```
<h1>Introducción a la Terminal y Línea de Comandos</h1>
<img src="readme_img/terminal.png" width="">
```

Tabla de contenido

Curso de Introducción a la Terminal y Línea de Comandos

Primeros pasos

¿Qué es la terminal?

La terminal es una herramienta indispensable que cualquier persona en la tecnología debe conocer . Es importante porque:

- Te da **flexibilidad** . Con unos pocos comandos, puedes hacer mucho.
- Es mucho más **veloz** que una interfaz .

- Es tu única opción si no hay interfaz , como para **configurar un servidor** remoto.
- Puedes invocar **demonios** . Hay que tener cuidado con los comandos.

Específicamente, la terminal es una interfaz gráfica muy sencilla que simula una línea de comandos:

- **Terminal**: Ventana que muestra el prompt. **Prompt** es el carácter o conjunto de caracteres que se muestran en una línea de comandos para indicar que está a la espera de órdenes. Este puede variar dependiendo del intérprete de comandos y suele ser configurable.
- **Shell (línea de comandos)**: Programa que ejecuta los comandos. Hay varios tipos de shell, pero sirven para lo mismo. La más comunes son **bash shell** o **Z shell**. En este curso, usaremos la primera .

Para conocer que shell tenemos:

- `$ echo $SHELL`

  Un comando es un programa que se puede ejecutar desde la terminal.

Aprendiendo a caminar en la terminal

Debemos entender como esta compuesto el sistema de archivos de linux, en la siguiente imagen podemos ver un arbol de la composición:

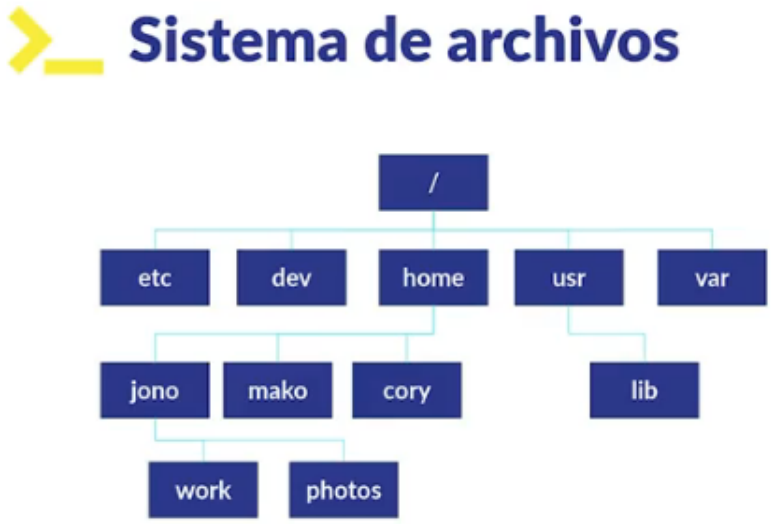

Figure 1: sistema-de-archivos-linux

En el siguiente gráfico tenemos un detalle mas amplio de lo que tenemos en el sistema de archivos de Linux.:

