**Introducción a Git y a la línea de comandos**

Hoy en día, estamos acostumbrados a interactuar con los programas a través de interfaces gráficas como botones, formularios y menús desplegables. Por ejemplo, para crear una carpeta en tu computadora, puedes hacer clic derecho y seleccionar la opción necesaria en el menú gráfico. Esta interfaz de "apuntar y hacer clic" puede ser sumamente práctica, pero existen otras interfaces.

Además de la interfaz gráfica de tu computadora, también puedes usar una interfaz basada en texto: la línea de comandos. Puedes utilizar la línea de comandos para gestionar muchas de las tareas que gestiona la interfaz gráfica, tales como crear, mover o eliminar archivos. Pero la línea de comandos también te da acceso a muchas herramientas de desarrollo que son difíciles, por no decir imposibles, de utilizar a través de una interfaz gráfica. Un ejemplo de esto es Git, un sistema de control de versiones que usarás a lo largo de este programa y en tu carrera como ingeniero o ingeniera de software.

En este capítulo, te presentaremos Git brevemente y te mostraremos cómo instalarlo en tu computadora. Después de eso, te explicaremos los conceptos básicos de la línea de comandos.

# ¿Qué es la línea de comandos?

Comencemos a explorar la línea de comandos explicando parte de la terminología. Como dijimos en la lección de introducción, la línea de comandos es una interfaz basada en texto para interactuar con una computadora. Podemos acceder a esta interfaz a través de programas llamados terminales o emuladores de terminal.

Más específicamente, el emulador de terminal interactúa con un shell, un programa que interpreta las entradas de la línea de comandos y las traduce en instrucciones para el sistema operativo.

Los programas shell más comunes en macOS y Linux son bash y zsh, los cuales comparten interfaces similares, así que sus comandos, normalmente, son interoperables.

Al emulador de terminal predeterminado en estos sistemas se le conoce, simplemente, como Terminal. Pero, los programas shell y terminal predeterminados en Windows son muy distintos de bash y zsh. Es por ello que tuviste que instalar Git en la lección anterior. Para los usuarios de Windows, Git Bash ya viene incluido en Git, lo que proporciona un shell bash y un emulador de terminal en Windows.

### Abrir la terminal

Ahora, manos a la obra. Para abrir tu emulador de terminal:

* En Windows, abre Git Bash.
* En macOS o Linux, abre Terminal.

💡 También puedes acceder a la línea de comandos directamente desde VS Code. El atajo para abrirlo es Ctrl + ` tanto en Windows como en macOS. Si estás en Windows, debes cambiar tu perfil de terminal predeterminada a Git Bash. En [este video](https://www.youtube.com/embed/3oF9diyu7hc) te explicamos cómo.

Cuando abras la terminal, lo que veas dependerá mucho del emulador de terminal que estés utilizando, pero las características generales de la línea de comandos siguen siendo las mismas. A continuación, puedes ver un ejemplo de un emulador de terminal de Linux.



1. A la izquierda, puedes ver el símbolo del sistema. La información que se incluye en el mensaje varía de sistema en sistema. En este caso, vemos el nombre de usuario student y el nombre de computadora tripleten, seguidos de los caracteres ~$. La ~ tiene un significado especial del cual hablaremos pronto. El carácter $ suele emplearse para denotar el final del símbolo del sistema.
2. A la derecha, puedes ver el campo de entrada donde escribes tus comandos.

Y eso es todo lo que necesitas saber para empezar a utilizar la línea de comandos: dónde escribir los comandos. Así que, ¡aprendamos sobre comandos!

**Familiarizándote con pwd y ls**

💡 En este capítulo, vamos a utilizar la interfaz English OS con nombres estándar de carpetas como *Desktop* ("Escritorio"), *Downloads* ("Descargas"), etc. El inglés es el idioma estándar para la tecnología, especialmente en desarrollo web. Emplear una interfaz de sistema operativo en inglés favorece tanto a la compatibilidad global como al uso de un idioma estandarizado para una comunicación efectiva en la industria de la tecnología.

