

Temporada 01: iniciándose en la terminal

Por [Diego Vizcarra](http://codingornot.com/author/d9ziv)

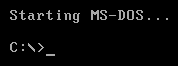
Ya sea que inicies desde cero o simplemente desees reafirmar tus conocimientos, en esta temporada te hablamos sobre los conceptos y comandos básicos que habría de conocer al momento de utilizar la terminal.

* [01. Terminal: “Hey, stranger…” – Conociendo la terminal](http://codingornot.com/conociendo-la-terminal)
* [02. Terminal: “It’s dangerous to go alone!” – Navegar por el sistema](http://codingornot.com/navegar-por-el-sistema/)
* [03. Terminal: creación de directorios](http://codingornot.com/creacion-de-directorios/)
* [04. Terminal: manipulando archivos y directorios](http://codingornot.com/manipulando-archivos-y-directorios/)
* [05. Terminal: tomando algunos atajos (shortcuts)](http://codingornot.com/tomando-algunos-atajos-shortcuts/)
* [06. Terminal: una visita al comando “history”](http://codingornot.com/06-terminal-una-visita-al-comando-history/)
* [07. Terminal: accediendo al interior de un archivo de texto](http://codingornot.com/07-terminal-accediendo-al-interior-de-un-archivo-de-texto/)
* [08. Terminal: comandos en el ambiente Windows](http://codingornot.com/08-terminal-comandos-en-el-ambiente-windows/)
* [09. Terminal: revisando el manual de usuario](http://codingornot.com/09-terminal-revisando-el-manual-de-usuario/)

01. Terminal: “Hey, stranger…” – Conociendo la terminal

por [Diego Vizcarra](http://codingornot.com/author/d9ziv)

Aún recuerdo cuando en casa tuvimos nuestra primer computadora. En aquellos días era sumamente alentador que, al encender el equipo, apareciera algo como esto:



Ammm… ¿qué?”

“¿Y eso qué hace?”

Aunque no lo podía decir con certeza no todo estaba perdido, el equipo venía con un conjunto de instrucciones: un post-it donde figuraban ciertas palabras que habría que teclear para poder realizar cosas como cambiar de carpeta o crear un nuevo archivo de texto.

Todo hasta ahí bien… aunque debo confesar que para mi, siendo aún niño, lo más divertido era poder jugar Pac-Man y olvidarme de esas líneas “sin sentido”.



Qué buenos recuerdos

Pasado el tiempo conocí al nuevo Windows, bienvenida seas interfaz gráfica, y por fin pude decirle adiós a ese extraño fondo negro nada llamativo. Quién necesitaba ya el teclado, más allá de escribir, si podías navegar por el sistema a base de un clic (o dos).

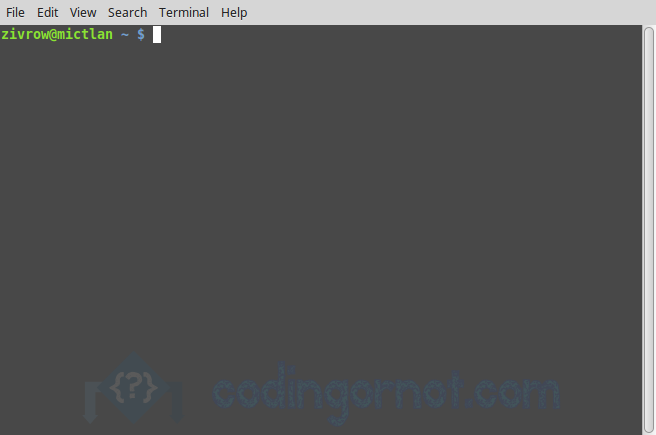
No fue sino hasta muchos, muchos años después que me di cuenta de lo que me estaba perdiendo. Uno de los instructores en un pequeño curso sobre cierto lenguaje de programación nos mostraba cómo crear un nuevo archivo para el código fuente; fue en ese instante donde quedé asombrado al ver cómo creaba dicho archivo, le cambiaba el nombre, creaba nuevas carpetas y accedía a otras partes del sistema sin siquiera tocar el mouse.

Espera… ¿no acabas de mencionar que todo fue mejor gracias a la interfaz gráfica y el mouse?  
Sí, pero ¡no había visto lo lento que era usarlos!

La velocidad con la que el instructor realizaba todas esas acciones superaba mi pericia donde tenía que buscar un editor de texto, abrirlo, escribir el programa, ir al escritorio, crear una carpeta dando clic derecho; abrir un IDE, compilar el programa… Todo esto tomaba poco tiempo pero, en estos momentos, estaba observando cómo podía ser realizado más ágilmente con unas simples líneas dentro de la, antes odiada e incomprendida, terminal.

Basta de historias. Dime, ¿qué hago?

Asumiendo que mi experiencia haya servido para aumentar el interés en ti, estimado lector, de conocer qué se esconde tras esa ventana aparentemente poco amigable, también conocida como consola, vamos conociendo cómo se ve.



Aquí tomaremos como ejemplo la terminal de un ambiente Linux donde podemos apreciar que aparece el llamado prompt, es decir, información útil que nos dice, por mencionar algo, quiénes somos o dónde nos encontramos. Este prompt aparecerá siempre en cada nueva sesión que iniciemos en alguna ventana de consola y cada vez que queramos escribir un nuevo comando.

En este caso en particular se nos informa que nuestro usuario es zivrow, que estamos ubicados en la máquina con nombre mictlan y que nos encontramos dentro de nuestra carpeta personal (o home) representada por la virgulilla ~.

Si se preguntan qué significa ese signo $ que aparece al final del prompt, no es más que una sutil forma que tiene la consola para decirnos que espera un comando. Es importante saber reconocer este signo ya que aparece en multitud de lugares donde piden que ejecutemos ciertas instrucciones en nuestra terminal. Para un ejemplo de esto, pueden visitar la página de [Homebrew](http://brew.sh/).

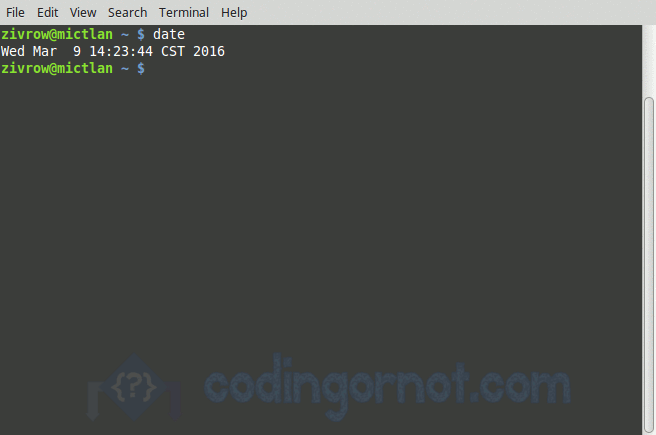
Ahora supongamos que queremos conocer qué día y qué hora es en estos momentos. Para realizar esta sencilla tarea basta con decirle amablemente a la Terminal:

$ dime, por favor, la fecha y hora actuales



Vaya… tal parece que no entendió lo que tratamos de decir y nos lo marcó con un pequeño error: command not found. No es que hayamos estado faltos de cortesía sino que, la verdad sea dicha, no basta con pedir amablemente las cosas, también hay que hablar en el mismo idioma. Veamos qué sucede si escribimos lo siguiente:

$ date



Mucho mejor. Ahora nos muestra el día, mes y hora actual ¿Lo mejor de todo? Solo tuvimos que escribir una sencilla palabra que además es fácil de recordar.

Para terminar

Conforme vayamos avanzando en el conocimiento de esta gran herramienta, nos iremos dando cuenta que nos desenvolvemos con mayor facilidad en nuestra área de trabajo y, asimismo, podemos conocer qué es lo que está haciendo realmente nuestra computadora para realizar lo que le pedimos. Contar con esta clase de conocimiento nos otorga un mayor dominio sobre lo que estamos realizando y nos ayuda a encontrar diversas y mejores formas de lograr nuestros objetivos.

Antes de poder realizar cualquier otra acción, es preciso que seamos capaces de movernos libremente por nuestros documentos; por tal razón, en la próxima nota trataremos exactamente cómo navegar en cortos y sencillos pasos a través de nuestro sistema.

02. Terminal: “It’s dangerous to go alone!” – Navegar por el sistema

por [Diego Vizcarra](http://codingornot.com/author/d9ziv)

En el título nos encontramos con una famosa frase que algunos recordarán haber leído en cierto videojuego. Me pareció adecuada usarla porque es este conjunto de palabras los que precedían al ofrecimiento de una “herramienta” que nos ayudaría a abrir el camino hacia un mundo lleno de obstáculos y de varias posibilidades.

Muy bien, hemos abierto nuestra aplicación para escribir en consola y ésta espera a que le demos instrucciones. ¿Qué es lo primero que deberíamos conocer antes que cualquier otra cosa?  
Como habíamos visto [anteriormente](http://codingornot.com/conociendo-la-terminal/), el prompt nos muestra información útil como quiénes somos y en qué parte del sistema nos encontramos. Suponiendo que la consola fuera un poco desconsiderada y no nos mostrara absolutamente nada de información, no cabe duda que estaríamos totalmente perdidos.

Si no conoces el lugar exacto en el que estás parado, qué tan seguro podrías estar de realizar alguna función; dónde podrías encontrar la carpeta que acabas de crear si te decidieras ir a buscarla a través de la interfaz gráfica; cómo sabrías a cuántas carpetas ingresar o de cuántas regresar para llegar a cierto documento. No hay ningún problema, existe un conjunto de comandos que te permitirán conocer dónde te encuentras, cómo moverte y qué cosas hay en el lugar.

Disculpe, ¿sabe dónde queda…?

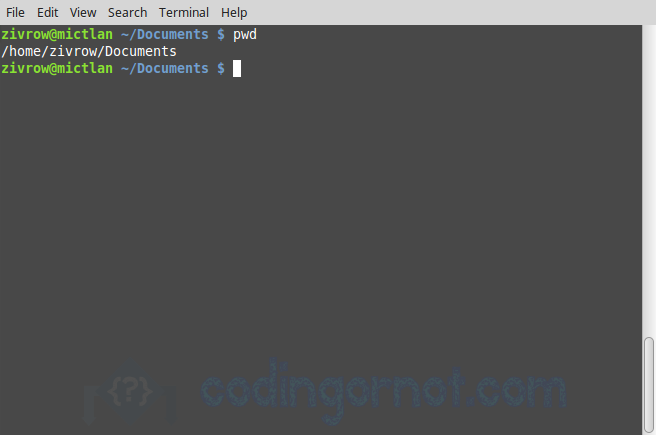
Suena lógico que si te has visto perdido en alguna ciudad lo primero que hagas es preguntarle a alguien sobre el área en la que te encuentras, situación no muy diferente a lo que ocurre en la terminal. Aquí el buen samaritano que te librará de tu confusión es un sencillo comando conocido como:

1. $ pwd

Puede que en estos momentos estés comenzando a pensar que vendrán nombres raros que seguramente olvidarás pronto y que, tal vez, terminarás por buscar de nuevo su significado ya sea a través de una búsqueda rápida en Google o visitando de nuevo esta nota. Nada más lejos de la realidad.

La tarea de la terminal no es complicarte la vida sino facilitarla, por eso, pwd no es más que una pequeña contracción de print working directory. Ahora no es tan difícil de recordar, ¿cierto?

Este comando hace exactamente lo que menciona su nombre, nos imprime la ruta en la que estamos situados; he aquí un pequeño ejemplo:



Después de haber ejecutado el comando se nos otorga una ruta completa que es muy fácil de leer: partiendo de la carpeta raíz del sistema /, has pasado por la carpeta principal home, continuaste por la carpeta de usuario zivrow para, finalmente, entrar a una carpeta llamada Documents.

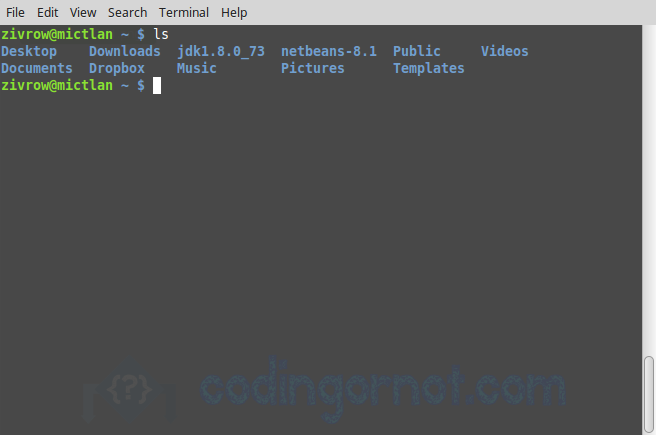
A decir verdad, la parte de zivrow (en este caso) hace referencia a nuestro directorio personal que es donde prácticamente realizamos todo lo de nuestro día a día, aquí es donde se encuentran los omnipresentes Documentos, Imágenes, Música, etc. Normalmente tu directorio personal llevará por nombre tu nombre de usuario y estará dentro del directorio (o directorios) que el sistema decida. Como vimos en la primera nota, el símbolo ~ es un “apodo” y hace referencia a toda la ruta desde la carpeta raíz hasta tu carpeta personal, en la imagen podemos darnos cuenta de esto ya que en el resultado del comando pwd aparece /home/zivrow/Documents y en el prompt únicamente aparece ~/Documents.

Todo hasta aquí muy bien pero, volviendo al papel de turista, después de llegar a cierto lugar deseas conocer qué hay alrededor y hacia dónde podrás dirigirte. Es aquí donde entra nuestro siguiente comando:

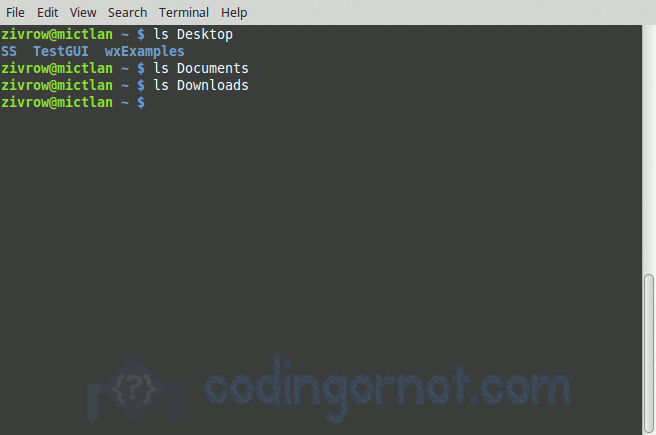
1. $ ls

Guía de turistas

ls es otro mnemónico para un comando cuya función es listar (de aquí el nombre) el contenido de algún directorio o carpeta que quieras, tan fácil como eso. Estando dentro de una carpeta cualquiera puedes ejecutar dicho comando para conocer qué hay dentro. Veamos entonces:

Parece que ls no es un mal guía de turistas. Este comando nos deja ver de forma ordenada (y hasta en orden alfabético) qué carpetas se encuentran dentro del directorio en el que estamos actualmente. Ahora puedes saber qué lugares visitar a partir de tu posición. Aún así, ¿no habíamos dicho que mostraba información sobre cualquier carpeta?

