4

Maniputating Files and directories
Manipulando archivos y directorios

At this point, we are ready for some real work! This chapter will introduce five of the most frequently used Linux commands. The following commands are used for manipulating both files and directories:
En este punto, ¡estamos listos para un trabajo real! Este capítulo presentará cinco de los comandos de Linux más utilizados. Los siguientes comandos se utilizan para manipular archivos y directorios:

- cp -- Copy files and directories
  cp -- copia archivos y directorios
- mv -- Move/rename files and directories
  mv -- Mover / renombrar archivos y directorios
- mkdir -- Create directories
  mkdir -- Crear directorios
- rm -- Remove files and directories
  rm -- Eliminar archivos y directorios
- ln -- Create hard and symbolic links
  ln -- crea vínculos físicos y simbólicos

Now, to be frank, some of the tasks performed by these commands are more easily done with a graphical file manager. With a file manager, we can drag and drop a file from one directory to another, cut and paste files, delete files, and so on. So why use these old command line programs?
Ahora, para ser franco, algunas de las tareas realizadas por estos comandos se realizan más fácilmente con un administrador de archivos gráfico. Con un administrador de archivos, podemos arrastrar y soltar un archivo de un directorio a otro, cortar y pegar archivos, eliminar archivos, etc. Entonces, ¿por qué usar estos viejos programas de línea de comandos?

The answer is power and flexibility. While it is easy to perform simple file manipulations with a graphical file manager, complicated tasks can be easier with the command line programs. For example, how could we copy all the HTML files from one directory to another but copy only files that do not exist in the destination directory or are newer than the versions in the destination directory? It's pretty hard with a file manager but pretty easy with the command line.
La respuesta es poder y flexibilidad. Si bien es fácil realizar manipulaciones simples de archivos con un administrador de archivos gráfico, las tareas complicadas pueden ser más fáciles con los programas de línea de comandos. Por ejemplo, ¿cómo podríamos copiar todos los archivos HTML de un directorio a otro pero copiar solo los archivos que no existen en el directorio de destino o son más recientes que las versiones en

el directorio de destino? Es bastante difícil con un administrador de archivos, pero bastante fácil con la línea de comandos.

```
cp -u *.html destination
```

Wildcards
Comodines

Before we begin using our commands, we need to talk about a shell feature that makes these commands so powerful. Because the shell uses filenames so much, it provides special characters to help you rapidly specify groups of filenames. These special characters are called wildcards. Using wildcards (which is also known as globbing) allows you to select filenames based on patterns of characters. Table 4-1 lists the wildcards and what they select.
Antes de comenzar a usar nuestros comandos, necesitamos hablar sobre una función de shell que hace que estos comandos sean tan poderosos. Debido a que el shell usa tanto nombres de archivo, proporciona caracteres especiales para ayudarlo a especificar rápidamente grupos de nombres de archivo. Estos caracteres especiales se denominan comodines. El uso de comodines (que también se conoce como globbing) le permite seleccionar nombres de archivos basados en patrones de caracteres. La Tabla 4-1 enumera los comodines y lo que seleccionan.

> Table 4-1: Wildcards
> Tabla 4-1: Comodines

Wildcard **Comodin**	**Meaning** **Significado**
*	Matches any characters Coincide con cualquier personaje
?	Matches any single character Coincide con cualquier carácter individual
[characters]	Matches any character that is a member of the set characters Coincide con cualquier personaje que sea miembro del conjunto de caracteres
[!characters]	Matches any character that is not a member of the set characters Coincide con cualquier personaje que no sea miembro del conjunto de caracteres
[[:class:]]	Matches any character that is a member of the specified class Coincide con cualquier personaje que sea miembro de la clase especificada

> Table 4-2 lists the most commonly used character classes.
> La Tabla 4-2 enumera las clases de caracteres más utilizadas.

> Table 4-2: Commonly Used Character Classes
> Tabla 4-2: Clases de caracteres más utilizadas

Character class	**Meaning**

Character class	Meaning
[:alnum:]	Matches any alphanumeric character Coincide con cualquier carácter alfanumérico
[:alpha:]	Matches any alphabetic character Coincide con cualquier carácter alfabético
[:digit:]	Matches any numeral Coincide con cualquier numeral
[:lower:]	Matches any lowercase letter Coincide con cualquier letra minúscula
[:upper:]	Matches any uppercase letter Coincide con cualquier letra mayúscula

Using wildcards makes it possible to construct sophisticated selection criteria for filenames. Table 4-3 provides some examples of patterns and what they match.
El uso de comodines permite construir criterios de selección sofisticados para nombres de archivos. La tabla 4-3 proporciona algunos ejemplos de patrones y sus coincidencias.