Muchas de las tareas más comunes de la línea de comandos implican interactuar con el sistema de archivos de tu computadora. Piensa en cómo has interactuado con el sistema de archivos a través de la interfaz gráfica de usuario (GUI). Crear archivos, eliminar archivos, copiar o cambiar el nombre de archivos: estas tareas también se pueden gestionar en la línea de comandos. Pero antes de interactuar con un archivo o carpeta específica, debemos saber cómo encontrarla en la estructura de archivos de nuestra computadora.

Siempre que utilizas una interfaz gráfica, estás en una carpeta. Lo mismo ocurre con la línea de comandos. Al ejecutar el comando pwd, podemos mostrar nuestra ubicación actual. Las letras del comando pwd significan Print Working Directory ("mostrar directorio de trabajo") y eso es exactamente lo que pasará al introducir este comando:

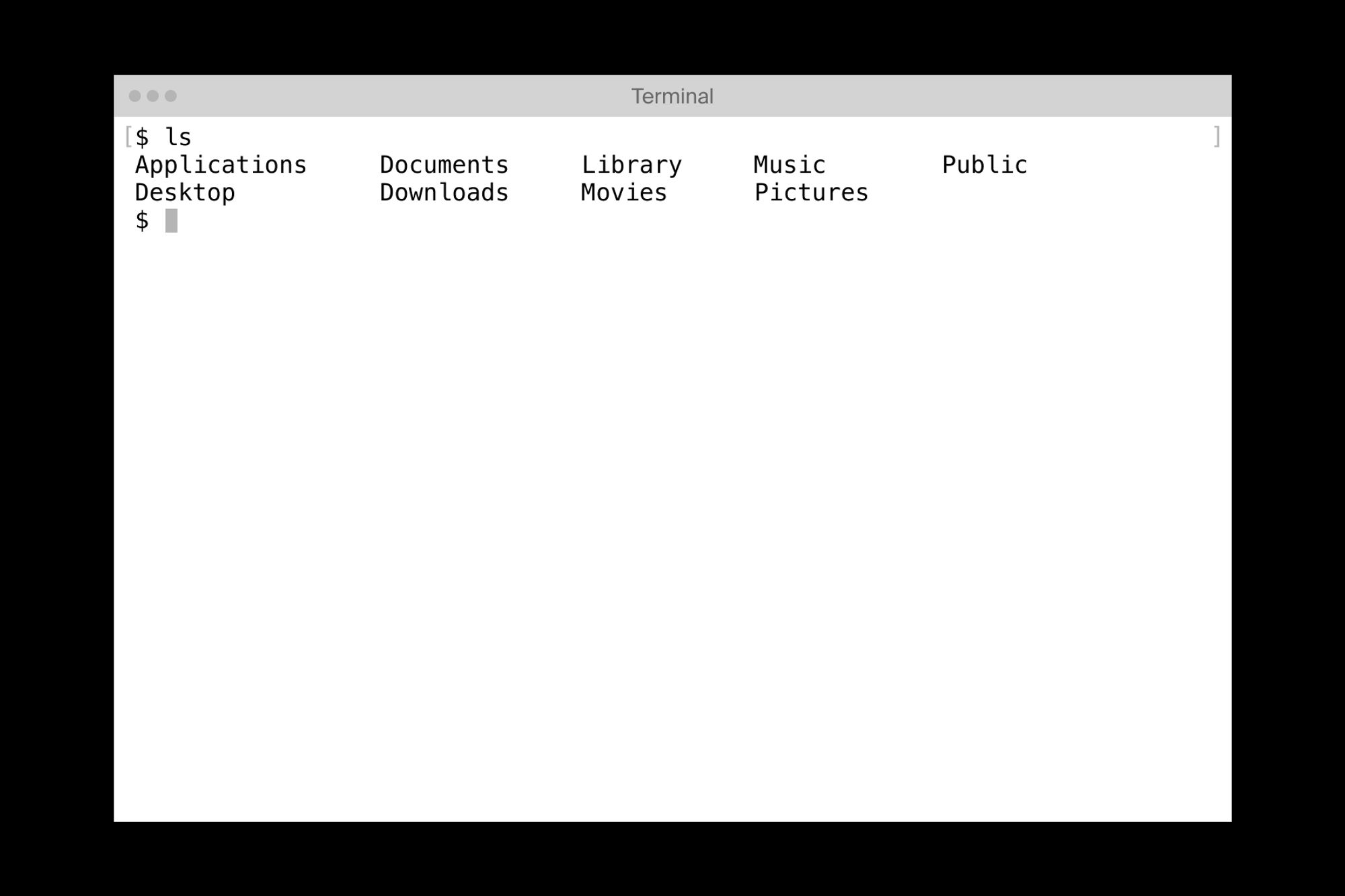
pwd

Después de introducir nuestro comando y presionar Enter, se mostrará la ruta de nuestra carpeta en la siguiente línea. Ahora verás tu nombre de usuario en lugar de TU\_NOMBREDEUSUARIO.

/Users/TU\_NOMBREDEUSUARIO

Este es tu directorio de usuario (o directorio de inicio), y siempre que abras la terminal, aquí es donde comenzarás por defecto.

En una interfaz gráfica, siempre que abras una carpeta, podrás ver el contenido de esa carpeta. Este no es el caso en la línea de comandos. En su lugar, tendremos que utilizar el comando ls para ver nuestros archivos cuando usemos la línea de comandos. Si introduces este comando, verás una lista de los archivos y carpetas dentro del directorio actual.



Pregunta

¿Qué comando mostrará la ruta hacia el directorio de trabajo actual?

pwd

¡Correcto! El comando ls se usa para enumerar el contenido del directorio actual. Y si bien a veces verás que se usa la abreviatura "cwd" para referirse al directorio de trabajo actual, el comando para mostrarlo es pwd.