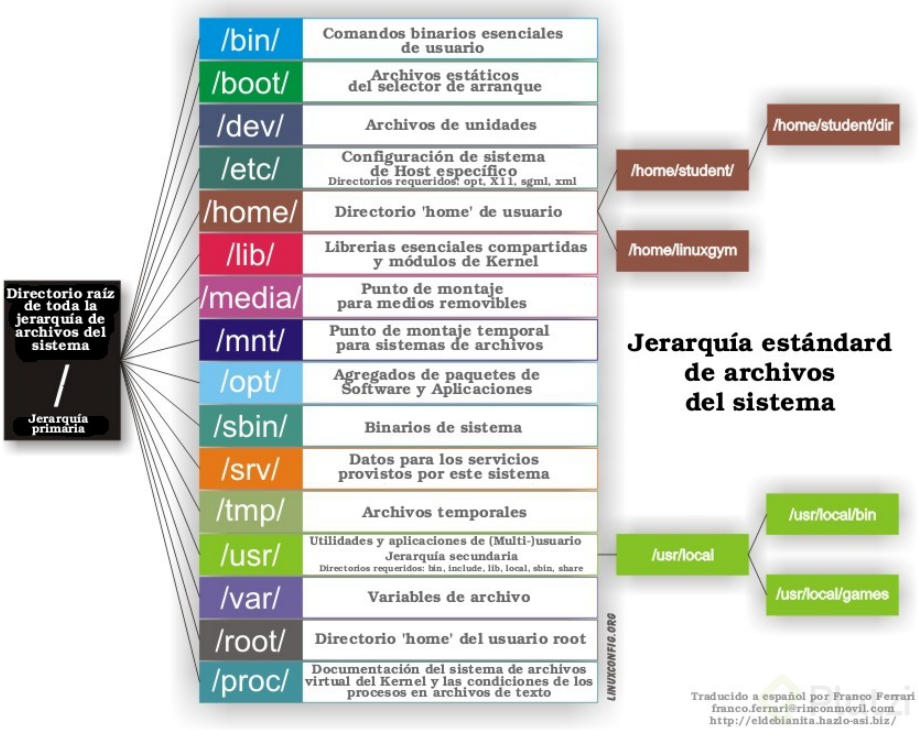

Figure 2: sistema-de-archivos-linux-detalles

Comandos Basicos de la terminal

- `ls`: Listar archivos
- `ls -lh`: Listar archivos para ver su peso de una manera mas legible
- `ls -a`: Listar archivos ocultos
- `pwd`: Identificar la ruta en la que estamos en nuestro sistema
- `cd`: Movernos entre directorios
- `mkdir`: Crear un directorio
- `cp`: Copiar un archivo
- `rm`: Borrar un archiv
- `mv`: Mover un archivo
- `rmdir`: Borrar un directorio:
- `clear` o `Ctrl + L`: Limpiar la terminal

Manipulando archivos y directorios

Manipular archivos

- `ls`: Algunas variantes:
  - `ls -la`: Nos muestra todo, incluso los ocultos.
  - `ls -lS`: Los ordena por tamaño.
  - `ls -lr`: Los muestra en reversa.
- `tree`: Nos muestra todo, lo que hay dentro de cada carpeta, en forma de arbol . Podemos elegir el nivel de profundidad con `tree -L <numero de niveles>`
- `mkdir <nombre_directorio>`: crea un directorio también puedes añadir la ruta
- `touch <nombre_archivo>`: crea un archivo, si no indicas la extension por defecto sera .txt
- `cp <archivo> <ruta>`: nos permite duplicar archivos, para duplicar directorios con su contenido añadir el modificador `-r` que indica recursividad
- `mv <archivo> <ruta>`: mover un archivo hacia un directorio. También se usa para renombrar archivos `mv <archivo> <nuevo nombre>` no olvidar la extension del archivo. Para mover directorios añadir el modificador `-r`.
- `rm <archivo>`: Elimina archivos. Hay una opción muy útil que es `rm -i <archivo>(interactivo)` que te pregunta explícitamente si en verdad si quiere borrarlo . Funciona de manera directa para directorios vacíos, pero si hay archivos dentro, debes usar la opción recursiva `rm -r <directorio>`, que lo que hace es ir borrando todo lo que hay dentro y al final borra el directorio . Te va preguntando por cada cosa.

  Ojo: Lo que borras desde la terminal se borra para siempre .

Explicacion detallada de la importancia de `rm` ¿Y para qué me sirve correrlo como interactivo?

Si te preguntaste esto es porque seguramente aún no te has dado cuenta de lo peligroso que es eliminar algo. Cuando tú eliminas algo desde la terminal, este archivo/carpeta se elimina… **PARA SIEMPRE *chan, chan chan.***

Con esto me refiero a que ese archivo que eliminaste NO se va a mover al basurero, sino que directamente se va a eliminar, y la única forma de recuperarlo podría ser con algún programa que lea sector por sector de tu disco duro.

Sobre cómo eliminar una carpeta, el comando que más comparten es `rm -rf` carpeta sin saber exactamente qué es lo que hace, poner esa `f` como parámetro es muy peligroso. Con poner `f` basta, esto porque el borrado es recursivo básicamente recorrerá cada subcarperta/archivo y las irá borrando uno por uno.

**##
DANGER ZONE ##**

Ahora quiero explicarte por qué este comando es peligroso, tanto que puedes llegar a eliminar tu sistema operativo, **NO CORRAS ESTE COMANDO POR NADA DEL MUNDO.**

Imagina que quieres eliminar alguna carpeta de tu computadora usando la terminal, y por alguna razón decides usar una ruta absoluta:

```
rm -rf / home/tuUsuario/carpetaAEliminar/
```

Todo bien… ¿verdad? Bien, si ejecutaras este comando todo tu sistema operativo desaparecería, ¿por qué? quiero que te fijes cómo entre el `/` y la palabra `home` hay un espacio… ¿recuerdas qué significaba esa `/`? Es la carpeta **raíz**, la carpeta **donde vive todo tu sistema operativo**, y le estás diciendo que la elimine a la fuerza, ese espacio indica que va a eliminar dos carpetas, primero eliminará `/` y luego eliminará `home/tuUsuario/carpetaAEliminar/`.

