**Trabajando con archivos y directorios**

**Hoja de ruta**

**Leccion:** 15 min   
**Prácticas:** 0 min

**Preguntas**

* ¿Cómo puedo crear, copiar y eliminar archivos y directorios?
* ¿Cómo puedo editar archivos?

**Objetivos**

* Crear una jerarquía de directorios que coincida con un diagrama dado.
* Crear archivos en esa jerarquía usando un editor o copiando y renombrando archivos existentes.
* Mostrar el contenido de un directorio utilizando la línea de comandos.
* Eliminar archivos y/o directorios específicos.

Ahora sabemos cómo explorar archivos y directorios, pero ¿cómo los creamos en primer lugar? Volvamos a nuestro directorio datos-shell en Desktop y utilicemos el comando ls -F para ver lo que contiene:

$ pwd

/Usuarios/nelle/Escritorio/datos-shell

$ ls -F

criaturas/ moléculas/ pizza.cfg

datos/ norte-pacífico/ solar.pdf

Escritorio/ notas.txt escritura/

Creemos un nuevo directorio llamado tesis usando el comando mkdir tesis (que no genera una salida):

$ mkdir tesis

Como su nombre sugiere, mkdir significa “make directory”, que significa “crear directorio” en inglés. Dado que tesis es una ruta relativa (es decir, no inicia con una barra oblicua /), el nuevo directorio se crea en la carpeta de trabajo actual:

$ ls -F

criaturas/ norte-pacífico/ tesis/

datos / notas.txt escritura/

Escritorio/ pizza.cfg

moléculas/ solar.pdf

**Dos maneras de hacer lo mismo**

Usar la terminal para crear un directorio no es diferente de usar un navegador de archivos gráfico. Si abres el directorio actual utilizando el explorador de archivos gráfico de tu sistema operativo, el directorio tesis aparecerá allí también. Si bien son dos formas diferentes de interactuar con los archivos, los archivos y los directorios con los que trabajamos son los mismos.

**Buena nomenclatura para archivos y directorios**

Usar nombres complicados para archivos y directorios pueden hacer tu vida muy complicada cuando se trabaja en la línea de comandos. Te ofrecemos algunos consejos útiles para nombrar tus archivos de ahora en adelante.

1. No uses espacios en blanco.

Los espacios en blanco pueden hacer un nombre más significativo, pero, dado que se utilizan para separar argumentos en la línea de comandos, es mejor evitarlos en nombres de archivos y directorios. Puedes utilizar - o \_ en lugar de espacios en blanco.

1. No comiences el nombre con un - (guión).

Los comandos tratan a los nombres que comienzan con - como opciones.

1. Utiliza únicamente letras, números, . (punto), - (guión) y \_ (guión bajo).

Muchos otros caracteres tienen un significado especial en la línea de comandos y los aprenderemos durante esta lección. Algunos sólo harán que tu comando no funcione, otros pueden incluso hacer que pierdas datos.

Si necesitas referirte a nombres de archivos o directorios que tengan espacios en blanco u otro carácter no alfanumérico, se debe poner el nombre entre comillas dobles ("").

Dado que acabamos de crear el directorio tesis, aún se encuentra vacío:

$ ls -F tesis

Cambiemos nuestro directorio de trabajo a tesis usando cd, y a continuación, ejecutemos un editor de texto llamado Nano para crear un archivo denominado draft.txt:

$ cd tesis

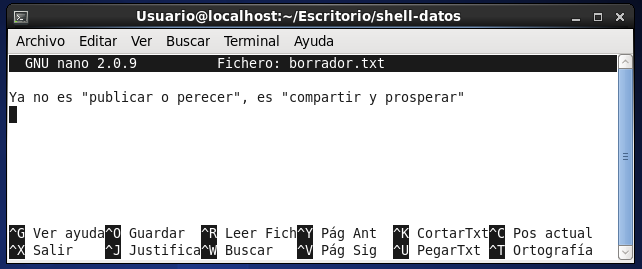
$ nano borrador.txt

**¿Qué editor usar?**

Cuando decimos, “nano es un editor de texto”, realmente queremos decir “texto”: sólo funciona con datos de caracteres simples, no con tablas, imágenes o cualquier otro formato amigable con el usuario. Lo utilizamos en ejemplos porque es un editor muy sencillo que permite funciones muy básicas. Sin embargo, por estas mismas cualidades, podría ser insuficiente para necesidades de la vida real. En los sistemas Unix (como Linux y Mac OS X) muchos programadores utilizan [Emacs](http://www.gnu.org/software/emacs/) o [Vim](http://www.vim.org/) (ambos requieren más tiempo para familiarizarse con ellos), o un editor gráfico como [Gedit](http://projects.gnome.org/gedit/). En Windows puedes utilizar [Notepad ++](http://notepad-plus-plus.org/). Windows también tiene un editor interno llamado notepad que se puede ejecutar desde la línea de comandos de la misma manera que nano para los propósitos de esta lección.

Sea cual sea el editor que uses, necesitarás saber dónde busca y guarda archivos. Si lo inicias desde la terminal, usará (probablemente) el directorio de trabajo actual como ubicación predeterminada. Si utilizas el menú de inicio de tu computadora puede ser que los archivos se guarden en tu **Escritorio** o el directorio de Documentos. Puedes cambiar de directorio destino navegando a otro directorio la primera vez guardes el archivo usando “Guardar como …”.

Escribamos algunas líneas de texto. Una vez que estemos contentos con nuestro texto, podemos presionar Ctrl-O (presiona la tecla Ctrl o Control y, mientras la mantienes presionada, oprime la tecla O) para escribir nuestros datos en el disco (se nos preguntará en qué archivo queremos guardar esto: presiona Enter para aceptar el valor predeterminado sugerido borrador.txt).



Una vez que nuestro archivo está guardado, podemos usar Ctrl-X para salir del editor y volver a la terminal.

**Tecla Control, Ctrl o ^**

La tecla Control también se denomina tecla “Ctrl”. Hay varias maneras de indicar el uso de la tecla Control. Por ejemplo, una instrucción para presionar la tecla Control y, mientras la mantienes pulsada, presionar la tecla X, puede ser descrita de cualquiera de las siguientes maneras:

* Control-X
* Control+X
* Ctrl-X
* Ctrl+X
* ^X
* C-x

En nano, a lo largo de la parte inferior de la pantalla se lee ^G Ver ayuda ^O Guardar. Esto significa que puedes usar Control-G para obtener ayuda y Control-O para guardar tu archivo.

nano no deja ninguna salida en la pantalla después de que salir del programa, pero ls ahora muestra que hemos creado un archivo llamado borrador.txt:

$ ls

borrador.txt

Limpiemos un poco ejecutando rm borrador.txt:

$ rm borrador.txt

Este comando elimina archivos (rm es la abreviatura de “remove”, “remover” en inglés). Si ejecutamos ls de nuevo, la salida estará vacía una vez más, indicándonos que nuestro archivo ha desaparecido:

$ ls

**Eliminar es para siempre**

La terminal de Unix no tiene una papelera de reciclaje desde donde podamos restaurar archivos eliminados (aunque la mayoría de las interfaces gráficas de Unix sí lo tienen). En su lugar, cuando eliminamos archivos, los mismos se desvinculan del sistema de archivos para que su espacio de almacenamiento en disco pueda ser reciclado. Existen herramientas para encontrar y recuperar archivos eliminados, pero no hay garantía de que funcionen en todas las situaciones, ya que la computadora puede reciclar el espacio en disco del archivo en cuestión inmediatamente, perdiéndose de manera permanente.

Creemos de nuevo el archivo y después subamos un directorio a /Usuarios/nelle/Escritorio/datos-shell usando cd ..:

$ pwd

/Usuarios/nelle/Escritorio/datos-shell/tesis

$ nano borrador.txt

$ ls

borrador.txt

$ cd ..

Si tratamos de eliminar todo el directorio tesis usando rm tesis, obtenemos un mensaje de error:

$ rm tesis

rm: no se puede borrar <<tesis/>>: Es un directorio

Esto ocurre porque rm normalmente trabaja sólo con archivos, no con directorios.