cwd

ls

¡Perfecto!

# Familiarizándote con cd

A estas alturas ya sabes cómo averiguar dónde te encuentras en el sistema de archivos y ver el contenido de esa ubicación. Así que ahora vas a aprender a desplazarte mediante el comando cd. Aquí, cd significa "cambiar de directorio". La sintaxis consiste en el comando seguido por el directorio deseado, de esta manera: cd NOMBRE\_CARPETA. Aquí tienes un ejemplo:

*# comprueba dónde estamos*

pwd

*# salida:*

/Users/TU\_NOMBREDEUSUARIO

*# mover a la carpeta de Escritorio*

cd Desktop

*# comprueba dónde estamos ahora*

pwd

*# salida:*

/Users/TU\_NOMBREDEUSUARIO/Desktop

*# intentar utilizar cd para entrar a una carpeta que no existe genera un error*

cd carpeta-no-existente

*# salida*

cd: carpeta-no-existente: No such file or directory

💡 En el bloque de código anterior, hacemos uso de los comentarios al estilo bash. En el lenguaje bash, los comentarios van precedidos por un sólo carácter #. Técnicamente, estos comentarios deberían colocarse al final de las líneas de comando correspondientes. Por ejemplo:

pwd *# se ignora todo lo que está a la derecha del hash.*

Pero los hemos colocado arriba de las líneas de comando para mejorar la legibilidad. También estamos utilizando los comentarios para distinguir entre los comandos y su salida.

Observa también que estamos empleando un guion bajo para separar las letras mayúsculas para representar el contenido de la variable, como NOMBRE\_CARPETA y TU\_NOMBREDEUSUARIO. Esta es una convención que usaremos para resaltar partes de los comandos que no deberían ejecutarse tal como se muestran. Por ejemplo, en el comando cd NOMBRE\_CARPETA, debes remplazar NOMBRE\_CARPETA con el nombre de un archivo real.