El comando correcto que deberíamos usar es:

```
rm -rf /home/tuUsuario/carpetaAEliminar/
```

Aquí ese espacio no existe, por lo que todo iría bien, quiero que te fijes cómo con un pequeño error, puedes eliminar todo. Por eso siempre ten cuidado al usar este comando, y de preferencia… **¡Usa el modo interactivo!**

Explorando el contenido de nuestros archivos

Podemos explorar el contenido de archivos sin la necesidad de abrirlos, desde la terminal . Esto para archivos de texto.

- `cat <documento de texto >`: Nos muestre el documento de texto completo.
- `head <documento de texto>`: Nos muestra las primeras 10 líneas de un archivo de texto. Para especificar el número de líneas `head -n <numero de lineas> <archivo>`.
- `tail <documento>`: Nos muestra las últimas 10 líneas.
- `less <archivo>`: Este es muy cool, es muy interactivo, nos permite hacer scroll, y nos permite hacer búsquedas haciendo `/<palabra a buscar>`. Para salir presionamos `q` .
- `xdg-open <archivo>`: Para abrir un archivo desde la terminal. Usa las aplicaciones predeterminadas. Esto para linux, para mac, es `open`. Esto crea un proceso en la terminal que no nos dejará hacer nada mas. Para terminar el proceso `ctrl+c`.
- `nautilus` nos permite abrir el explorador de archivos en una posición dada (en linux) .

¿Qué es un comando?

Un comando puede ser 4 cosas:

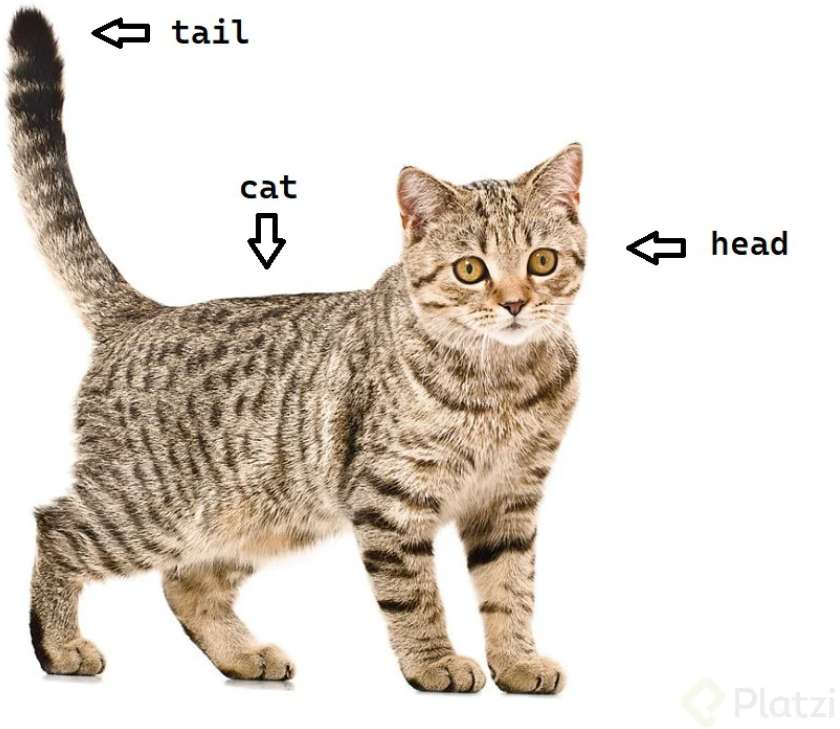

Figure 3: cat-head-tail-commands

- Un programa ejecutable que se compilo en algun lenguaje de programación y se puede ejecutar.
- Un comando de utilidad de la shell.
- Una función de shell
- Un alias

Comandos

- `type` nos permite saber que clase es un comando. Por ejemplo `type cd` (es una funcion de shell), `ls` (es un alias). Ejemplos:

```
$ type cd
cd is a shell builtin

$ type mkdir
mkdir is hashed (/usr/bin/mkdir)

$ type ls
ls is aliased to `ls --color=auto'
```

- Para crear un alias: `alias 'nombreDelAlias' = 'comandoQueInvoca'`. Por ejemploalias `l="ls -lh"`. Temporales, se borra cuando cerramos la terminal.
- Con `-help` o `help`, puedes tener una ayuda sobre los comandos.
- `man 'comando'` : hace referencia al manual de usuario de un comando, otro similar es `info 'comando'`.
- `whatis 'comando'`: nos da una descripcion muy corta de que hace ese comando. Pero no funciona con todos.