> Table 4-3: Wildcard Examples
> Tabla 4-3: Ejemplos de comodines

Pattern Patrón	Matches Partidos
*	All files Todos los ficheros
g*	Any file beginning with g Cualquier archivo que comience con g
b*.txt	Any file beginning with b followed by any characters and ending with .txt Cualquier archivo que comience con b seguido de cualquier carácter y termine con .txt
Data???	Any file beginning with Data followed by exactly three characters Cualquier archivo que comience con Data seguido de exactamente tres caracteres
[abc]*	Any file beginning with either an a, a b, or a c
BACKUP.[0-9][0-9][0-9]	Any file beginning with BACKUP. followed by exactly three numerals Cualquier archivo que comience con BACKUP. seguido de exactamente tres números
[[:upper:]]*	Any file beginning with an uppercase letter Cualquier archivo que comience con una letra mayúscula

Pattern Patrón	Matches Partidos
[![:digit:]]*	Any file not beginning with a numeral Cualquier archivo que no comience con un número
*[[:lower:]123]	Any file ending with a lowercase letter or the numerals 1, 2, or 3 Cualquier archivo que termine con una letra minúscula o los números 1, 2 o 3

Wildcards Work in the GUI, Too
Los comodines también funcionan en la GUI

Wildcards are especially valuable not only because they are used so frequently on the command line but because they are also supported by some graphical file managers.
Los comodines son especialmente valiosos no solo porque se usan con mucha frecuencia en la línea de comandos, sino porque también son compatibles con algunos administradores de archivos gráficos.

- In Nautilus (the file manager for GNOME), you can select files using the Edit4Select Pattern menu item. Just enter a file selection pattern with wildcards and the files in the currently viewed directory will be highlighted for selection.
  En Nautilus (el administrador de archivos para GNOME), puede seleccionar archivos usando el elemento de menú Edit4Select Pattern. Simplemente ingrese un patrón de selección de archivos con comodines y los archivos en el directorio visualizado actualmente se resaltarán para su selección.

- In some versions of Dolphin and Konqueror (the file managers for KDE), you can enter wildcards directly on the location bar. For example, if you want to see all the files starting with a lowercase u in the /usr/bin directory, enter /usr/bin/u* in the location bar and it will display the result.
  En algunas versiones de Dolphin y Konqueror (los administradores de archivos de KDE), puede ingresar comodines directamente en la barra de ubicación. Por ejemplo, si desea ver todos los archivos que comienzan con una u minúscula en el directorio /usr/bin, ingrese /usr/bin/u* en la barra de ubicación y mostrará el resultado.

Many ideas originally found in the command line interface make their way into the graphical interface, too. It is one of the many things that make the Linux desktop so powerful.
Muchas ideas que se encuentran originalmente en la interfaz de línea de comandos también se abren paso en la interfaz gráfica. Es una de las muchas cosas que hacen que el escritorio de Linux sea tan poderoso.

Wildcards can be used with any command that accepts filenames as arguments, but we'll talk more about that in Chapter 7.
Los comodines se pueden usar con cualquier comando que acepte nombres de archivo como argumentos, pero hablaremos más sobre eso en el Capítulo 7.

Character Ranges
Rangos de caracteres

> If you are coming from another Unix-like environment or have been reading some other books on this subject, you may have encountered the [A-Z] and [a-z] character range notations. These are traditional Unix notations and worked in older versions of Linux as well. They can still work, but you have to be careful with them because they will not produce the expected results unless properly configured. For now, you should avoid using them and use character classes instead.
> Si viene de otro entorno similar a Unix o ha estado leyendo algunos otros libros sobre este tema, es posible que haya encontrado las notaciones de rango de caracteres [A-Z] y [a-z]. Estas son notaciones tradicionales de Unix y también funcionaban en versiones anteriores de Linux. Todavía pueden funcionar, pero debe tener cuidado con ellos porque no producirán los resultados esperados a menos que estén configurados correctamente. Por ahora, debes evitar usarlos y usar clases de caracteres en su lugar.

mkdir -- Create Directories
mkdir - Crear directorios

The mkdir command is used to create directories. It works like this:
El comando mkdir se usa para crear directorios. Funciona así:

```
mkdir directory...
```

Note that when three periods follow an argument in the description of a command (as in the preceding example), it means that the argument can be repeated. Thus, the following command:
Tenga en cuenta que cuando tres puntos siguen a un argumento en la descripción de un comando (como en el ejemplo anterior), significa que el argumento se puede repetir. Así, el siguiente comando:

```
mkdir dir1
```

would create a single directory named dir1. This command
crearía un único directorio llamado dir1. Este comando

```
mkdir dir1 dir2 dir3
```

would create three directories named dir1, dir2, and dir3.
crearía tres directorios llamados dir1, dir2 y dir3.

cp -- Copy Files and Directories
cp - Copiar archivos y directorios

The cp command copies files or directories. It can be used two different ways. The following:
El comando cp copia archivos o directorios. Se puede utilizar de dos formas diferentes. La siguiente:

```
cp item1 item2
```

copies the single file or directory item1 to the file or directory item2. This command:
copia el único archivo o directorio item1 en el archivo o directorio item2. Este comando:

```
cp item... directory
```

copies multiple items (either files or directories) into a directory.
copia varios elementos (archivos o directorios) en un directorio.

Useful Options and Examples
Opciones y ejemplos útiles

Table 4-4 describes some of the commonly used options (the short option and the equivalent long option) for cp.
La Tabla 4-4 describe algunas de las opciones más utilizadas (la opción corta y la opción larga equivalente) para cp.

