8

Advanced Keyboard Tricks
Trucos avanzados de teclado

I often kiddingly describe Unix as “the operating system for people who like to type.” Of course, the fact that it even has a command line is a testament to that; however, command line users don’t like to type that much. Why else would so many commands have such short names like cp , ls , mv , and rm ? In fact, one of the most cherished goals of the command line is laziness—doing the most work with the fewest number of keystrokes. Another goal is never having to lift your fingers from the keyboard and reach for the mouse. In this chapter, we will look at bash features that make keyboard use faster and more efficient.
A menudo, en broma, describo Unix como "el sistema operativo para las personas a las que les gusta escribir". Por supuesto, el hecho de que incluso tenga una línea de comandos es un testimonio de eso; sin embargo, a los usuarios de la línea de comandos no les gusta escribir tanto. ¿Por qué, si no, tantos comandos tendrían nombres tan cortos como cp, ls, mv y rm? De hecho, uno de los objetivos más preciados de la línea de comandos es la pereza: hacer la mayor parte del trabajo con el menor número de pulsaciones de teclas. Otro objetivo es no tener que levantar los dedos del teclado y alcanzar el mouse. En este capítulo, veremos las características de bash que hacen que el uso del teclado sea más rápido y eficiente.

The following commands will make an appearance:
Aparecerán los siguientes comandos:

- clear -- Clear the terminal screen
  clear - Limpia la pantalla del terminal
- history -- Display or manipulate the history list
  historial: muestra o manipula la lista del historial

Command Line Editing
Edición de línea de comandos

bash uses a library (a shared collection of routines that different programs can use) called Readline to implement command line editing. We have already seen some of this. We know, for example, that the arrow keys move the cursor, but there are many more features. Think of these as additional tools that we can employ in our work. It’s not important to learn all of them, but many of them are very useful. Pick and choose as desired.
bash usa una biblioteca (una colección compartida de rutinas que pueden usar diferentes programas) llamada Readline para implementar la edición de la línea de comandos. Ya hemos visto algo de esto. Sabemos, por ejemplo, que las teclas de flecha mueven el cursor, pero hay muchas más funciones. Piense en

estos como herramientas adicionales que podemos emplear en nuestro trabajo. No es importante aprenderlos todos, pero muchos de ellos son muy útiles. Elija y elija como desee.

Note
Nota

Some of the key sequences covered in this chapter (particularly those that use the alt key) may be intercepted by the GUI for other functions. All of the key sequences should work properly when using a virtual console.
Algunas de las secuencias de teclas cubiertas en este capítulo (particularmente aquellas que usan la tecla alt) pueden ser interceptadas por la GUI para otras funciones. Todas las secuencias de teclas deberían funcionar correctamente cuando se usa una consola virtual.

Cursor Movement
Movimiento del cursor

Table 8-1 describes the keys used to move the cursor.
La Tabla 8-1 describe las teclas que se utilizan para mover el cursor.

> Table 8-1: Cursor Movement Commands
> Tabla 8-1: Comandos de movimiento del cursor

Key	Action
ctrl-A	Move cursor to the beginning of the line. Mover el cursor al principio de la línea
ctrl-E	Move cursor to the end of the line. Mueva el cursor al final de la línea.
ctrl-F	Move cursor forward one character; same as the right arrow key. Mueve el cursor un carácter hacia adelante; igual que la tecla de flecha derecha.
ctrl-B	Move cursor backward one character; same as the left arrow key.
alt-F	Move cursor forward one word. Mueve el cursor una palabra hacia adelante.
alt-B	Move cursor backward one word. Mueve el cursor una palabra hacia atrás.
ctrl-L	Clear the screen and move the cursor to the top-left corner. The clear command does the same thing. Limpia la pantalla y mueve el cursor a la esquina superior izquierda. El comando clear hace lo mismo.