Wildcards

Son una serie de caracteres especiales que nos permiten realizar búsquedas muy avanzadas utilizando `ls` . Puedes utilizar wildcards con otros comandos que realicen manipulación de archivos como `mv`, `cp` o `rm` . También se conocen como **comodines**.

- `ls *.txt`: Lo que hace este comando es listar todos los archivos que tengan extensión .txt, independientemente del nombre que tengan.
- `ls datos*`: Lo que hace este comando es listar todos los archivos que inicien con la palabra "datos", independientemente de cómo siga el nombre o la extensión que tenga.
- `ls datos?`: Lo que hace este comando es listar todos los archivos que inicien con la palabra "datos" y solamente tengan un caracter más luego de esa palabra.
- `ls datos???`: Lo que hace este comando es listar todos los archivos que inicien con la palabra "datos" y solamente tengan tres caracteres más luego de esa palabra.
- `ls [[:upper:]]*`: Lo que hace este comando es listar todos los archivos

que inicien con una mayúscula, independientemente de cómo siga el nombre o la extensión que tenga.

- `ls -d [[:upper:]]*`: Lo que hace este comando es listar todos los DIRECTORIOS que inicien con una mayúscula, independientemente de cómo siga el nombre que tenga.
- `ls [[:lower:]]*`: Lo que hace este comando es listar todos los DIRECTORIOS que inicien con una minúscula, independientemente de cómo siga el nombre o la extensión que tenga.
- `ls [ad]*`: Lo que hace este comando es listar todos los archivos que inicien con la letra "a" o con la letra "d", independientemente de cómo siga el nombre o la extensión que tenga.

Empezando a correr

Redirecciones: cómo funciona la shell

Las redirecciones nos permiten trasladar información, haciendo uso del símbolo `>`. ¿Como funciona?

`stdin (0)`: Entrada estándar. `stdout (1)`: Salida estándar. `stderr (2)`: Salida de errores.

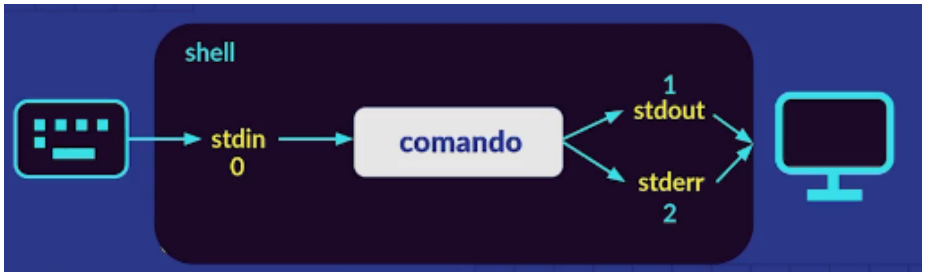

Figure 4: how-to-shell work

Normalmente, cuando pones un comando en la terminal, la salida se muestra ahí mismo, pero se puede redirigir la salida a un archivo . Si la salida es correcta tiene `file descriptor 1`, si no, `file descriptor 2`.

La entrada estándar es nuestro teclado que tiene `file descriptor 0`, pero también puede venir de otro lado .

- Para redirigir algo usamos `>`. Por ejemplo `ls > misarchivos.txt`, entonces la salida del comando se guarda en ese archivo de texto. Siempre crea este archivo (si ya existe, lo reescribe).
- Para que se concatene la salida en un archivo preexistente usa `"comando" >> "archivo"`. Esto ambos solo redirigen los `stdout`.
- Para redirigir `stderr`, agregas su file descriptor `"comando" 2> "archivo"`.

- Si quiere redirigir cualquiera de las dos opciones `"comando" >> "archivo" 2>&1`. Esto nos puede servir para, por ejemplo, guardar los mensajes de error que manda un servidor .
- Para redirigir `stdin` se usa `<`. Esto te permite tener de entrada de comandos algún archivo.

