6

Redireccionamiento.

In this lesson we are going to unleash what may be the coolest feature of the command line. It's called I/O redirection. The "I/O" stands for input/output, and with this facility you can redirect the input and output of commands to and from files, as well as connect multiple commands together into powerful command pipelines. To show off this facility, we will introduce the following commands:
En esta lección vamos a desatar lo que puede ser la característica más interesante de la línea de comandos. Se llama redireccionamiento de E / S. La "E / S" significa entrada / salida, y con esta función puede redirigir la entrada y salida de comandos hacia y desde archivos, así como conectar varios comandos juntos en potentes canalizaciones de comandos. Para mostrar esta facilidad, introduciremos los siguientes comandos:

- cat -- Contatenate files
  cat - Concatenar archivos
- sort -- Sort lines of text
  sort - ordena las líneas de texto
- uniq -- Report or omit repeated lines
  uniq - Informar u omitir líneas repetidas
- grep -- Print lines matching a pattern
  grep - líneas de impresión que coinciden con un patrón
- wc -- Print newline, word, and byte counts for each file
  wc - imprime recuentos de nuevas líneas, palabras y bytes para cada archivo
- head -- Output the first part of a file
  head - muestra la primera parte de un archivo
- tail -- Output the last part of a file
  tail - muestra la última parte de un archivo
- tee -- Read from standard input and write to standard output and files
  tee - lee desde la entrada estándar y escribe en archivos y salidas estándar

Standard Input, Output, and Error
Entrada, salida y error estándar

Many of the programs that we have used so far produce output of some kind. This output often consists of two types.

Muchos de los programas que hemos utilizado hasta ahora producen algún tipo de salida. Esta salida a menudo consta de dos tipos.

- The program's results; that is, the data the program is designed to produce
  Los resultados del programa; es decir, los datos que el programa está diseñado para producir
- Status and error messages that tell us how the program is getting along
  Mensajes de estado y error que nos dicen cómo va el programa

If we look at a command like ls , we can see that it displays its results and its error messages on the screen.
Si miramos un comando como ls, podemos ver que muestra sus resultados y sus mensajes de error en la pantalla.

Keeping with the Unix theme of "everything is a file," programs such as ls actually send their results to a special file called standard output (often expressed as stdout) and their status messages to another file called standard error (stderr). By default, both standard output and standard error are linked to the screen and not saved into a disk file.
Siguiendo el tema de Unix de "todo es un archivo", los programas como ls envían sus resultados a un archivo especial llamado salida estándar (a menudo expresado como stdout) y sus mensajes de estado a otro archivo llamado error estándar (stderr). De forma predeterminada, tanto la salida estándar como el error estándar están vinculados a la pantalla y no se guardan en un archivo de disco

In addition, many programs take input from a facility called standard input (stdin), which is, by default, attached to the keyboard.Además, muchos programas toman la entrada de una instalación llamada entrada estándar (stdin), que está, por defecto, adjunta al teclado.

I/O redirection allows us to change where output goes and where input comes from. Normally, output goes to the screen and input comes from the keyboard, but with I/O redirection, we can change that.
La redirección de E/S nos permite cambiar a dónde va la salida y de dónde viene la entrada. Normalmente, la salida va a la pantalla y la entrada proviene del teclado, pero con la redirección de E/S, podemos cambiar eso.

Redirecting Standard Output
Redirigir la salida estándar

I/O redirection allows us to redefine where standard output goes. To redirect standard output to another file instead of the screen, we use the > redirection operator followed by the name of the file. Why would we want to do this? It's often useful to store the output of a command in a file. For example, we could tell the shell to send the output of the ls command to the file ls-output.txt instead of the screen.
La redirección de E / S nos permite redefinir dónde va la salida estándar. Para redirigir la salida estándar a otro archivo en lugar de la pantalla, usamos el operador de redirección> seguido del nombre del archivo. ¿Por qué querríamos hacer esto? Suele ser útil almacenar la salida de un comando en un archivo. Por ejemplo, podríamos decirle al shell que envíe la salida del comando ls al archivo ls-output.txt en lugar de a la pantalla.

```
[me@linuxbox ~]$ ls -l /usr/bin > ls-output.txt
```

Here, we created a long listing of the /usr/bin directory and sent the results to the file ls-output.txt. Let's examine the redirected output of the command, shown here:
Aquí, creamos una lista larga del directorio / usr / bin y enviamos los resultados al archivo ls-output.txt. Examinemos la salida redirigida del comando, que se muestra aquí:

```
[me@linuxbox ~]$ ls -l ls-output.txt
-rw-rw-r-- 1 me me 167878 2018-02-01 15:07 ls-output.txt
```

