador la archivos dependiendo del S.O.

¿Qué es un comando?

- **GLOSARIO**

★ IDEAS PRINCIPALES

★ Los alias se eliminan al cerrar y abrir la terminal, para guardarlos se deben configurar las variables de entorno.

❑ PREGUNTAS

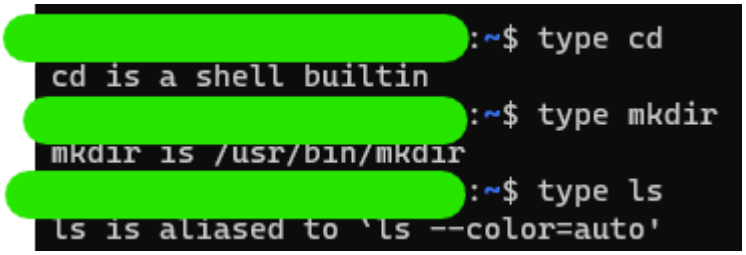
❑ ¿Dónde se guardan los comandos de función? En la carpeta .bin del S.O.

❑ ¿Funciona el comando *help* en todas las terminales? *help* pertenece a la Bash Shell, por lo que puede no servir en otra terminal como Z Shell.

Los comandos pueden ser básicamente 4 cosas:

1. Un programa ejecutable: Es un código compilado en algún lenguaje de programación que se puede ejecutar de forma autónoma.
2. Un comando de utilidad de la Shell: Es un comando que la Terminal trae por sí misma y que permite realizar ciertas acciones especiales.
3. Una función de la Shell: Son comandos instalados como archivos binarios para funcionar en la Shell que le dan de mayor funcionalidad.
4. Un alias: Otro nombre o forma de llamar a un comando específico.

La instrucción *type* permite categorizar un comando en una de estas 4 opciones.



Para crear un alias para un comando se crea usa el comando:

- *alias <nombre del alias>="<comando>"*

Otros comandos útiles son:

- *help <comando>* - Muestra información de ayuda técnica sobre un comando.
- *<comando> --help* - Funciona igual que el comando help.
- *man <comando>* - Muestra la sinopsis, descripción y opciones internas de un comando, para salir del menú se usa q.
- *info <comando>* - Permite ver de forma resumida la información sobre un comando.
- *whatis <comando>* - Brinda una descripción muy corta del comando, sin embargo no funciona para todos los comandos.

RESUMEN

Los comandos se clasifican en 4 tipos que son programas ejecutables, comandos de la Shell, funcionalidades externas y alias; los comandos tienen distintas formas de brindar información sobre su uso como con los comandos *man*, *help*, *info* y la instrucción *whatis*.

Wildcards

- **GLOSARIO**

★ IDEAS PRINCIPALES

❑ PREGUNTAS

❑ ¿Existe alguna guía para la terminal? Si, Platzi dispone de un archivo guía que nos ayuda a comprender los comandos y sus acciones.

Las Wildcards son una herramienta muy útil para realizar búsquedas de archivos y directorios de forma avanzada.

Las Wildcards nos permiten filtrar la información de archivos y directorios usando ciertos caracteres especiales de una forma concreta.

Así pueden aplicarse a cualquier comando que realice manipulación de archivos como *mv*, *cp* y *rm*; pero principalmente se usan con *ls*.

Wildcard:

- **.<extensión>* - Sirve para buscar archivos por una extensión.
- *<nombre>** - Sirve para buscar archivos que inicien por un mismo nombre.
- <nombre>? - Buscar archivos que inicien por un nombre pero que solo tengan un carácter después.
- *[<caracteres>]* - Permite hacer una búsqueda según una lista de caracteres.
- *[!<caracteres>]* - Permite hacer una búsqueda que excluya una lista de caracteres.
- *[[:clase:]]* - Permite hacer una búsqueda según una clase de caracteres.
  - *[:alnum:]* - Busca coincidencias con cualquier carácter alfanumérico.
  - *[:alpha:]* - Busca coincidencias con cualquier letra del alfabeto.
  - *[:digit:]* - Busca coincidencias con cualquier número.
  - *[:upper:]* - Busca coincidencias con mayúsculas.
  - *[:lower:]* - Busca coincidencias con minúsculas.