Redirecciones: pipe operator

El `pipe operator |` permite enviar la salida de un comando `stdout(1)` como entrada del siguiente `stdin(0)`.

- `echo <texto>` genera un `stdout` con el texto que tenemos.
- `cat <archivo1> <archivo2>` muestra los dos archivos concatenados .
- El pipe operator `|` hace que el `stdout` de un comando sea el `stdin` de otro comando. Por ejemplo `ls -lh | less`
- `tee` hace algo parecido a `>`, pero dentro de los pipe´s, por ejemplo `ls -lh | tee output.txt |less`. Se puede poner en medio, pero se ignora porque se sigue pasando.
- `cowsay "Texto"` es un comando que imprime una vaca que dice algo .
  - `sudo apt install cowsay` para instalar el comando.
- `lolcat "texto"` imprime a color la salida o le da color al comando que imprimimos.
  - `sudo apt install lolcat` para instalar lolcat

  Es resumen: Es uno de los operadores mas útiles que existen, ya que nos permite poner varios comandos, tales que la salida de uno es la entrada del siguiente .

Encadenando comandos: operadores de control

Los **operadores de control** permiten ejecutar más de un comando, encadenando los mismos.

- Son símbolos reservados por la terminal que nos permiten ejecutar varios comandos seguidos, e incluso agregar condicionales .
- **Síncronos**: Se corre uno detrás de otro, en orden. Se hace esto con `;` , por ejemplo `ls; mkdir carpeta1; cal`
- **Asíncrono**: Por cada comando, se abre una nueva terminal, y cada comando se corre de manera paralela, esto es con `&`, por ejemplo `ls & date & cal`
- **Condicionales**: Podemos agregar lógica a como se corren los comandos:
  - **AND**: Si se cumple un comando, entonces se ejecuta el siguiente, se usa `&&`, un ejemplo es `mkdir carpeta1 && cd carpeta1 && echo "Si se pudo"` . Si no se puede ejecutar el primer comando, no se ejecuta el siguiente.
  - **OR**: Se ejecuta el primer comando que se pueda ejecutar, y se usa `||`, por ejemplo `cd carpeta || echo "No hay carpeta"`.

Cómo se manejan los permisos

- Cuando listamos con `ls -l` se muestran varias cosas. Los **tipos de archivos**:
  - `-`: archivo normal.
  - `d`: directorio.
  - `l`: link simbólico.
  - `b`: archivo de bloque especial.
- **Tipos de modos**: `rwx` corresponde con `read, write y execute`.

Permiso	Archivo	Directorio
r	Permite abrir y leer un archivo.	Permite listar el contenido de un directorio solo si el permiso de ejecución (x) también está activo.
w	Permite escribir en un archivo; sin embargo, este atributo no permite cambiar el nombre de los archivos o eliminarlos. La capacidad de eliminar o cambiar el nombre de los archivos está determinado por los atributos del directorio.	Permite que los archivos dentro de un directorio sean creados, eliminados y renombrados si también se establece el atributo de ejecución.
x	Permite que un archivo sea tratado como un programa y pueda ser ejecutado.	Permite entrar al directorio.

Figure 5: rwx-permissions

- Se representan con 3 bits, y los podemos manejar a través de un modo **octal**, esto es, pasar de binario a número.
  - `rwx (1,1,1)` dueño. En modo octal es 7.
  - `r-x (1,0,1)` grupo. En modo octal es 5.
  - `r-x (1,0,1)` world. Octal 5.

Linux permission generator

- **Modo simbólico**: Esto es para asignar los permisos a los diferentes posibles usuarios.
  - `u`: Solo para el usuario.
  - `g`: Solo para el grupo.
  - `o`: Solo para otros (world).
  - `a`: Aplica para todos.

Octal	Binario	Permisos
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Figure 6: modo-octal-binario-permisos

Modificando permisos en la terminal

Existen diversos usuarios con permisos cada uno; el usuario root es especial y puede hacer de todo .

