Part IV

Writing Shell Scripts
Escribiendo Scripts de ventana

24

writing your first script.
Escribiendo tu primer script

In the preceding chapters, we assembled an arsenal of command line tools. While these tools can solve many kinds of computing problems, we are still limited to manually using them one by one on the command line. Wouldn't it be great if we could get the shell to do more of the work? We can! By joining our tools together into programs of our own design, the shell can carry out complex sequences of tasks all by itself. We can enable it to do this by writing shell scripts.
En los capítulos anteriores, reunimos un arsenal de herramientas de línea de comandos. Si bien estas herramientas pueden resolver muchos tipos de problemas informáticos, todavía estamos limitados a usarlos manualmente uno por uno en la línea de comandos. ¿No sería fantástico si pudiéramos conseguir que el caparazón hiciera más trabajo? ¡Podemos! Al unir nuestras herramientas en programas de nuestro propio diseño, el shell puede realizar secuencias complejas de tareas por sí mismo. Podemos habilitarlo para hacer esto escribiendo scripts de shell.

What Are Shell Scripts?
¿Qué son los scripts de Shell?

In the simplest terms, a shell script is a file containing a series of commands.
En los términos más simples, un script de shell es un archivo que contiene una serie de comandos.

The shell reads this file and carries out the commands as though they have been entered directly on the command line.
El shell lee este archivo y ejecuta los comandos como si se hubieran introducido directamente en la línea de comandos.

The shell is somewhat unique, in that it is both a powerful command line interface to the system and a scripting language interpreter. As we will see, most of the things that can be done on the command line can be done in scripts, and most of the things that can be done in scripts can be done on the command line.
El shell es algo único, ya que es tanto una poderosa interfaz de línea de comandos para el sistema como un intérprete de lenguaje de secuencias de comandos. Como veremos, la mayoría de las cosas que se pueden hacer en la línea de comandos se pueden hacer en scripts, y la mayoría de las cosas que se pueden hacer en los scripts se pueden hacer en la línea de comandos.

We have covered many shell features, but we have focused on those features most often used directly on the command line. The shell also provides a set of features usually (but not always) used when writing programs.
Hemos cubierto muchas funciones del shell, pero nos hemos centrado en aquellas funciones que se utilizan con más frecuencia directamente en la línea de comandos. El shell también proporciona un conjunto de características que normalmente (pero no siempre) se utilizan al escribir programas.

How to Write a Shell Script
Cómo escribir un script de shell

To successfully create and run a shell script, we need to do three things.
Para crear y ejecutar correctamente un script de shell, necesitamos hacer tres cosas.

> Write a script. Shell scripts are ordinary text files. So, we need a text editor to write them. The best text editors will provide syntax highlighting, allowing us to see a color-coded view of the elements of the script. Syntax highlighting will help us spot certain kinds of common errors. vim, gedit, kate, and many other editors are good candidates for writing scripts.
> Escribe un guión. Los scripts de Shell son archivos de texto normales. Entonces, necesitamos un editor de texto para escribirlos. Los mejores editores de texto proporcionarán resaltado de sintaxis, lo que nos permitirá ver una vista codificada por colores de los elementos del script. El resaltado de sintaxis nos ayudará a detectar ciertos tipos de errores comunes. vim, gedit, kate y muchos otros editores son buenos candidatos para escribir guiones.

> Make the script executable. The system is rather fussy about not letting any old text file be treated as a program, and for good reason! We need to set the script file's permissions to allow execution.
> Haga que el script sea ejecutable. El sistema es bastante quisquilloso para no permitir que ningún archivo de texto antiguo sea tratado como un programa, ¡y por una buena razón! Necesitamos configurar los permisos del archivo de secuencia de comandos para permitir la ejecución.

> Put the script somewhere the shell can find it. The shell automatically searches certain directories for executable files when no explicit pathname is specified. For maximum convenience, we will place our scripts in these directories.
> Coloque el script en algún lugar donde el shell pueda encontrarlo. El shell busca automáticamente archivos ejecutables en ciertos directorios cuando no se especifica un nombre de ruta explícito. Para mayor comodidad, colocaremos nuestros scripts en estos directorios.