Modifying Text
Modificar texto

Because it's possible we might make a mistake while composing commands, we need a way to correct them efficiently. Table 8-2 describes keyboard commands that are used to edit characters on the command line.
Debido a que es posible que cometamos un error al redactar comandos, necesitamos una forma de corregirlos de manera eficiente. La Tabla 8-2 describe los comandos del teclado que se utilizan para editar caracteres en la línea de comandos.

> Table 8-2: Text Editing Commands
> Tabla 8-2: Comandos de edición de texto

Key	Action
ctrl-D	Delete the character at the cursor location. Elimina el carácter en la ubicación del cursor.
ctrl-T	Transpose (exchange) the character at the cursor location with the one preceding it. Transponer (intercambiar) el carácter en la ubicación del cursor con el que lo precede.
alt-T	Transpose the word at the cursor location with the one preceding it. Transponga la palabra en la ubicación del cursor con la que la precede.
alt-L	Convert the characters from the cursor location to the end of the word to lowercase. Convierta los caracteres desde la ubicación del cursor hasta el final de la palabra a minúsculas.
alt-U	Convert the characters from the cursor location to the end of the word to uppercase. Convierta los caracteres desde la ubicación del cursor hasta el final de la palabra a mayúsculas.

Cutting and Pasting (Killing and Yanking) Text
Cortar y pegar (matar y tirar) texto

The Readline documentation uses the terms killing and yanking to refer to what we would commonly call cutting and pasting (see Table 8-3). Items that are cut are stored in a buffer (a temporary storage area in memory) called the kill-ring.
La documentación de Readline usa los términos matar y tirar para referirse a lo que comúnmente llamaríamos cortar y pegar (ver Tabla 8-3). Los elementos que se cortan se almacenan en un búfer (un área de almacenamiento temporal en la memoria) llamado kill-ring.

> Table 8-3: Cut-and-Paste Commands
> Tabla 8-3: Comandos de cortar y pegar

Key	Action
ctrl-K	Kill text from the cursor location to the end of line. Elimina el texto desde la ubicación del cursor hasta el final de la línea.

Key	Action
ctrl-U	Kill text from the cursor location to the beginning of the line. Elimina el texto desde la ubicación del cursor hasta el principio de la línea.
alt-D	Kill text from the cursor location to the end of the current word. Elimina el texto desde la ubicación del cursor hasta el final de la palabra actual.
alt-backspace	Kill text from the cursor location to the beginning of the current word. if the cursor is at the beginning of a word, kill the previous word. Elimina el texto desde la ubicación del cursor hasta el comienzo de la palabra actual. si el cursor está al principio de una palabra, elimine la palabra anterior.
ctrl-Y	Yank text from the kill-ring and insert it at the cursor location. Extraiga el texto del kill-ring e insértelo en la ubicación del cursor.

The Meta Key
La clave meta

If you venture into the Readline documentation, which can be found in the “READLINE” section of the bash man page, you will encounter the term meta key. On modern keyboards, this maps to the alt key; however, this wasn’t always the case.
Si se aventura en la documentación de Readline, que se puede encontrar en la sección "READLINE" de la página de manual de bash, encontrará el término meta clave. En los teclados modernos, esto se asigna a la tecla alt; sin embargo, este no fue siempre el caso.

Back in the dim times (before PCs but after Unix), not everybody had their own computer. What they might have had was a device called a terminal. A terminal was a communication device that featured a text display screen and a keyboard and just enough electronics inside to display text characters and move the cursor around. It was attached (usually by serial cable) to a larger computer or the communication network of a larger computer.
En los tiempos oscuros (antes de las PC pero después de Unix), no todo el mundo tenía su propia computadora. Lo que podrían haber tenido era un dispositivo llamado terminal. Una terminal era un dispositivo de comunicación que presentaba una pantalla de visualización de texto y un teclado, y en su interior solo había elementos electrónicos suficientes para mostrar caracteres de texto y mover el cursor. Se conectó (generalmente mediante un cable serie) a una computadora más grande o la red de comunicación de una computadora más grande.