- Puedes crear archivos de texto también con `> archivo.txt` y también podemos editarlo con `cat > archivo.txt`
- En un archivo, se muestran: `[tipo de archivo][rwx usuario][rwx grupo][rwx mundo]`, por ejemplo, `-rw-r--r-- mitexto.txt` .
- `chmod <permiso en octal para usuario><para grupo><para mundo> <archivo>`: **change mode** nos sirve para cambiar los permisos de un archivo. Si hacemos por ejemplo `chmod 755 mitexto.txt` tendremos ahora `-rwxr-xr-x mitexto.txt`, esto no cambia para nada el contenido del archivo.
- Para quitarle los permisos a alguien en particular, usamos el modo simbólico y usando la resta, por ejemplo quitando el permiso de lectura al usuario `chmod u-r mitexto.txt`. Para agregar, se usa la suma.
- Podemos hacer configuraciones mas avanzadas, por ejemplo, podemos asignar varios permisos al mismo tiempo `chmod u-x,go=w mitexto.txt`.
- `whoami` Para saber que usuario somos, y también podemos obtener el ID del usuario con id.
- `su root` para cambiar de usuario hacía **root**, hay que tener cuidado al usar este usuario . Su home es incluso distinto. Los archivos que crea

root (o otro usuario) no se pueden eliminar por un usuario normal.
  - `sudo <comando>` nos otorga temporalmente los permisos de root para ejecutar algún comando que ocupe permisos especiales. Nunca dejes el usuario root por defecto, y ponle una contraseña distinta.

Cómo configurar variables de entorno

- La terminal tiene una configuración con diferentes valores, que se pueden acceder con las variables de entorno. Estas son muy importantes para la configuración general del sistema .
- Podemos guardar alias para que se queden de manera permanente con esto .
- `ln -s <ruta> <Nombre>` Esto para hacer link simbólicos, que son un tipo de archivo que hacen referencia a otro lugar, básicamente es un acceso directo desde terminal .
- `printenv` nos muestra todas las variables de entorno que tenemos configuradas .
- `echo $<variables>` esto nos sirve para imprimir una variable en particular.
- Algunas variables son:
  - `HOME` es nuestro HOME de usuario .
  - `PATH` tiene todas las rutas donde se encuentran los binarios en los que se ejecuta nuestro sistema. Hay varios manejadores de paquetes para binarios, pero no todas las veces se agregan a PATH, y se deben agregar a mano.
- En `HOME`, existe un archivo que se llama `.bashrc` que es donde está nuestra configuración de **Bash**. Lo podemos abrir con VS Code para modificarlo. En este archivo podemos crear alias.
- `alias <nombre>="comando"` para crear un alias útil .
- `code <archivo>` para abrir un archivo de texto en VS Code desde la terminal.
- Para modificar o crear una variable de entorno, se hace, por ejemplo `PLATZI_MESSAGE='Hola amigos"`.
- Para agregar una ruta a la variable PATH ponemos en **.bashrc** `PATH=$PATH:<ruta>`, guardamos, cargamos bash en la terminal, y listo .

  Es muuuy importante tener cuidado con los alias, nunca hay que nombrar un alias como un comando ya existente .

Comandos de búsqueda

Es una de las partes mas interesantes de la terminal, ya que nos permite buscar archivos de manera eficiente y específica .

- `which <programa>`: Busca en todas las rutas del PATH para encontrar donde está alojado algún archivo binario .
- `find <ruta inicial> -name <archivo>`: Nos permite encontrar un

archivo a partir de una ruta inicial, y dentro de todas las carpetas que surjan de ese inicio .

Algo interesante es que podemos usar wildcards para hacer mas eficiente la búsqueda . - `find <ruta inicial> -type <tipo> -name <nombre>`: podemos especificar el tipo de archivo, **d → directorio**, **f → documento**. - `find <ruta inicial> -size <tamaño><unidad>` podemos buscar tamaños mayores a un determinado tamaño, por ejemplo, de **20M (megas)**.

Su majestad: grep

Es uno de los comandos mas útiles, y de los mas potentes dentro de Linux.

El comando grep nos permite encontrar texto que contenga un patrón dentro de uno o varios archivos de manera rápida.

- `grep <Expresión regular> <archivo>`: El primer parámetro es una expresión regular, y es diferente a las wildcarts; es muy versátil para realizar búsquedas.

Tenemos varías opciones:

- `-i`: para ignorar case-sensitive.
- `-c`: cuenta el número de elementos.
- `-v`: para hacer búsqueda complementaria, esto es, todos los elementos que no coincidan.
- `wc <archivo>`: cuenta el número de palabras. Opciones:
  - `-l`: cuenta el número de lineas.
  - `-w`: cuenta el número de palabras.
  - `-c`: número de bits.

  Las expresiones regulares pueden ser útiles en otros contextos, por ejemplo, en otros lenguajes de programación que las soporten.