> Table 4-4: cp Options
> Tabla 4-4: Opciones de cp

Option	Meaning
-a , --archive	Copy the files and directories and all of their attributes, including ownerships and permissions. Normally, copies take on the default attributes of the user performing the copy. We'll take a look at file permissions in Chapter 9. Copie los archivos y directorios y todos sus atributos, incluidas las propiedades y los permisos. Normalmente, las copias adoptan los atributos predeterminados del usuario que realiza la copia. Echaremos un vistazo a los permisos de archivos en el Capítulo 9.
-i , --interactive	Before overwriting an existing file, prompt the user for confirmation. If this option is not specified, cp will silently (meaning there will be no warning) overwrite files. Antes de sobrescribir un archivo existente, solicite confirmación al usuario. Si no se especifica esta opción, cp sobrescribirá los archivos en silencio (lo que significa que no habrá advertencias).
-r , --recursive	Recursively copy directories and their contents. This option (or the -a option) is required when copying directories. Copie de forma recursiva directorios y su contenido. Esta opción (o la opción -a) es necesaria al copiar directorios.
-u , --update	When copying files from one directory to another, only copy files that either don't exist or are newer than the existing corresponding files in the destination directory. This is useful when copying large numbers of files as it skips files that don't need to be copied. Al copiar archivos de un directorio a otro, solo copie los archivos que no existen o son más nuevos que los archivos correspondientes existentes en el directorio de destino. Esto es útil al copiar una gran cantidad de archivos, ya que omite archivos que no necesitan copiarse.
-v , --verbose	Display informative messages as the copy is performed. Muestra mensajes informativos a medida que se realiza la copia.

Table 4-5 provides some examples of these commonly used options.
La Tabla 4-5 proporciona algunos ejemplos de estas opciones de uso común.

Table 4-5: cp Examples
Tabla 4-5: Ejemplos de cp

Command	Results
cp file1 file2	Copy file1 to file2 . If file2 exists, it is overwritten with the contents of file1 . If file2 does not exist, it is created. Copie el archivo1 al archivo2. Si el archivo2 existe, se sobrescribe con el contenido del archivo1. Si el archivo2 no existe, se crea.
cp -i file1 file2	Same as previous command, except that if file2 exists, the user is prompted before it is overwritten. Igual que el comando anterior, excepto que si el archivo2 existe, se solicita al usuario antes de sobrescribirlo.
cp file1 file2 dir1	Copy file1 and file2 into directory dir1 . The directory dir1 must already exist. Copie el archivo1 y el archivo2 en el directorio dir1. El directorio dir1 ya debe existir.
cp dir1/* dir2	Using a wildcard, copy all the files in dir1 into dir2. The directory dir2 must already exist. Usando un comodín, copie todos los archivos en dir1 en dir2. El directorio dir2 ya debe existir.
cp -r dir1 dir2	Copy the contents of directory dir1 to directory dir2 . If directory dir2 does not exist, it is created and, after the copy, will contain the same contents as directory dir1 . If directory dir2 does exist, then directory dir1 (and its contents) will be copied into dir2. Copie el contenido del directorio dir1 al directorio dir2. Si el directorio dir2 no existe, se crea y, después de la copia, contendrá el mismo contenido que el directorio dir1. Si el directorio dir2 existe, entonces el directorio dir1 (y su contenido) se copiará en dir2.

mv -- Move and Rename Files
mv -- Mover y cambiar el nombre de archivos

The mv command performs both file moving and file renaming, depending on how it is used. In either case, the original filename no longer exists after the operation. mv is used in much the same way as cp , as shown here:
El comando mv realiza tanto el movimiento de archivos como el cambio de nombre, dependiendo de cómo se use. En cualquier caso, el nombre de archivo original ya no existe después de la operación. mv se usa de la misma manera que cp, como se muestra aquí:

```
mv item1 item2
```

to move or rename the file or directory item1 to item2. It's also used as follows:
para mover o cambiar el nombre del archivo o directorio item1 a item2. También se utiliza de la siguiente manera:

```
mv item... directory
```

to move one or more items from one directory to another.
para mover uno o más elementos de un directorio a otro.

Useful Options and Examples
Opciones y ejemplos útiles

mv shares many of the same options as cp , as described in Table 4-6.
mv comparte muchas de las mismas opciones que cp, como se describe en la Tabla 4-6.

> Table 4-6: mv Options
> Tabla 4-6: Opciones de mv

Option	Meaning
-i , --interactive	Before overwriting an existing file, prompt the user for confirmation. If this option is not specified, mv will silently overwrite files. Antes de sobrescribir un archivo existente, solicite confirmación al usuario. Si no se especifica esta opción, mv sobrescribirá silenciosamente los archivos.
-u , --update	When moving files from one directory to another, only move files that either don't exist or are newer than the existing corresponding files in the destination directory. Al mover archivos de un directorio a otro, solo mueva los archivos que no existen o que son más nuevos que los archivos correspondientes existentes en el directorio de destino.
-v , --verbose	Display informative messages as the move is performed. Muestra mensajes informativos a medida que se realiza el movimiento.

Table 4-7 provides some examples of using the mv command.
La Tabla 4-7 proporciona algunos ejemplos del uso del comando mv.

> Table 4-7: mv Examples
> Tabla 4-7: Ejemplos de mv

Command	Results
mv file1 file2	Move file1 to file2 . If file2 exists, it is overwritten with the contents of file1 . If file2 does not exist, it is created. In either case, file1 ceases to exist. Mueva file1 a file2. Si el archivo2 existe, se sobrescribe con el contenido del archivo1. Si el archivo2 no existe, se crea. En cualquier caso, file1 deja de existir.
mv -i file1 file2	Same as the previous command, except that if file2 exists, the user is prompted before it is overwritten. Igual que el comando anterior, excepto que si el archivo2 existe, se solicita al usuario antes de sobrescribirlo.
mv file1 file2 dir1	Move file1 and file2 into directory dir1 . The directory dir1 must already exist. Mueva file1 y file2 al directorio dir1. El directorio dir1 ya debe existir.

