Clases del Curso de Introducción a la Terminal y Línea de Comandos

Comandos Basicos de la terminal:

Listar archivos:

```
ls
```

Listar archivos para ver su peso de una manera mas mas legible

```
ls -lh
```

Listar archivos ocultos:

```
ls -a
```

Identificar la ruta en la que estamos en nuestro sistema:

```
pwd
```

Movernos entre directorios:

```
cd
```

Crear un directorio:

```
mkdir
```

Copiar un archivo:

```
cp
```

Borrar un archivo:

```
rm
```

Mover un archivo:

```
mv
```

Borrar un directorio:

```
rmdir
```

Limpiar la terminal

```
clear
```

Anatomía de un CLI o Command Line Interface

Definición

- Un comando es una acción escrita, generalmente, a través de una terminal y emulada por una shell para ser ejecutada por nuestro sistema operativo.

Comandos de la clase

- **pwd o print working directory:** Comando para encontrar el ruta actual de trabajo de manera absoluta o desde el directorio
- **cd o change directory** Comando para navegar entre directorios requiriendo, dependiendo, del directorio o ruta a navegar. Hay varios shortcuts para navegar rápidamente: **cd ..** para mover a un directorio inmediato anterior **cd** para ir al directorio home **cd -** para mover a un directorio previo
- **ls o list directory contents:** Comando para ver el contenido de un directorio. Por defecto, se mostrará el contenido del directorio actual. Algunas opciones son: **ls -R** listar archivos de sub-directorios **ls -al** listar archivos ocultos y de manera detallada file . Comando para reconocer el tipo descriptivo del archivo

Manipulando archivos y directorios.

Todo lo que hacemos normalmente en una interfaz gráfica, lo podemos hacer en la terminal. 1. **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 mkdir : Para crear un directorio nuevo 🗂️. No es buena idea en general poner espacios en el nombre de los directorios. Si lo quieres hacer, usa comillas "Nombre con espacios". touch : Para crear archivos vacíos. Puedes crear varios con un solo comando 📄. cp : Para copiar un archivo. Si no se pone una ruta en el nombre del nuevo, se va a copiar en el directorio donde estés 🖨. mv

: Para mover archivos a otra ruta. También te sirve para renombrarlos 💫, funciona con directorios y con archivos. rm : Elimina archivos. Hay una opción muy útil que es rm -i (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 , 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 😢.

¿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 recuerarlo podría ser con algún programa que lea sector por sector de tu disco duro.

He visto por ahí cuando alguien pregunta por cómo eliminar una carpeta, el comando que más comparten es rm -rf carpeta sin saber exactamente qué es lo que hace, ahora ya saben que poner esa “f” como parámetro es muy peligroso. Con poner “r” 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 /? Ya lo vimos 👀… Así 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 la a eliminar dos carpetas, primero eliminará / y luego eliminará home/tuUsuario/carpetaAEliminar/. El comando correcto que deberíamos usar es:

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

Explorando el contenido de nuestro archivos.

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

- **head <documento de texto>:** Nos muestra las primeras 10 líneas de un archivo de texto. Para especificar el número de líneas head -n
- **cat <documento>:** ver todo el contenido
- **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 4️⃣:

1. Un programa ejecutable.
2. Un comando de utilidad de la shell. Esto es un programa en si mismo, que puede tener funciones. Ejemplo cd
3. Una función de shell. Son funciones de shell externas al comando de utilidad. Ejemplo mkdir
4. Un alias. Un ejemplo es ls

- **type <comando>:** Nos permite conocer que tipo de comando es.
- **alias l="<secuencia de comandos>":** Nos permite crear comandos. Son temporales, se borran al cerrar la terminal.
- **help <comando>:** Nos permite consultar un poco de documentación de un comando 📄.
- **man <comando>:** De manual, nos permite conocer mucha mas información de un comando.
- **info <comando>:** Similar al anterior, pero un poco resumido y con otro formato.
- **whatis <comando>:** Describe un comando en una sola línea ☺. 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 * <texto>: Nos muestra todos los archivos que tengan en su nombre dicho texto al final. Si haces ls * lo busca al final. El asterisco significa cualquier string. *ls <text>? El signo de interrogación sustituye a cualquier carácter, por ejemplo, buscamos todos los archivos que tengan una palabra y un número dado de caracteres después. El signo de interrogación significa cualquier carácter.
- **ls [[:upper:]]*** Para filtrar cosas que inicien que una mayúscula; esto busca también dentro de los directorios.
- **ls [[:lower:]]*** Lo mismo pero con minúsculas.
- **ls -d** Se muestran solo directorios.
- **ls [ad]*** Todo lo que inicie con a o con b

Redirecciones: Cómo funciona la shell.

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, 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.

- 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 🤳.
- **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** . introduce todos los metadatos de los archivos dentro de output.txt (read from standard input and write to standard output)
- **cowsay "Texto"** es un comando que imprime una vaca que dice algo JAJAJAJAJ 🐮.
- **lolcat "Texto"** es un comando que imprime texto con colores. "sudo apt install lolcat" para instalar

Encadenando comandos y operadores de control.

- 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:

    1. 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. 🚆

    2. 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. 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 (1,1,1) grupo. En modo octal es 5.
r-x (1,0,1) world. Octal 5.
```

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.
```

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 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 laresta, 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!!

Variables de entorno.