Script File Format
Formato de archivo de secuencia de comandos

In keeping with programming tradition, we'll create a "Hello World" program to demonstrate an extremely simple script. Let's fire up our text editors and enter the following script:
De acuerdo con la tradición de la programación, crearemos un programa "Hello World" para demostrar un guión extremadamente simple. Encienda nuestros editores de texto e ingresemos el siguiente script:

```
#!/bin/bash
# This is our first script.
echo 'Hello World!'
```

The last line of our script is pretty familiar; it's just an echo command with a string argument. The second line is also familiar. It looks like a comment that we have seen used in many of the configuration files we have examined and edited. One thing about comments in shell scripts is that they may also appear at the ends of lines provided they are preceded by at least one whitespace character, like so:
La última línea de nuestro guión es bastante familiar; es solo un comando de eco con un argumento de cadena. La segunda línea también es familiar. Parece un comentario que hemos visto utilizado en muchos de los archivos de configuración que hemos examinado y editado. Una cosa sobre los comentarios en los scripts de shell es que también pueden aparecer al final de las líneas siempre que estén precedidos por al menos un carácter de espacio en blanco, así:

```
echo 'Hello world!' # this is a comment too
```

Everything from the # symbol onward on the line is ignored. Like many things, this works on the command line, too.
Todo desde el símbolo # en adelante en la línea se ignora. Como muchas cosas, esto también funciona en la línea de comandos.

```
[~]$ echo 'Hello World!' # This is a comment too
Hello World!
```

Though comments are of little use on the command line, they will work.
Aunque los comentarios son de poca utilidad en la línea de comandos, funcionarán.

The first line of our script is a little mysterious. It looks as if it should be a comment since it starts with # , but it looks too purposeful to be just that.
La primera línea de nuestro guión es un poco misteriosa. Parece que debería ser un comentario, ya que comienza con #, pero parece demasiado útil para ser solo eso.

The #! character sequence is, in fact, a special construct called a shebang.
Los #! La secuencia de caracteres es, de hecho, una construcción especial llamada shebang.

The shebang is used to tell the kernel the name of the interpreter that should be used to execute the script that follows. Every shell script should include this as its first line.
El shebang se usa para decirle al kernel el nombre del intérprete que debe usarse para ejecutar el script que sigue. Cada script de shell debe incluir esto como su primera línea.

Let's save our script file as hello_world.
Guardemos nuestro archivo de secuencia de comandos como hello_world.

Executable Permissions
Permisos de ejecutables

The next thing we have to do is make our script executable. This is easily done using chmod.
Lo siguiente que tenemos que hacer es hacer que nuestro script sea ejecutable. Esto se hace fácilmente usando chmod.

```
[me@linuxbox~]$ ls -l hello_world
-rw-r--r-- 1 me me 63 2018-03-07 10:10 hello_world
[~]$ chmod 755 hello_world
[~]$ ls -l hello_world
-rwxr-xr-x 1 me me 63 2018-03-07 10:10 hello_world
```

There are two common permission settings for scripts: 755 for scripts that everyone can execute and 700 for scripts that only the owner can execute. Note that scripts must be readable to be executed.
Hay dos configuraciones de permisos comunes para los scripts: 755 para scripts que todos pueden ejecutar y 700 para scripts que solo el propietario puede ejecutar. Tenga en cuenta que las secuencias de comandos deben ser legibles para su ejecución.

Script File Location
Ubicación del archivo de script

With the permissions set, we can now execute your script.
Con los permisos establecidos, ahora podemos ejecutar su script.

```
[~]$ ./hello_world
Hello World!
```

For the script to run, we must precede the script name with an explicit path. If we don't, we get this:
Para que se ejecute el script, debemos preceder el nombre del script con una ruta explícita. Si no lo hacemos, obtenemos esto:

```
[me@linuxbox ~]$ hello_world
bash: hello_world: command not found
```

Why is this? What makes our script different from other programs? As it turns out, nothing. Our script is fine. Its location is the problem.
¿Por qué es esto? ¿Qué hace que nuestro script sea diferente de otros programas? Resulta que nada. Nuestro guión está bien. Su ubicación es el problema.