Good -- a nice, large, text file. If we look at the file with less , we will see that the file ls-output.txt does indeed contain the results from our ls command.
Bien: un archivo de texto grande y agradable. Si miramos el archivo con menos, veremos que el archivo ls-output.txt de hecho contiene los resultados de nuestro comando ls.

```
[me@linuxbox ~]$ less ls-output.txt
```

Now, let's repeat our redirection test, but this time with a twist. We'll change the name of the directory to one that does not exist.
Ahora, repitamos nuestra prueba de redireccionamiento, pero esta vez con un giro. Cambiaremos el nombre del directorio por uno que no existe.

```
[me@linuxbox ~]$ ls -l /bin/usr > ls-output.txt
ls: cannot access /bin/usr: No such file or directory
```

We received an error message. This makes sense since we specified the nonexistent directory /bin/usr, but why was the error message displayed on the screen rather than being redirected to the file ls-output.txt? The answer is that the ls program does not send its error messages to standard output.
Recibimos un mensaje de error. Esto tiene sentido ya que especificamos el directorio inexistente / bin / usr, pero ¿por qué se mostró el mensaje de error en la pantalla en lugar de ser redirigido al archivo ls-output.txt? La respuesta es que el programa ls no envía sus mensajes de error a la salida estándar.

Instead, like most well-written Unix programs, it sends its error messages to standard error. Because we redirected only standard output and not standard error, the error message was still sent to the screen. We'll see how to redirect standard error in just a minute, but first let's look at what happened to our output file.
En cambio, como la mayoría de los programas Unix bien escritos, envía sus mensajes de error al error estándar. Debido a que redirigimos solo la salida estándar y no el error estándar, el mensaje de error aún se envió a la pantalla. Veremos cómo redirigir el error estándar en solo un minuto, pero primero veamos qué sucedió con nuestro archivo de salida.

```
[me@linuxbox ~]$ ls -l ls-output.txt
-rw-rw-r-- 1 me me 0 2018-02-01 15:08 ls-output.txt
```

The file now has zero length! This is because when we redirect output with the > redirection operator, the destination file is always rewritten from the beginning. Because our ls command generated no results and

only an error message, the redirection operation started to rewrite the file and then stopped because of the error, resulting in its truncation. In fact, if we ever need to actually truncate a file (or create a new, empty file), we can use a trick like this:
¡El archivo ahora tiene una longitud cero! Esto se debe a que cuando redirigimos la salida con el operador de redirección, el archivo de destino siempre se reescribe desde el principio. Debido a que nuestro comando ls no generó resultados y solo un mensaje de error, la operación de redirección comenzó a reescribir el archivo y luego se detuvo debido al error, lo que resultó en su truncamiento. De hecho, si alguna vez necesitamos truncar un archivo (o crear un archivo nuevo y vacío), podemos usar un truco como este:

```
[me@linuxbox ~]$ > ls-output.txt
```

Simply using the redirection operator with no command preceding it will truncate an existing file or create a new, empty file.
El simple hecho de usar el operador de redirección sin ningún comando precedente truncará un archivo existente o creará un nuevo archivo vacío.

So, how can we append redirected output to a file instead of overwriting the file from the beginning? For that, we use the >> redirection operator, like so:
Entonces, ¿cómo podemos agregar la salida redirigida a un archivo en lugar de sobrescribir el archivo desde el principio? Para eso, usamos el >> operador de redirección, así:

```
[me@linuxbox ~]$ ls -l /usr/bin >> ls-output.txt
```

Using the >> operator will result in the output being appended to the file. If the file does not already exist, it is created just as though the > operator had been used. Let's put it to the test.
El uso del operador >> dará como resultado que la salida se agregue al archivo. Si el archivo no existe, se crea como si se hubiera utilizado el operador >. Pongámoslo a prueba.

```
[me@linuxbox ~]$ ls -l /usr/bin >> ls-output.txt
[me@linuxbox ~]$ ls -l /usr/bin >> ls-output.txt
[me@linuxbox ~]$ ls -l /usr/bin >> ls-output.txt
[me@linuxbox ~]$ ls -l ls-output.txt
-rw-rw-r-- 1 hernani hernani 807716 feb 13 00:34 ls-output.txt
```

We repeated the command three times, resulting in an output file three times as large.
Repetimos el comando tres veces, lo que resultó en un archivo de salida tres veces más grande.