## Espacios en los nombres del archivo o carpeta

Te recomendamos que evites dejar espacios en los nombres de archivos o carpetas que vayas a crear, ya que te será difícil trabajar con ese tipo de archivos en la línea de comandos y rutas de archivos de tus proyectos.

*# cd no funcionará si intentas emplear un nombre de carpeta que tenga un espacio*

cd My Videos

*# salida*

cd: too many arguments

Pero, si no pueden evitarse los espacios, tienes dos opciones para manejarlos. Primero, puedes utilizar comillas:

cd "My Videos"

Por otra parte, puedes emplear el carácter de diagonal invertida \ para escapar el espacio en el nombre de la carpeta:

cd My\ Videos

## Volver al directorio padre

Volvamos a donde dejamos el primer bloque de código en esta lección.

pwd

*# salida*

/Users/TU\_NOMBREDEUSUARIO/Desktop

Hemos utilizado cd para entrar en nuestro directorio Desktop. Pero ¿qué tal si queremos volver al directorio padre, /Users/TU\_NOMBREDEUSUARIO?

Para emplear cd para entrar al padre inmediato del directorio actual de trabajo, escribe dos puntos .. en lugar del nombre de carpeta.

pwd

*# salida*

/Users/TU\_NOMBREDEUSUARIO/Desktop

*# usar dos puntos nos hace subir un nivel*

cd ..

*# veamos dónde estamos ahora*

pwd

*# salida*

/Users/TU\_NOMBREDEUSUARIO

También podemos agregarlos a los ... Por ejemplo, el comando cd ../UNA\_CARPETA intentaría cambiar el directorio en un directorio padre y, de ahí, en UNA\_CARPETA.

## Volviendo al inicio

Cada sesión de la línea de comandos empieza en el directorio de inicio, así que es una tarea común cambiar directorios de vuelta a este directorio. Pero, supón que estás en lo más profundo de la estructura de archivos.

pwd

*# salida*

/Users/TU\_NOMBREDEUSUARIO/Desktop/pics/kittens

Podrías volver al directorio de inicio con cd de esta manera:

*# utiliza cd para subir tres niveles en la estructura de archivos*

cd ../../..

pwd

*# salida*

/Users/TU\_NOMBREDEUSUARIO

Pero es tedioso y susceptible a errores. Afortunadamente, existe un atajo: cd ~. La ~ representa el directorio de inicio del usuario (o directorio de usuario), y ejecutar cd ~ te devolverá a él desde donde sea que estés en ese momento. También puedes emplear ~ como un segmento en rutas de archivo más largas. Por ejemplo, cd ~/Desktop utilizará cd para entrar en el directorio Desktop que está dentro del directorio de inicio.

Sólo ten en cuenta que existe un atajo aún más corto para volver al directorio de inicio, ya que ejecutar cd sin ningún otro argumento tiene el mismo efecto que ejecutar cd ~.

## Volver al directorio anterior

A veces resulta útil emplear cd para volver a entrar a la última ubicación en que estuviste. Por ejemplo, supongamos que estás en tu directorio web\_project\_library, pero por alguna razón, debes usar cd para entrar en tu directorio de inicio.

pwd

*# salida*

/Users/TU\_NOMBREDEUSUARIO/projects/web\_project\_library

*# volver a inicio*

cd

pwd

*# salida*

/Users/TU\_NOMBREDEUSUARIO

Puedes utilizar cd para volver a entrar a tu directorio del proyecto con cd /Users/TU\_NOMBREDEUSUARIO/projects/web\_project\_library, pero también existe un atajo.