In Chapter 11, we discussed the PATH environment variable and its effect on how the system searches for executable programs. To recap, the system searches a list of directories each time it needs to find an executable program, if no explicit path is specified. This is how the system knows to execute /bin/ls when we type ls at the command line. The /bin directory is one of the directories that the system automatically searches. The list of directories is held within an environment variable named PATH. The PATH variable contains a colon-separated list of directories to be searched. We can view the contents of PATH.
En el Capítulo 11, discutimos la variable de entorno PATH y su efecto sobre cómo el sistema busca programas ejecutables. En resumen, el sistema busca en una lista de directorios cada vez que necesita encontrar un programa ejecutable, si no se especifica una ruta explícita. Así es como el sistema sabe ejecutar /bin/ls cuando escribimos ls en la línea de comandos. El directorio /bin es uno de los directorios que el sistema busca automáticamente. La lista de directorios se encuentra dentro de una variable de entorno denominada PATH. La variable PATH contiene una lista de directorios separados por dos puntos para buscar. Podemos ver el contenido de PATH.

```
[~]$ echo $PATH
/home/me/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/
usr/games
```

Here we see our list of directories. If our script were located in any of the directories in the list, our problem would be solved. Notice the first directory in the list, /home/me/bin. Most Linux distributions configure the PATH variable to contain a bin directory in the user's home directory to allow users to execute their own programs. So, if we create the bin directory and place our script within it, it should start to work like other programs.
Aquí vemos nuestra lista de directorios. Si nuestro script estuviera ubicado en cualquiera de los directorios de la lista, nuestro problema estaría resuelto. Observe el primer directorio de la lista, / home / me / bin. La mayoría de las distribuciones de Linux configuran la variable PATH para que contenga un directorio bin en el directorio de inicio del usuario para permitir que los usuarios ejecuten sus propios programas. Entonces, si creamos el directorio bin y colocamos nuestro script dentro de él, debería comenzar a funcionar como otros programas.

```
[me@linuxbox ~]$ mkdir bin
[me@linuxbox ~]$ mv hello_world bin
```

```
[me@linuxbox ~]$ hello_world
Hello World!
```

And so it does.
Y así es.

If the PATH variable does not contain the directory, we can easily add it by including this line in our .bashrc file:
Si la variable PATH no contiene el directorio, podemos agregarlo fácilmente incluyendo esta línea en nuestro archivo .bashrc:

```
export PATH=~/bin:"$PATH"
```

After this change is made, it will take effect in each new terminal session.
Una vez realizado este cambio, entrará en vigor en cada nueva sesión de terminal.

To apply the change to the current terminal session, we must have the shell reload the .bashrc file. This can be done by “sourcing” it.
Para aplicar el cambio a la sesión de terminal actual, debemos hacer que el shell vuelva a cargar el archivo .bashrc. Esto se puede hacer "abasteciéndolo".

```
[~]$ . .bashrc
```

The dot (.) command is a synonym for the source command, a shell builtin that reads a specified file of shell commands and treats it like input from the keyboard.
El comando dot (.) Es un sinónimo del comando fuente, un intérprete de comandos de shell que lee un archivo específico de comandos de shell y lo trata como una entrada del teclado.

> Note
> Nota

> Ubuntu (and most other Debian-based distributions) automatically adds the ~/bin directory to the PATH variable if the ~/bin directory exists when the user’s .bashrc file is executed. So, on Ubuntu systems, if we create the ~/bin directory and then log out and log in again, everything works.
> Ubuntu (y la mayoría de las otras distribuciones basadas en Debian) agrega automáticamente el directorio ~/bin a la variable PATH si el directorio ~/bin existe cuando se ejecuta el archivo .bashrc del usuario. Entonces, en los sistemas Ubuntu, si creamos el directorio ~/bin y luego cerramos y volvemos a iniciar sesión, todo funciona.