There were many different brands of terminals, and they all had different keyboards and display feature sets. Because they all tended to at least understand ASCII, software developers wanting portable applications wrote to the lowest common denominator. Unix systems have an elaborate way of dealing with terminals and their different display features. Because the developers of Readline could not be sure of the presence of a dedicated extra control key, they invented one and called it meta. While the alt key serves as the meta key on modern keyboards, you can also press and release the esc key to get the same effect as holding down the alt key if you’re still using a terminal (which you can still do in Linux!).
Había muchas marcas diferentes de terminales, y todas tenían distintos teclados y conjuntos de

> funciones de pantalla. Debido a que todos tendían a comprender al menos ASCII, los desarrolladores de software que querían aplicaciones portátiles escribieron con el mínimo común denominador. Los sistemas Unix tienen una forma elaborada de tratar con terminales y sus diferentes características de visualización. Debido a que los desarrolladores de Readline no podían estar seguros de la presencia de una clave de control adicional dedicada, inventaron una y la llamaron meta. Si bien la tecla alt sirve como tecla meta en los teclados modernos, también puede presionar y soltar la tecla esc para obtener el mismo efecto que mantener presionada la tecla alt si todavía está usando una terminal (¡lo que aún puede hacer en Linux!).

Completion
Terminación

Another way that the shell can help you is through a mechanism called completion. Completion occurs when you press the tab key while typing a command. Let's see how this works. Given a home directory that looks like this:
Otra forma en que el caparazón puede ayudarlo es a través de un mecanismo llamado finalización. La finalización ocurre cuando presiona la tecla de tabulación mientras escribe un comando. Veamos cómo funciona esto. Dado un directorio de inicio que se ve así:

```
[me@linuxbox ~]$ ls
Desktop ls-output.txt  Pictures Templates Videos
Documents Music Public
```

try typing the following, but don't press enter:
intente escribir lo siguiente, pero no presione Intro:

```
[me@linuxbox ~]$ ls l
```

Now press tab.
Ahora presione el tabulador.

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

See how the shell completed the line for you? Let's try another one.
¿Ves cómo el caparazón completó la línea por ti? Probemos con otro.

Again, don't press enter.
Nuevamente, no presione Intro.

```
[me@linuxbox ~]$ ls D
```

Press tab.

```
[me@linuxbox ~]$ ls D
```

No completion, just nothing. This happened because D matches more than one entry in the directory. For completion to be successful, the "clue" you give it has to be unambiguous. If we go further, as with the

following:
Sin finalización, simplemente nada. Esto sucedió porque D coincide con más de una entrada en el directorio. Para que la finalización sea exitosa, la “pista” que le des debe ser inequívoca. Si vamos más allá, como ocurre con lo siguiente:

```
[me@linuxbox ~]$ ls Do
```

and then press tab:
y luego presione tab:

```
[me@linuxbox ~]$ ls Documents
```

the completion is successful.
la finalización es exitosa.

While this example shows completion of pathnames, which is its most common use, completion will also work on variables (if the beginning of the word is a $), usernames (if the word begins with ~), commands (if the word is the first word on the line), and hostnames (if the beginning of the word is @).
Si bien este ejemplo muestra la finalización de los nombres de ruta, que es su uso más común, la finalización también funcionará en variables (si el comienzo de la palabra es $), nombres de usuario (si la palabra comienza con ~), comandos (si la palabra es el primera palabra en la línea) y nombres de host (si el comienzo de la palabra es @).

Hostname completion works only for hostnames listed in /etc/hosts.
La finalización del nombre de host solo funciona para los nombres de host enumerados en / etc / hosts.

A number of control and meta key sequences are associated with completion, as listed in Table 8-4.
Una serie de secuencias de teclas de control y meta están asociadas con la finalización, como se enumera en la Tabla 8-4.