En realidad el * representa cualquier texto, frase o nombre; mientras ? representa cualquier carácter.

Además las Wildcards buscan dentro de las carpetas y directorios por lo menos en 2 niveles.

Archivo Guía de la Terminal

RESUMEN

Las Wildcard hacen uso de caracteres especiales y de notaciones específicas para permitir una búsqueda avanzada de archivos y caracteres usando en * para representar cualquier cadena y el *?* como cualquier carácter, pudiéndose crear agrupaciones con *[<grupo>]* y clases con *[[:<clase>:]]*.

Introducción a la Terminal y Línea de Comandos

Redirecciones: ¿Cómo funciona la shell?

- **GLOSARIO**
- File descriptor: Es el número que representa el proceso sea entrada (0), salida (1) o error (2).

★ IDEAS PRINCIPALES

❑ PREGUNTAS

Los comandos funcionan como una entrada en la consola que luego es procesada por el sistema operativo y produce una salida o un error.

El proceso de entrada se conoce como Standar Input (stdin - 0); el de salida es el Standar Output (stout - 1) y el de error es Standar Error (stderr - 2).

Dentro de la shell la información producida por un comando puede redirigirse, por ejemplo, a un archivo de texto, para esto se usa el signo **>**.

```
ls > archivos.txt
```

Esta redirección siempre sobreescribe el archivo sin guardar la información; si lo que queremos es concatenar la información, se usa el operador **>>**.

Sin embargo el operador **>** por sí solo no permite redirigir errores, para esto se debe colar el file descriptor del error, es decir: ***2>***.

Si se quieren redirigir tanto el standar output como el standar error se coloca al final del comando el operador ***2>&1***.

```
ls VsCodo >> archivos.txt 2>&1
ls VsCode >> archivos.txt 2>&1
```

Esto redirige la salida error (2) a una salida output (1) y lo procesa normalmente.

Estas redirecciones son útiles para guardar errores y salidas producidos por scripts que se ejecuten en una terminal, por ejemplo dentro de un server.

Por otro lado, la redirección del input de un comando se realiza con el operador **<**.

Es decir, con este operador se pasa desde un archivo el input de algún comando en específico que requiere de una entrada desde teclado.

Este operador generalmente no se puede usar sin una posterior redirección del input.

RESUMEN

Dentro de la consola cada comando tiene un input identificado con el 0, un output identificado con el 1 y un posible error identificado con el 2. Dentro de los comandos se pueden redireccionar las salidas o / y los errores a un archivo de texto; de la misma manera se puede redireccionar desde un archivo las entradas para un comando.

Redirecciones: Pipe operator

- GLOSARIO

★ IDEAS PRINCIPALES

❑ PREGUNTAS

El pipe operator nos permite, entre otras cosas, encadenar distintos comandos por medio de stdin y stdout.

Un comando útil es *echo <mensaje>* que nos permite imprimir como output un mensaje en consola.

Otro comando interesante es *cat <archivo 1> <archivo 2>* que nos permite obtener como output la suma del primer archivo con el segundo.

El operador **<** sirve para redirigir el input de un comando, internamente muchas instrucciones hace uso de este operador para obtener información de un archivo.

El pipe operator **|** nos permite convertir el stdout de un comando en el stdin de otro, encadenando la funcionalidad de distintos comandos.

Por ejemplo concatenar archivos y enviarlos a un menú cómo less:

```
cat errores.txt archivos.txt | less
```

Así, no existe por ejemplo la necesidad de crear un archivo.

Otra instrucción interesante es *tee <nombre de archivo>* que permite crear un archivo en base a la información que viene del pipe operator.

Dentro del pipe operator se pueden realizar distintos filtros:

- *sort <líneas de un archivo>* - Ordena alfabéticamente las líneas de un archivo.