Good Locations for Scripts
Buenas ubicaciones para scripts

The ~/bin directory is a good place to put scripts intended for personal use. If we write a script that everyone on a system is allowed to use, the traditional location is /usr/local/bin. Scripts intended for use by the system administrator are often located in /usr/local/sbin. In most cases, locally supplied software, whether scripts or compiled programs, should be placed in the /usr/local hierarchy and not in /bin or /usr/bin. These directories are specified by the Linux Filesystem Hierarchy Standard to contain only files supplied and maintained by the Linux distributor.
El directorio ~/bin es un buen lugar para colocar scripts destinados a uso personal. Si escribimos un script que todos en un sistema pueden usar, la ubicación tradicional es /usr/local/bin. Los scripts destinados a ser utilizados por el administrador del sistema a menudo se encuentran en /usr/local/sbin. En la mayoría de los casos, el software suministrado localmente, ya sean scripts o programas compilados, debe colocarse en la jerarquía /usr/local y no en /bin o /usr/bin. Estos directorios están especificados por el estándar de jerarquía del sistema de archivos de Linux para contener solo archivos proporcionados y mantenidos por el distribuidor de Linux.

More Formatting Tricks
Más trucos de formato

One of the key goals of serious script writing is ease of maintenance; that is, the ease with which a script may be modified by its author or others to adapt it to changing needs. Making a script easy to read and understand is one way to facilitate easy maintenance.
Uno de los objetivos clave de la escritura de guiones serios es la facilidad de mantenimiento; es decir, la facilidad con la que su autor u otros pueden modificar un guión para adaptarlo a las necesidades cambiantes. Hacer que un guión sea fácil de leer y comprender es una forma de facilitar el mantenimiento.

Long Option Names
Nombres de opciones largos

Many of the commands we have studied feature both short and long option names. For instance, the ls command has many options that can be expressed in either short or long form. For example, the following:
Muchos de los comandos que hemos estudiado tienen nombres de opciones tanto cortos como largos. Por ejemplo, el comando ls tiene muchas opciones que se pueden expresar en forma corta o larga. Por ejemplo, lo siguiente:

```
[~]$ ls -ad
```

---

is equivalent to this:
es equivalente a esto:

```
[~]$ ls --all --directory
```

In the interests of reduced typing, short options are preferred when entering options on the command line, but when writing scripts, long options can provide improved readability.
En aras de la escritura reducida, se prefieren las opciones cortas al ingresar opciones en la línea de comandos, pero al escribir scripts, las opciones largas pueden proporcionar una mejor legibilidad.

Indentation and Line Continuation
Sangría y continuación de línea

When employing long commands, readability can be enhanced by spreading the command over several lines. In Chapter 17, we looked at a particularly long example of the find command.
Al emplear comandos largos, la legibilidad se puede mejorar extendiendo el comando en varias líneas. En el Capítulo 17, vimos un ejemplo particularmente largo del comando de búsqueda.

```
[~]$ find playground \( -type f -not -perm 0600 -exec chmod 0600 '{}' ';'
\) -or \( -type d -not -perm 0700 -exec chmod 0700 '{}' ';' \)
```

Obviously, this command is a little hard to figure out at first glance. In a script, this command might be easier to understand if written this way:
Obviamente, este comando es un poco difícil de entender a primera vista. En un script, este comando puede ser más fácil de entender si se escribe de esta manera:

```
find playground \
    \( \
        -type f \
        -not -perm 0600 \
        -exec chmod 0600 '{}' ';' \
    \) \
    -or \
    \( \
        -type d \
        -not -perm 0700 \
        -exec chmod 0700 '{}' ';' \
    \)
```