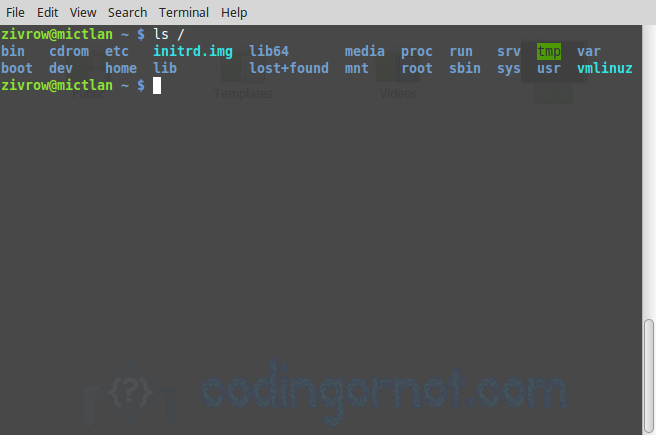
Si utilizamos el comando solo, sin indicarle de qué carpeta mostrarnos información, nos dirá qué hay dentro del directorio actual. Con esto en cuenta, suena lógico que si queremos se nos muestre información sobre cualquier otro lugar, habrá que nombrárselo. Investiguemos qué hay dentro de algunas carpetas que nos acaba de mostrar.

a primera vez mostró resultados y en las dos últimas no ocurrió nada ¿por qué? No es que el comando haya dejado de funcionar, simplemente me está diciendo que efectivamente no hay nada dentro de esas carpetas. Igual podríamos haber mencionado las tres carpetas así:

1. $ ls Desktop Documents Downloads

Y nos mostraría el contenido de las tres dentro de un mismo resultado. ¿Recuerdas cuál era el símbolo de la carpeta raíz? Veamos qué nos muestra si le decimos nos despliegue el contenido de ese directorio.

1. $ ls /

Ahora que ya sabes dónde estás y qué lugares de interés hay alrededor, necesitas ir directamente a conocer esos lugares. El siguiente comando te transportará hacia donde tú le indiques, solo necesita una única cosa: saber el nombre del sitio que deseas visitar.

¡Taxi!

Varios son los lugares lo suficientemente placenteros como para disfrutarlos con una agradable caminata mientras se llega la hora de comer o de descansar. Llegado tal momento, nos damos cuenta que ese buen restaurante que nos recomendaron o el hotel donde nos alojamos queda retirado de donde estamos actualmente. Sería sumamente genial que alguien se ofreciera a llevarnos con solo pedírselo ¿no es así?

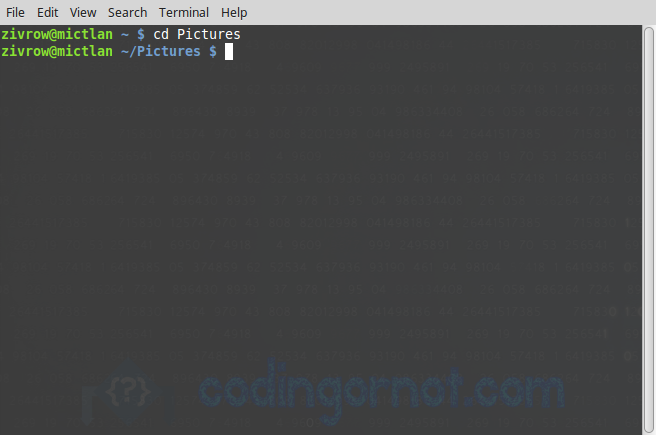
Te presento a:

1. $ cd

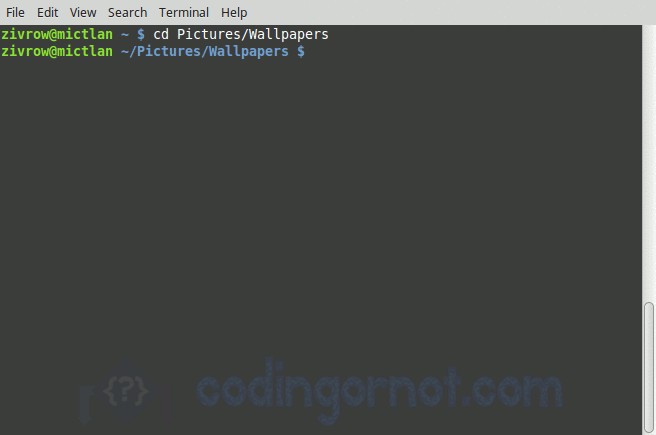
Este comando satisface la necesidad de transportarnos hacia una dirección en particular y, además, de forma totalmente gratuita. El significado de este comando es sumamente predecible; navegar por el sistema no es más que realizar cambios entre diversos directorios, por tanto, podrías decirme ¿qué quieren decir estas dos letras?

cd significa change directory. Este comando, como habíamos mencionado, solo necesita que le digas hacia dónde quieres ir. Imaginemos que deseas moverte hacia la carpeta de imágenes, digámosle entonces a cd el nombre del directorio.

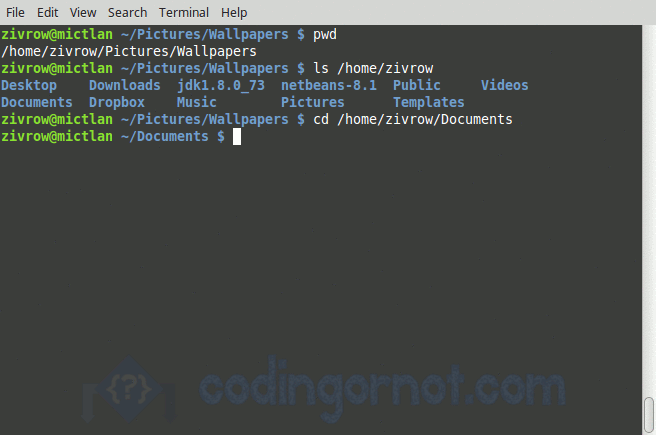
1. $ cd Pictures

¡Servido, señor!  
Puedes darte cuenta cómo Pictures se encontraba dentro de la carpeta principal (que es donde estábamos anteriormente) y únicamente bastó con agregarle al comando ese nombre. Ya sea por las letras azules, como aparecen en la imagen, o si usas el comando pwd, verás que efectivamente estás justo donde lo pediste.

Si, por ejemplo, fueras consciente que Pictures alberga otra carpeta y quisieras moverte hacia ella directamente, habría que indicar la ruta a seguir a partir de la que nos encontramos en este momento (este tipo de rutas se conocen como rutas relativas). Lo único que te queda entonces por hacer es decirle a cd que se mueva a Pictures y, de ahí, vaya hacia la otra carpeta que alberga.

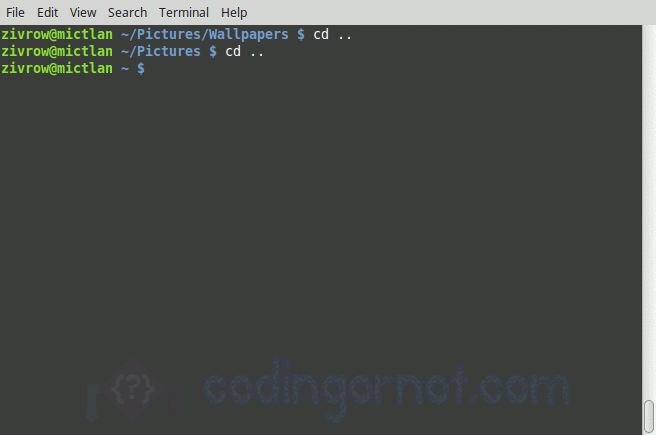
Si lo que quisieras es moverte hacia un directorio que sabes no se encuentra ni en tu posición actual ni en ninguna otra carpeta interior, basta con indicar la dirección completa del directorio comenzando desde la carpeta raíz. Al ver el comando pwd vimos que nos muestra esta dirección completa, conocida como ruta absoluta.

Para realizar el siguiente ejercicio nos ayudaremos de los 3 comandos que hemos visto hasta el momento y notarás cómo resalta la importancia de cada uno para que logremos nuestra meta: Llegar a la carpeta de Documents.

Comencemos a examinar lo que se acaba de hacer:

1. Estando en la carpeta Wallpapers utilizamos el comando pwd para conocer la dirección exacta de donde estamos, o en otras palabras, conocer la ruta absoluta.
2. Tenemos la idea que Documents se encuentra dentro de la misma carpeta donde reside Pictures, es decir, en zivrow. Usando el comando ls podemos confirmar que esta suposición es cierta.
3. Sabiendo ya dónde se encuentra el directorio que queremos y su dirección (ruta), solo queda decirle a cd hacia dónde queremos ir.

Llegado el momento puede que desees volver al directorio padre, que contiene al directorio donde estás, conocido como directorio hijo. Para hacerlo, existe un alias en particular para referirnos a ese directorio padre, solo hay que escribir punto punto.

En el GIF de arriba se aprecia como, en un primer intento, nos movimos hacia Pictures, directorio padre de Wallpapers, y luego a la carpeta principal ~, directorio padre de Pictures.

Para terminar

Los comandos que acabas de ver forman la base de todo lo que iremos conociendo de ahora en adelante. Es gracias a estas sencillas instrucciones que podrás entender con mayor seguridad qué es lo que realmente sucede cuando, por ejemplo, se da un doble clic en alguna carpeta para abrirla.

Dominar estos comandos lleva un poco de práctica pero no es razón alguna para desanimarse, verás como con el tiempo podrás moverte libremente por las diferentes partes de tu sistema y pedir información a la consola de manera casi automática.

No olvides que si tienes alguna duda con lo que se ha expuesto puedes expresarla libremente dejando un comentario. Estamos para ayudarte.

En notas posteriores aprenderás cómo crear, mover e incluso eliminar carpetas, otras partes importante en el uso del terminal. Hasta entonces.

03. Terminal: creación de directorios

Por [Diego Vizcarra](http://codingornot.com/author/d9ziv)

Qué bien saber de ti… En esta ocasión, trataremos el tema de los directorios: ¿Qué son? ¿De dónde vienen? ¿Acaso el nombre “nueva carpeta” guarda un mensaje oculto?

Como bien sabrás, ha habido momentos en nuestra vida donde llegamos a tener cientos y cientos de papeles, algunos de relevante importancia y otros de relevantes ganas de solo ocupar un lugar en el espacio. ¿Qué hacer con tanto papel? Bueno, si no tenemos tiempo de ordenarlo (o ganas) simplemente lo juntamos todo y lo hacemos a un lado para que ya no nos moleste.

Todo sería perfecto si tan solo no volviéramos a necesitar de esos documentos pero sabemos que no será así. Llegará el momento donde, por una razón o por otra, tengamos que enfrentar esa gran pila de celulosa y en nuestra mente aparecerá una única y existencial pregunta: ¿Por qué?

Afortunadamente existen objetos que pueden ayudarnos a cambiar nuestra situación y a mantener nuestros documentos de forma organizada (si así lo deseamos): Las carpetas.

En el mundo de la informática existe un análogo a estos almacenes de documentos importantes (o no tan importantes) y que son mejor conocidos como directorios. Estos contenedores virtuales existen con la intención de guardar archivos, o más directorios, atendiendo a la idea de que aquello que reunamos tenga cierta relación; aunque igual existe la opción de que podamos crear un bonito desastre.

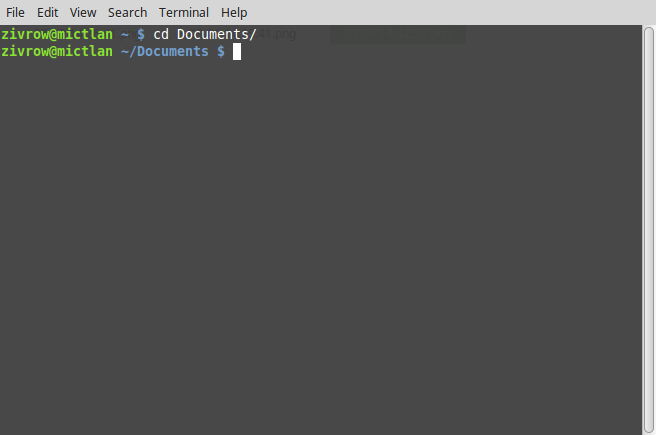
Dada esta pequeña introducción veamos, ya de una vez, cómo es que se crea un nuevo directorio utilizando solo la terminal.

Un directorio para aquí, por favor

Por sorprendente que pueda parecer, nuestra computadora cuenta con un gran número de directorios o carpetas (ambas palabras son tratadas como sinónimos), de los cuales, el más importante (y que ya hemos mencionado) es… El directorio raíz. En este directorio se encuentran almacenados todos y cada uno de los archivos de nuestro sistema, desde aquellos muy importantes como los propios archivos del sistema operativo, hasta aquellos que nosotros mismos creamos al trabajar en nuestra máquina.

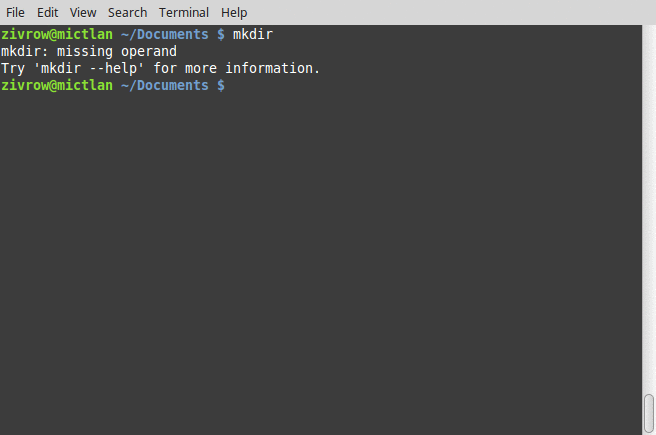
Para crear nuestro primer directorio artesanal necesitamos ubicarnos en el lugar donde queremos que aparezca. Básicamente podría ser en cualquier parte del sistema pero para esta ocasión, nos trasladaremos a la carpeta de Documents. ¿Recuerdas como cambiar de directorio?