Se debe tener cuidado con la redirección del pipe operator, pues en ocasiones el stdout de un comando no es adecuado como stdin del otro.

RESUMEN

El pipe operator | permite encadenar comandos al pasar las stdout de un comando como stdin de otra instrucción; algunos comandos para ejecutar con el pipe operator son sort que filtra las líneas de los archivos de texto y el comando tee que crea un archivo.

Encadenando comandos: Operadores de control

Los operadores de control, al igual que el pipe operator, permiten relacionar 2 comandos por medio de ciertos caracteres especiales.

Es muy útil para por ejemplo correr 2 instrucciones al tiempo.

Para ejecutar comandos de forma secuencial o síncrona se usa un punto y coma.

Por otro lado, se pueden ejecutar comandos asincrónicamente, es decir, en terminales en segundo plano.

Para esto se usa el operador & que permite crear una terminal secundaria.

Así mismo se puede ejecutar comandos de forma condicional; si se ejecuta el primer comando entonces se ejecutara el segundo.

Para eso se hace uso del operador &&.

En oposición al operador && esta el operador || que permite correr un comando o el otro pero no ambos.

RESUMEN

Existen otros operadores para relacionar y ejecutar comandos; el operador ; ejecuta un comando tras otro; por otro lado el operador & ejecuta asíncronamente y en segundo plano los comandos; el operador && procesa un segundo comando solo si el primero fue exitoso mientras el operador || opera el segundo comando solo si el primero no funciona.

¿Cómo se manejan los permisos?

Cada archivo tienen ciertos permisos y ciertas propiedades; estas se pueden ver usando *ls -l*.

El primer carácter simboliza que tipo de dato es:

- - : Es un archivo normal
- d : Es un directorio
- l : Es un link simbólico
- b : Archivos de información de un bloque de datos.

Posterior al carácter correspondiente al tipo de dato están los caracteres correspondientes a los permisos.

Existen 3 usuarios distintos con 3 permisos distintos, estos usuarios son:

1. Admin: Es el superusuario, obtiene permisos con la “u”.
2. Team: Es el grupo de usuarios cercanos al admin y con algún tipo de gestión sobre los archivos, obtiene permisos con la “g”.
3. Público: Es el público general, obtiene permisos con la “o”.

Si se quieren asignar permisos a todos se representa con la “a”.

Cada uno de estos usuarios puede o no tener 3 principios:

1. Leer (r): Hace referencia a la lectura y revisión del archivo.
2. Escribir (w): Hace referencia a la modificación del archivo.
3. Ejecutar (x): Hace referencia a una ejecución.

Por lo general cada archivo tiene los siguientes permisos:

Admin	Team	Público
rwx	r-x	r-x
111	101	101
7	5	5

Siendo 1 si el permiso está activo y 0 si no; y su traducción a octal el cambio de base de los bits a decimal.

RESUMEN

En Linux existen 3 tipos de usuarios: Admin(u), Team(g) y Public(o); con 3 permisos posibles: Read (r), Write(w) y Execute(x). Cada archivo representa estos permisos con 9 letras, 1 por permiso, 3 por usuario, aunque también se puede representar con 3 números en la representación octal.

Introducción a la Terminal y Línea de Comandos

Modificando permisos en la terminal

- **GLOSARIO**

★ **IDEAS PRINCIPALES**

❑ **PREGUNTAS**

❑ ¿Cómo podemos editar texto desde la terminal? Hacemos uso del comando *cat > <nombre del archivo>*; esto nos permitirá editar la salida del cat, es decir, el archivo internamente. De este editor se puede salir con Ctrl + D.

❑ ¿Que puedo hacer si no funciona *su root*? Se puede emplear el comando *sudo su root* que permitirá cambiar de user.

Desde la terminal se pueden cambiar los permisos que tienen ciertos usuarios, para esto es importante el usuario root.

Este usuario tiene muchos permisos sobre la terminal.