*# cambia el directorio al último directorio visitado*

cd -

pwd

*# resultado*

/Users/TU\_NOMBREDEUSUARIO/projects/web\_project\_library

## Mantenerse orientados fácilmente

Te hemos mostrado cómo orientarte en la línea de comandos a través del comando pwd. Pero, en muchos sistemas, esta información está disponible de un solo vistazo. Recordemos el ejemplo de un símbolo del sistema que te mostramos en una lección anterior:

student@tripleten ~$

La ~ representa el directorio actual de trabajo, que en este caso, es el directorio de inicio.

student@tripleten ~$ pwd

*# salida*

/Users/student

Si cambiamos de directorio, podemos ver su cambio en el símbolo del sistema:

student@tripleten ~$ cd Downloads

student@tripleten ~/Downloads$

## Trabajar con rutas de archivo en la línea de comandos

Imagina que estás en tu directorio de usuario, y te gustaría usar cd para entrar en una carpeta llamada notes que está en tu escritorio. Podrías hacer esto:

cd Desktop

cd notes

pwd

*# salida*

/Users/TU\_NOMBREDEUSUARIO/Desktop/notes

Sin embargo, podemos lograr esto en un solo comando, de esta manera:

cd Desktop/notes

Observa que la ruta de archivo Desktop/notes comienza con el nombre de un directorio dentro del directorio actual de trabajo. Probablemente recuerdes del capítulo sobre sistemas de archivos, que el directorio actual de trabajo suele representarse con ./. Esta convención viene de la sintaxis de la línea de comandos, y el siguiente comando se comportaría igual que el anterior.

cd ./Desktop/notes

Pero, tal y como vimos en el capítulo sobre sistemas de archivos, una ruta de archivo que comience con una / es algo completamente diferente.

*# ups, no existe tal directorio*

cd /Desktop/notes

*# salida*

cd: /Desktop/notes: No such file or directory

Esto da como resultado un error, ya que la primera / representa el directorio raíz, no el directorio de inicio donde se ubica el directorio Desktop/notes.

Pregunta

1. Verdadero o falso. Cuando abres la terminal por primera vez, empiezas en tu directorio de usuario.

Verdadero

¡Correcto! Es importante reconocer este hecho, pues no entenderlo podría dar lugar a muchos errores.

Falso

¡Buen trabajo!

Pregunta

2. Imagina que estás en tu directorio de usuario y quieres utilizar **cd** para entrar en tu directorio de descargas, **Downloads**, que está dentro del directorio de usuario. ¿Qué comandos harán la magia?

Elige tantas como quieras

cd Downloads

Correcto.

cd /Downloads

cd ~/Downloads

Correcto.

cd /.Downloads

cd ./Downloads

Correcto.

¡Bien hecho!

Pregunta

3. Imagina que estás en el directorio **~/projects/sample-project/images**. Deseas volver al directorio **projects**. ¿Cuál de los siguientes comandos lo hará?

Elige tantas como quieras

cd ..

cd ../..

Correcto. Este comando te llevará al padre del padre del directorio images, que es projects.

cd ~/projects

Correcto. Este comando utiliza una ruta de archivo absoluta para llegar a la ubicación deseada. A veces, es más fácil hacer esto que contar mentalmente el número de segmentos ../ que debes utilizar.

cd projects

¡Buen trabajo!

# Crear archivos y carpetas

Bien, ahora que ya sabes cómo navegar por la estructura de archivos mediante la línea de comandos, llegó el momento de aprender cómo realizar algunas tareas simples del sistema de archivos con ella.

💡 En las próximas lecciones, vamos a trabajar con una carpeta de muestra llamada sample-project. Esta carpeta es sólo para fines ilustrativos. Te recomendamos que vayas siguiendo estas lecciones al tiempo que ejecutas los comandos en tu computadora. Sin embargo, recuerda que no necesitarás ninguno de los archivos ni carpetas de este capítulo en tus proyectos futuros.

En este video, te mostraremos cómo crear y eliminar archivos o carpetas con la línea de comandos.