Command	Results
mv dir1 dir2	If directory dir2 does not exist, create directory dir2 and move the contents of directory dir1 into dir2 and delete directory dir1. If directory dir2 does exist, move directory dir1 (and its contents) into directory dir2. Si el directorio dir2 no existe, cree el directorio dir2 y mueva el contenido del directorio dir1 a dir2 y elimine el directorio dir1. Si el directorio dir2 existe, mueva el directorio dir1 (y su contenido) al directorio dir2.

rm -- Remove Files and Directories
rm - Eliminar archivos y directorios

The rm command is used to remove (delete) files and directories, as shown here:
El comando rm se usa para eliminar (eliminar) archivos y directorios, como se muestra aquí:

```
rm item...
```

where item is one or more files or directories.
donde item es uno o más archivos o directorios.

Useful Options and Examples
Opciones y ejemplos útiles

Table 4-8 describes some of the common options for rm.
La Tabla 4-8 describe algunas de las opciones comunes para rm.

> Table 4-8: rm Options
> Tabla 4-8: Opciones rm

Option	Meaning
-i , --interactive	Before deleting an existing file, prompt the user for confirmation. If this option is not specified, rm will silently delete files. Antes de eliminar un archivo existente, solicite confirmación al usuario. Si no se especifica esta opción, rm eliminará los archivos de forma silenciosa.
-r , --recursive	Recursively delete directories. This means that if a directory being deleted has subdirectories, delete them too. To delete a directory, this option must be specified. Eliminar directorios de forma recursiva. Esto significa que si un directorio que se está eliminando tiene subdirectorios, elimínelos también. Para eliminar un directorio, se debe especificar esta opción.
-f , --force	Ignore nonexistent files and do not prompt. This overrides the --interactive option. Ignore los archivos inexistentes y no pregunte. Esto anula la opción --interactive.
-v , --verbose	Display informative messages as the deletion is performed. Muestra mensajes informativos a medida que se realiza la eliminación.

Be Careful with rm!
¡Tenga cuidado con rm!

Unix-like operating systems such as Linux do not have an undelete command. Once you delete something with rm , it's gone. Linux assumes you're smart and you know what you're doing.
Los sistemas operativos similares a Unix, como Linux, no tienen un comando de recuperación. Una vez que eliminas algo con rm, desaparece. Linux asume que eres inteligente y sabes lo que estás haciendo.

Be particularly careful with wildcards. Consider this classic example. Let's say you want to delete just the HTML files in a directory. To do this, you type the following:
Tenga especial cuidado con los comodines. Considere este ejemplo clásico. Supongamos que desea eliminar solo los archivos HTML de un directorio. Para hacer esto, escribe lo siguiente:

```
rm *.html
```

This is correct, but if you accidentally place a space between the * and the .html like so:
Esto es correcto, pero si accidentalmente coloca un espacio entre * y .html así:

```
rm * .html
```

the rm command will delete all the files in the directory and then complain that there is no file called .html.
el comando rm eliminará todos los archivos en el directorio y luego se quejará de que no hay ningún archivo llamado .html.

Here is a useful tip: whenever you use wildcards with rm (besides carefully checking your typing!), test the wildcard first with ls . This will let you see the files that will be deleted. Then press the up arrow to recall the command and replace ls with rm.
Aquí hay un consejo útil: siempre que use comodines con rm (¡además de verificar cuidadosamente su escritura!), Pruebe el comodín primero con ls. Esto le permitirá ver los archivos que se eliminarán. Luego presione la flecha hacia arriba para recuperar el comando y reemplace ls con rm.

Table 4-9 provides some examples of using the rm command.
La tabla 4-9 proporciona algunos ejemplos de uso del comando rm

Table 4-9: rm Examples
Tabla 4-9: Ejemplos de rm

Command	Results
rm file1	Delete file1 silently. Elimina el archivo1 en silencio.
rm -i file1	Same as the previous command, except that the user is prompted for confirmation before the deletion is performed. Igual que el comando anterior, excepto que se solicita al usuario que confirme antes de realizar la eliminación.

Command	Results
rm -r file1 dir1	Delete file1 and dir1 and its contents. Elimina file1 y dir1 y su contenido.
rm -rf file1 dir1	Same as the previous command, except that if either file1 or dir1 does not exist, rm will continue silently. Igual que el comando anterior, excepto que si file1 o dir1 no existe, rm continuará silenciosamente.

ln -- Create Links
ln - Crear enlaces

The ln command is used to create either hard or symbolic links. It is used in one of two ways. The following creates a hard link:
El comando ln se utiliza para crear vínculos físicos o simbólicos. Se utiliza de dos formas. Lo siguiente crea un vínculo físico:

```
ln file link
```

The following creates a symbolic link:
Lo siguiente crea un enlace simbólico:

```
ln -s item link
```

where item is either a file or a directory.
donde item es un archivo o un directorio.