1. $ cd Documents/

Bien, ahora hagamos que una nueva carpeta vea la luz de este mundo. Recuerda que la terminal siempre está a la espera de que le otorguemos una instrucción y, como nosotros queremos decirle que haga una nueva carpeta, tenemos que darle exactamente esa orden y en su lenguaje. Coloca en la terminal:

1. $ mkdir

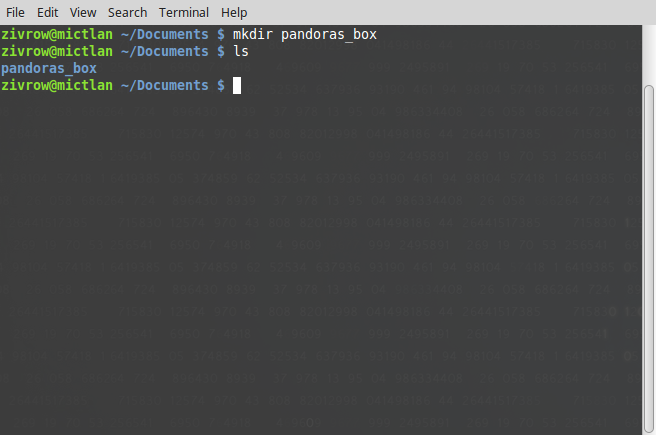
Este mnemónico bastante sencillo guarda el significado de make directories. ¿Es esto suficiente para que la terminal nos cree un nuevo directorio? Veamos…

Parece que no. La terminal nos indica por medio de un mensaje que no es suficiente con introducir el nombre del comando, le hace falta una parte pero ¿cuál podría ser?

Normalmente cuando creamos una nueva carpeta utilizando el mouse, o alguna función dentro del famoso menú Archivo, esta lleva un nombre por defecto como —¡sorpresa!— “nueva carpeta”; aún así, la idea es que nosotros quitemos ese nombre y le demos uno que corresponda al propósito por el que la creamos. La terminal no desea que tengamos decenas de nombres genéricos y por eso, desde el inicio, nos pide un nombre al momento de pedirle crear un nuevo directorio.

¿Cuántas veces hemos tenido la oportunidad de bautizar a una carpeta recién nacida? Puede que muchas, pero ahora lo haremos usando nuestro nuevo comando. Ingresa de nuevo el comando y enseguida coloquemos un nombre para nuestra carpeta:

1. $ mkdir pandoras\_box

Como puedes apreciar en la imagen, se ha creado una carpeta nueva con el nombre que hemos indicado en la instrucción y gracias al comando ls reafirmamos que tal acción ha sido un éxito. Podrías preguntarte ¿por qué incluir un guión bajo en el nombre? Te contesto enseguida:

Resulta que un comando, como mkdir, puede recibir no solo uno sino varios valores a la vez, cada uno de los distintos valores tendrán que ir separados por un espacio para que el comando sea capaz de distinguir un valor de otro. En el caso de mkdir, dependiendo el número de valores que le pasemos, será el mismo número de carpetas que nos creará.

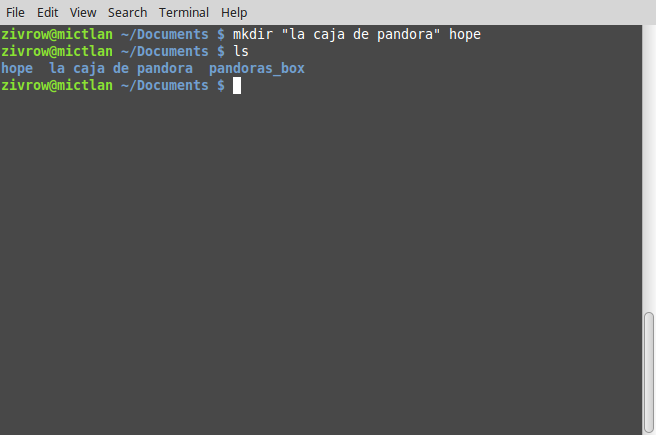
¿Qué crees que suceda si escribimos la siguiente instrucción?

1. $ mkdir la caja de pandora

Quisiera pensar que el comando sería lo bastante inteligente para saber que mi intención es crear una carpeta con el nombre la caja de pandora pero no es así, lo que hará es crear cuatro carpetas distintas cuyos nombres serán la, caja, de y pandora (¿!Pero qué rayos!?).

“No os preocupéis”. Lo que acabamos de ver no quiere decir que tengamos que vivir en un mundo con carpetas que incluyan guiones bajos en sus nombres en lugar de espacios (aunque nada nos impide que lo hagamos), lo único que se nos pedirá de extra es que le indiquemos explícitamente al comando que nuestro nombre con espacios forman una y solo una entidad; para hacer lo anterior basta con encerrar el nombre entre comillas dobles y ya está:

1. $ mkdir "la caja de pandora" hope

Un nombre entre comillas como el que acabamos de utilizar no es más que un valor más, podemos agregar varios o ninguno de estos o también combinarlos con las palabras simples que ya hemos estado utilizando, como hope que aparece en el ejemplo, para crear las carpetas que queramos con los nombres que queramos.

Para terminar

La creación de directorios es el primer paso que daremos en esta ocasión dentro del área de los directorios. Ya habíamos visto con el comando cd que algunos comandos requieren de algún valor extra para que puedan trabajar correctamente, este tipo de valores se suelen conocer como parámetros, y dependiendo del comando que utilicemos, se necesitará agregar cierto número de ellos y cada uno tendrá un significado especial dentro de la instrucción.

Continuando con esta serie, le seguiremos la pista a tres nuevos comandos que trabajan con directorios (al igual que con archivos) y que nos ayudarán a comprender mejor todo esto que ahora denominamos parámetros. Recuerda que si tienes dudas puedes expresarlas en los comentarios y que una parte muy importante en esto es que practiques.

04. Terminal: manipulando archivos y directorios

Por [Diego Vizcarra](http://codingornot.com/author/d9ziv)

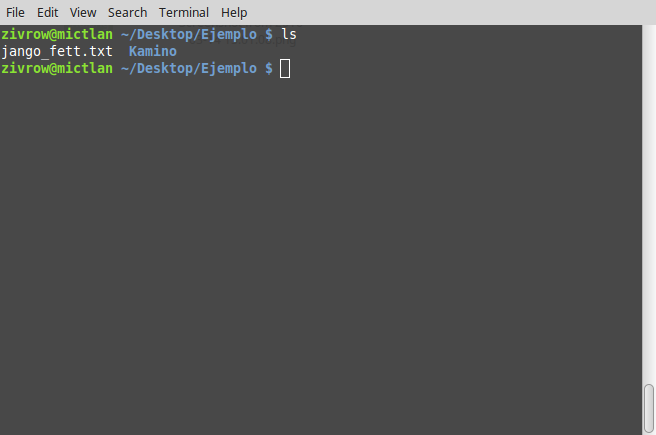
A lo largo de todo este camino has aprendido a obtener información y a crear, por el momento, directorios por medio de la consola. En esta nota, nos apropiaremos de un grupo más de herramientas que justamente nos ayudarán a avanzar en la manipulación de nuestros archivos y nos darán nuestro siguiente pequeño gran paso para convertirnos en un verdadero maestro.

Creando un ejército de clones

Por una razón o por otra, en ocasiones debemos crear una copia a partir de un archivo o un directorio que elijamos. Ya es por todos bien conocido el usar la combinación de teclas Ctrl+C seguida de un Ctrl+V para realizar la tarea en cuestión pero ¿cómo se realiza lo mismo por medio de una instrucción en consola? Para poder darle una respuesta a este cuestionamiento conozcamos el siguiente comando:

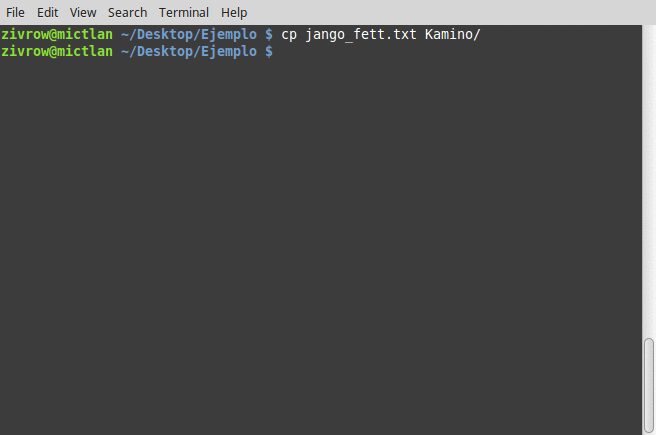
1. $ cp

El nombre de cp le viene perfecto ya que precisamente hace referencia a la palabra copy. Para lograr usar este comando se requiere que le pasemos 2 parámetros: el primero refiriéndose al archivo o directorio que deseamos copiar, y el segundo a la ubicación donde queremos se sitúe esta nueva copia. Veamos un primer ejemplo de cómo copiar un archivo (en este caso un archivo de texto) a otra carpeta:

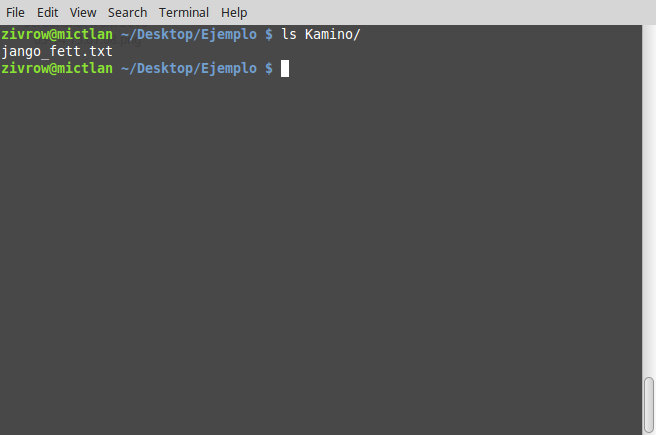


Observando el entorno anterior te darás cuenta de que justo tenemos los materiales que necesitamos para nuestro pequeño ejercicio (¡que conveniente!) vayamos pues a copiar nuestro archivo usando la siguiente instrucción:

1. $ cp jango\_fett.txt Kamino/

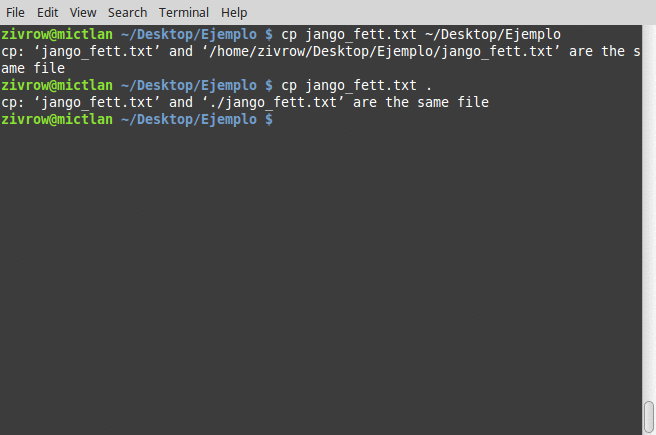


¿Qué ha pasado? ¿Funcionó o no? Veámoslo con ayuda de ls.

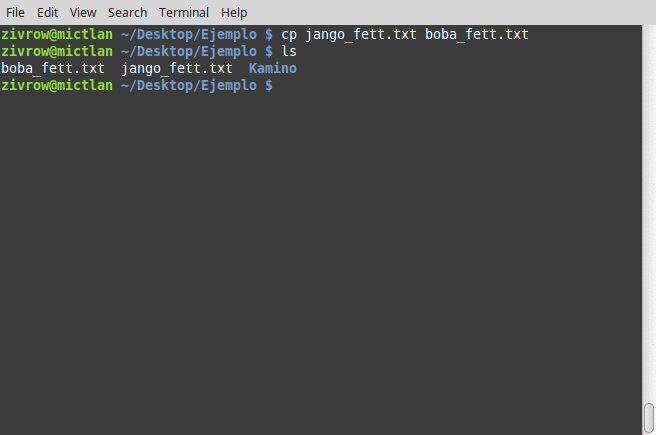


¡Genial, sí que ha funcionado! Podemos ver cómo un clon de nuestro archivo ahora reside en la carpeta Kamino. Podemos usar cualquier carpeta como destino pero habrá que tomar muy en cuenta lo referente a las rutas, ¿recuerdas lo que era una ruta relativa y una ruta absoluta? si no, puedes checarlo de nuevo volviendo a la [nota no. 2](http://codingornot.com/navegar-por-el-sistema/) de esta misma serie.

Ahora imaginemos que no quieres guardar una copia en otro lugar sino que quieres hacerlo justo donde está el original; existen dos formas en particular para hacer mención de la carpeta donde estamos situados actualmente: una es escribiendo la dirección absoluta de la carpeta o, de manera más sencilla, escribiendo un único punto. A diferencia de lo que se había dicho en una [nota previa](http://codingornot.com/creacion-de-directorios/) donde dos puntos seguidos hacían referencia a la carpeta padre, un solo punto hace referencia a la carpeta actual. Veamos ambos métodos y el resultado que se nos muestra

Como puedes ver, el comando no permite crear una copia en el mismo directorio si tiene el mismo nombre que el archivo original (¿cómo los diferenciaría?), en el anterior caso sí que se pudo mantener el mismo nombre porque la ubicación era diferente. Habiendo dicho lo anterior, necesitamos darle otro nombre al archivo para que se pueda copiar en el mismo directorio. Ya que haremos la copia en el mismo lugar, qué te parece si solamente le indicamos el nuevo nombre como destino:

1. $ cp jango\_fett.txt boba\_fett.txt

Como no indicamos nada más que el nombre como segundo parámetro, el comando deduce que queremos hacer la copia en el mismo directorio; si quisieras crear la copia en otro lugar con nombre diferente, basta que coloques ese nuevo nombre inmediatamente después de la ruta de la carpeta.

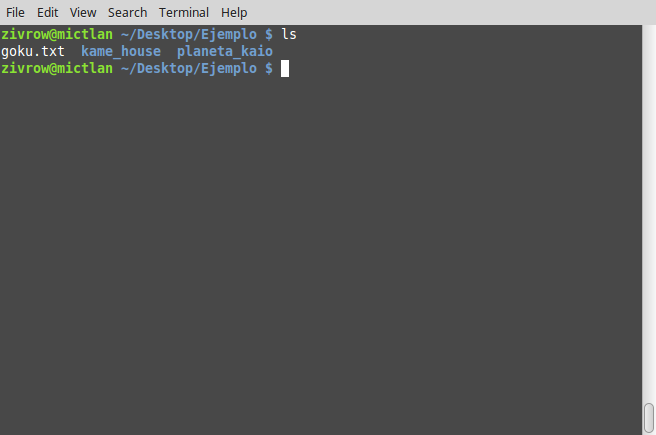
Pasemos ahora a algo parecido pero diferente: mover archivos.