## Crear directorios con mkdir

Con el comando mkdir (se pronuncia "méik-dir"), podemos crear una nueva carpeta justo desde dentro de la terminal. Vamos a crear una carpeta llamada projects dentro de tu directorio de usuario. Para ello, primero asegúrate de estar en tu directorio de usuario ejecutando cd ~. Después utiliza el comando mkdir para crear un nuevo directorio:

*# ve a tu directorio de usuario*

cd ~

*# mkdir seguido del nombre de directorio deseado (projects) crea un directorio*

mkdir projects

Después, vamos a ingresar cd para entrar a nuestro directorio recién creado, y luego crearemos otra carpeta dentro de esta a la que llamaremos sample-project. Y entonces, utilizaremos nuevamente cd para entrar en ella.

*# entra a la carpeta projects*

cd projects

*# comprueba dónde estamos*

pwd

*# salida*

/Users/YOUR\_USERNAME/projects

*# crea una nueva carpeta*

mkdir sample-project

*# utiliza cd para entrar en esa carpeta*

cd sample-project

*# comprueba dónde estamos*

pwd

*# salida*

/Users/YOUR\_USERNAME/projects/sample-project

## Crear archivos con touch

A continuación, vamos a crear algunos de los archivos que necesitaremos. Hay algunas maneras en que podemos crear un nuevo archivo en la línea de comandos. La primera es el comando touch.

*# este comando creará un nuevo archivo llamando a index.html en el*

*# directorio actual de trabajo, si aún no existe.*

touch index.html

Ahora, usa el comando touch para crear un archivo llamado style.css en el mismo directorio.

touch style.css

En este punto, deberías tener una estructura de archivos que puede visualizarse así:

~/

projects/

sample-project

index.html

style.css

En otras palabras, tu directorio de inicio ahora contiene un directorio projects con un directorio sample-project dentro de él. Dentro del directorio sample-project, debe haber dos archivos: index.html y style.css.

## Trabajar con rutas de archivo en la línea de comandos

Tanto touch como mkdir te permiten crear archivos en cualquier lugar, no sólo en el directorio actual de trabajo. Todo lo que debes hacer es proporcionar una ruta relativa o absoluta en la que deseas crearlo. Así que, en realidad no teníamos que utilizar el comando cd en nuestro ejemplo anterior. En cambio, pudimos haber hecho esto:

*# ir al directorio de inicio*

cd ~

*# crear un directorio en el directorio actual de trabajo*

mkdir projects

*# crear un directorio dentro del directorio projects*

mkdir projects/sample-project

*# crear un archivo dentro de projects/sample-project*

touch projects/sample-project/index.html

Ten en cuenta que todos estos comandos sólo funcionarán si existen todas las carpetas intermedias. Por ejemplo, ejecutar mkdir projects/sample-project fallaría si no hay un directorio projects en el directorio actual de trabajo. Pero, si aún no existen las carpetas, puedes indicarle a mkdir que las cree todas en la ruta de archivos a través del flag -p.

*# con el flag -p, se creará el directorio projects si este aún no existe*

mkdir -p projects/sample-project

Pregunta

1. ¿Qué condiciones deben cumplirse antes de crear un nuevo directorio mediante la línea de comandos?

Piensa en un nombre adecuado para tu directorio.

¡Correcto! Un directorio debe tener un nombre que describa su propósito.

Navega hacia una carpeta padre de tu nuevo directorio.

Poder hacer clic derecho para que aparezca el menú contextual.

¡Perfecto!

Pregunta

2. Escribe un comando para crear un directorio llamado **my-folder** en el directorio actual de trabajo.

¡Correcto!

¡Lo has entendido bien!

Pregunta

3. Ahora, escribe un comando para crear un archivo **my-file** dentro del directorio **my-folder**. No uses **cd** para entrar primero en el directorio **my-folder**.

¡Correcto!

¡Buen trabajo!