Ejemplos de grep

- Buscar algún paquete en específico que tengas instalado:

```
dpkg --get-selections | grep nombreDelPaquete
# dpkg --get-selections te dirá todos tus paquetes instalados
# grep filtrará esa lista con el paquete que te interesa
```

- Filtrar algún archivo en específico después de un ls:

```
ls -al | grep myFile.txt

# ls te dará la lista de todos tus archivos
# grep filtrará todos y te mostrará únicamente el que deseas
```

- Buscar algún contenido en específico dentro de algún archivo:

```
cat unArchivoLargo.txt | grep "La línea que busco"

# cat Te listará todo el contenido de ese archivo
# grep te filtrará únicamente lo que quieres ver
```

- Buscar una línea en específico en diferentes archivos por medio de un patrón:

```
grep "string" archivo_*

# grep buscará la palabra "string" en todos los archivos que comienzen por "archivo_" y te l
```

- Buscar usando expresiones regulares:

Imagina que tienes un archivo llamado test.txt y adentro contiene la siguiente frase. Imagina que quieres buscar algo, entonces, podemos usar grep así:

```
grep "Imagina .* algo" test.txt

# grep buscará alguna coincidencia, la expresion .* indica que ahí dentro puede haber una o
```

15 Practical Grep Command Examples In Linux / UNIX

Utilidades de la terminal

Utilidades de red

Existen comandos que nos dan información sobre la red :

- `ifconfig`: Nos da información general sobre nuestra red .
- `ping <url>`: Nos dice si una página está activa a no . Lo revisa continuamente, y podemos usarla para ver la velocidad de nuestra conexión.
- `curl <url>`: Nos trae un archivo de texto a través de la red . (El index.html).
- `wget <url>`: Web get, trae un archivo de la web, descarga el archivo directamente a nuestra computadora . (El index.html con mejor formato).
- `traceroute <url>`: Nos da la lista de todas las computadoras (direcciones IP) por la que nuestra conexión pasa para llegar a un sitio web .
- `netstat -i`: Nos muestra los dispositivos de red. Similar a ifconfig pero más resumido .

Comprimiendo archivos

Comprimiendo archivos.

Podemos crear archivos comprimidos `.zip` o `.tar` desde la terminal.

- `.tar`: se usa mucho en repositorios. Para comprimir `tar -cvf <nombre>.tar <archivos>`, donde `c → compress`, `v → verbose`, `f→ file`.

- `.gz`: es un poco mejor, se usa el mismo comando pero con la bandera `z→zip tar -cvzf <nombre>.gz <archivos>`. Usa el algoritmo gzip que es muy eficiente para comprimir. Para descomprimirlo, usamos el mismo comando pero con la bandera `x → decompress` en lugar de `c → compress`. Para que funcione, debemos descomprimir debemos usar el mismo tipo de compresión (tar o zip).
- `.zip`: es uno muy común. Es necesario instalarlo en linux. `zip -r <nombre>.zip <directorio>`. Para descomprimir usamos `unzip <nombre>.zip`.
- `.rar`: funciona igual que `.zip`, pero con `rar` y `unrar` de comandos .

Manejo de procesos

Cuando se traba nuestro OS, normalmente terminamos procesos con el administrador de tareas , en la terminal se puede hacer, pero es un poco diferente. - `ps` nos muestra los procesos que están corriendo actualmente. Cada proceso tiene un **PID**. Podemos ver los procesos que estén en el background (por ejemplo, CAT). - `kill <PID>` nos ayuda a terminar procesos fuera de nuestra terminal. - `top <PID>` nos muestra los procesos que están usando más recursos de nuestra computadora. Podemos filtrar los procesos (para ver como, usamos bandera `h` → help).

> La terminal, sabiéndola usar bien, es más eficiente que el administrador de tareas.

- `htop` es como `top` pero con esteroides. Debemos instalarlo. Tiene muchas más opciones

Nivel de poder de algunos de programas para gestionar procesos:

- 4to: `ps` (Dios griego: Hephaestus)
- 3ro: `top` (Dios griego: Apollo)
- 2do: `htop` (Dios griego: Poseidon)
- 1ro: `glanses` (Dios griego: Zeus) lo malo de este es que consume bastante CPU, pero se ve genial y te dice que problemas ocurren.

Procesos en foreground y background

Los procesos que están corriendo pero no se muestran en terminal se dice que están en **background**. Los que si se muestran están en **foregroung**.