La técnica de teletransportación

Cambiar archivos de lugar es sumamente fácil si utilizamos la interfaz gráfica, solo se necesita arrastrar y soltar el objeto en cuestión ayudándonos del mouse. ¿Qué sucede cuando queramos hacerlo mediante consola? Es igual de sencillo, es más, es prácticamente igual que usar el comando que vimos hace un momento, solo que en lugar de cp usaremos:

1. $ mv

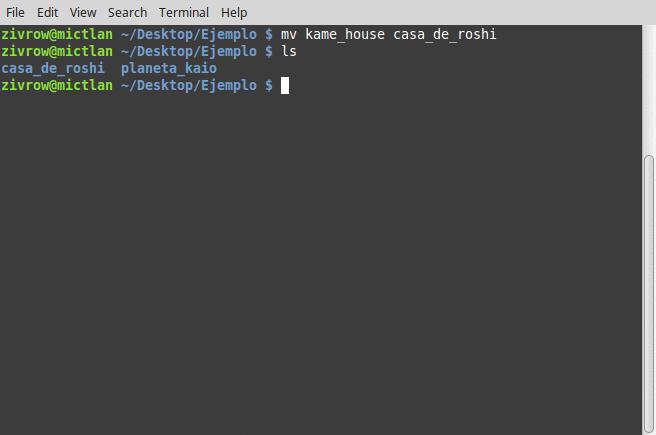
Este comando nos dice exactamente lo que nos imaginamos, move. ¿Mover qué y mover a dónde? Esas son las preguntas que contestaremos igual que la vez pasada usando los mismos dos parámetros, la fuente y el objetivo. Crearemos otro entorno donde podamos practicar esta técnica (comando):

Queremos que el archivo se mueva de su posición actual a la carpeta planeta\_kaio y, de ahí, se mueva a kame\_house. Realicemos entonces lo anterior y veamos los contenidos en cada caso para estar seguros de que nuestro trabajo se realizó de forma adecuada:

Con el GIF anterior es posible ver que nuestro archivo se ha movido justo como lo queríamos. De igual forma, esta ha sido una excelente oportunidad para refrescar los conocimientos que has ido adquiriendo.

El comando mv tiene otro uso común en el ambiente de consola: puede cambiarle el nombre a un archivo. Con cp vimos que si usamos otro nombre diferente al del archivo original para el parámetro destino se creará una copia en el mismo directorio con ese nuevo nombre. Con mv realizamos el mismo procedimiento pero la diferencia será que en lugar de tener una copia, tendremos el mismo archivo con distinto nombre, veamos un ejemplo pero ahora usando un directorio:

1. $ mv kame\_house casa\_de\_roshi

Un gran poder conlleva una gran responsabilidad

ADVERTENCIA:  
El mal uso de lo explicado en la siguiente sección podría ocasionar pérdida de material importante en su sistema ocasionando depresión o hasta pérdida de la conciencia. Se recomienda discreción.

Ok, ok, no era la intención asustarte pero de alguna forma tenía que llamar tu atención para estar alerta. Eliminar archivos haciendo uso de la interfaz gráfica es una labor relativamente segura, siempre aparecerá la, a veces molesta, “ventanita” que nos preguntará si estamos seguros de eliminar aquello que, por error o no, pedimos desaparecer de nuestro disco duro.

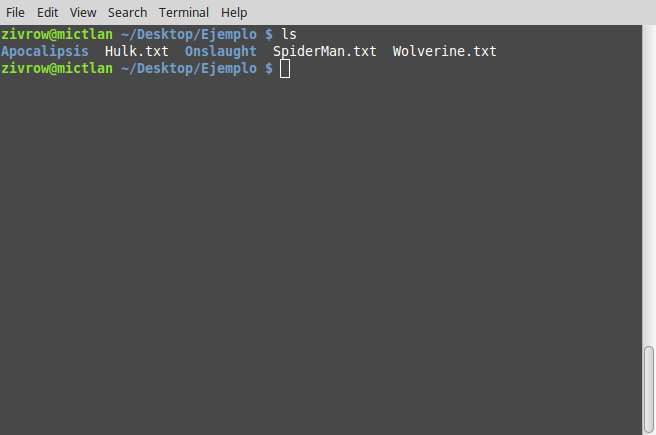
Realizar la misma labor en la terminal requiere pensar en nuestra decisión unas dos o tres veces más (tal vez 10), esto a causa de que no tendremos un mensaje de advertencia y la consola simplemente hará lo que le hayamos pedido explícitamente:

— “¿Eliminar todas las fotos de tus vacaciones? ¡Concedido!”

Esta gran arma que ahora aprenderemos a usar se conoce como:

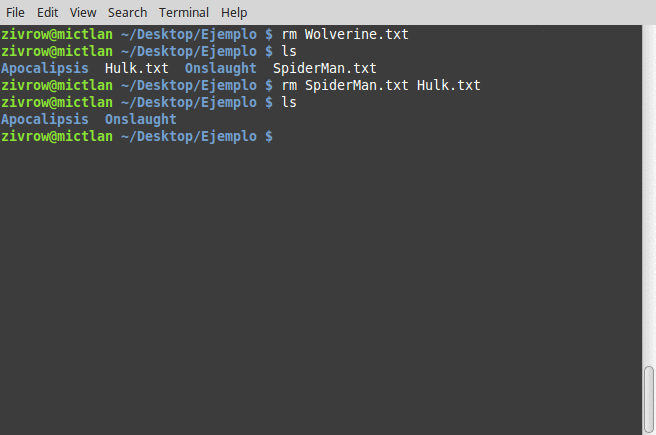
1. $ rm

El comando rm es el alias de remove. Eliminar, borrar, remover… da igual la forma en que lo quieras llamar, este comando eliminará de tu disco (sin preguntar) lo que le pases como parámetro, así que, tómalo con calma antes de utilizarlo. Demos un vistazo a nuestro nuevo escenario para utilizar este nuevo comando:

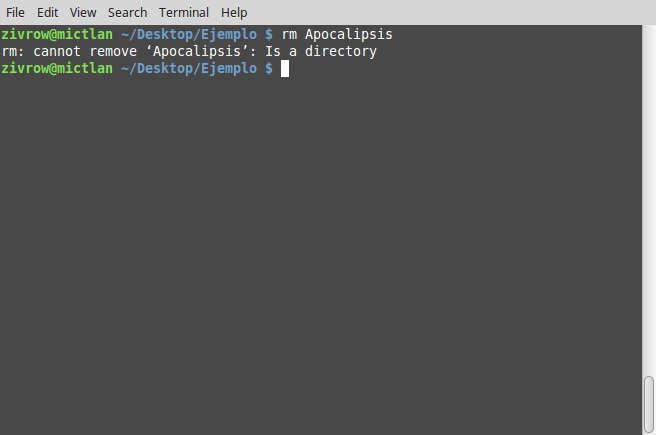


Tenemos una combinación de archivos de texto y directorios que deseamos eliminar de nuestro disco, utilicemos primero nuestro comando con los .txt para ver qué sucede:

1. $ rm Wolverine.txt
2. $ rm SpiderMan.txt Hulk.txt

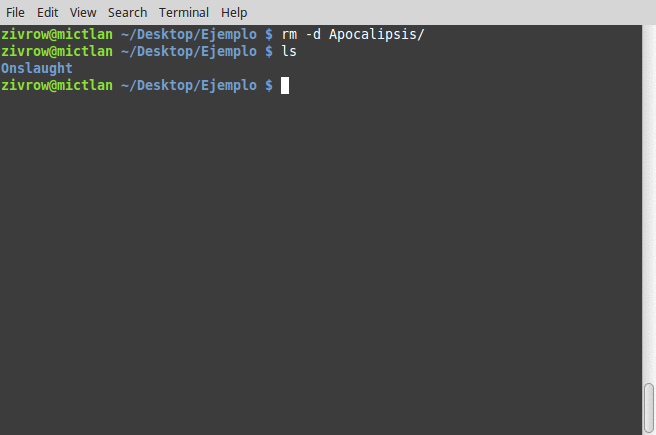
Parece que no ofrecieron la más mínima resistencia. Veamos ahora qué sucede si decidimos utilizar el comando en un directorio como el de Apocalipsis.

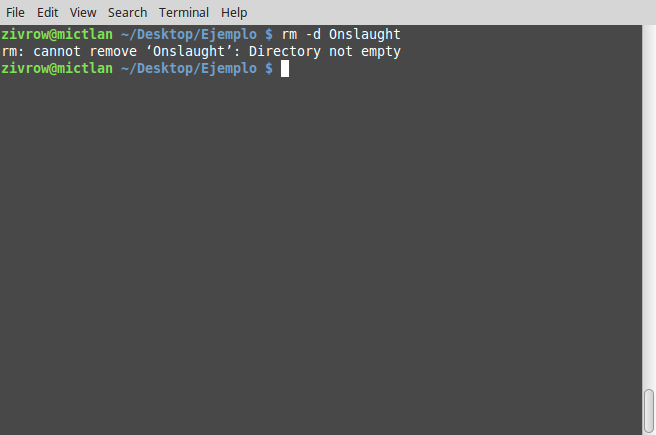
$ rm Apocalipsis



Por defecto rm no elimina directorios tal y como lo hace con un archivo común y corriente, necesita que le indiquemos que aquello a borrar es un directorio (parece que el comando no es tan malo como te hice creer al principio). A fin de realizar esta acción se requiere agregar una parte conocida como opción, y en este caso en particular, necesita que escribamos -d antes de nuestros parámetros.

1. $ rm -d Apocalipsis

Ya sabiendo esto, borremos de una vez el directorio restante:

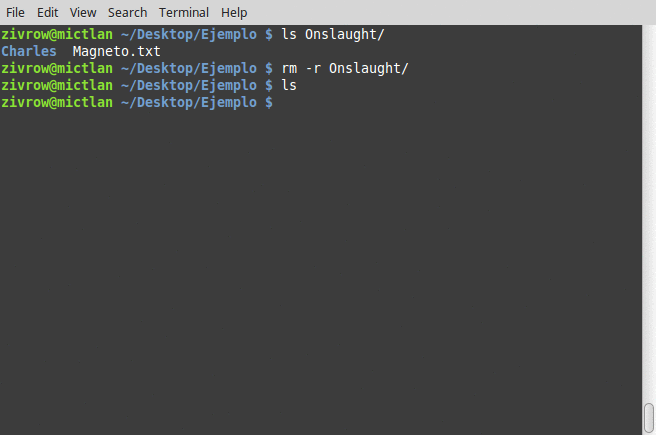


Otra buena nueva de nuestro peligroso comando. El mensaje nos dice que no es posible borrar el directorio porque no está vacío, ¿qué piensas que significa esto con respecto a lo que hicimos poco antes? La opción -d solo funciona si el directorio a borrar es un directorio vacío, en caso contrario, necesitaremos de otra opción que indique que deseamos borrar el directorio y todo lo que haya en su interior.

¡CUIDADO!  
Este último mensaje será de igual manera la última clase de advertencia antes de decirle adiós a un directorio y su contenido. Recuerda que una vez colocada la instrucción y haber presionado ‘Enter’, el comando realizará su función y no habrá marcha atrás, tus archivos serán eliminados definitivamente sin pasar siquiera por la Papelera de Reciclaje.

Teniendo la completa certeza de lo que vamos a hacer debemos agregar la opción -r en nuestra instrucción para que rm borre el directorio y su contenido (un archivo o directorio tras otro). Vamos allá entonces:

1. $ rm -r Onslaught

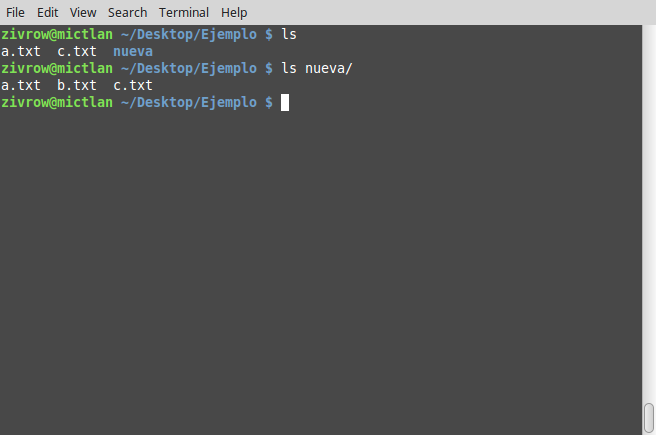
Primero observamos el contenido del directorio Onslaught, nos aseguramos que su contenido no es precisamente algo que queramos conservar y entonces ejecutamos la instrucción. Al final comprobamos que efectivamente todo se ha eliminado sin dejar nada.

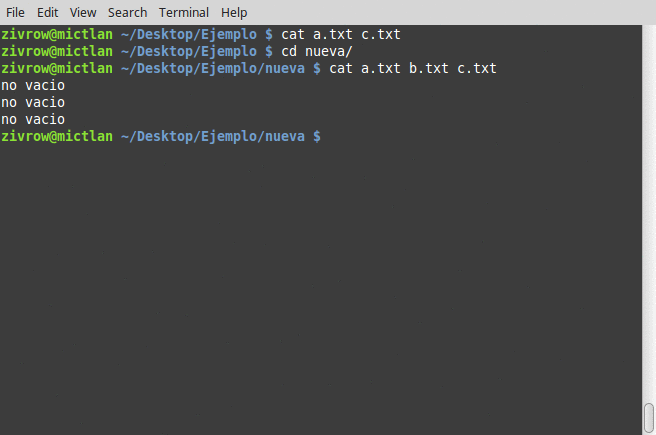
Es muy importante, disculpa que sea tan insistente, que uses este comando con absoluta precaución. Cualquier error podría causar que dañes tu sistema si llegaras a actuar sobre un archivo o directorio sobre el que ignoras totalmente su función y/o contenido. La buena noticia es que ahora tienes un buen conocimiento sobre esto y puedes actuar con seguridad e incluso a ayudar a otros a entender mejor esta herramienta.

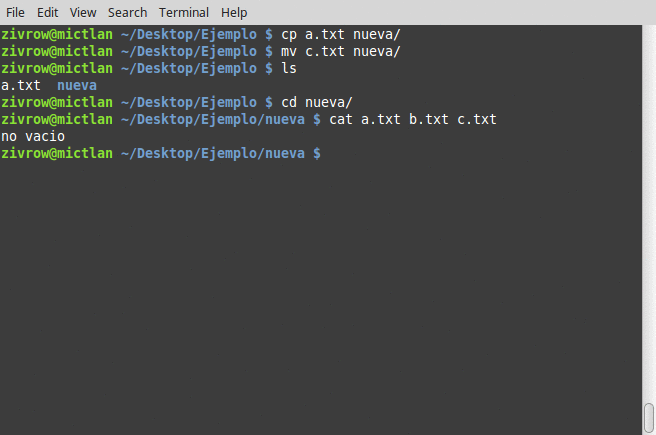
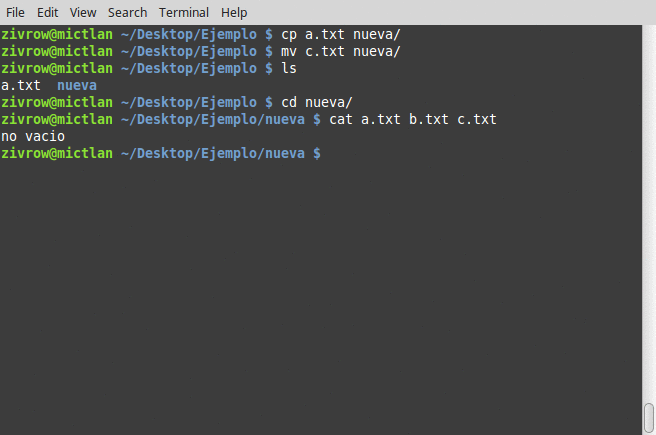
Dr. Jekyll y Mr. Hyde

Ammm ¿no habías dicho que lo anterior era la última parte de la nota? En cierta forma sí, pero hay algo más que debes de saber antes de que terminemos. Hasta ahora hemos visto que el comando rm es uno con el que hay que tener mucho cuidado pero ¿y si te dijera que también deberías tener cuidado con cp y mv?

