Part II
Parte II

Configuration and the environment
Configuracion y el Entorno

11

The Environment
El Entorno

As we discussed earlier, the shell maintains a body of information during our shell session called the environment. Programs use the data stored in the environment to determine facts about the system's configuration.
Como discutimos anteriormente, el shell mantiene un cuerpo de información durante nuestra sesión de shell llamado entorno. Los programas utilizan los datos almacenados en el entorno para determinar hechos sobre la configuración del sistema.

While most programs use configuration files to store program settings, some programs also look for values stored in the environment to adjust their behavior. Knowing this, we can use the environment to customize our shell experience.
Si bien la mayoría de los programas utilizan archivos de configuración para almacenar la configuración del programa, algunos programas también buscan valores almacenados en el entorno para ajustar su comportamiento. Sabiendo esto, podemos usar el entorno para personalizar nuestra experiencia de shell.

In this chapter, we will work with the following commands:
En este capítulo trabajaremos con los siguientes comandos:

- printenv -- Print part or all of the environment
  printenv: imprime parte o todo el entorno
- set -- Set shell options
  set - Establecer opciones de shell
- export -- Export environment to subsequently executed programs
  exportar: entorno de exportación a programas ejecutados posteriormente

- alias -- Create an alias for a command
  alias: crea un alias para un comando

What Is Stored in the Environment?
¿Qué se almacena en el medio ambiente?

The shell stores two basic types of data in the environment; though, with bash, the types are largely indistinguishable. They are environment variables and shell variables. Shell variables are bits of data placed there by bash , and environment variables are everything else. In addition to variables, the shell stores some programmatic data, namely, aliases and shell functions. We covered aliases in Chapter 5 and we will cover shell functions (which are related to shell scripting) in Part IV of this book.
El shell almacena dos tipos básicos de datos en el entorno; aunque, con bash, los tipos son en gran parte indistinguibles. Son variables de entorno y variables de shell. Las variables de shell son bits de datos colocados allí por bash, y las variables de entorno son todo lo demás. Además de las variables, el shell almacena algunos datos programáticos, a saber, alias y funciones del shell. Cubrimos los alias en el Capítulo 5 y cubriremos las funciones de shell (que están relacionadas con el script de shell) en la Parte IV de este libro.

Examining the Environment
Examinar el medio ambiente

To see what is stored in the environment, we can use either the set builtin in bash or the printenv program. The set command will show both the shell and environment variables, while printenv will display only the latter. Because the list of environment contents will be fairly long, it is best to pipe the output of either command into less.
To see what is stored in the environment, we can use either the set builtin in bash or the printenv program. The set command will show both the shell and environment variables, while printenv will display only the latter. Because the list of environment contents will be fairly long, it is best to pipe the output of either command into less.

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

Doing so, we should get something that looks like this:
Al hacerlo, deberíamos obtener algo parecido a esto:

```
SHELL=/bin/bash
SESSION_MANAGER=local/EliteDesk:@/tmp/.ICE-
unix/3496,unix/EliteDesk:/tmp/.ICE-unix/3496
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
HISTCONTROL=ignoredups
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
HISTSIZE=1000
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
```

```
JAVA_HOME=/usr/lib/jvm/java-1.14.0-openjdk-amd64
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=3454
GTK_MODULES=gail:atk-bridge
PWD=/home/hernani
LOGNAME=hernani
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
WINDOWPATH=2
HOME=/home/hernani
USERNAME=hernani
IM_CONFIG_PHASE=1
LANG=es_ES.UTF-8
```

What we see is a list of environment variables and their values. For example, we see a variable called USER , which contains the value me . The printenv command can also list the value of a specific variable.
Lo que vemos es una lista de variables de entorno y sus valores. Por ejemplo, vemos una variable llamada USUARIO, que contiene el valor yo. El comando printenv también puede enumerar el valor de una variable específica.

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

The set command, when used without options or arguments, will display both the shell and environment variables, as well as any defined shell functions. Unlike printenv , its output is courteously sorted in alphabetical order.
El comando set, cuando se usa sin opciones o argumentos, mostrará tanto el shell como las variables de entorno, así como cualquier función de shell definida. A diferencia de printenv, su salida se ordena cortésmente en orden alfabético.

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

It is also possible to view the contents of a variable using the echo command, like this:
También es posible ver el contenido de una variable usando el comando echo, así:

```
[me@linuxbox ~]$ echo $HOME
/home/me
```

One element of the environment that neither set nor printenv displays is aliases. To see them, enter the alias command without arguments.

Un elemento del entorno que ni set ni printenv muestra son los alias. Para verlos, ingrese el comando alias sin argumentos.

```
[me@linuxbox ~]$ alias
alias l.='ls -d .* --color=tty'
alias ll='ls -l --color=tty'
alias ls='ls --color=tty'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --
show-tilde'
```

Some Interesting Variables
Algunas variables interesantes

The environment contains quite a few variables, and though the environment will differ from the one presented here, we will likely see the variables listed in Table 11-1 in our environment.
El entorno contiene bastantes variables, y aunque el entorno será diferente del que se presenta aquí, es probable que veamos las variables enumeradas en la Tabla 11-1 en nuestro entorno.

Table 11-1: Environment Variables
Tabla 11-1: Variables de entorno

Variable	Contenido
DISPLAY	El nombre de su pantalla si está ejecutando un entorno gráfico. Por lo general, esto es: 0, lo que significa la primera pantalla generada por el servidor X.
EDITOR	El nombre del programa que se utilizará para la edición de texto.
SHELL	El nombre de su programa de shell.
HOME	El nombre de ruta de su directorio personal.
LANG	Define el juego de caracteres y el orden de clasificación de su idioma.
OLDPWD	El directorio de trabajo anterior.
PAGER	El nombre del programa que se utilizará para la salida de paginación. Suele estar configurado en /usr/bin/less.
PATH	RUTA, una lista de directorios separados por dos puntos que se buscan cuando ingresa nombre de un programa ejecutable.
PS1	Significa cadena de solicitud 1. Esto define el contenido del indicador de shell. Como veremos más adelante, esto se puede personalizar ampliamente.
PWD	El directorio de trabajo actual.