> Table 8-4: Completion Commands
> Tabla 8-4: Comandos de finalización

Key	Action
alt-?	Display a list of possible completions. On most systems, you can also do this by pressing the tab key a second time, which is much easier. Muestra una lista de posibles finalizaciones. En la mayoría de los sistemas, también puede hacer esto presionando la tecla de tabulación por segunda vez, lo cual es mucho más fácil.
alt-*	Insert all possible completions. This is useful when you want to use more than one possible match. Inserte todas las terminaciones posibles. Esto es útil cuando desea utilizar más de una coincidencia posible.

There are quite a few more that are rather obscure. A list appears in the bash man page under **"READLINE."**
Hay bastantes más que son bastante oscuros. Aparece una lista en la página de manual de bash en **"READLINE".**

Programmable Completion
Finalización programable

> Recent versions of bash have a facility called programmable completion.
> Las versiones recientes de bash tienen una función llamada finalización programable.

> Programmable completion allows you (or more likely, your distribution provider) to add more completion rules. Usually this is done to add support for specific applications. For example, it is possible to add completions for the option list of a command or match particular file types that an application supports. Ubuntu has a fairly large set defined by default. Programmable completion is implemented by shell functions, a kind of mini shell script that we will cover in later chapters. If you are curious, try the following:
> La finalización programable le permite a usted (o más probablemente, a su proveedor de distribución) agregar más reglas de finalización. Por lo general, esto se hace para agregar soporte para aplicaciones específicas. Por ejemplo, es posible agregar terminaciones para la lista de opciones de un comando o hacer coincidir tipos de archivos particulares que admite una aplicación. Ubuntu tiene un conjunto bastante grande definido por defecto. La finalización programable se implementa mediante funciones de shell, una especie de mini script de shell que cubriremos en capítulos posteriores. Si tiene curiosidad, intente lo siguiente:

```
set | less
```

> Now see whether you can find them. Not all distributions include them by default.
> Ahora vea si puede encontrarlos. No todas las distribuciones los incluyen de forma predeterminada.

Using History
Usando el historial

As we discovered in Chapter 1, bash maintains a history of commands that have been entered. This list of commands is kept in your home directory in a file called bash_history. The history facility is a useful resource for reducing the amount of typing you have to do, especially when combined with command line editing.
Como descubrimos en el Capítulo 1, bash mantiene un historial de comandos que se han ingresado. Esta lista de comandos se guarda en su directorio de inicio en un archivo llamado bash_history. La función de historial es un recurso útil para reducir la cantidad de escritura que tiene que hacer, especialmente cuando se combina con la edición de la línea de comandos.

Searching History
Historial de búsqueda

At any time, we can view the contents of the command history list by doing the following:
En cualquier momento, podemos ver el contenido de la lista del historial de comandos haciendo lo siguiente:

```
[me@linuxbox ~]$ history | less
```

By default, bash stores the last 500 commands we have entered, though most modern distributions set this value to 1,000. We will see how to adjust this value in Chapter 11. Let's say we want to find the commands we used to list /usr/bin. This is one way we could do this:
Por defecto, bash almacena los últimos 500 comandos que hemos ingresado, aunque la mayoría de las distribuciones modernas establecen este valor en 1,000. Veremos cómo ajustar este valor en el Capítulo 11. Digamos que queremos encontrar los comandos que usamos para listar / usr / bin. Esta es una forma en que podríamos hacer esto:

```
[me@linuxbox ~]$ history | grep /usr/bin
```

And let's say that among our results we got a line containing an interesting command like this:
Y digamos que entre nuestros resultados obtuvimos una línea que contiene un comando interesante como este:

```
88 ls -l /usr/bin > ls-output.txt
```

The **88** is the **line number** of the command in the **history list**. We could use this immediately using another type of expansion called history expansion.
El **88** es el **número de línea** del comando en la **lista de historial**. Podríamos usar esto inmediatamente usando otro tipo de expansión llamada expansión histórica.