Tal y como tratamos con este par de comandos en su momento, pareciera que no hay nada que temer con ellos, sus funciones son aparentemente inocentes y además aparecen mensajes de error si hacemos algo mal. La verdad es que si, en cualquier caso, tú decidieras copiar o mover un archivo a una ubicación donde reside otro con el mismo nombre, éste último sería reemplazado por el primero, ¿qué significa esto? Significa que también podrías perder información. Veamos cómo y de paso aprendamos un nuevo comando:

Observa que tenemos un par de archivos con nombres a.txt y c.txt en nuestra carpeta actual, así también, dentro de la carpeta nueva existen dos con el mismo nombre más uno extra llamado b.txt. Averigüemos qué contienen los archivos en cada una de los directorios usando el comando cat para que se muestren en pantalla:

En el primer intento vemos que no se mostró nada por la razón de que los archivos están vacíos, en el segundo sí aparece algo y de hecho el comando cat concatena estos datos y los muestra de una sola vez en pantalla. Sabiendo que los archivos de la carpeta nueva sí contienen algo, investiguemos qué sucede si movemos o copiamos uno con el mismo nombre desde otra carpeta a esta:

Lo que nos temíamos… Las operaciones han sustituido a los archivos que residían en carpeta y, por tanto, solo uno de ellos (b.txt) ha conservado su contenido ya que no se realizó ningún operación sobre él. Por lo que hemos aprendido, es importante que seas cuidadoso al usar cualquiera de estos tres comandos.

Para terminar

La nota en esta ocasión ha sido algo larga, puede que incluso hayas saltado directamente a esta parte para ver si puedes continuar sin saber esto pero créeme, el tiempo invertido en comprender cómo estas herramientas funcionan es de gran valor. Este tipo de herramientas, sin el debido conocimiento, pueden ocasionar problemas donde solo una copia de seguridad podría salvarnos, por eso, si deseas practicar con ellas, crea un entorno propio como los que has visto para que no tengas nada de qué preocuparte.

Has logrado un buen avance en esta serie sobre la terminal, aún así, puede que llegados a este punto te esté inquietando el escribir tantas cosas y hasta te resulte cansado presionar tantas teclas, no pasa nada, en la siguiente nota hablaré sobre cómo ser más ágiles al momento de introducir comandos y aprenderemos un poco más de ellos de forma más genera

05. Terminal: tomando algunos atajos (shortcuts)

Por [Diego Vizcarra](http://codingornot.com/author/d9ziv)

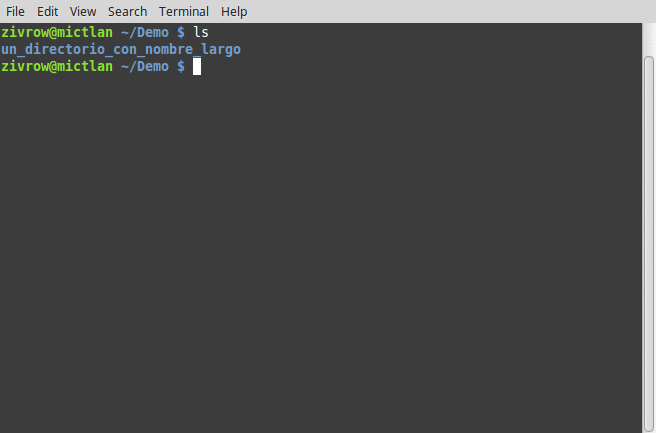
En la [nota anterior](http://codingornot.com/manipulando-archivos-y-directorios/) aprendimos a realizar acciones tan elementales como copiar, mover o eliminar directorios y archivos. Seguro durante los distintos ejemplos pudiste notar que el hecho de introducir algunas instrucciones por medio del teclado daba la sensación de tomar más tiempo del debido. ¿Quién dijo que escribir mv un\_archivo\_con\_nombre\_largo.txt ~/carpeta\_destino/ era más rápido que simplemente tomar el archivo con el mouse y arrastrarlo hacia el escritorio?

No desesperes, en esta nota hablaremos precisamente de cómo agilizar nuestra interacción con la terminal. Con ayuda de unos cuantos atajos verás que el introducir comandos es igual o más rápido que realizar acciones con el mouse.

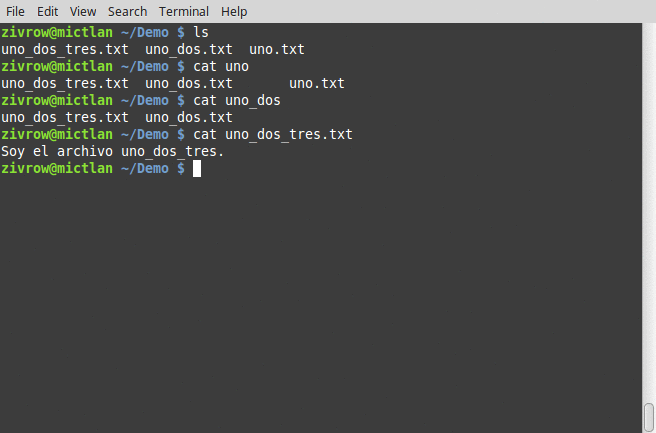
Mejor… que lo haga tabulador

Como veníamos comentando, usar la terminal es básicamente escribir comandos o nombres de archivos. ¿Qué sucede cuando estos nombres son muy largos? ¿Qué pasa si se llegaran a escribir incorrectamente? Las respuestas que vendrán primero a nuestra cabeza son: pérdida de tiempo y/o la aparición de algún error (esto último en el mejor de los casos).

No hay problema, nuestro mismo teclado contiene una pieza clave que nos será de gran ayuda: la tecla tabulador. Esta tecla se encargará de completar, de manera automática, el nombre del archivo o del comando que queramos usar. Veamos un ejemplo:

Como puedes ver, solo fue necesario escribir el principio del nombre de la carpeta para, inmediatamente después, presionar tabulador y que la terminal agregara el resto del nombre por nosotros. Fácil, ¿no?

Aunque lo anterior representa una gran ventaja, hay ocasiones en que tendremos que intervenir para ayudar un poco a la terminal a comprender qué es lo que queremos. Este otro ejemplo ilustrará lo que quiero decir:

Queremos averiguar el contenido del archivo llamado uno\_dos\_tres.txt, vemos que en la carpeta donde nos encontramos existen además dos archivos cuyos nombres empiezan igual que nuestro archivo objetivo, ¿qué sucede en este caso? Solo debemos hacer lo siguiente (toma en cuenta que el símbolo [TAB] hace referencia a presionar la tecla tabulador):

1. $ cat u[TAB] -> cat uno
2. $ cat uno[TAB][TAB] -> Muestra archivos que comienzan con "uno"
3. $ cat uno\_[TAB] -> cat uno\_dos
4. $ cat uno\_[TAB][TAB] -> Muestra archivos que comienzan con "uno\_dos"
5. $ cat uno\_dos\_[TAB] -> cat uno\_dos\_tres.txt

Después de nuestro comando, escribimos la letra u y enseguida presionamos tabulador para realizar un autocompletado pero… ¿Por qué no se completó todo y se detuvo hasta la palabra uno? Resulta que la terminal no sabe cómo continuar debido a que son 3 los archivos que inician con la palabra uno, por tanto, necesita de nuestra ayuda para salir de tal confusión. Si nosotros mismos al estar completando un nombre no sabemos que letra o símbolo seguía, basta con volver a presionar tabulador y la terminal nos mostrará las opciones que tenemos (como se puede ver en el GIF).

Continuamos entonces agregando un guión bajo y presionamos tabulador; la terminal sigue sin completar todo el nombre y requiere que le ayudemos de nuevo, agregamos el siguiente caracter y volvemos a presionar tabulador; con esto último, la terminal sabe que no hay ningún otro archivo que comience con uno\_dos\_ y termina por agregar el resto del nombre que queríamos. Finalmente ejecutamos la instrucción y vemos su resultado.

Aún con esta particularidad que se acaba de explicar, es claro observar que la tecla tabulador nos da una mayor rapidez al ir escribiendo instrucciones.

Agilizando el proceso de edición

Imagina que estás escribiendo una nueva línea en tu terminal, te das cuenta que hay un error al principio y es necesario volver hasta esa parte, usas las flechas del teclado (o el mouse) para llegar hasta ahí. Ahora es necesario ir al final para continuar escribiendo tu instrucción, vuelves a moverte hacia allá.

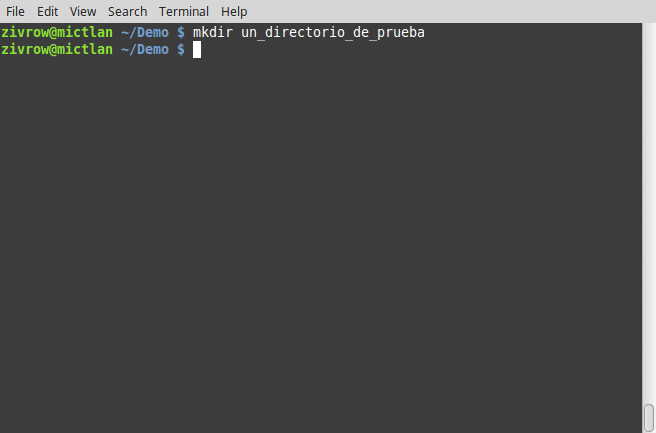
En otro momento ves que la línea que escribes no es la que en verdad querías y necesitas borrarla toda (o una parte). Ahora ves tu terminal llena de instrucciones que se llevaron con éxito, mezcladas con mensajes de error, y quisieras que desaparecieran para sentirte con más comodidad al ver una pantalla limpia, no queda de otra más que cerrar y volver a abrir el programa o presionar varios Enter.

¿No sería mejor si se pudieran realizar todas estas acciones de una forma fácil? Pues déjame decirte que sí. A continuación te describo 5 atajos de teclado que te harán la vida más sencilla.

Ir al inicio de una línea: Ctrl+A

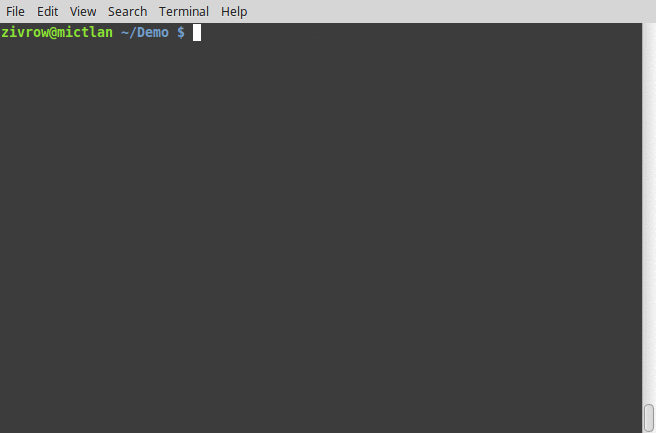
Ir al final de la línea: Ctrl+E

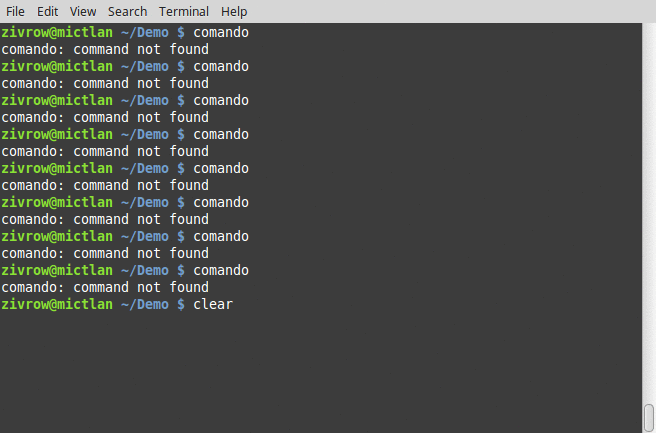
1. $ un\_directorio\_de[Ctrl+A]
2. $ mkdir [Ctrl+E]un\_directorio\_de
3. $ mkdir un\_directorio\_de\_prueba

Borrar desde el cursor hasta el inicio de la línea: Ctrl+U  
Borrar desde el cursor hasta el final de la línea: Ctrl+K

1. $ una\_linea\_para\_borrar
2. $ una\_linea\_pa[Ctrl+U]ra\_borrar
3. $ [Ctrl+K]ra\_borrar

Aquí nos movemos, con ayuda de las flechas, a medias de la palabra para y presionamos la combinación de teclas correspondiente.

Para limpiar el área de nuestra terminal existe un comando llamado clear. Aún así, lo mejor (y más rápido) es utilizar la combinación Ctrl+L que realiza la misma función.

Para terminar

Los atajos que vimos en esta ocasión son básicos para tener una mejor interacción con nuestra terminal. Verás que su uso aumentará de forma considerable tu rapidez al momento de escribir una o varias instrucciones, al igual que permite una manera sencilla de corregir errores que lleguemos a cometer.

06. Terminal: una visita al comando “history”

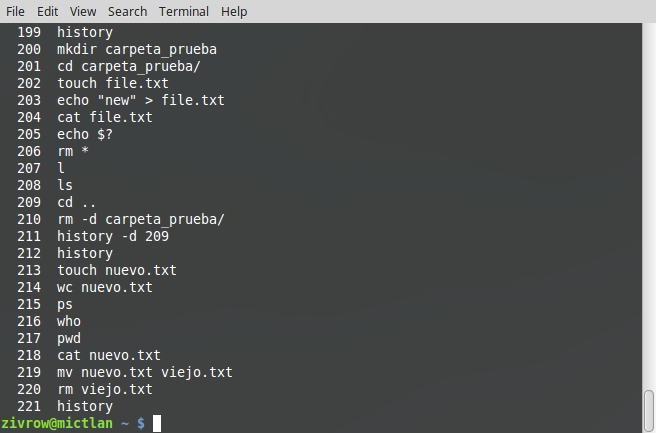
Por [Diego Vizcarra](http://codingornot.com/author/d9ziv)

¿Sabías que la terminal guarda un registro de todos los comandos que has ejecutado desde cierto punto en el que la has estado utilizando? ¿Qué ventajas nos puede otorgar el saber que contamos con esta clase de memoria?