Redirecting Standard Error
Error estándar de redireccionamiento

Redirecting standard error lacks the ease of a dedicated redirection operator. To redirect standard error, we must refer to its file descriptor. A program can produce output on any of several numbered file streams. While we have referred to the first three of these file streams as standard input, output, and error, the shell references them internally as file descriptors 0, 1, and 2, respectively. The shell provides a notation for redirecting files using the file descriptor number. Because standard error is the same as file descriptor

number 2, we can redirect standard error with this notation:
El error estándar de redireccionamiento carece de la facilidad de un operador de redireccionamiento dedicado. Para redirigir el error estándar, debemos consultar su descriptor de archivo. Un programa puede producir resultados en cualquiera de varios flujos de archivos numerados. Si bien nos hemos referido a los tres primeros de estos flujos de archivos como entrada, salida y error estándar, el shell los hace referencia internamente como descriptores de archivo 0, 1 y 2, respectivamente. El shell proporciona una notación para redirigir archivos utilizando el número de descriptor de archivo. Debido a que el error estándar es el mismo que el descriptor de archivo número 2, podemos redirigir el error estándar con esta notación:

```
[me@linuxbox ~]$ ls -l /bin/usr 2> ls-error.txt
```

The file descriptor 2 is placed immediately before the redirection operator to perform the redirection of standard error to the file ls-error.txt.
El descriptor de archivo 2 se coloca inmediatamente antes del operador de redirección para realizar la redirección del error estándar al archivo ls-error.txt.

Redirecting Standard Output and Standard Error to One File
Redirigir la salida estándar y el error estándar a un archivo

There are cases in which we may want to capture all of the output of a command to a single file. To do this, we must redirect both standard output and standard error at the same time. There are two ways to do this. Shown here is the traditional way, which works with old versions of the shell:
Hay casos en los que es posible que deseemos capturar toda la salida de un comando en un solo archivo. Para hacer esto, debemos redirigir tanto la salida estándar como el error estándar al mismo tiempo. Hay dos maneras de hacer esto. Aquí se muestra la forma tradicional, que funciona con versiones antiguas del shell:

```
[me@linuxbox ~]$ ls -l /bin/usr > ls-output.txt 2>&1
```

Using this method, we perform two redirections. First we redirect standard output to the file ls-output.txt, and then we redirect file descriptor 2 (standard error) to file descriptor 1 (standard output) using the notation 2>&1.
Con este método, realizamos dos redirecciones. Primero redirigimos la salida estándar al archivo ls-output.txt, y luego redirigimos el descriptor de archivo 2 (error estándar) al descriptor de archivo 1 (salida estándar) usando la notación 2> & 1.

> # Notice That the Order of the Redirections Is Significant
> # Observe que el orden de las redirecciones es significativo

> The redirection of standard error must always occur after redirecting standard output or it doesn't work. The following example redirects standard error to the file ls-output.txt:
> La redirección del error estándar siempre debe ocurrir después de redireccionar la salida estándar o no funcionará. El siguiente ejemplo redirige el error estándar al archivo ls-output.txt:

```
ls-output.txt 2>&1
```

If the order is changed to the following, then standard error is directed to the screen:
Si el orden se cambia a lo siguiente, el error estándar se dirige a la pantalla:

```
2>&1 >ls-output.txt
```

Recent versions of bash provide a second, more streamlined method for performing this combined redirection, shown here:
Las versiones recientes de bash proporcionan un segundo método más simplificado para realizar esta redirección combinada, que se muestra aquí:

```
[me@linuxbox ~]$ ls -l /bin/usr &> ls-output.txt
```

In this example, we use the single notation &> to redirect both standard output and standard error to the file ls-output.txt. You may also append the standard output and standard error streams to a single file like so:
En este ejemplo, usamos la notación simple &> para redirigir tanto la salida estándar como el error estándar al archivo ls-output.txt. También puede agregar la salida estándar y los flujos de error estándar a un solo archivo de la siguiente manera:

```
[me@linuxbox ~]$ ls -l /bin/usr &>> ls-output.txt
```

Disposing of Unwanted Output
Eliminación de resultados no deseados

Sometimes "silence is golden" and we don't want output from a command; we just want to throw it away. This applies particularly to error and status messages. The system provides a way to do this by redirecting output to a special file called /dev/null. This file is a system device often referred to as a bit bucket, which accepts input and does nothing with it. To suppress error messages from a command, we do this:
A veces, "el silencio es oro" y no queremos salida de un comando; solo queremos tirarlo a la basura. Esto se aplica especialmente a los mensajes de error y estado. El sistema proporciona una forma de hacerlo redirigiendo la salida a un archivo especial llamado / dev / null. Este archivo es un dispositivo del sistema al que a menudo se hace referencia como un cubo de bits, que acepta entradas y no hace nada con ellas. Para suprimir los mensajes de error de un comando, hacemos esto:

```
[me@linuxbox ~]$ ls -l /bin/usr 2> /dev/null
```

> # /dev/null in Unix Culture
> # /dev/null en la cultura Unix

> The bit bucket is an ancient Unix concept, and because of its universality, it has appeared in many parts of Unix culture. When someone says they are sending your comments to /dev/null, now you know what it means. For more examples, see the Wikipedia article on /dev/null.
> El cubo de bits es un concepto antiguo de Unix y, debido a su universalidad, ha aparecido en muchas partes de la cultura Unix. Cuando alguien dice que está enviando sus comentarios a /dev/null, ahora sabe lo que significa. Para obtener más ejemplos, consulte el artículo de Wikipedia sobre /dev/null.