To use our discovered line, we could do this:
Para usar nuestra línea descubierta, podríamos hacer esto:

```
[me@linuxbox ~]$ !88
```

bash will expand !88 into the contents of the 88th line in the history list. There are other forms of history expansion that we will cover in the next section.
bash expandirá !88 al contenido de la línea 88 en la lista del historial. Hay otras formas de expansión de la historia que cubriremos en la siguiente sección.

bash also provides the ability to search the history list incrementally. This means we can tell bash to search the history list as we enter characters, with each additional character further refining our search. To start incremental search, press ctrl-R followed by the text you are looking for.
bash también ofrece la posibilidad de buscar en la lista del historial de forma incremental. Esto significa que podemos decirle a bash que busque en la lista del historial a medida que ingresamos caracteres, con cada carácter adicional refinando aún más nuestra búsqueda. Para iniciar la búsqueda incremental, presione ctrl-R seguido del texto que está buscando.

When you find it, you can either press enter to execute the command or press ctrl-J to copy the line from the history list to the current command line.
Cuando lo encuentre, puede presionar enter para ejecutar el comando o presionar ctrl-J para copiar la línea de la lista del historial a la línea de comando actual.

To find the next occurrence of the text (moving "up" the history list), press ctrl-R again. To quit searching, press either ctrl-G or ctrl-C. Here we see it in action:
Para encontrar la siguiente aparición del texto (moviendo "hacia arriba" la lista del historial), presione ctrl-R nuevamente. Para dejar de buscar, presione ctrl-G o ctrl-C. Aquí lo vemos en acción:

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

First press ctrl-R.
Primero presione ctrl-R.

```
(reverse-i-search)`':
```

The prompt changes to indicate that we are performing a reverse incremental search. It is “reverse” because we are searching from “now” to sometime in the past. Next, we start typing our search text. In this example, we're searching for /usr/bin:
El mensaje cambia para indicar que estamos realizando una búsqueda incremental inversa. Es "inverso" porque estamos buscando desde "ahora" hasta algún momento en el pasado. A continuación, comenzamos a escribir nuestro texto de búsqueda. En este ejemplo, buscamos /usr/bin:

```
(reverse-i-search)`/usr/bin': ls -l /usr/bin > ls-output.txt
```

Immediately, the search returns our result. With our result, we can execute the command by pressing enter , or we can copy the command to our current command line for further editing by pressing ctrl-J. Let’s copy it. Press ctrl-J.
Inmediatamente, la búsqueda devuelve nuestro resultado. Con nuestro resultado, podemos ejecutar el comando presionando enter, o podemos copiar el comando a nuestra línea de comando actual para editarlo más presionando ctrl-J. Copiemos. Presione ctrl-J.

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

Our shell prompt returns, and our command line is loaded and ready for action!
¡Nuestro indicador de shell regresa y nuestra línea de comandos está cargada y lista para la acción!

Table 8-5 lists some of the keystrokes used to manipulate the history list.
La Tabla 8-5 enumera algunas de las pulsaciones de teclas utilizadas para manipular la lista del historial.

> Table 8-5: History Commands
> Tabla 8-5: Comandos de historial

Key	Action
ctrl-P	Move to the previous history entry. This is the same action as the up arrow. Ir a la entrada anterior del historial. Esta es la misma acción que la flecha hacia arriba.
ctrl-N	Move to the next history entry. This is the same action as the down arrow. Pasar a la siguiente entrada del historial. Esta es la misma acción que la flecha hacia abajo.
alt-<	Move to the beginning (top) of the history list. Ir al principio (parte superior) de la lista del historial.
alt->	Move to the end (bottom) of the history list, i.e., the current command line. Ir al final (parte inferior) de la lista del historial, es decir, la línea de comando actual.