En una nota anterior conocimos algunos atajos de teclado que nos ayudan a agilizar tanto la escritura como la edición de las instrucciones que le damos a la terminal; así pues, en esta ocasión hablaremos del comando history, un comando que accede precisamente al registro de la terminal mencionado al principio para permitirnos repetir comandos que ya hemos utilizado y así evitarnos la necesidad de volverlos a escribir.

¿Qué fue lo último que hice?

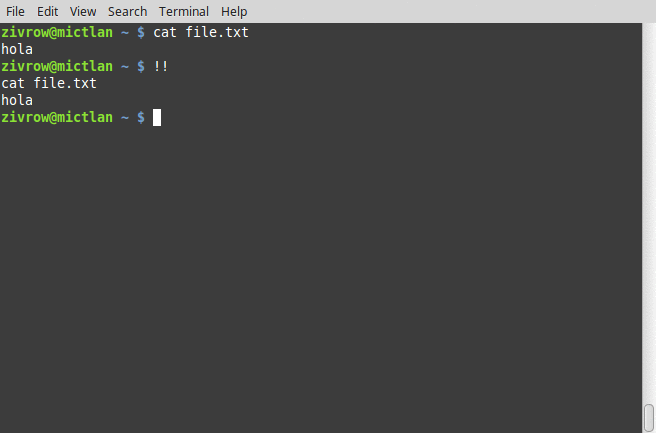
Como habíamos mencionado, history accede al registro que lleva la terminal sobre los comandos que hemos utilizado y, además, los despliega en pantalla. Empecemos por ejecutar el comando y ver qué sucede:

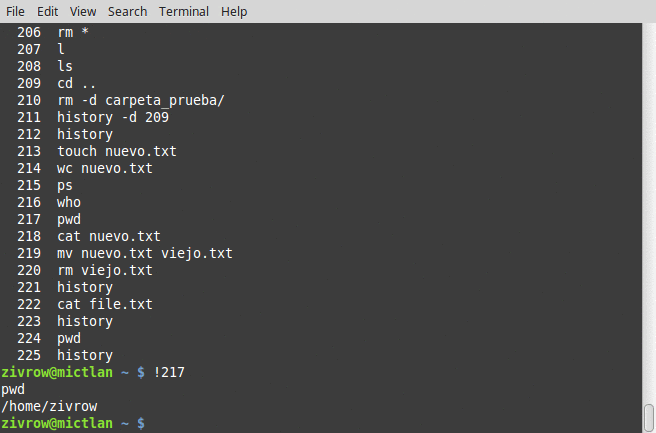
Se nos ha desplegado una lista numerada con todos los comandos que hemos usado hasta el momento. Tomando en cuenta esta lista, conozcamos la primera forma de acceder a un comando que ya hemos escrito anteriormente: presiona la flecha arriba de tu teclado y aparecerá el último comando de la lista, es decir, el último que usaste; presiona varias veces la misma flecha y seguirás accediendo, uno por uno, a los comandos anteriores. Ahora, si presionas la flecha abajo, lograrás ir a los comandos recientes (hasta llegar al último que usaste).

Aunque el método anterior es la forma más común de acceder a los comandos del historial, existen otras más que son igualmente sencillas y, dependiendo qué tengamos que hacer, será más conveniente usar una que la otra.

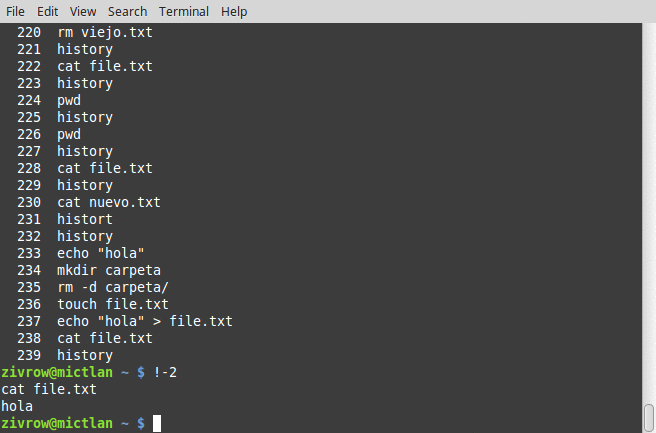
Acceso rápido

Para acceder al último comando que hemos utilizado bastará con escribir !!

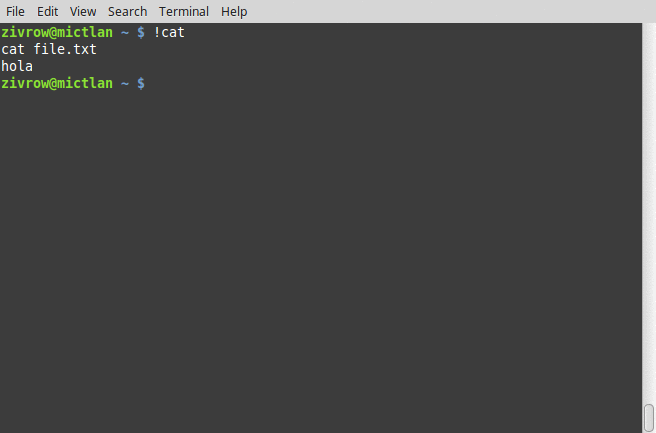
Si lo que deseas es usar un comando en específico basta con escribir !n, siendo n el número que ese mismo comando ocupa en la lista del historial.

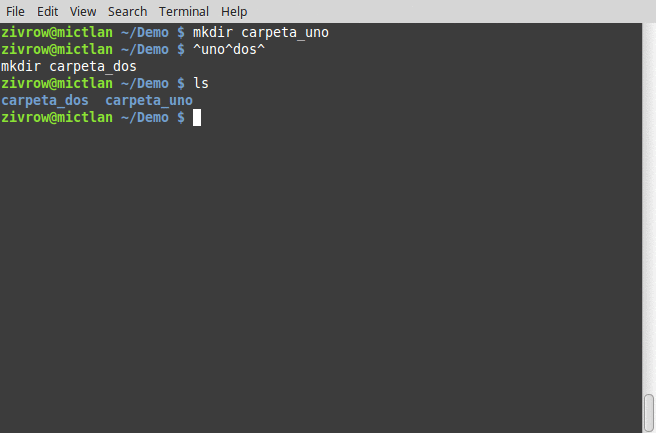


Usando !-n, accederemos al comando que se ejecutó n comandos antes en el historial (!-1 sería lo mismo que !! o pulsar la flecha arriba una vez).



Con !texto accederemos al último comando que utilizamos y que inicia con texto (texto vendría siendo cualquier palabra que elijamos).

En caso de que quisieras acceder al último comando pero necesitaras cambiar una parte de él, basta con ingresar lo siguiente ^texto1^texto2^ siendo texto1 la palabra que quieres cambiar y texto2 el reemplazo.

Cabe mencionar que al usar cualesquiera de estos métodos accedemos al comando y, además, se ejecuta automáticamente. Es importante tener esto en cuenta para evitar llamar de nuevo algún comando que realizó una operación delicada.

Por último, si queremos borrar nuestro historial y que el registro quede totalmente vacío, basta con escribir lo siguiente:

1. $ history -c

Nota: La operación anterior no te pedirá confirmación alguna, es decir, se ejecutará al instante. Piensa bien antes de llegar a utilizarlo.

Más atajos

Antes de llegar al final de esta nota, quisiera mostrarte tres atajos de teclado que tienen que ver con el comando que acabamos de revisar. Si utilizas la combinación Ctrl+P obtendrás el mismo resultado que si presionas la flecha arriba de tu teclado; por otra parte, si utilizas Ctrl+N, se realizará el mismo trabajo que si presionas la flecha abajo.

Por último, si desearas reutilizar un comando en específico y quisieras evitarte el consultar toda la lista para ver qué número necesitas llamar, puedes presionar Ctrl+R y enseguida aparecerá un buscador de tu historial; lo único que tienes que hacer es ir escribiendo alguna palabra que está incluida en el comando que buscas y te aparecerá una opción que coincida con tu escrito, puedes seguir presionando Ctrl+R para que la búsqueda continúe ofreciéndote opciones.

Te sugiero visites la nota [Edición de comandos](http://codingornot.com/01-shell-edicion-de-comandos/) para revisar ejemplos sobre estos y otros atajos.

Para terminar

Hasta aquí llegamos con las utilidades que nos facilitan el trabajo cuando de escribir en terminal se trata. En la próxima nota incluiremos otro conjunto de comandos a nuestro arsenal que trabajan con archivos. A diferencia de los pequeños archivos que hemos tratado en la serie, seguro en tu sistema existen muchos que contienen grandes cantidades de información, ¿cómo manejar dicha información de forma fácil en la consola?

07. Terminal: accediendo al interior de un archivo de texto

Por [Diego Vizcarra](http://codingornot.com/author/d9ziv)

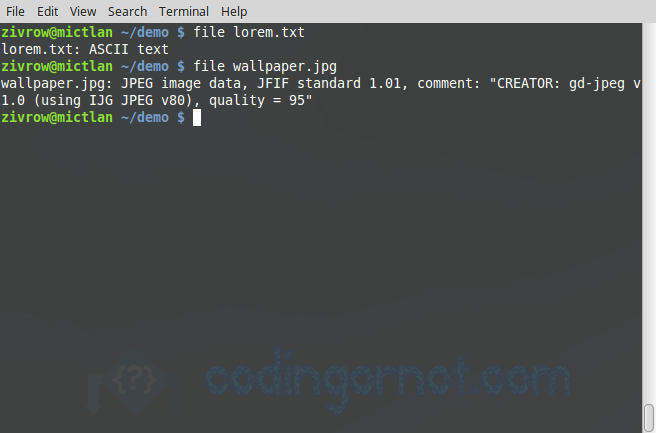
Si tengo un archivo de texto que quiero leer ¿qué debería de hacer para lograrlo? Simple: abro mi editor de texto favorito y problema resuelto. Ahora, ¿qué sucede si estoy trabajando en la terminal y quiero ver rápidamente el contenido de algún archivo? Salir de la terminal (o minimizarla) y acceder al editor de textos ya no suena tan buena idea, esas pequeñas acciones toman tiempo y si queremos revisar varios archivos seguro que se nos irán varias unidades valiosas de tiempo en el seleccionar algún archivo y cargarlo, seleccionar otro y cargarlo, seleccionar otro y…

En esta nota hablaremos sobre unos útiles comandos que precisamente nos permitirán visualizar el contenido de un archivo sin tener que abrir ningún programa gráfico. No importa que cantidad de información alberguen nuestros archivos, estos comandos nos mostrarán toda o parte de ella. Vamos allá entonces.

Conociendo el tipo de archivo

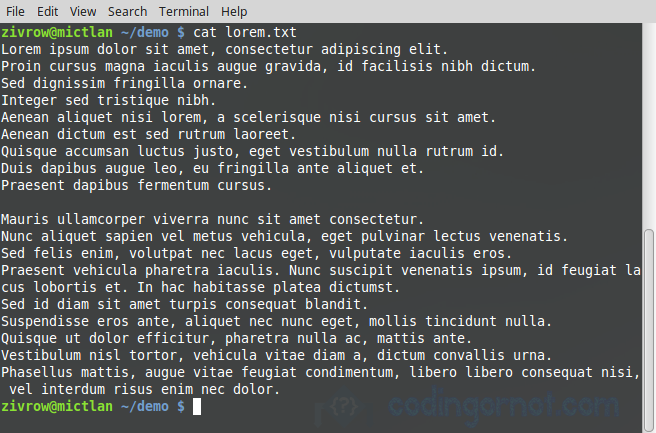
Antes que cualquier otra cosa, visitemos primero el comando file. Si queremos ver el contenido de algún archivo de texto, es necesario que este sea de este tipo, ¿por qué? Porque de esto dependerá de si lo que veremos en pantalla será texto común y legible o un montón de símbolos extraños. Uno podría pensar que basta con observar que la extensión del archivo sea .txt pero, a decir verdad, un archivo de texto puede llevar cualquier tipo de extensión, por ejemplo, los archivos que se utilizan para guardar los mensajes que genera un programa (conocidos como log files) suelen contar con la extensión .log.

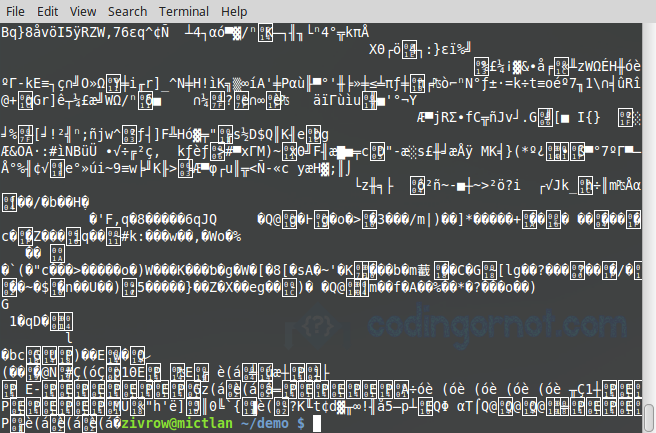
Veamos dos archivos de ejemplo para utilizar este comando, uno será un archivo de texto y otro será un archivo de imagen:

Como puedes apreciar, file nos indica correctamente qué tipo de archivo es cada uno. Para el archivo de texto nos despliega ASCII text que básicamente significa que el archivo contiene texto con caracteres “ASCII”. file puede recibir cualquier número de parámetros, de esta forma podemos saber (usando una única línea) el tipo de varios archivos.