La termina 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 🗂️.
- **print env** 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. **> code .bashrc**

- **alias <nombre>="comando"** para crear un alias útil 👀. code 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 o programa ejecutable.
- **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 muy cool es que podemos usar wildcards para hacer mas eficiente la búsqueda . 1. **find <ruta inicial> -type <tipo> -name <nombre>** podemos especificar el tipo de archivo, d → directorio, f → documento. 2. **find <ruta inicial> -size <tamaño><unidad>** podemos buscar tamaños mayores a un determinado tamaño, por ejemplo, de 20M (megas). 3. Solución al reto:

```
        find ./ -name *.txt -type f -size 1M > mis_archivos_texto.txt |
echo "archivos guardadados exitosamente"
```

Banderas del comando find

Banderas básicas:

- -name: Realiza una búsqueda por nombre de archivo.
- -iname: Realiza una búsqueda por nombre de archivo sin tomar en cuenta las mayúsculas o minúsculas.
- -type: Realiza una búsqueda por tipo de archivo, f(files) y d(directories) que son los más comunes.
- -size: Realiza una búsqueda por el tamaño de archivo y/o directorio.

Banderas de tiempo⏱️

- mmin: Búsqueda por tiempo en minutos.
- mtime: Búsqueda por tiempo en días.

Más banderas👀

- **-maxdepth** Después de está bandera se pone el número de niveles de profundidad en los que queremos realizar la búsqueda
- **-empty** Realiza una búsqueda de archivos y/o directorios vacíos.
- **-perm** Búsqueda de archivos por permisos.
- **-not** Retorna los resultados que no coinciden con la búsqueda.
- **-delete** Está bandera se coloca al final del comando, eliminara los resultados de la busqueda(⚠Hay que tener mucho cuidado al usarla).

Hay muchas más banderas, pero esas son las que me parecieron más útiles.

- Ejemplos: A continuación pondré unos cuantos ejemplos de las banderas mencionadas anteriormente.

Búsqueda de todos los archivos con el permiso 644 a partir del directorio actual con una profundidad de 2 niveles.

```
find . -perm 644 -maxdepth 2
```

Búsqueda de archivos vacíos a partir del directorio actual.

```
find . -type f -empty
```

Búsqueda de todos los archivos .log, todos los resultados serán eliminados.

```
find . -type d -iname "*.log" -delete
```

Su majestad grep

Permite encontrar coincidencias en la búsqueda dentro de un archivo de texto, no necesariamente un archivo puede ser incluso la salida de info de la terminal ya que después de todo es texto.

- grep <palabara a buscar // aquí estoy buscando una coincidencia, y me va a mostrar por pantalla ejemplo: $ grep Towers movies.csv, busco en el archivo movies todas las películas que tengan las palabra towers
- grep // vamos a buscar todas las coincidencias que tengan “the” y que estén en el archivo movies, pero con el parámetro “-i” que nos permite que tanto las palabras que empiecen con mayúscula y minúscula se muestren, ya que con el anterior solo buscaba por defecto los “the” que empezaban con minuscula, ejemplo $ grep -i the movies.csv
- grep –i the movies.csv | less // nos permite hacer scrol, es decir verlo de manera adecuada o poco a poco.
- grep -c the movies.csv // si el parámetro lo cambiamos por –c en este caso nos muestra el número de veces que se encuentra una coincidencia y si le agregamos el parámetro “-ci” básicamente mostrara más numero de coincidencias debido a lo ya explicado con -i“
- grep -v the movies.csv // nos muestra las opciones que no coinciden con “the” si agregamos el parámetro por ejemplo –vic excluirá the pero tanto las coincidencias en mayúscula y minúscula y además las contara.
- **Bonus:** El comando wc nos indica varios datos útiles en un archivo entonces si le pasamos el comando al archivo: $ wc movies.csv, nos mostrara lo sigt. 9126 30006 477779 movies.csv, donde 9126 significa las líneas, 30006 cuantos caracteres, 477779 numero de bits y finalmente el nombre del archivo. Este comando tiene varios parametrs también como: -l, -w, -c donde respectivamente nos muestra solo las líneas, caracteres y numero de bits.

Utilidades de red.

Existen comandos que nos dan información sobre la red 🥅:

- ifconfig: Nos da información general sobre nuestra red 🔍.
- ping : Nos dice si una página está activa a no 🏃. Lo revisa continuamente, y podemos usarla para ver la velocidad de nuestra conexión.
- curl : Nos trae un archivo de texto a través de la red 🌎. (El index.html).
- wget : Web get, trae un archivo de la web, descarga el archivo directamente a nuestra computadora 💻. (El index.html con mejor formato).
- traceroute : 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.

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

- tar se usa mucho en repositorios. Para comprimir tar -cvf .tar , 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 .gz . 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 .zip . Para descomprimir usamos unzip .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 nos ayuda a terminar procesos fuera de nuestra terminal. 🛑
- top 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 💪🏽

Procesos de 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 nos permite traer un proceso al foreground. Es importante notar que el número de trabajo no es lo mismo que el PID.
- bg nos permite llevar un proceso al background, pero sin suspender el proceso.⭐

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 /. 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.

```
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
- Windows Subsystem for linux no permite GUI ☹ por lo que no se puede usar Ubuntu en windows con esta terminal. Pero podemos usar WINDOWS TERMINAL y esta hermosaaaa 💕. Este blog me resultó muy útil para instalarlo:
- 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 🔥.