Hard Links
Enlaces duros

Hard links are the original Unix way of creating links, compared to symbolic links, which are more modern. By default, every file has a single hard link that gives the file its name. When we create a hard link, we create an additional directory entry for a file. Hard links have two important limitations.
Los enlaces duros son la forma original de Unix de crear enlaces, en comparación con los enlaces simbólicos, que son más modernos. De forma predeterminada, cada archivo tiene un único vínculo físico que le da su nombre. Cuando creamos un enlace físico, creamos una entrada de directorio adicional para un archivo. Los enlaces duros tienen dos limitaciones importantes.

- A hard link cannot reference a file outside its own file system. This means a link cannot reference a file that is not on the same disk partition as the link itself.
  Un vínculo físico no puede hacer referencia a un archivo fuera de su propio sistema de archivos. Esto significa que un vínculo no puede hacer referencia a un archivo que no esté en la misma partición de disco que el vínculo en sí.

- A hard link may not reference a directory.
  Es posible que un vínculo físico no haga referencia a un directorio.

A hard link is indistinguishable from the file itself. Unlike a symbolic link, when you list a directory containing a hard link, you will see no special indication of the link. When a hard link is deleted, the link is removed, but the contents of the file itself continue to exist (that is, its space is not deallocated) until all links to the file are deleted.
Un enlace físico es indistinguible del archivo en sí. A diferencia de un enlace simbólico, cuando enumera un directorio que contiene un enlace físico, no verá ninguna indicación especial del enlace. Cuando se elimina un vínculo físico, el vínculo se elimina, pero el contenido del archivo en sí continúa existiendo (es decir, su espacio no se desasigna) hasta que se eliminan todos los vínculos al archivo.

It is important to be aware of hard links because you might encounter them from time to time, but modern practice prefers symbolic links, which we will cover next.
Es importante estar al tanto de los enlaces físicos porque es posible que los encuentre de vez en cuando, pero la práctica moderna prefiere los enlaces simbólicos, que veremos a continuación.

Symbolic Links
Enlaces simbólicos

Symbolic links were created to overcome the limitations of hard links. They work by creating a special type of file that contains a text pointer to the referenced file or directory. In this regard, they operate in much the same way as a Windows shortcut, though of course they predate the Windows feature by many years.
Los enlaces simbólicos se crearon para superar las limitaciones de los enlaces duros. Funcionan creando un tipo especial de archivo que contiene un puntero de texto al archivo o directorio referenciado. En este sentido, operan de la misma manera que un acceso directo de Windows, aunque, por supuesto, son anteriores a la función de Windows en muchos años.

A file pointed to by a symbolic link and the symbolic link itself are largely indistinguishable from one another. For example, if you write something to the symbolic link, the referenced file is written to. When you delete a symbolic link, however, only the link is deleted, not the file itself. If the file is deleted before the symbolic link, the link will continue to exist but will point to nothing. In this case, the link is said to be broken. In many implementations, the ls command will display broken links in a distinguishing color, such as red, to reveal their presence.
Un archivo al que apunta un enlace simbólico y el enlace simbólico en sí mismo son en gran medida indistinguibles entre sí. Por ejemplo, si escribe algo en el enlace simbólico, se escribe en el archivo al que se hace referencia. Sin embargo, cuando elimina un enlace simbólico, solo se elimina el enlace, no el archivo en sí. Si el archivo se elimina antes del enlace simbólico, el enlace seguirá existiendo pero no apuntará a nada. En este caso, se dice que el vínculo está roto. En muchas implementaciones, el comando ls mostrará los enlaces rotos en un color distintivo, como el rojo, para revelar su presencia.

The concept of links can seem confusing, but hang in there. We're going to try all this stuff, and it will, ideally, become clear.
El concepto de enlaces puede parecer confuso, pero aguanta. Vamos a probar todas estas cosas y, idealmente, quedará claro.

Building a Playground
Construyendo un patio de juegos

Because we are going to do some real file manipulation, let's build a safe place to "play" with our file manipulation commands. First we need a directory to work in. We'll create one in our home directory and call it playground.
Debido a que vamos a realizar una manipulación real de archivos, creemos un lugar seguro para "jugar" con nuestros comandos de manipulación de archivos. Primero, necesitamos un directorio para trabajar. Crearemos uno en nuestro directorio de inicio y lo llamaremos área de juegos.

Creating Directories
Creando Directorios

The mkdir command is used to create a directory. To create our playground directory, we will first make sure we are in our home directory and will then create the new directory.
El comando mkdir se usa para crear un directorio. Para crear nuestro directorio de juegos, primero nos aseguraremos de estar en nuestro directorio de inicio y luego crearemos el nuevo directorio.

```
[me@linuxbox ~]$ cd
[me@linuxbox ~]$ mkdir playground
```

To make our playground a little more interesting, let's create a couple of directories inside it called dir1 and dir2. To do this, we will change our current working directory to playground and execute another mkdir.
Para hacer que nuestro patio de juegos sea un poco más interesante, creemos un par de directorios dentro de él llamados dir1 y dir2. Para hacer esto, cambiaremos nuestro directorio de trabajo actual a playground y ejecutaremos otro mkdir.

```
[me@linuxbox ~]$ cd playground
[me@linuxbox playground]$ mkdir dir1 dir2
```

Notice that the mkdir command will accept multiple arguments allowing us to create both directories with a single command.
Tenga en cuenta que el comando mkdir aceptará varios argumentos, lo que nos permitirá crear ambos directorios con un solo comando.