Concatenando archivos

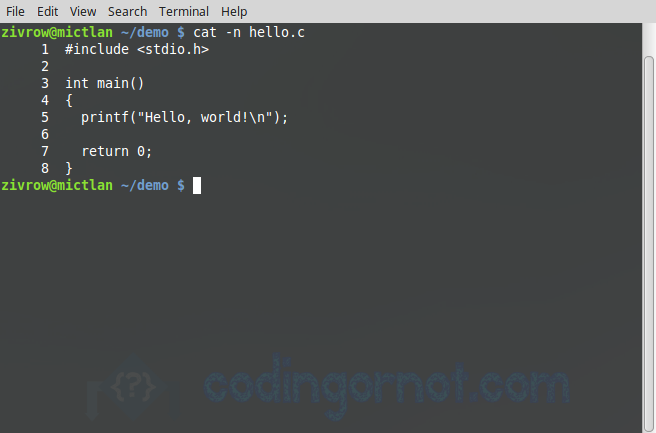
Recordando la [nota no. 4](http://codingornot.com/manipulando-archivos-y-directorios/) de esta serie, vimos de forma rápida el comando cat que nos permitía desplegar el contenido de un archivo de texto directamente en pantalla. Si colocamos varios parámetros, cat concatenará los contenidos y nos los mostrará todos de una vez (como si de un solo archivo se tratara). Como curiosidad, veamos qué se nos muestra en pantalla al usar los dos archivos del apartado anterior:

Uso del comando cat en texto

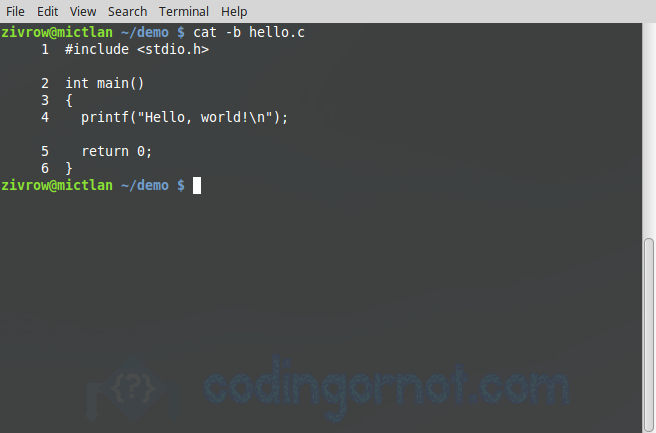
Uso del comando cat en una imagen

Como habíamos comentado, el archivo de texto nos muestra información legible y la imagen solo consta de símbolos extraños.

Pasemos ahora a conocer dos opciones para el comando cat que pueden resultar muy útiles en algunas circunstancias. Tomemos como ejemplo un código fuente, de este queremos que, además del contenido, se nos despliegue también el número de cada línea; podemos lograr esto simplemente añadiendo -n:

De esta manera a cada línea del archivo le antecederá un número, el cual, en este caso en particular, nos sería de gran ayuda si estábamos buscando lo que existe en la línea número X.

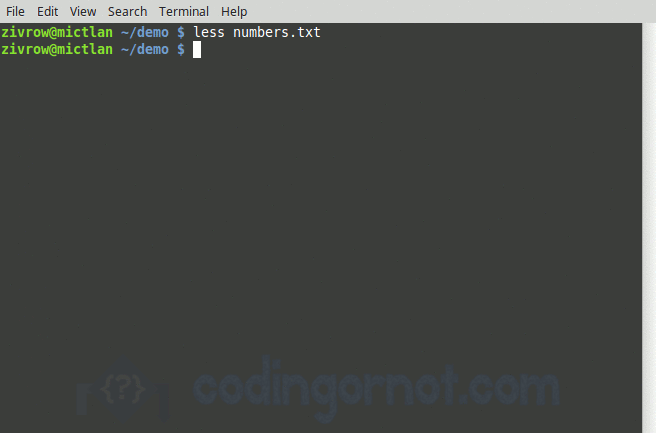
Si quisiéramos que se nos muestre un número pero solo en aquellas líneas que contengan texto, basta con añadir la opción -b:

Menos es más

Como acabamos de ver, cat es una herramienta muy útil pero tiene un pequeño inconveniente, si el archivo que queremos leer tiene mucha información, cat nos mostrará toda esa información de una vez, sin importarle detenerse en algún momento para que vayamos leyendo cada parte.

less es un comando muy útil que permite desplegar el contenido de un archivo pero con la gran ventaja de detenerse cuando el texto ha llenado la pantalla, si esto sucede basta con presionar la letra D para seguir leyendo el archivo, o presionar U si queremos regresar. Para salir del modo lectura queda presionar la tecla Q y volveremos a nuestro prompt.

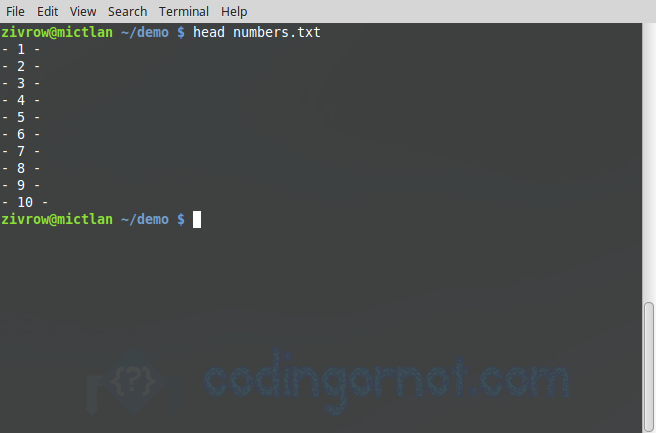
Veamos cómo funciona este comando utilizando un archivo que contiene los números del 1 al 4000:

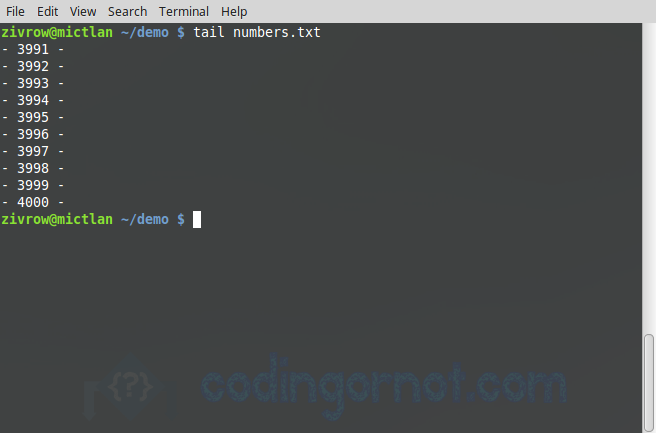
Oprimiendo “D” vamos avanzando en el archivo y salimos oprimiendo “Q”

Desde la cabeza hasta la cola

Existen otro tipo de situaciones en que no queremos ver todo el archivo sino solo una parte de él; si esta parte se encuentra al inicio o al final del archivo, estamos de suerte.

El comando head nos permitirá conocer las primeras líneas del archivo que le pasemos como parámetro, por defecto las primeras 10 líneas. Por otra parte, el comando tail nos mostrará las últimas líneas del archivo que elijamos, igual las última 10 líneas por defecto. Utilicemos el archivo de números para demostrar cómo funcionan estas instrucciones:

head: Nos muestra las primeras 10 líneas

tail: Nos muestra las últimas 10 líneas

Si queremos cambiar el número de líneas que se nos presenten, en lugar de las 10 que mencionamos, basta con colocar -n # donde # será el numero de líneas que queremos desplegar. Una forma más corta de hacerlo es únicamente colocar -# y obtendremos el mismo resultado:

“head” nos muestra 5 líneas, mientras “tail” nos muestra 7 líneas.

Para terminar

En esta nota aprendimos como visualizar el contenido de un archivo con la ayuda de unos prácticos comandos. Ya sea que quisiéramos ver toda la información, o solo una parte de ella, ahora tenemos las herramientas para lograrlo y todo desde la comodidad de la misma terminal.

08. Terminal: comandos en el ambiente Windows

Por [Diego Vizcarra](http://codingornot.com/author/d9ziv)

Como te comenté en la [primera nota](http://codingornot.com/conociendo-la-terminal/) de esta serie, el primer sistema con el que interactué fue MS-DOS. En aquellos días era sumamente común el tener que aprender diversos comandos para poder realizar tareas (desde las más sencillas hasta las más complejas) en tu computadora. Tiempo después, la interfaz gráfica hizo que el escribir todas estas sentencias fuera casi dejado en el olvido (sobre todo para los usuarios comunes), y que el mouse se convirtiera en la principal forma de comunicación con nuestro equipo; se llegó incluso al punto en el que si una persona llegaba a perder su mouse le resultaba casi imposible hacer nada con su computadora.

Aunque no fue el primero que las incluía, Windows fue el sistema operativo más vendido y famoso cuando las interfaces gráficas llegaron. Comenzando con Windows 3.1 y llegando al actual\* Windows 10, la interfaz gráfica ha ido evolucionando y adecuándose a las necesidades del usuario. Aún así, la consola sigue estando presente para cualquiera que desee usarla.

En notas anteriores hemos ido conociendo varios comandos para diversas tareas, pero estos solo se encuentran disponibles si nos encontramos en un sistema operativo que tenga una base UNIX. Te llegué a comentar que estaríamos trabajando en un entorno Linux pero creo que vale la pena hacer mención del equivalente que existe de estos comandos en el ambiente de Windows. Aunque las tareas realizadas son prácticamente las mismas, la instrucción que debemos escribir para ejecutarlas es diferente en ambos entornos.

De acuerdo a lo que fui tratando en la serie, iremos viendo las alternativas a los comandos de Linux en el siguiente orden:

1. Comandos de navegación: pwd, ls y cd.
2. Directorios y archivos: mkdir, cp, mv y rm.
3. Otros comandos útiles: history, clear, cat y less.

Vamos allá entonces.

Comandos de navegación

Seguro conoces de sobra cómo abrir la consola de comandos en Windows pero me tomaré el atrevimiento de comentarte que una forma sencilla de acceder a ella es realizando lo siguiente:

1. Usa la combinación de teclas Win+R (recuerda que Win es la tecla con el símbolo de Windows).
2. En la ventana que aparece, escribe cmd.
3. Presiona enter o da clic en OK.

Esto es parecido a lo que comúnmente se realiza en Linux, ahí puedes presionar Ctrl+Alt+T y la terminal aparecerá de igual manera.

Si deseas conocer en Windows sobre qué directorio te encuentras basta con elegir una de dos opciones disponibles, puedes escribir cd o chdir. Por motivos de ahorrarte tiempo al escribir estoy seguro que elegirás la primera opción, por tanto, puedes tomar el significado de dicho comando como “current directory”.

Si lo que quieres es listar el contenido de un directorio basta con escribir el comando dir. La forma en que ls y dir despliegan los contenidos es distinta pero no tanto como para que te llegues a confundir. El significado que guarda este comando podría ser algo como “display files”.

A diferencia del cd que nos muestra el directorio donde nos encontramos (y que no lleva ningún parámetro) para este caso sí que requerimos colocar un parámetro: el directorio al que deseamos movernos. La funcionalidad del comando, en este último caso, es el mismo que en Linux y podemos tomar también su mismo significado: “change directory”.

Directorios y archivos

Existen comandos en Linux más sencillos de recordar (o que no crean tanta confusión) que en Windows; de igual forma, existen comandos en Windows que son más sencillos (o descriptivos) que en Linux. En esta parte veremos unos buenos ejemplos de esto.

Igual que sucede en el sistema operativo del pingüino Tux, si deseas crear un directorio basta con usar el comando mkdir, obtendrás el mismo resultado. Aún así, existe una alternativa más abreviada: md. Cualquiera que decidas utilizar (y otorgándole el parámetro adecuado) tendrás un nuevo y resplandeciente directorio.

Para lograr copiar archivos o directorios teníamos a cp, en Windows tenemos una alternativa más descriptiva: copy.

En cuanto a la tarea de mover archivos tenemos a move en lugar de mv. ¿Recuerdas que si queríamos renombrar un archivo o directorio podíamos usar el mismo comando mv para lograrlo? En Windows no es la excepción, pero existe otro comando mucho más descriptivo: rename. La forma de utilizarlo es exactamente igual que en move.

Por último, vayamos con la eliminación de archivos o directorios. En el caso de los archivos solo debíamos usar rm; para el caso de los directorios debíamos agregar ciertas opciones dependiendo de si estos se encontraban vacíos o no. En Windows aparecen dos comandos que realizan la acción de eliminación.

Comenzando con el comando del, este suele utilizarse sobre archivos que queramos eliminar; no obstante, también puede utilizarse sobre algún directorio pero tendrá un comportamiento distinto: independientemente del estado del directorio, lo que del hará es eliminar todo el contenido de dicha carpeta (no sin antes pedirnos una confirmación). Es lógico pensar que si el directorio se encuentra vacío, el comando hará prácticamente nada.

El segundo comando es: rd. Este comando está hecho específicamente para ser usado con directorios. Si el directorio se encuentra vacío, simplemente lo eliminará, de lo contrario, dirá que no es posible realizar la acción porque el directorio alberga cierto contenido. Para lograr eliminar un directorio de las características mencionadas, debemos agregar la opción /S y, después de una confirmación, se eliminará todo rastro de la carpeta y su contenido.

Otros comandos útiles

¿Recuerdas que en Linux tenemos al comando history para desplegar un historial de los comandos que hemos utilizado? En la consola de Windows no existe tal comando pero podemos recrear la misma funcionalidad si presionamos la tecla F7. De esta forma aparecerá un mini menú donde podemos elegir uno de los comandos que hemos utilizado en nuestra sesión actual. Si deseas salir de él, basta con que presiones Esc.

Si tu área de trabajo en la consola se ha ido llenando de varias instrucciones y de los resultados obtenidos por ejecutarlas, solo tenemos que utilizar cls. Tenemos otro nombre más corto que el de clear que utilizamos en Linux.

Para lograr ver el contenido de un archivo de texto, en lugar de utilizar cat, debemos utilizar el comando type. Si queremos utilizar la funcionalidad de less, tendremos algo parecido llamando al comando more. Este último comando necesitará de que presionemos enter para ir avanzando en la lectura de nuestro archivo; si deseásemos salir, basta con presionar la tecla Q.

Para terminar

Con la llegada de Windows 10 se introdujo la posibilidad de utilizar los mismos comandos que hemos ido viendo a lo largo de la serie haciendo uso de un entorno como el que se encuentra en Linux. Puedes ver más información [aquí](https://msdn.microsoft.com/en-us/commandline/wsl/about).

Atendiendo a lo dicho en el párrafo anterior, podríamos pensar que ya no es necesario aprender los comandos que hemos visto en esta nota, no obstante, creo que aprender un poco más sobre la diversidad de sistemas que nos rodean nos convierte en usuarios con un mayor control sobre las herramientas que utilizamos en el día a día. Aún cuando cada cierto tiempo aparecen nuevas ediciones de sistemas operativos, las versiones viejas se siguen utilizando en algunas partes y es cuando este tipo de conocimientos se vuelven útiles.

09. Terminal: revisando el manual de usuario

Por [Diego Vizcarra](http://codingornot.com/author/d9ziv)

Seguro recuerdas que en algunos comandos vistos logré mencionar algo sobre las opciones. Estas partes que acompañan a un comando permiten que este último ofrezca una funcionalidad extra a la que normalmente tiene. Por ejemplo, cuando vimos el comando rm, notaste que no funciona si lo usamos sobre un directorio vacío, en cambio, si le añadíamos la opción -d, el directorio era eliminado sin problema alguno.

Toda esta información no es algo que debas aprenderte estrictamente de memoria, para nuestra suerte existe un manual de usuario que nos dirá exactamente qué hace un comando y qué otras acciones puede llegar a realizar con las opciones adecuadas. En esta última nota conoceremos a man, un comando que precisamente nos mostrará el manual de aquellos comandos que queramos.