Para cambiar los permisos de un archivo se usa el siguiente comando:

- *chmod <permisos admin octal><permisos Team octal><permisos público octal> <archivo>* - Asigna los permisos de los 3 usuarios sobre un archivo por medio del código octal.

Sin embargo, el chmod puede usarse con el modo simbólico:

- *chmod <símbolo del tipo de usuario><+,- o =><letra del permiso> <archivo>* - Sirve para asignar (+), retirar (-) o sobreescribir (=) los permisos de un grupo específico.

Otros comandos relacionados con los usuarios son:

- *whoami* - Imprime en consola el usuario que se está usando.
- *id* - Imprime en consola distintos datos sobre el usuario como los grupos a los que pertenece.
- *su <usuario>* - Permite cambiar de usuario.
  - *su root* - Permite usar el usuario root.
- *sudo <comando>* - Permite realizar comandos con los permisos de root.
- *passwd* - Permite cambiar la contraseña de un usuario.

El home del usuario root habita directamente en la raíz del sistema (/root) y no en el home como cualquier otro usuario (/home/miusuario).

Un usuario normal no puede borrar archivos creados por el root, para esto se usa el comando sudo.

RESUMEN

Es importante poder cambiar los permisos de un archivo usando *chmod* usa el sistema octal para las acciones así como los símbolos para los tipos de users; así mismo, conocer el usuario root que tiene permisos especiales, como cambiar de usuario con el comando *su* y cambiar la contraseña con la instrucción *passwd*.

Introducción a la Terminal y Línea de Comandos

¿Cómo configurar variables de entorno?

- **GLOSARIO**
- Variables de entorno: Es la configuración de la terminal y del sistema, aquí se asignan ciertos valores y alías que se ejecutan de forma autónoma.

★ **IDEAS PRINCIPALES**

★ Las variables de entorno son una herramienta poderosa para configurar la terminal y su funcionamiento.

★ Para acceder al valor de una variable de entorno se antecede el signo *$* al nombre de la misma.

❑ **PREGUNTAS**

Saber manejar los valores de las variables de entorno es muy útil pues permiten asegurar y configurar el sistema y la terminal.

También es conveniente poder crear links simbólicos que enlacen hacia archivos y directorios, esto se realiza usando el comando:

- *ln -s <ruta del directorio> <nombre del link>* - Asigna un nombre a una ruta, de forma que es más fácil acceder a esta dirección.

Estos links permiten abrir carpetas y modificar archivos de forma más rápida.

Entendiendo las variables de entorno se puede usar el comando *printenv* para listar las variables de entorno.

Para imprimir una variable de entorno en específico se usa:

- *echo $<nombre de la variable de entorno>*

Algunas variables de entorno importantes son:

- HOME - Es la ruta del home de la terminal.
- PATH - Contiene la ruta de todos los binarios que ejecuta el sistema.

En ocasiones al descargar un binario desde un gestor de paquetes este no se asigna en el PATH, por eso es importante saber modificar estas variables.

Para esto se accede a la dirección HOME y allí se busca como archivo oculto *.bashrc* (o *.zshrc* si se corre en este tipo de terminal).

Dentro de este archivo se pueden crear alias:

alias <alias del comando>='<comando como tal>'

(Es importante no dejar espacios)

También se pueden crear variables de entorno, estas se escriben con mayúsculas y se les asigna un valor entre comillas simples (').

PLATZI_MENSSAGE='Hola compañeros de Platzi'

Para modificar una variable de entorno como PATH agregando contenido se usan los dos puntos, tal que:

PATH=$PATH:<ruta de binario>

RESUMEN

Las variables de entorno son una herramienta muy importante para configurar y agilizar nuestro desarrollo en la terminal; estas variables se acceden con el comando *printenv* y se modifican desde el archivo *.bashrc* asignadolas sin dejar espacios; o modificandolas llamando al mismo valor con el signo *:*.