- Para mover un proceso al background, usamos `Ctrl+z`. Esto lo suspende, pero sigue corriendo (como con Cat). Para matar un proceso se usa `Ctrl+c`
- `fg <numero de trabajo>` nos permite traer un proceso al `foreground`. Es importante notar que el número de trabajo no es lo mismo que el PID.
- `bg <numero de trabajo>` nos permite llevar un proceso al background, pero sin suspender el proceso.

Ejemplo muy sencillo que servirà para controlar procesos es el siguiente:

Crear un listado recursivo (Que liste todos los archivos y directorios)

Para eso seguimos los sigueintes pasos: 1. Primero hacemos el comando `cd /` para dirigirnos a la raíz de nuestro sistema. 2. Después ejecutamos el comando `ls -R` y comenzará a listar TODO lo que existe dentro de nuestro S.O. 3. Ahora lo que hacermos será confirmar lo aprendido, utilizamos `Ctrl + C` y veremos que se cancela el proceso, pero lo interesante viene cuando hacemos lo siguiente 4. Ahora ejecutamos el mismo comando para listar TODO, `ls -R` y lo detenemos con `Ctrl + Z` a lo que nos saldrá lo siguiente: `[Número del proceso] + Id del proceso +` En donde se detuvo el proceso. Y se verá algo como esto: `[1] + 40751 suspended (signal) ls --color=tty -R` 5. Y repetiremos el paso 4 otras 3 veces. 6. Si ejecutamos el comando `jobs` nos mostrará todos los procesos suspendidos y si tenemos ejecutando alguno en segundo plano. 7. Para volver a activar algún proceso podemos hacerlo con el comando `fg %Número del proceso`

Por ejemplo en bash con `fg %1` y en zsh con `fg %1` para volver a correr el primer proceso que suspendimos, y ahora le damos `Crl + C` para ahora sí matarlo, ahora el proceso 1 ya no existirá al ejecutar jobs pero el 2, 3 y 4 ahí seguirán

Esto es muy importante para cuando queremos manejar diferentes procesos, por ejemplo con el comando `sleep 10000 &` pondrá un proceso en segundo plano, entonces no lo podremos ver, pero nos marca el Número del proceso entre los `[]` también lo podemos ver con `jobs` junto con su número de proceso, si lo queremos traer a primer plano lo podemos hacer con el comando `fg %1` por ejemplo y de ahí ya cancelarlo o suspenderlo, o una manera más rápida, solo ejecutar el comando `kill %Número del proceso` por ejemplo: con el mismo ejemplo de `sleep 10000 &` si nos da el `[1]` lo podremos terminar con `kill 1` o `kill %1` y nos saldrá algo como `[1] + 41723 terminated sleep 10000` y ahora ya saber como manejar y exterminar todos los procesos como terminator .

Editores de texto en la terminal

Una de las utilidades más importantes de la terminal es el editor de texto. Hay diferentes opciones, pero **Vim** es uno de los mas sencillos y populares. También está **Emacs** y **Nano** . - `vi <archivo>` es la versión vieja. - `vim <archivo>`: Vi modern. Tenemos dos modos, el normal o de inserción, para instertar presionamos la `tecla i` y para salir presionamos `Esc`. Para salir del editor y guardar `:wq`. - Este editor tiene un resaltador de sintaxis depende del tipo de archivo. - Al igual que con `less` para buscar una palabra, podemos hacerlo en Vim con `/<palabra>`. Te lleva a la primera coincidencia. - Para eliminar una línea, desde el modo normal, nos ponemos al inicio de la línea y presionamos `dd`.

Personalizar la terminal de comandos

Podemos personalizar la terminal para que quedé bonita, profesional y sea muy cómoda .

- Para esto, podemos usar un emulador llamado Tilix. En Tilix podemos tener varias terminales activas .
- Podemos instalar **ZSH**, y luego `chsh -s <> $(which zsh)`, con este comando podemos cambiar de shell.
- Ya en ZSH, podemos instalar un enhancer que incrementa las capacidades de la shell:
  - Oh My Zsh - a delightful & open source framework for Zsh
  - Oh My Zsh - Github
- Customize Windows Terminal with WSL2
- Para regresar a bash `exec bash` y para ir a ZSH `exec zsh`.
- Puedes mejorar aún más tu terminal con PowerLevel10k :
  - GitHub - romkatv/powerlevel10k: A Zsh theme
- Es importante que instales las fonts necesarias para usar la funcionalidad máxima de esto .
- Personalizar la terminal por codevars