Copying Files
Copiando documentos

Next, let's get some data into our playground. We'll do this by copying a file. Using the cp command, we'll copy the passwd file from the /etc directory to the current working directory.
A continuación, introduzcamos algunos datos en nuestro campo de juego. Haremos esto copiando un archivo. Usando el comando cp, copiaremos el archivo passwd del directorio /etc al directorio de trabajo actual.

```
[me@linuxbox playground]$ cp /etc/passwd .
```

Notice how we used shorthand for the current working directory, the single trailing period. So now if we perform an ls , we will see our file.
Observe cómo usamos la abreviatura para el directorio de trabajo actual, el período final único. Entonces, si realizamos un ls, veremos nuestro archivo.

```
[me@linuxbox playground]$ ls -l
```

Now, just for fun, let's repeat the copy using the -v option (verbose) to see what it does.
Ahora, solo por diversión, repitamos la copia usando la opción -v (detallada) para ver qué hace.

```
[me@linuxbox playground]$ cp -v /etc/passwd .
`/etc/passwd' -> `./passwd'
```

The cp command performed the copy again but this time displayed a concise message indicating what operation it was performing. Notice that cp overwrote the first copy without any warning. Again, this is a case of cp assuming that we know what we're doing. To get a warning, we'll include the -i (interactive) option.
El comando cp volvió a realizar la copia, pero esta vez mostró un mensaje conciso que indicaba qué operación estaba realizando. Observe que cp sobrescribió la primera copia sin ninguna advertencia. Nuevamente, este es un caso de cp asumiendo que sabemos lo que estamos haciendo. Para recibir una advertencia, incluiremos la opción -i (interactiva).

```
[me@linuxbox playground]$ cp -i /etc/passwd .
cp: overwrite `./passwd'?
```

Responding to the prompt by entering a y will cause the file to be overwritten; any other character (for example, n) will cause cp to leave the file alone.
Si responde al mensaje ingresando una y, el archivo se sobrescribirá; cualquier otro carácter (por ejemplo, n) hará que cp deje el archivo en paz.

Moving and Renaming Files
Mover y renombrar archivos

Now, the name passwd doesn't seem very playful and this is a playground, so let's change it to something else.
Ahora, el nombre passwd no parece muy divertido y esto es un patio de recreo, así que cambiémoslo por otro.

```
[me@linuxbox playground]$ mv passwd fun
```

Let's pass the fun around a little by moving our renamed file to each of the directories and back again. The following moves it first to the directory dir1:
Pasemos la diversión un poco moviendo nuestro archivo renombrado a cada uno de los directorios y viceversa. Lo siguiente lo mueve primero al directorio dir1:

```
[me@linuxbox playground]$ mv fun dir1
```

The following then moves it from dir1 to dir2:
A continuación, lo siguiente lo mueve de dir1 a dir2:

```
[me@linuxbox playground]$ mv dir1/fun dir2
```

Finally, this command brings it back to the current working directory:
Finalmente, este comando lo devuelve al directorio de trabajo actual:

```
[me@linuxbox playground]$ mv dir2/fun .
```

Next, let's see the effect of mv on directories. First we will move our data file into dir1 again, like this:
A continuación, veamos el efecto de mv en los directorios. Primero, volveremos a mover nuestro archivo de datos a dir1, así:

```
[me@linuxbox playground]$ mv fun dir1
```

Then we move dir1 into dir2 and confirm it with ls.
Luego movemos dir1 a dir2 y lo confirmamos con ls.

```
[me@linuxbox playground]$ mv dir1 dir2
[me@linuxbox playground]$ ls -l dir2
total 4
drwxrwxr-x 2 me me 4096 2018-01-11 06:06 dir1
[me@linuxbox playground]$ ls -l dir2/dir1
total 4
-rw-r--r-- 1 me me 1650 2018-01-10 16:33 fun
```

Note that because dir2 already existed, mv moved dir1 into dir2. If dir2 had not existed, mv would have renamed dir1 to dir2. Lastly, let's put everything back.
Tenga en cuenta que debido a que dir2 ya existía, mv movió dir1 a dir2. Si dir2 no hubiera existido, mv habría cambiado el nombre de dir1 a dir2. Por último, devolvamos todo.

```
[me@linuxbox playground]$ mv dir2/dir1.
[me@linuxbox playground]$ mv dir1/fun.
```

Creating Hard Links
Crear enlaces duros

Now we'll try some links. We'll first create some hard links to our data file, like so:
Ahora probaremos algunos enlaces. Primero crearemos algunos vínculos físicos a nuestro archivo de datos, así:

```
[me@linuxbox playground]$ ln fun fun-hard
[me@linuxbox playground]$ ln fun dir1/fun-hard
[me@linuxbox playground]$ ln fun dir2/fun-hard
```

So now we have four instances of the file fun. Let's take a look at our playground directory.
Así que ahora tenemos cuatro instancias del archivo divertido. Echemos un vistazo a nuestro directorio de juegos.

```
[me@linuxbox playground]$ ls -l
```