Comandos de búsqueda

- **GLOSARIO**
- Archivos .log: Es un archivo txt que guarda la información de funcionamiento de una aplicación

---

★ IDEAS PRINCIPALES

❑ PREGUNTAS

Los comandos de búsqueda permiten rastrear archivos y carpetas a partir de sus distintas características como nombre, extensión y su ubicación.

Algunos de los comandos de búsqueda útiles son:

- *which <comando>* - Busca un comando de tipo binario dentro de las distintas rutas de PATH.
- *find <ruta> -name <nombre>* - Permite buscar archivos por su nombre partiendo desde una ruta.
  - *find <ruta> -type f* - Realiza búsquedas de solo archivos.
  - *find <ruta> -type s* - Realiza búsquedas de solo directorios.
  - *find <ruta> -size <Peso en unidades de información>* - Busca archivos con un peso mayor al indicado.
  - *find <ruta> -maxdepth <Niveles>* - Busca archivos teniendo en cuenta ciertos niveles de profundidad.
  - *find <ruta> -empty* - Busca archivos y directorios vacíos.
  - *find <ruta> -perm <Permisos en sistema octal>* - Busca recursos de acuerdo a sus permisos escritos en octal.
  - *find <ruta> -delete* - Borra los resultados de la búsqueda y se ubica al final del comando, **se debe usar con precaución**.

El comando *find* puede hacer uso de wildcards para dinamizar la búsqueda de archivos.

RESUMEN

Existen comandos que facilitan la búsqueda de archivos, directorios y binarios; por ejemplo el comando *which* localiza la ubicación del binario de un comando; mientras que la instrucción *find* ubica los recursos en función de nombre y tamaño pudiendo usarse Wildcards.

Su majestad: grep

- **GLOSARIO**
- Expresiones regulares: Son una herramienta para examinar cadenas de texto de forma compleja y rica en detalles.

★ IDEAS PRINCIPALES

❑ PREGUNTAS

El comando grep es supremamente útil en el mundo de Linux.

Esta instrucción nos permite hallar coincidencias dentro de un archivo de texto incluyendo Standar Outputs.

Este comando utiliza el formato:

- *grep <Coincidencia> <Archivo>* - Imprime las líneas del archivo donde se encuentre la coincidencia buscada.
  - *grep <Expresión regular> <Archivo>* - Realiza búsquedas avanzadas dentro de un archivo usando las expresiones regulares.
  - *grep -i <Coincidencia> <Archivo>* - Encuentra coincidencias ignorando las mayúsculas y minúsculas.
  - *grep -c <Coincidencia> <Archivo>* - Cuenta el número de líneas encontradas con la coincidencia.
  - *grep -v <Coincidencia> <Archivo>* - Coincide con las líneas que no tienen el término buscado.

Otro comando útil es la instrucción *wc:*

- *wc <archivo>* - Cuenta las líneas, las palabras y los caracteres dentro de un archivo, así como los bits que ocupa.
  - *wc -l <archivo>* - Cuenta exclusivamente las líneas en el archivo.
  - *wc -w <archivo>* - Imprime el número de palabras del archivo.
  - *wc -c <archivo>* - Imprime el número de caracteres o bytes del archivo.

Líneas	Palabras	# de bytes

```
  9126  30006 477779 movies.csv
```

El comando grep nos ayuda a filtrar nuestro código y las instrucciones en él.

RESUMEN

El comando *grep* es muy útil para examinar archivos de forma rápida y completa, ayuda buscando coincidencias línea por línea usando una búsqueda literal o una expresión regular; también es interesante la instrucción *wc* que permite contar líneas, palabras y caracteres entre otros datos.

Utilidades de red

- GLOSARIO

★ IDEAS PRINCIPALES

❑ PREGUNTAS

La terminal también tiene la capacidad de darnos información sobre la red.

Algunos comandos que son útiles en este sentido son:

- *ifconfig* - Muestra diversos datos sobre la red y la conexión como dirección ip, la máscara de la red, la ip privada, etc.
- *ping <Link de una página>* - Verifica si una determinada página está activa y transmite información.
- *curl <Link de una página>* - Carga el documento html de la página indicada.
- *wget <Link de una página>* - Descarga de forma directa un recurso desde la página indicada.
- *traceroute <Link de una página>* - Muestra un registro de todos los puntos de acceso que recorre la conexión a una página.
- *netstat -i* - Muestra en una pequeña tabla la información de los dispositivos de red locales.