Para realmente deshacernos de tesis también debemos eliminar el archivo borrador.txt. Podemos hacer esto con la opción [recursiva](https://en.wikipedia.org/wiki/Recursion) para rm:

$ rm -r tesis

**Un gran poder conlleva una gran responsabilidad**

Eliminar los archivos en un directorio recursivamente puede ser una operación muy peligrosa. Si nos preocupa lo que podríamos eliminar, podemos añadir la opción “interactiva” -i a rm, que nos pedirá confirmar cada paso.

$ rm -r -i tesis

rm: ¿descender al directorio <<tesis>>? (s/n) s

rm: ¿borrar el fichero regular <<tesis/borrador.txt>>? (s/n) s

rm: ¿borrar el directorio <<tesis>>? (s/n) s

Esto elimina todo el contenido en el directorio y después el directorio mismo, preguntando en cada paso para que se confirme la eliminación.

Vamos a crear el directorio y el archivo una vez más. (Ten en cuenta que esta vez estamos ejecutando nano con la ruta de acceso tesis/borrador.txt, en lugar de ir al directorio tesis y ejecutar nano en borrador.txt.)

$ pwd

/Usuarios/nelle/Escritorio/datos-shell

$ mkdir tesis

$ nano tesis/borrador.txt

$ ls tesis

borrador.txt

borrador.txt no es un nombre particularmente informativo, así que cambiemos el nombre del archivo usando el comando mv, que es la abreviatura de “move” (mover):

$ mv tesis/borrador.txt tesis/citas.txt

El primer parámetro dice a mv lo que estamos “moviendo””, mientras que el segundo indica a dónde hay que moverlo. En este caso estamos moviendo tesis/borrador.txt a tesis/citas.txt, que tiene el mismo efecto que cambiar el nombre del archivo. Como esperamos, ls nos muestra que tesis ahora contiene un archivo llamado citas.txt:

$ ls tesis

citas.txt

Hay que tener cuidado al especificar el nombre del archivo destino, ya que mv remplaza silenciosamente cualquier archivo existente con el mismo nombre, provocando pérdida de datos. Un indicador adicional, mv -i (o mv --interactive), se puede utilizar para hacer que mv te pida confirmación antes de sobrescribir.

Sólo por el gusto de la consistencia, mv también funciona en directorios, es decir, no existe un comando separado mvdir. Vamos a mover citas.txt al directorio de trabajo actual. Utilizamos mv una vez más, pero esta vez sólo usaremos el nombre de un directorio como el segundo parámetro para indicar a mv que queremos mantener el nombre de archivo, pero poner el archivo en algún lugar nuevo. (es por eso que el comando se llama “mover”.) En este caso, el nombre de directorio que usamos es el nombre de directorio especial . que mencionamos anteriormente.

$ mv tesis/citas.txt .

El resultado es mover el archivo desde el directorio en el que estaba en el directorio de trabajo actual. ls ahora nos muestra que tesis está vacío:

$ ls tesis

Además, ls con un nombre de archivo o un nombre de directorio como parámetro sólo lista ese archivo o directorio. Podemos usar esto para ver que quotes.txt todavía está en nuestro directorio actual:

$ ls citas.txt

citas.txt

El comando cp funciona muy bien como mv, excepto que copia un archivo en lugar de moverlo. Podemos comprobar que hizo lo correcto usando ls con dos rutas como parámetros — como la mayoría de los comandos Unix, ls puede recibir múltiples rutas a la vez:

$ cp citas.txt tesis/anotaciones.txt

$ ls citas.txt tesis/anotaciones.txt

citas.txt tesis/anotaciones.txt

Para probar que hicimos una copia, eliminemos el archivo citas.txt del directorio actual y después ejecutemos el mismo ls de nuevo.

$ rm citas.txt

$ ls citas.txt tesis/anotaciones.txt

ls: no se puede acceder citas.txt: No existe el fichero o el directorio

tesis/anotaciones.txt

Esta vez el error nos dice que no se puede encontrar citas.txt en el directorio actual, pero encuentra la copia en tesis que no hemos borrado.