Redirecting Standard Input
Redirigir la entrada estándar

Up to now, we haven't encountered any commands that make use of standard input (actually we have, but we'll reveal that surprise a little bit later), so we need to introduce one.
Hasta ahora, no hemos encontrado ningún comando que haga uso de la entrada estándar (en realidad lo hemos hecho, pero revelaremos esa sorpresa un poco más adelante), por lo que debemos introducir uno.

cat -- Concatenate Files
cat - Concatenar archivos

The cat command reads one or more files and copies them to standard output like so:
El comando cat lee uno o más archivos y los copia en la salida estándar de la siguiente manera:

```
cat filename
```

In most cases, you can think of cat as being analogous to the TYPE command in DOS. You can use it to display files without paging. For example, the following will display the contents of the file ls-output.txt:
En la mayoría de los casos, puede pensar en cat como algo análogo al comando TYPE en DOS. Puede usarlo para mostrar archivos sin paginar. Por ejemplo, lo siguiente mostrará el contenido del archivo ls-output.txt:

```
[me@linuxbox ~]$ cat ls-output.txt
```

cat is often used to display short text files. Because cat can accept more than one file as an argument, it can also be used to join files together. Suppose we have downloaded a large file that has been split into multiple parts (multimedia files are often split this way on Usenet), and we want to join them back together. If the files were named as follows:
cat se utiliza a menudo para mostrar archivos de texto cortos. Debido a que cat puede aceptar más de un archivo como argumento, también se puede usar para unir archivos. Supongamos que hemos descargado un archivo grande que se ha dividido en varias partes (los archivos multimedia a menudo se dividen de esta manera en Usenet) y queremos volver a unirlos. Si los archivos se nombraron de la siguiente manera:

```
movie.mpeg.001 movie.mpeg.002 ... movie.mpeg.099
```

we could join them back together with this command:
podríamos volver a unirlos con este comando:

```
cat movie.mpeg.0* > movie.mpeg
```

Because wildcards always expand in sorted order, the arguments will be arranged in the correct order.
Dado que los comodines siempre se expanden en orden ordenado, los argumentos se organizarán en el

orden correcto.

This is all well and good, but what does this have to do with standard input? Nothing yet, but let's try something else. What happens if we enter cat with no arguments?
Todo esto está muy bien, pero ¿qué tiene esto que ver con la entrada estándar? Todavía no hay nada, pero intentemos algo más. ¿Qué pasa si entramos gato sin argumentos?

```
[me@linuxbox ~]$ cat
```

Nothing happens; it just sits there like it's hung. It might seem that way, but it's really doing exactly what it's supposed to do.
No pasa nada; simplemente se queda ahí como si estuviera colgado. Puede parecer así, pero realmente está haciendo exactamente lo que se supone que debe hacer.

If cat is not given any arguments, it reads from standard input, and since standard input is, by default, attached to the keyboard, it's waiting for us to type something! Try adding the following text and pressing enter:
Si no se le da ningún argumento a cat, lee de la entrada estándar, y dado que la entrada estándar está, por defecto, adjunta al teclado, ¡está esperando a que escribamos algo! Intente agregar el siguiente texto y presione enter:

```
[me@linuxbox ~]$ cat
The quick brown fox jumped over the lazy dog.
```

Next, type ctrl-D (i.e., hold down the ctrl key and press D) to tell cat that it has reached end of file (EOF) on standard input.
Luego, escriba ctrl-D (es decir, mantenga presionada la tecla ctrl y presione D) para decirle a cat que ha llegado al final del archivo (EOF) en la entrada estándar.

```
[me@linuxbox ~]$ cat
The quick brown fox jumped over the lazy dog.
The quick brown fox jumped over the lazy dog.
```

In the absence of filename arguments, cat copies standard input to standard output, so we see our line of text repeated. We can use this behavior to create short text files. Let's say we wanted to create a file called lazy_dog.txt containing the text in our example. We would do this:
En ausencia de argumentos de nombre de archivo, cat copia la entrada estándar en la salida estándar, por lo que vemos nuestra línea de texto repetida. Podemos usar este comportamiento para crear archivos de texto cortos. Supongamos que queremos crear un archivo llamado lazy_dog.txt que contiene el texto de nuestro ejemplo. Haríamos esto:

```
[me@linuxbox ~]$ cat > lazy_dog.txt
The quick brown fox jumped over the lazy dog.
```