RESUMEN

Desde la terminal puede accederse a distintos datos sobre la red y la conexión a ciertas páginas web en específico; por ejemplo, con *ipconfig* y *netstat* se pueden revisar los dispositivos de conexión local, con *curl* y con *wget* acceder a archivos de un sitio web; finalmente *ping* y *traceroute* para analizar la conexión a páginas web en específico.

Comprimiendo archivos

- **GLOSARIO**
- Archivos .tar: Son usados para comprimir archivos e información dentro de repositorios.
- Archivos .gz: Hacen uso de un algoritmo de compresión muy eficiente que les permite ahorrar mucha memoria.

★ IDEAS PRINCIPALES

❑ PREGUNTAS

Desde la terminal se pueden crear archivos con extensión .zip y .tar como archivos comprimidos.

Para crear archivos .tar se utiliza el comando:

tar -cvf <Nombre del archivo comprimido>.tar <Documento o directorio>

Este comando usa las banderas *c* para crear un archivo, *v* para obtener una salida en la terminal y *f* para referirse a un archivo.

Con otra bandera en este comando pueden comprimirse archivos en formato gz:

tar -cvzf <Nombre del archivo comprimido>.tar.gz <Documento o directorio>

Para descomprimir este tipo de archivos desde la terminal se sigue un proceso muy similar.

tar -xzvf <Nombre del archivo comprimido>.tar.gz

Para conocer el índice de archivos comprimidos en un tar se usa la instrucción:

tar -tvf <Nombre del archivo comprimido>.tar

Por otra parte, se pueden crear archivos en formato zip usando un comando similar:

zip -r <Nombre del archivo comprimido>.zip <Directorio>

Para descomprimir en cambio se usa un comando distinto llamado unzip:

unzip <Nombre del archivo comprimido>.zip

También pueden realizarse compresiones en formato rar de forma similar al formato zip.

Esto es muy útil junto con técnicas de filtrado de archivos para lograr extraer información de forma rápida y sencilla.

RESUMEN

Desde la terminal pueden comprimirse archivos en distintos formatos, por ejemplo el comando *tar* permite comprimir y descomprimir archivos en formato tar y en formato gz; por otro lado la instrucción *zip* comprime carpetas en zip y *unzip* las descomprime.

Manejo de procesos

- **GLOSARIO**

★ IDEAS PRINCIPALES

★ La terminal es más efectiva y poderosa que los administradores de tareas.

❑ PREGUNTAS

❑ ¿Cómo abrir el administrador de tareas en Windows? Se utiliza la combinación de teclas Ctrl + alt + supr

❑ ¿Desde el WSL se ven los procesos de Windows? No, el sistema operativo no lo permite.

❑ ¿Cómo puedo ver y matar procesos en el CMD de Windows? Se usan los comandos tasklist y taskkill

Desde la terminal se pueden visualizar los procesos, manejarlos, filtrarlos y de ser necesario matarlos.

La instrucción *ps* muestra los comandos y procesos que corren en la terminal actualmente.

Los procesos tienen un Id conocido como PID.

El comando *ps* permite identificar aquellas instrucciones que la terminal sigue corriendo de manera asíncrona o en el background.

La instrucción *kill <PID del proceso>* ayuda a matar procesos que no corran directamente en la terminal.

Otra forma de acceder a los procesos es por medio del comando *top* que muestra todos los procesos que se están ejecutando.

top crea toda una interfaz con la que se puede interactuar a través del teclado. Con la tecla *h* se puede acceder al menú de ayuda.

Por otro lado, con la tecla *U* se pueden filtrar los procesos por usuarios.