Key	Action
ctrl-R	Reverse incremental search. This searches incrementally from the current command line up the history list. Búsqueda incremental inversa. Esto busca de forma incremental desde la línea de comandos actual hasta la lista del historial.
alt-P	Reverse search, non incremental. With this key, type in the search string and press enter before the search is performed. Búsqueda inversa, no incremental. Con esta tecla, escriba la cadena de búsqueda y presione Intro antes de realizar la búsqueda.
alt-N	Forward search, non incremental. Búsqueda hacia adelante, no incremental.
ctrl-O	Execute the current item in the history list and advance to the next one. This is handy if you are trying to re-execute a sequence of commands in the history list. Ejecute el elemento actual en la lista del historial y avance al siguiente. Esto es útil si está intentando volver a ejecutar una secuencia de comandos en la lista del historial.

History Expansion
Expansión de la historia

The shell offers a specialized type of expansion for items in the history list by using the ! character. We have already seen how the exclamation point can be followed by a number to insert an entry from the history list. There are a number of other expansion features, as described in Table 8-6.
El shell ofrece un tipo de expansión especializada para los elementos de la lista del historial mediante el uso de! personaje. Ya hemos visto cómo el signo de exclamación puede ir seguido de un número para insertar una entrada de la lista del historial. Hay otras características de expansión, como se describe en la Tabla 8-6.

> Table 8-6: History Expansion Commands
> Tabla 8-6: Comandos de expansión del historial

Sequence	Action
!!	Repeat the last command. It is probably easier to press the up arrow and enter. Repite el último comando. Probablemente sea más fácil presionar la flecha hacia arriba e ingresar.
!number	Repeat history list item number. Repita el número de elemento de la lista de historial.
!string	Repeat last history list item starting with string. Repite el último elemento de la lista del historial que comienza con una cadena.
!?string	Repeat last history list item containing string. Repite el último elemento de la lista del historial que contiene la cadena.

I caution against using the !string and !?string forms unless you are absolutely sure of the contents of the history list items.
Advierto contra el uso de las formas! String y!? String a menos que esté absolutamente seguro del contenido de los elementos de la lista del historial.

Many more elements are available in the history expansion mechanism, but this subject is already too arcane, and our heads may explode if we continue. The “HISTORY EXPANSION” section of the bash man page goes into all the gory details. Feel free to explore!
Hay muchos más elementos disponibles en el mecanismo de expansión de la historia, pero este tema ya es demasiado arcano y nuestras cabezas pueden explotar si continuamos. La sección "HISTORIA EXPANSIÓN" de la página de manual de bash entra en todos los detalles sangrientos. ¡Siéntete libre de explorar!

---

script
guión

In addition to the command history feature in bash , most Linux distributions include a program called script that can be used to record an entire shell session and store it in a file. The basic syntax of the command is as follows:
Además de la función de historial de comandos en bash, la mayoría de las distribuciones de Linux incluyen un programa llamado script que se puede usar para grabar una sesión de shell completa y almacenarla en un archivo. La sintaxis básica del comando es la siguiente:

```
script file
```

where file is the name of the file used for storing the recording. If no file is specified, the file typescript is used. See the script man page for a complete list of the program’s options and features.
donde archivo es el nombre del archivo utilizado para almacenar la grabación. Si no se especifica ningún archivo, se utiliza el archivo mecanografiado. Consulte la página del manual del script para obtener una lista completa de las opciones y características del programa.

Summing Up
Resumen

In this chapter, we covered some of the keyboard tricks that the shell provides to help hardcore typists reduce their workloads. As time goes by and we become more involved with the command line, we can refer back to this chapter to pick up more of these tricks. For now, consider them optional and potentially helpful.
En este capítulo, cubrimos algunos de los trucos de teclado que proporciona el shell para ayudar a los mecanógrafos expertos a reducir sus cargas de trabajo. A medida que pasa el tiempo y nos involucramos más con la línea de comandos, podemos consultar este capítulo para aprender más de estos trucos. Por ahora, considérelos opcionales y potencialmente útiles.