Type the command followed by the text we want to place in the file. Remember to type ctrl -D at the end. Using the command line, we have implemented the world's dumbest word processor! To see our results, we can use cat to copy the file to stdout again.
Escriba el comando seguido del texto que queremos colocar en el archivo. Recuerde escribir ctrl -D al final. Utilizando la línea de comandos, hemos implementado el procesador de texto más tonto del mundo. Para ver nuestros resultados, podemos usar cat para copiar el archivo a stdout nuevamente.

```
[me@linuxbox ~]$ cat lazy_dog.txt
The quick brown fox jumped over the lazy dog.
```

Now that we know how cat accepts standard input, in addition to filename arguments, let's try redirecting standard input.
Ahora que sabemos cómo acepta cat la entrada estándar, además de los argumentos del nombre de archivo, intentemos redirigir la entrada estándar.

```
[me@linuxbox ~]$ cat < lazy_dog.txt
The quick brown fox jumped over the lazy dog.
```

Using the < redirection operator, we change the source of standard input from the keyboard to the file lazy_dog.txt. We see that the result is the same as passing a single filename argument. This is not particularly useful compared to passing a filename argument, but it serves to demonstrate using a file as a source of standard input. Other commands make better use of standard input, as we will soon see.
Usando el operador de redirección <, cambiamos la fuente de la entrada estándar del teclado al archivo lazy_dog.txt. Vemos que el resultado es el mismo que pasar un solo argumento de nombre de archivo. Esto no es particularmente útil en comparación con pasar un argumento de nombre de archivo, pero sirve para demostrar el uso de un archivo como fuente de entrada estándar. Otros comandos hacen un mejor uso de la entrada estándar, como veremos pronto.

Before we move on, check out the man page for cat because it has several interesting options.
Antes de continuar, consulte la página de manual de cat porque tiene varias opciones interesantes.

Pipelines
Canalizaciones

The capability of commands to read data from standard input and send to standard output is utilized by a shell feature called pipelines. Using the pipe operator | , the standard output of one command can be piped into the standard input of another.
La capacidad de los comandos para leer datos de la entrada estándar y enviarlos a la salida estándar es utilizada por una función de shell llamada canalizaciones. Usando el operador de tubería | , la salida estándar de un comando se puede canalizar a la entrada estándar de otro.

```
command1 | command2
```

To fully demonstrate this, we are going to need some commands. Remember how we said there was one we already knew that accepts standard input? It's less . We can use less to display, page by page, the output of any command that sends its results to standard output.
Para demostrar completamente esto, necesitaremos algunos comandos. ¿Recuerda que dijimos que ya sabíamos que aceptaba entradas estándar? Es menos. Podemos usar menos para mostrar, página por página, la salida de cualquier comando que envíe sus resultados a la salida estándar.

```
[me@linuxbox ~]$ ls -l /usr/bin | less
```

This is extremely handy! Using this technique, we can conveniently examine the output of any command that produces standard output.
¡Esto es extremadamente útil! Usando esta técnica, podemos examinar convenientemente la salida de cualquier comando que produzca una salida estándar.

Filters
Filtros

Pipelines are often used to perform complex operations on data. It is possible to put several commands together into a pipeline. Frequently, the commands used this way are referred to as filters. Filters take input, change it somehow, and then output it. The first one we will try is sort . Imagine we wanted to make a combined list of all the executable programs in /bin and /usr/bin, put them in sorted order, and view the resulting list.
Las canalizaciones se utilizan a menudo para realizar operaciones complejas con datos. Es posible poner varios comandos juntos en una canalización. Con frecuencia, los comandos utilizados de esta manera se denominan filtros. Los filtros toman la entrada, la cambian de alguna manera y luego la emiten. El primero que probaremos es sort. Imagine que queremos hacer una lista combinada de todos los programas ejecutables en / bin y / usr / bin, ponerlos en orden y ver la lista resultante.

```
[me@linuxbox ~]$ ls /bin /usr/bin | sort | less
```

Because we specified two directories (/bin and /usr/bin), the output of ls would have consisted of two sorted lists, one for each directory. By including sort in our pipeline, we changed the data to produce a single, sorted list.
Debido a que especificamos dos directorios (/bin y /usr/bin), la salida de ls habría consistido en dos listas ordenadas, una para cada directorio. Al incluir la ordenación en nuestra canalización, cambiamos los datos para producir una única lista ordenada.