Una forma de matar procesos por su nombre y no por su PID es con el comando *pkill <Nombre del proceso>*

Alternativamente con la instrucción *killall <Nombre de los procesos>* se mataran todos los procesos que concuerden con el nombre.

Adicional al *top* existen otros manejadores de procesos más avanzados instalables desde el gestor de paquetes, por ejemplo *htop*, *bpytop* y *glanses*.

RESUMEN

Desde la consola de Linux pueden visualizarse y matarse procesos; por ejemplo, el comando *ps* lista los procesos que corren sobre la terminal, la instrucción *kill* permite matar procesos a partir de PID mientras *pkill* los mata a partir de su nombre, finalmente *top* permite crear una interfaz donde filtrar y analizar los procesos.

Procesos en foreground y background

- **GLOSARIO**

★ **IDEAS PRINCIPALES**

❑ **PREGUNTAS**

❑ ¿Cómo matar un proceso que se corre directamente en la shell? Se usa la combinación de teclas Ctrl + C

❑ ¿Cúal es la diferencia entre Ctrl + C y Ctrl + D? Mientras Ctrl + C mata el proceso, Ctrl + D sólo inhabilita la entrada de texto desde la consola.

❑ ¿Cómo suspendemos un proceso sin matarlo? Se usa la combinación de letras Ctrl + Z, esto coloca el comando en background.

En el sistema se pueden estar corriendo comandos y procesos de forma asíncrona que pueden quedar activos si no se finalizan.

A este comportamiento de seguir corriendo pero no mostrarse se le conoce como “proceso en background”.

En cambio si el comando se muestra y termina satisfactoriamente se dice que es un “proceso en foreground”.

Con el comando *jobs* podemos listar los procesos que se ejecutan en el background.

```
[1]+  Stopped                 cat
```

Con el número del trabajo (que aparece a la izquierda en el comando *jobs*) podemos pasar el comando como foreground.

Para esto se usa el comando *fg <Num. de trabajo>*

Si bien Ctrl + C mata el proceso, otra forma de mandar un comando al background es colocando el operador asincrónico (&) al final del mismo.

Con el comando *bg <Num. de trabajo>* podemos traer un proceso suspendido y ejecutarlo de forma que no afecte la ejecución de la terminal.

RESUMEN

Algunos procesos están activos pero no se muestran, esto se conoce como background, los procesos en background se pueden ver con el comando *jobs*, para pasar un proceso a background se puede usar Ctrl + Z, el comando *bg* o el operador *&*; por otro lado, para que un proceso se muestre (foreground) se usa la instrucción *fg*.

Editores de texto en la terminal

- GLOSARIO

★ IDEAS PRINCIPALES

❑ PREGUNTAS

Dentro de la terminal se pueden correr distintos editores de texto.

El más popular y usado es Vim por sencillez y facilidad, aunque existen otros como Emacs y Nano.

Vim procede de un editor de texto conocido como Vi.

Para abrirlo se ejecuta el comando *vim*, y para salir se colocan : y q.

Colocando un nombre al comando, se puede crear un archivo en Vim, así:

vim <Nombre del archivo>

Dentro de Vim existen 2 modos, el llamado Normal y el de Inserción. Con el modo de Inserción se puede escribir normalmente en el archivo.

Estando en el modo Normal se puede pasar al modo Inserción con la tecla i.

Para pasar del modo Inserción al modo Normal se usa la tecla esc.

Vim cuenta con un resaltador de sintaxis en función del lenguaje o tipo del archivo.

Dentro del modo Inserción con un slash (*/*) se puede buscar coincidencia por coincidencia de un término.

Al pulsar 2 veces la tecla d se borrara una línea completa de texto.

Usando :w se puede guardar el texto de un archivo.

RESUMEN

Dentro de la terminal pueden editarse archivos de texto con Vim, este editor cuenta con un modo Normal donde ejecutar las características del editor, y un modo de Inserción donde escribir; para salir se usa *:q*, para guardar *:w* y 2 veces *d* para borrar una línea. Por otro lado, *esc* permite cambiar de Inserción a Normal, mientras *i* permite lo opuesto.