One thing we notice is that both the second fields in the listings for fun and fun-hard contain a 4, which is the number of hard links that now exist for the file. Remember that a file will always have at least one link because the file's name is created by a link. So, how do we know that fun and fun-hard are, in fact, the same file? In this case, ls is not very helpful. While we can see that fun and fun-hard are both the same size (field 5), our listing provides no way to be sure. To solve this problem, we're going to have to dig a little deeper.
Una cosa que notamos es que tanto el segundo campo en los listados para diversión como para diversión contienen un 4, que es el número de enlaces físicos que existen ahora para el archivo. Recuerde que un archivo siempre tendrá al menos un enlace porque el nombre del archivo se crea mediante un enlace. Entonces, ¿cómo sabemos que diversión y diversión son, de hecho, el mismo archivo? En este caso, ls no es muy útil. Si bien podemos ver que la diversión y la diversión tienen el mismo tamaño (campo 5), nuestra lista no proporciona ninguna forma de estar seguro. Para resolver este problema, tendremos que profundizar un poco más.

When thinking about hard links, it is helpful to imagine that files are made up of two parts.
Al pensar en enlaces físicos, es útil imaginar que los archivos se componen de dos partes.

- The data part containing the file's contents
  La parte de datos que contiene el contenido del archivo.
- The name part that holds the file's name
  La parte del nombre que contiene el nombre del archivo.

When we create hard links, we are actually creating additional name parts that all refer to the same data part. The system assigns a chain of disk blocks to what is called an inode, which is then associated with the name part. Each hard link therefore refers to a specific inode containing the file's contents.
Cuando creamos enlaces físicos, en realidad estamos creando partes de nombre adicionales que se refieren a la misma parte de datos. El sistema asigna una cadena de bloques de disco a lo que se llama un inodo, que luego se asocia con la parte del nombre. Por lo tanto, cada enlace físico se refiere a un inodo específico que contiene el contenido del archivo.

The ls command has a way to reveal this information. It is invoked with the -i option.
El comando ls tiene una forma de revelar esta información. Se invoca con la opción -i.

```
[me@linuxbox playground]$ ls -li
```

In this version of the listing, the first field is the inode number and, as we can see, both fun and fun-hard share the same inode number, which confirms they are the same file.
En esta versión de la lista, el primer campo es el número de inodo y, como podemos ver, tanto divertido como divertido comparten el mismo número de inodo, lo que confirma que son el mismo archivo.

Creating Symbolic Links
Creación de enlaces simbólicos

Symbolic links were created to overcome the two disadvantages of hard links:
Los enlaces simbólicos se crearon para superar las dos desventajas de los enlaces duros:

- Hard links cannot span physical devices.
  Los enlaces duros no pueden abarcar dispositivos físicos.
- Hard links cannot reference directories, only files.
  Los enlaces duros no pueden hacer referencia a directorios, solo archivos.

Symbolic links are a special type of file that contains a text pointer to the target file or directory.
Los enlaces simbólicos son un tipo especial de archivo que contiene un puntero de texto al archivo o directorio de destino.

Creating symbolic links is similar to creating hard links.
La creación de enlaces simbólicos es similar a la creación de enlaces físicos.

```
[me@linuxbox playground]$ ln -s fun fun-sym
[me@linuxbox playground]$ ln -s ../fun dir1/fun-sym
[me@linuxbox playground]$ ln -s ../fun dir2/fun-sym
```

The first example is pretty straightforward; we simply add the -s option to create a symbolic link rather than a hard link. But what about the next two? Remember that when we create a symbolic link, we are creating a text description of where the target file is relative to the symbolic link. It's easier to see if we look at the ls output, shown here:
El primer ejemplo es bastante sencillo; simplemente agregamos la opción -s para crear un enlace simbólico en lugar de un enlace físico. Pero, ¿qué pasa con los dos siguientes? Recuerde que cuando creamos un enlace simbólico, estamos creando una descripción de texto de dónde está el archivo de destino en relación con el enlace simbólico. Es más fácil ver si miramos la salida de ls, que se muestra aquí:

```
[me@linuxbox playground]$ ls -l dir1
```

The listing for fun-sym in dir1 shows that it is a symbolic link by the leading l in the first field and that it points to ../fun, which is correct. Relative to the location of fun-sym, fun is in the directory above it. Notice, too, that the length of the symbolic link file is 6, the number of characters in the string ../fun rather than the length of the file to which it is pointing.
La lista de fun-sym en dir1 muestra que es un enlace simbólico por la l inicial en el primer campo y que apunta a ../fun, lo cual es correcto. En relación con la ubicación de fun-sym, fun está en el directorio de

arriba. Observe también que la longitud del archivo de enlace simbólico es 6, el número de caracteres en la cadena ../fun en lugar de la longitud del archivo al que apunta.

When creating symbolic links, you can either use absolute pathnames, as shown here:
Al crear enlaces simbólicos, puede utilizar nombres de ruta absolutos, como se muestra aquí:

```
[me@linuxbox playground]$ ln -s /home/me/playground/fun dir1/fun-sym
```

or relative pathnames, as we did in our earlier example. In most cases, using relative pathnames is more desirable because it allows a directory tree containing symbolic links and their referenced files to be renamed and/or moved without breaking the links.
o nombres de ruta relativos, como hicimos en nuestro ejemplo anterior. En la mayoría de los casos, es más deseable usar nombres de ruta relativos porque permite que un árbol de directorios que contiene enlaces simbólicos y sus archivos referenciados se renombre y / o mueva sin romper los enlaces.