The Difference Between > and |
La diferencia entre > y |

At first glance, it may be hard to understand the redirection performed by the pipeline operator | versus the redirection operator > . Simply put, the redirection operator connects a command with a file, while the pipeline operator connects the output of one command with the input of a second command.
A primera vista, puede resultar difícil comprender la redirección que realiza el operador de la canalización | versus el operador de redirección>. En pocas palabras, el operador de redirección conecta un comando con

un archivo, mientras que el operador de canalización conecta la salida de un comando con la entrada de un segundo comando.

```
command1 > file1

command1 | command2
```

A lot of people will try the following when they are learning about pipelines, “just to see what happens”:
Mucha gente intentará lo siguiente cuando aprendan sobre las canalizaciones, "solo para ver qué sucede":

```
command1 > command2
```

Answer: sometimes something really bad. Here is an actual example submitted by a reader who was administering a Linux-based server appliance. As the superuser, he did this:
Respuesta: a veces algo realmente malo. A continuación, se muestra un ejemplo real enviado por un lector que administraba un dispositivo de servidor basado en Linux. Como superusuario, hizo esto:

```
# cd /usr/bin
# ls > less
```

The first command put him in the directory where most programs are stored, and the second command told the shell to overwrite the file less with the output of the ls command. Since the /usr/bin directory already contained a file named less (the less program), the second command overwrote the less program file with the text from ls , thus destroying the less program on his system.
El primer comando lo puso en el directorio donde se almacenan la mayoría de los programas, y el segundo comando le dijo al shell que sobrescribiera menos el archivo con la salida del comando ls. Dado que el directorio / usr / bin ya contenía un archivo llamado less (el programa less), el segundo comando sobrescribió el archivo de programa less con el texto de ls, destruyendo así el programa less en su sistema.

The lesson here is that the redirection operator silently creates or overwrites files, so you need to treat it with a lot of respect.
La lección aquí es que el operador de redirección crea o sobrescribe archivos de manera silenciosa, por lo que debe tratarlos con mucho respeto.

uniq: Report or Omit Repeated Lines
uniq: Informar u omitir líneas repetidas

The uniq command is often used in conjunction with sort . uniq accepts a sorted list of data from either standard input or a single filename argument (see the uniq man page for details) and, by default, removes any duplicates from the list. So, to make sure our list has no duplicates (that is, any programs of the same name that appear in both the /bin and /usr/bin directories), we will add uniq to our pipeline.
El comando uniq se usa a menudo junto con sort. uniq acepta una lista ordenada de datos de la entrada estándar o de un solo argumento de nombre de archivo (consulte la página del manual de uniq para obtener más detalles) y, de forma predeterminada, elimina los duplicados de la lista. Entonces, para asegurarnos de que nuestra lista no tenga duplicados (es decir, cualquier programa

> con el mismo nombre que aparezca en los directorios /bin y /usr/bin), agregaremos uniq a nuestra canalización.

```
[me@linuxbox ~]$ ls /bin /usr/bin | sort | uniq | less
```

In this example, we use uniq to remove any duplicates from the output of the sort command. If we want to see the list of duplicates instead, we add the -d option to uniq like so:
En este ejemplo, usamos uniq para eliminar cualquier duplicado de la salida del comando sort. Si queremos ver la lista de duplicados en su lugar, agregamos la opción -d a uniq así:

```
[me@linuxbox ~]$ ls /bin /usr/bin | sort | uniq -d | less
```

wc - Print Line, Word, and Byte Counts
wc - Imprimir recuentos de líneas, palabras y bytes

The wc (word count) command is used to display the number of lines, words, and bytes contained in files. Here's an example:
El comando wc (recuento de palabras) se utiliza para mostrar el número de líneas, palabras y bytes contenidos en los archivos. Aquí tienes un ejemplo:

```
[me@linuxbox ~]$ wc ls-output.txt
7902 64566 503634 ls-output.txt
```

In this case, it prints out three numbers: lines, words, and bytes contained in ls-output.txt. Like our previous commands, if executed without command line arguments, wc accepts standard input. The -l option limits its output to report only lines. Adding it to a pipeline is a handy way to count things. To see the number of items we have in our sorted list, we can do this:
En este caso, imprime tres números: líneas, palabras y bytes contenidos en ls-output.txt. Como nuestros comandos anteriores, si se ejecuta sin argumentos de línea de comando, wc acepta entrada estándar. La opción -l limita su salida para informar solo líneas. Agregarlo a una canalización es una forma práctica de contar cosas. Para ver la cantidad de elementos que tenemos en nuestra lista ordenada, podemos hacer esto:

```
[me@linuxbox ~]$ ls /bin /usr/bin | sort | uniq | wc -l
2728
```

grep -- Print Lines Matching a Pattern
grep - Imprimir líneas que coinciden con un patrón

grep is a powerful program used to find text patterns within files. It's used like this:
grep es un programa poderoso que se usa para encontrar patrones de texto dentro de archivos. Se usa así:

```
grep pattern filename
```

When grep encounters a “pattern” in the file, it prints out the lines containing it. The patterns that grep can match can be very complex, but for now we will concentrate on simple text matches. We’ll cover the advanced patterns, called regular expressions, in Chapter 19.
Cuando grep encuentra un "patrón" en el archivo, imprime las líneas que lo contienen. Los patrones que grep puede hacer coincidir pueden ser muy complejos, pero por ahora nos concentraremos en coincidencias de texto simples. Cubriremos los patrones avanzados, llamados expresiones regulares, en el Capítulo 19.