Introducción a la Terminal y Línea de Comandos

Personalizar la terminal de comandos

- **GLOSARIO**

★ **IDEAS PRINCIPALES**

❑ **PREGUNTAS**

❑ ¿Si uso la nueva Windows Terminal debo instalar Tilix? Al usar la nueva terminal, esta se puede personalizar de forma nativa.

❑ ¿Cómo instalar Z Shell en WSL? Se usa el comando *sudo apt install zsh*

Es importante personalizar la terminal para así apropiarnos de ella y sentirla como un espacio natural para nuestro trabajo.

Personalizar la terminal también nos permitirá ser más eficientes y efectivos partiendo de la ayuda visual.

Para personalizar la terminal de forma fácil y sencilla es útil usar un emulador de terminal, en el curso se parte de Tilix pero existen otros.

Tilix se descarga desde el manejador de paquetes del respectivo OS.

Una vez descargado Tilix, se debe instalar una terminal distinta a Bash la cual es Z Shell.

Si se instala bien se puede ejecutar el comando:

zsh --version

Entonces se cambia de la Bash a ZSH con la orden:

chsh -s $(which zsh)

Con *$(<Comando>)* se puede ejecutar un mando dentro de otro comando.

Ya habiendo cambiado de línea de comandos, se debe reiniciar la terminal y configurar el Zshell con la opción *0*.

Dentro de Zsh se instala la dependencia Oh My Zsh usando el script:

sh -c "$(wget https://raw.github.com/ohmyzsh/ohmyzsh/master/tools/install.sh -O -)"

Si bien esta dependencia ayudará mucho a estilar y configurar la terminal, se debe agregar un tema a la terminal como powerLevel10K.

Al asignar este tema se descarga por un momento un repositorio y se asigna en la dirección del tema de Oh My Zsh.

En el archivo *~/.zshrc* se configura con Vim el tema de ZSH cambiando la variable como: *ZSH_THEME="powerlevel10k/powerlevel10k"*.

Entonces se descargan las fuentes recomendadas por el tema, se abren e instalan y se configuran en la terminal como fuente.

De nuevo se reinicia la terminal y se configuran como tal los estilos de la terminal.

RESUMEN

Para personalizar nuestra terminal debemos primero cambiar de Bash a ZSH, luego se instala la dependencia Oh My ZSH fundamental para estilar la terminal y a está se le asigna el estilo de powerLevel10K, dentro de este estilo se incluyen algunas fuentes recomendadas para instalar y configurar en la terminal. Una vez configurado OhMyZSH, powerLevel10K y las fuentes, la terminal se reinicia y se personaliza como tal.

Nunca pares de hackear

- GLOSARIO

★ IDEAS PRINCIPALES

❑ PREGUNTAS

Los hackers siempre encuentran formas distintas de hacer las cosas, de mejorar los procesos y de sacar provecho de la eficiencia.

La terminal permite facilitar y sobre todo agilizar procesos por eso, aprender a usarla implica ser un completo hacker.

Para ahondar más en los temas relacionados con la terminal, son recomendables algunos libros:

- Linux Basic for Hackers: Enfocado en la ciberseguridad, contiene muchos comandos y tips para agilizar los procesos.
- The Linux Command Line: Es la más excelente guía de la terminal de comandos.
- grep Pocket Reference: Es un libro dedicado enteramente al comando grep y a su uso avanzado.
- Regular Expression Pocket Reference: Está especializado como una referencia para construir expresiones regulares para la búsqueda.
- Linux Pocket Guide: Un resumen de 80 páginas sobre los fundamentos más básicos de Linux.
- Learning the vi and Vim Editors: Cubre todas las dudas, bases y utilidades de los editores de texto Vi y Vim.

RESUMEN

Usar la terminal realmente es hackear por que nos permite optimizar los procesos y sacar de ellos el máximo provecho; por eso es interesante seguir ampliando nuestro conocimiento con libros sobre la terminal, sobre sus comandos y sobre Linux.