In addition to regular files, symbolic links can also reference directories.
Además de los archivos normales, los enlaces simbólicos también pueden hacer referencia a directorios.

```
[me@linuxbox playground]$ ln -s dir1 dir1-sym
[me@linuxbox playground]$ ls -l
```

Removing Files and Directories
Eliminar archivos y directorios

As we covered earlier, the rm command is used to delete files and directories. We are going to use it to clean up our playground a little bit. First, let's delete one of our hard links.
Como cubrimos anteriormente, el comando rm se usa para eliminar archivos y directorios. Lo usaremos para limpiar un poco nuestro patio de recreo. Primero, eliminemos uno de nuestros enlaces físicos.

```
[me@linuxbox playground]$ rm fun-hard
[me@linuxbox playground]$ ls -l
```

That worked as expected. The file fun-hard is gone, and the link count shown for fun is reduced from four to three, as indicated in the second field of the directory listing. Next, we'll delete the file fun, and just for enjoyment, we'll include the -i option to show what that does.
Eso funcionó como se esperaba. El archivo fun-hard desapareció y el recuento de enlaces que se muestra por diversión se reduce de cuatro a tres, como se indica en el segundo campo de la lista del directorio. A continuación, eliminaremos el archivo de diversión y, solo por diversión, incluiremos la opción -i para mostrar lo que hace.

```
[me@linuxbox playground]$ rm -i fun
rm: remove regular file `fun'?
```

Enter y at the prompt and the file is deleted. But let's look at the output of ls now. Notice what happened to fun-sym ? Because it's a symbolic link pointing to a now-nonexistent file, the link is broken.
Introduzca y cuando se le solicite y el archivo se eliminará. Pero veamos ahora la salida de ls. ¿Observa lo que pasó con fun-sym? Debido a que es un enlace simbólico que apunta a un archivo que ahora no existe, el enlace está roto.

```
[me@linuxbox playground]$ ls -l
```

Most Linux distributions configure ls to display broken links. The presence of a broken link is not in and of itself dangerous, but it is rather messy. If we try to use a broken link, we will see this:
La mayoría de las distribuciones de Linux configuran ls para mostrar enlaces rotos. La presencia de un enlace roto no es peligrosa en sí misma, pero es bastante complicada. Si intentamos usar un enlace roto, veremos esto:

```
[me@linuxbox playground]$ less fun-sym
fun-sym: No such file or directory
```

Let's clean up a little. We'll delete the symbolic links here:
Limpiemos un poco. Eliminaremos los enlaces simbólicos aquí:

```
[me@linuxbox playground]$ rm fun-sym dir1-sym
[me@linuxbox playground]$ ls -l
```

One thing to remember about symbolic links is that most file operations are carried out on the link's target, not the link itself. rm is an exception. When you delete a link, it is the link that is deleted, not the target.
Una cosa para recordar acerca de los enlaces simbólicos es que la mayoría de las operaciones de archivos se llevan a cabo en el destino del enlace, no en el enlace en sí. rm es una excepción. Cuando elimina un enlace, es el enlace el que se elimina, no el destino.

Finally, we will remove our playground. To do this, we will return to our home directory and use rm with the recursive option (-r) to delete playground and all of its contents, including its subdirectories.
Finalmente, eliminaremos nuestro patio de recreo. Para hacer esto, regresaremos a nuestro directorio de inicio y usaremos rm con la opción recursiva (-r) para eliminar el patio de recreo y todo su contenido, incluyendo sus subdirectorios.

```
[me@linuxbox playground]$ cd
[me@linuxbox ~]$ rm -r playground
```

Creating Symlinks with the GUI
Creando Symlinks con la GUI

> The file managers in both GNOME and KDE provide an easy and automatic method of creating symbolic links. With GNOME, holding the ctrl - shift keys while dragging a file will create a link rather than copying (or moving) the file. In KDE, a small menu appears whenever a file is dropped, offering a choice of copying, moving, or linking the file.
> Los administradores de archivos de GNOME y KDE proporcionan un método sencillo y automático para crear enlaces simbólicos. Con GNOME, si mantiene presionadas las teclas ctrl - shift mientras arrastra un archivo, se creará un enlace en lugar de copiar (o mover) el archivo. En KDE, aparece un pequeño menú cada vez que se suelta un archivo, que ofrece la opción de copiar, mover o vincular el archivo.

Summing Up
Resumen

We've covered a lot of ground here, and it will take a while for it all to fully sink in. Perform the playground exercise over and over until it makes sense. It is important to get a good understanding of basic file manipulation commands and wildcards. Feel free to expand on the playground exercise by adding more files and directories, using wildcards to specify files for various operations. The concept of links is a little confusing at first, but take the time to learn how they work. They can be a real lifesaver!
Hemos cubierto mucho terreno aquí y nos llevará un tiempo asimilarlo por completo. Realice el ejercicio del patio de recreo una y otra vez hasta que tenga sentido. Es importante comprender bien los comandos básicos de manipulación de archivos y los comodines. Siéntase libre de ampliar el ejercicio del patio de juegos agregando más archivos y directorios, utilizando comodines para especificar archivos para varias operaciones. El concepto de enlaces es un poco confuso al principio, pero tómese el tiempo para aprender cómo funcionan. ¡Pueden ser un verdadero salvavidas!