Suppose we wanted to find all the files in our list of programs that had the word zip embedded in the name. Such a search might give us an idea of some of the programs on our system that had something to do with file compression. We would do this:
Supongamos que queremos encontrar todos los archivos en nuestra lista de programas que tienen la palabra zip incrustada en el nombre. Tal búsqueda podría darnos una idea de algunos de los programas de nuestro sistema que tienen algo que ver con la compresión de archivos. Haríamos esto:

```
[me@linuxbox ~]$ ls /bin /usr/bin | sort | uniq | grep zip
bunzip2
bzip2
gunzip
gzip
unzip
zip
zipcloak
zipgrep
zipinfo
zipnote
zipsplit
```

There are a couple of handy options for grep.
Hay un par de opciones útiles para grep

- -i , which causes grep to ignore case when performing the search (normally searches are case sensitive)
  -i, que hace que grep ignore las mayúsculas y minúsculas al realizar la búsqueda (normalmente las búsquedas distinguen entre mayúsculas y minúsculas)

- -v , which tells grep to print only those lines that do not match the pattern
  -v, que le dice a grep que imprima solo aquellas líneas que no coinciden con el patrón

head/tail: Print First/Last Part of Files
head / tail: Imprime la primera / última parte de los archivos

Sometimes you don’t want all the output from a command. You might want only the first few lines or the last few lines. The head command prints the first 10 lines of a file, and the tail command prints the last 10 lines. By default, both commands print 10 lines of text, but this can be adjusted with the -n option.
A veces no desea todo el resultado de un comando. Es posible que desee solo las primeras líneas o las

últimas líneas. El comando head imprime las primeras 10 líneas de un archivo y el comando tail imprime las últimas 10 líneas. De forma predeterminada, ambos comandos imprimen 10 líneas de texto, pero esto se puede ajustar con la opción -n.

```
[me@EliteDesk:~]$ head -n 5 ls-output.txt
total 1072480
-rwxr-xr-x 1 root root          59736 sep  5  2019 [
-rwxr-xr-x 1 root root             96 ago  4  2020 2to3-2.7
-rwxr-xr-x 1 root root          10104 abr 23  2016 411toppm
-rwxr-xr-x 1 root root             39 ago  9  2019 7z
[me@EliteDesk:~]$ tail -n 5 ls-output.txt
-rwxr-xr-x 1 root root          93584 abr 21  2017 zipsplit
-rwxr-xr-x 1 root root          26952 ene 30  2020 zjsdecode
-rwxr-xr-x 1 root root           2206 dic 13  2019 zless
-rwxr-xr-x 1 root root           1842 dic 13  2019 zmore
-rwxr-xr-x 1 root root           4553 dic 13  2019 znew
```

These can be used in pipelines as well:
Estos también se pueden usar en tuberías:

```
[me@linuxbox ~]$ ls /usr/bin | tail -n 5
znew
zonetab2pot.py
zonetab2pot.pyc
zonetab2pot.pyo
zsoelim
```

tail has an option that allows you to view files in real time. This is useful for watching the progress of log files as they are being written. In the following example, we will look at the messages file in /var/log (or the /var/log/syslog file if messages is missing). Superuser privileges are required to do this on some Linux distributions because the /var/log/messages file might contain security information.
tail tiene una opción que le permite ver archivos en tiempo real. Esto es útil para observar el progreso de los archivos de registro a medida que se escriben. En el siguiente ejemplo, veremos el archivo de mensajes en /var/log (o el archivo /var/log/syslog si faltan mensajes). Se requieren privilegios de superusuario para hacer esto en algunas distribuciones de Linux porque el archivo /var/log/messages puede contener información de seguridad.

```
[me@linuxbox ~]$ tail -f /var/log/messages
```

Using the -f option, tail continues to monitor the file, and when new lines are appended, they immediately appear on the display. This continues until you type ctrl-C.
Usando la opción -f, tail continúa monitoreando el archivo, y cuando se agregan nuevas líneas, aparecen inmediatamente en la pantalla. Esto continúa hasta que escribe ctrl-C.

tee -- Read from Stdin and Output to Stdout and Files
tee - Leer desde Stdin y enviar a Stdout y archivos

In keeping with our plumbing metaphor, Linux provides a command called tee that creates a “tee” fitting on our pipe. The tee program reads standard input and copies it to both standard output (allowing the data to continue down the pipeline) and to one or more files. This is useful for capturing a pipeline’s contents at an intermediate stage of processing.
De acuerdo con nuestra metáfora de la plomería, Linux proporciona un comando llamado tee que crea una conexión en "tee" en nuestra tubería. El programa tee lee la entrada estándar y la copia tanto en la salida estándar (permitiendo que los datos continúen en la tubería) como en uno o más archivos. Esto es útil para capturar el contenido de una canalización en una etapa intermedia de procesamiento.