El manual del manual

Comencemos con el principio, veamos cómo es que se debe de usar el comando man:

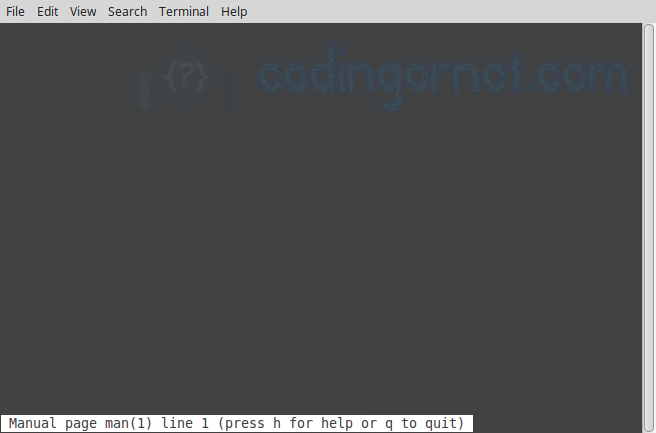
1. $ man <comando>

Como puedes observar, lo único que debemos hacer es reemplazar la parte de <comando> por el nombre del programa del que deseamos leer su documentación. De hecho, el mismo man cuenta con su propia página que describe lo que hace y las opciones que podemos llegar a utilizar en él.

Al momento de que la página de documentación se nos muestra en la pantalla, es importante identificar 3 partes que yo considero las más importantes:

1. Name: Muestra el nombre del comando junto a una resumida descripción.
2. Synopsis: Nos menciona la forma correcta de utilizar el comando.
3. Description: Otorga una completa descripción de lo que el comando en cuestión puede realizar y las opciones (con su propia descripción) que podemos añadirle.

En algunos casos puede y aparezca en la parte inferior de la pantalla de la terminal una pequeña área donde se te informe de cuántas líneas consta la documentación que estás leyendo y en qué línea te encuentras. Mientras estás en el manual, puedes oprimir la tecla H para recibir ayuda sobre cómo realizar algunas acciones tal como moverte por la documentación o buscar a través de ella.

¿Recuerdas al comando less y la forma para navegar entre sus páginas? Lo mismo puedes hacer en este caso para ir leyendo toda la documentación de un comando, simplemente oprime D para ir hacia adelante o U para ir hacia atrás. Si decides ir navegando línea por línea oprime la flecha de teclado correspondiente (arriba o abajo) y para salir solo presiona Q.

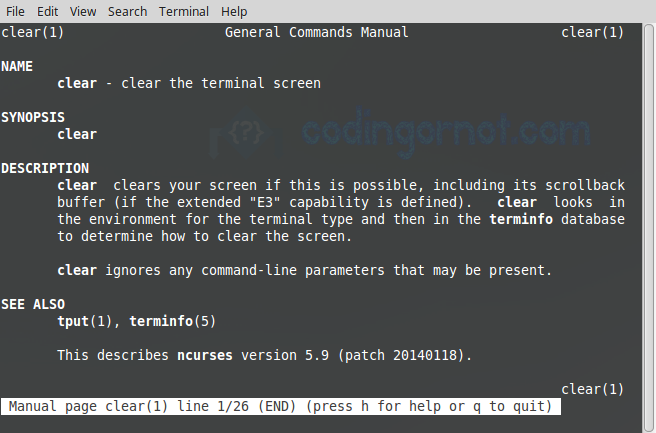
Ya con esta información en nuestro poder, ¿qué te parece si vemos algunos ejemplos?

Eligiendo entre diferentes opciones

En esta parte trataremos los comandos clear, ls y rm a fin de ejemplificar lo que se ha explicado hasta el momento.

Iniciando con el comando más sencillo de los tres, veamos qué información se nos muestra en pantalla al escribir lo siguiente:

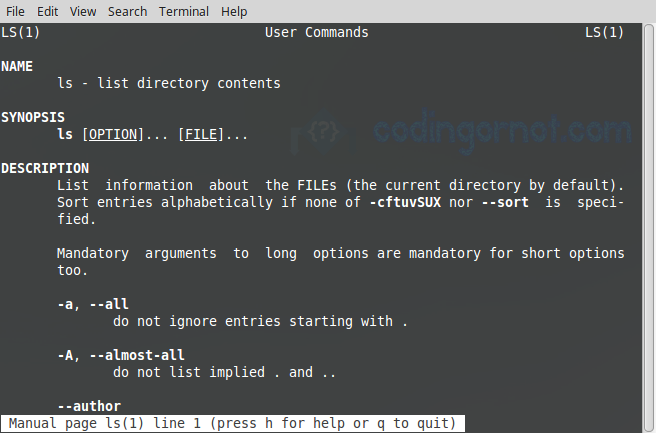
1. $ man clear

En la imagen aparecen las partes que comenté al principio, puedes observar que el comando nos sirve para limpiar la pantalla de la terminal, que para utilizarlo basta con escribir únicamente la palabra clear y que no tiene ninguna opción que le podamos agregar para que realice una acción diferente.

Veamos ahora algo más interesante, conozcamos la documentación de ls:

1. $ man ls

Ya vimos en otra nota para qué sirve ls, así que enfoquémonos en la parte de la sinopsis y en la descripción:

En sinopsis puedes ver lo siguiente:

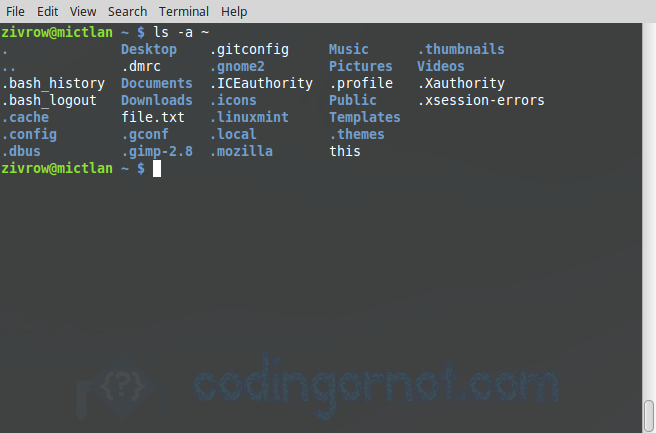
1. ls [OPTION]... [FILE]...

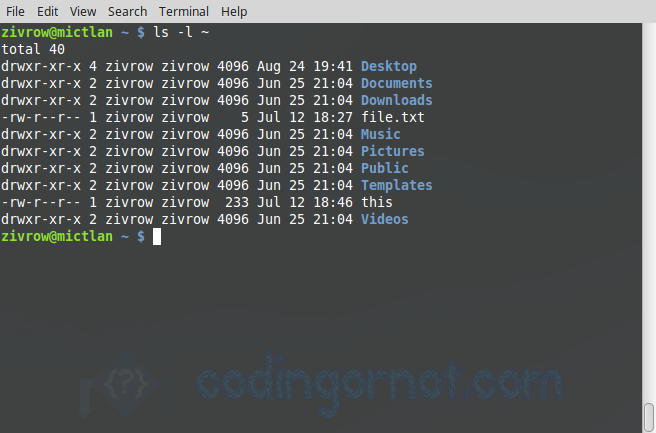
Para poder leer esta línea es preciso conocer qué significan algunos símbolos: si algo aparece entre corchetes significa que esa parte la podemos omitir; si aparecen puntos suspensivos se nos indica que puede haber cualquier número de argumentos que podemos escribir.

Sabiendo lo anterior, basta decir que al momento de usar ls, debemos escribir el nombre del comando; cero, uno o más opciones y cero, uno o más archivos.

Usemos ahora un par de opciones que vienen en la descripción y tomemos como ejemplo al directorio del usuario:

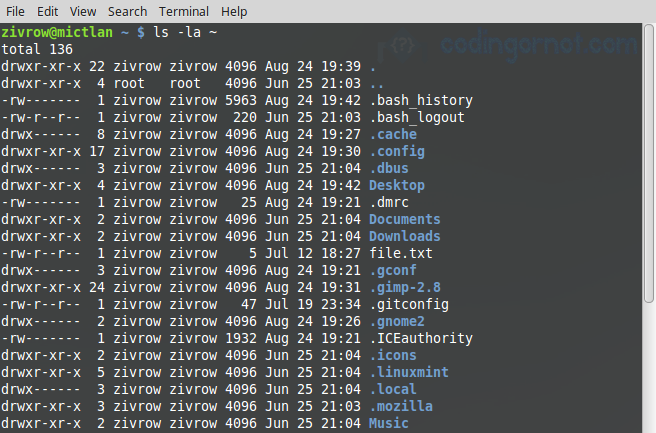
1. $ ls -a ~

Muestra todos los archivos que incluye el directorio, incluidos los archivos ocultos (estos llevan un punto al principio del nombre).



Muestra una lista más detallada de los archivos que incluye el directorio.

$ ls -la ~

Es fácil apreciar que la forma de utilizar a help es la misma que utilizamos con man; incluso también puedes llegar a conocer más información sobre este comando si escribes:

1. $ help help

En caso de que desees conocer más sobre lo que es la shell, te invito a que revises otra de nuestras series que con gusto ponemos a tu disposición en este blog, solo da clic [aquí](http://codingornot.com/temporada-01-fundamentos-basicos-de-la-shell/) y listo.

Para terminar

Hemos llegado al fin de esta primera temporada de la serie Terminal 101. Recuerda que el propósito de esta serie reside en ayudarte a tener un conocimiento básico de la terminal de tu sistema y que experimentes en carne propia lo que esta gran herramienta puede hacer por ti. Te garantizo que si decides incluir en tu flujo de trabajo todo lo que has aprendido hasta el momento, notarás que tu productividad se verá afectada de forma positiva.

Es fácil apreciar que la forma de utilizar a help es la misma que utilizamos con man; incluso también puedes llegar a conocer más información sobre este comando si escribes:

1. $ help help

En caso de que desees conocer más sobre lo que es la shell, te invito a que revises otra de nuestras series que con gusto ponemos a tu disposición en este blog, solo da clic [aquí](http://codingornot.com/temporada-01-fundamentos-basicos-de-la-shell/) y listo.

Para terminar

Hemos llegado al fin de esta primera temporada de la serie Terminal 101. Recuerda que el propósito de esta serie reside en ayudarte a tener un conocimiento básico de la terminal de tu sistema y que experimentes en carne propia lo que esta gran herramienta puede hacer por ti. Te garantizo que si decides incluir en tu flujo de trabajo todo lo que has aprendido hasta el momento, notarás que tu productividad se verá afectada de forma positiva.

Realiza las dos acciones anteriores de una sola vez.

Toma en cuenta que puedes concatenar varias opciones si las colocas juntas (no importa el orden), de esta manera evitas escribir algo como lo siguiente que te otorgará el mismo resultado:

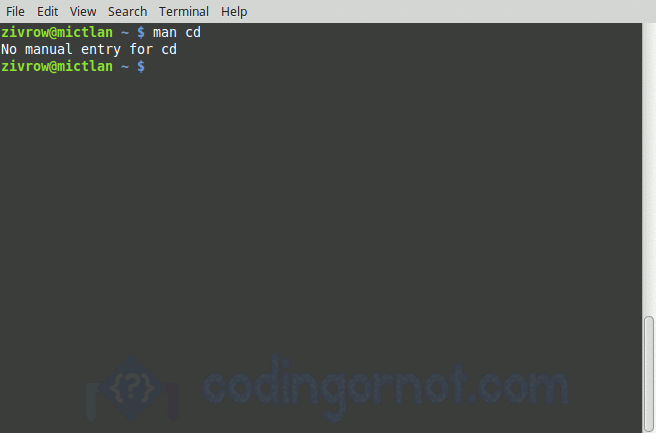
1. $ ls -l -a ~

Para finalizar, te dejo como ejercicio al comando rm; para esto te invito a que revises de nuevo esta nota y busques en la documentación (por tu propia cuenta) para qué nos sirven las opciones que ahí se expusieron.

¿Comandos sin manual?

Tal vez sientas curiosidad por conocer más sobre otros comandos diferentes a los que usamos como ejemplo en esta ocasión, pero también puede que logres llevarte una no tan grata sorpresa al descubrir que no existe una entrada que muestre la funcionalidad de cierto comando. Tomemos como ejemplo al comando cd:

1. $ man cd

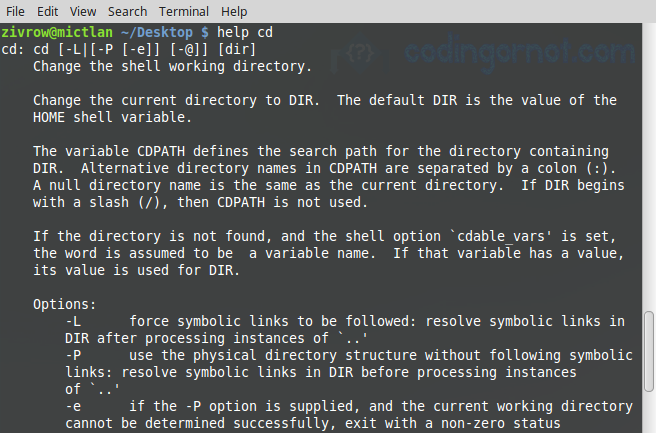
Puedes notar que se nos muestra un mensaje indicándonos que no existe alguna entrada de manual que tenga información sobre cd. ¿Por qué sucede esto y qué hacemos en estos casos?

Resulta que varios de los comandos que llegamos a utilizar son programas externos; estos programas que residen en nuestro sistema son localizados, llamados y ejecutados por el shell para que realicen su función. En el caso de estos comandos, sí podemos utilizar man para que nos muestre su descripción.

Existe otro grupo de comandos conocidos como shell builtins, estos se diferencian de los primeros en que se encuentran implementados dentro del mismo shell, es decir, no son programas que se deban cargar desde otra parte y luego ejecutarse, simplemente se llaman para que realicen lo propio.

cd es un buen ejemplo de shell builtin. Para poder conocer información sobre él (o cualquier otro del mismo tipo) necesitamos del siguiente comando que también es builtin: help

1. $ help cd

Es fácil apreciar que la forma de utilizar a help es la misma que utilizamos con man; incluso también puedes llegar a conocer más información sobre este comando si escribes:

1. $ help help

En caso de que desees conocer más sobre lo que es la shell, te invito a que revises otra de nuestras series que con gusto ponemos a tu disposición en este blog, solo da clic [aquí](http://codingornot.com/temporada-01-fundamentos-basicos-de-la-shell/) y listo.

Para terminar

Hemos llegado al fin de esta primera temporada de la serie Terminal 101. Recuerda que el propósito de esta serie reside en ayudarte a tener un conocimiento básico de la terminal de tu sistema y que experimentes en carne propia lo que esta gran herramienta puede hacer por ti. Te garantizo que si decides incluir en tu flujo de trabajo todo lo que has aprendido hasta el momento, notarás que tu productividad se verá afectada de forma positiva.