By using line continuations (backslash-linefeed sequences) and indentation, the logic of this complex command is more clearly described to the reader. This technique works on the command line, though it is seldom used, as it is awkward to type and edit. One difference between a script and a command line is that the script may employ tab characters to achieve indentation, whereas the command line cannot since tabs are used to activate completion.
Mediante el uso de continuaciones de línea (secuencias de avance de línea con barra invertida) y sangría, la lógica de este comando complejo se describe más claramente al lector. Esta técnica funciona en la línea de comandos, aunque rara vez se usa, ya que es difícil de escribir y editar. Una diferencia entre una secuencia de comandos y una línea de comando es que la secuencia de comandos puede emplear caracteres de

tabulación para lograr la sangría, mientras que la línea de comando no puede, ya que las tabulaciones se utilizan para activar la finalización.

Configuring vim for Script Writing
Configuración de vim para escritura de scripts

The vim text editor has many, many configuration settings. There are several common options that can facilitate script writing.
El editor de texto vim tiene muchas, muchas opciones de configuración. Hay varias opciones comunes que pueden facilitar la escritura de guiones.

The following turns on syntax highlighting:
Lo siguiente activa el resaltado de sintaxis:

```
:syntax on
```

With this setting, different elements of shell syntax will be displayed in different colors when viewing a script. This is helpful for identifying certain kinds of programming errors. It looks cool, too. Note that for this feature to work, you must have a complete version of vim installed, and the file you are editing must have a shebang indicating the file is a shell script. If you have difficulty with the previous command, try :set syntax=sh instead.
Con esta configuración, los diferentes elementos de la sintaxis de shell se mostrarán en diferentes colores al visualizar un script. Esto es útil para identificar ciertos tipos de errores de programación. También se ve genial. Tenga en cuenta que para que esta característica funcione, debe tener instalada una versión completa de vim y el archivo que está editando debe tener un shebang que indique que el archivo es un script de shell. Si tiene dificultades con el comando anterior, intente: establecer syntax = sh en su lugar.

This turns on the option to highlight search results:
Esto activa la opción para resaltar los resultados de la búsqueda:

```
:set hlsearch
```

Say we search for the word echo. With this option on, each instance of the word will be highlighted.
Digamos que buscamos la palabra eco. Con esta opción activada, cada instancia de la palabra se resaltará.

The following sets the number of columns occupied by a tab character:
Lo siguiente establece el número de columnas ocupadas por un carácter de tabulación:

```
:set tabstop=4
```

The default is eight columns. Setting the value to 4 (which is a common practice) allows long lines to fit more easily on the screen.
El valor predeterminado es ocho columnas. Establecer el valor en 4 (que es una práctica común) permite que las líneas largas quepan más fácilmente en la pantalla.

The following turns on the “auto indent” feature:
Lo siguiente activa la función de "sangría automática":

```
:set autoindent
```

This causes vim to indent a new line the same amount as the line just typed.
Esto hace que vim aplique una sangría a una nueva línea en la misma cantidad que la línea que acaba de escribir.

This speeds up typing on many kinds of programming constructs. To stop indentation, press ctrl-D.
Esto acelera la escritura en muchos tipos de construcciones de programación. Para detener la sangría, presione ctrl-D.

These changes can be made permanent by adding these commands (without the leading colon characters) to your ~/.vimrc file.
Estos cambios pueden hacerse permanentes agregando estos comandos (sin los dos puntos iniciales) a su archivo ~ /.vimrc.

Summing Up
Resumen

> In this first chapter of scripting, we looked at how scripts are written and made to easily execute on our system. We also saw how we can use various formatting techniques to improve the readability (and thus the maintainability) of our scripts. In future chapters, ease of maintenance will come up again and again as a central principle in good script writing.
> En este primer capítulo de secuencias de comandos, analizamos cómo se escriben y hacen las secuencias de comandos para que se ejecuten fácilmente en nuestro sistema. También vimos cómo podemos usar varias técnicas de formato para mejorar la legibilidad (y por lo tanto la capacidad de mantenimiento) de nuestros scripts. En capítulos futuros, la facilidad de mantenimiento aparecerá una y otra vez como un principio central en una buena escritura de guiones.