Here we repeat one of our earlier examples, this time including tee to capture the entire directory listing to the file ls.txt before grep filters the pipeline’s contents:
Aquí repetimos uno de nuestros ejemplos anteriores, esta vez incluyendo tee para capturar la lista completa del directorio en el archivo ls.txt antes de que grep filtre el contenido de la canalización:

```
[me@linuxbox ~]$ ls /usr/bin | tee ls.txt | grep zip
bunzip2
bzip2
gunzip
gzip
unzip
zip
zipcloak
zipgrep
zipinfo
zipnote
zipsplit
```

Summing Up
Resumen

As always, check out the documentation of each of the commands we have covered in this chapter. We have seen only their most basic usage. They all have a number of interesting options. As we gain Linux experience, we will see that the redirection feature of the command line is extremely useful for solving specialized problems. There are many commands that make use of standard input and output, and almost all command line programs use standard error to display their informative messages.
Como siempre, consulte la documentación de cada uno de los comandos que hemos cubierto en este capítulo. Solo hemos visto su uso más básico. Todos tienen una serie de opciones interesantes. A medida que ganemos experiencia en Linux, veremos que la función de redirección de la línea de comandos es extremadamente útil para resolver problemas especializados. Hay muchos comandos que utilizan entradas y salidas estándar, y casi todos los programas de línea de comandos utilizan el error estándar para mostrar sus mensajes informativos.

Linux Is About Imagination
Linux se trata de imaginación

When I am asked to explain the difference between Windows and Linux, I often use a toy analogy.
Cuando me piden que explique la diferencia entre Windows y Linux, suelo utilizar una analogía de juguetes.

Windows is like a Game Boy. You go to the store and buy one all shiny new in the box. You take it home, turn it on, and play with it. Pretty graphics, cute sounds. After a while, though, you get tired of the game that came with it, so you go back to the store and buy another one. This cycle repeats over and over. Finally, you go back to the store and say to the person behind the counter, “I want a game that does this!” only to be told that no such game exists because there is no “market demand” for it. Then you say, “But I only need to change this one thing!” The person behind the counter says you can’t change it. The games are all sealed up in their cartridges. You discover that your toy is limited to the games that others have decided you need.
Windows es como un Game Boy. Vas a la tienda y compras uno nuevo y brillante en la caja. Te lo llevas a casa, lo enciendes y juegas con él. Gráficos bonitos, sonidos bonitos. Sin embargo, después de un tiempo, te cansas del juego que venía con él, así que vuelves a la tienda y compras otro. Este ciclo se repite una y otra vez. Finalmente, regresa a la tienda y le dice a la persona detrás del mostrador: "¡Quiero un juego que haga esto!" sólo para que se les diga que tal juego no existe porque no hay "demanda de mercado" para él. Luego dices: "¡Pero solo necesito cambiar esto!" La persona detrás del mostrador dice que no puedes cambiarlo. Los juegos están todos sellados en sus cartuchos. Descubre que su juguete se limita a los juegos que otros han decidido que necesita.

Linux, on the other hand, is like the world’s largest Erector Set. You open it, and it’s just a huge collection of parts. There’s a lot of steel struts, screws, nuts, gears, pulleys, motors, and a few suggestions on what to build. So, you start to play with it. You build one of the suggestions and then another. After a while you discover that you have your own ideas of what to make. You don’t ever have to go back to the store, as you already have everything you need.
Linux, por otro lado, es como el juego de construcción más grande del mundo. Lo abres y es solo una gran colección de piezas. Hay muchos puntales de acero, tornillos, tuercas, engranajes, poleas, motores y algunas sugerencias sobre qué construir. Entonces, empiezas a jugar con eso. Construyes una de las sugerencias y luego otra. Después de un tiempo, descubre que tiene sus propias ideas sobre qué hacer. No es necesario que vuelva nunca a la tienda, ya que ya tiene todo lo que necesita.

The Erector Set takes on the shape of your imagination. It does what you want.
El Erector Set toma la forma de tu imaginación. Hace lo que quiere.

Your choice of toys is, of course, a personal thing, so which toy would you find more satisfying?
La elección de los juguetes es, por supuesto, algo personal, así que, ¿qué juguete encontraría más satisfactorio?