Variable	Contenido
TERM	El nombre de su tipo de terminal. Los sistemas similares a Unix admiten muchos protocolos de terminal; esta variable establece el protocolo que se utilizará con su emulador de terminal.
TZ	Especifica su zona horaria. La mayoría de los sistemas similares a Unix mantienen el reloj interno de la computadora en la hora universal coordinada (UTC) y luego muestran la hora local aplicando un desplazamiento especificado por esta variable.
USER	USUARIO, Su nombre de usuario.

Don't worry if some of these values are missing. They vary by distribution.
No se preocupe si faltan algunos de estos valores. Varían según la distribución.

How Is the Environment Established?
¿Cómo se establece el medio ambiente?

When we log on to the system, the bash program starts and reads a series of configuration scripts called startup files, which define the default environment shared by all users. This is followed by more startup files in our home directory that define our personal environment. The exact sequence depends on the type of shell session being started. There are two kinds.
Cuando iniciamos sesión en el sistema, el programa bash se inicia y lee una serie de scripts de configuración llamados archivos de inicio, que definen el entorno predeterminado compartido por todos los usuarios. A esto le siguen más archivos de inicio en nuestro directorio de inicio que definen nuestro entorno personal. La secuencia exacta depende del tipo de sesión de shell que se inicie. Hay dos tipos.

- **A login shell session** This is one in which we are prompted for our username and password. This happens when we start a virtual console session, for example.
  Una sesión de shell de inicio de sesión Esta es una en la que se nos solicita nuestro nombre de usuario y contraseña. Esto sucede cuando iniciamos una sesión de consola virtual, por ejemplo.

- **A non-login shell session** This typically occurs when we launch a terminal session in the GUI.
  ** Una sesión de shell sin inicio de sesión ** Esto suele ocurrir cuando iniciamos una sesión de terminal en la GUI.

Login shells read one or more startup files, as shown in Table 11-2.
Los shells de inicio de sesión leen uno o más archivos de inicio, como se muestra en la Tabla 11-2.

> Table 11-2: Startup Files for Login Shell Sessions
> Tabla 11-2: Archivos de inicio para sesiones de shell de inicio de sesión

File	Contents
/etc/profile	A global configuration script that applies to all users. Un script de configuración global que se aplica a todos los usuarios.

File	Contents
~/.bash_profile	A user's personal startup file. It can be used to extend or override settings in the global configuration script. Archivo de inicio personal de un usuario. Se puede utilizar para ampliar o anular la configuración en el script de configuración global.
~/.bash_login	If ~/.bash_profile is not found, bash attempts to read this script. Si no se encuentra ~ / .bash_profile, bash intenta leer este script.
~/.profile	If neither ~/.bash_profile nor ~/.bash_login is found, bash attempts to read this file. This is the default in Debian-based distributions, such as Ubuntu. Si no se encuentran ~ / .bash_profile ni ~ / .bash_login, bash intenta leer este archivo. Este es el valor predeterminado en las distribuciones basadas en Debian, como Ubuntu.

Non-login shell sessions read the startup files listed in Table 11-3.
Las sesiones de shell que no son de inicio de sesión leen los archivos de inicio que se enumeran en la Tabla 11-3.

> Table 11-3: Startup Files for Non-Login Shell Sessions
> Tabla 11-3: Archivos de inicio para sesiones de shell sin inicio de sesión

File	Contents
/etc/bash.bashrc	A global configuration script that applies to all users. Un script de configuración global que se aplica a todos los usuarios.
~/.bashrc	A user's personal startup file. It can be used to extend or override settings in the global configuration script. Archivo de inicio personal de un usuario. Se puede utilizar para ampliar o anular la configuración en el script de configuración global.

In addition to reading the startup files listed in Table 11-3, non-login shells inherit the environment from their parent process, usually a login shell.
Además de leer los archivos de inicio enumerados en la Tabla 11-3, los shells que no son de inicio de sesión heredan el entorno de su proceso principal, generalmente un shell de inicio de sesión.

Take a look and see which of these startup files are installed. Remember, because most of the filenames listed start with a period (meaning that they are hidden), we will need to use the -a option when using ls.
Eche un vistazo y vea cuáles de estos archivos de inicio están instalados. Recuerde, debido a que la mayoría de los nombres de archivo enumerados comienzan con un punto (lo que significa que están ocultos), necesitaremos usar la opción -a cuando use ls.

The ~/.bashrc file is probably the most important startup file from the ordinary user's point of view, because it is almost always read. Non-login shells read it by default, and most startup files for login shells are written in such a way as to read the ~/.bashrc file as well.
El archivo ~ / .bashrc es probablemente el archivo de inicio más importante desde el punto de vista del usuario común, porque casi siempre se lee. Los shells que no son de inicio de sesión lo leen de forma

predeterminada, y la mayoría de los archivos de inicio para los shells de inicio de sesión se escriben de tal manera que también pueden leer el archivo ~ / .bashrc.

What's in a Startup File?
¿Qué hay en un archivo de inicio?

If we take a look inside a typical .bash_profile (taken from a CentOS 6 system), it looks something like this:
Si echamos un vistazo dentro de un típico .bash_profile (tomado de un sistema CentOS 6), se ve así:

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi
# User specific environment and startup programs

PATH=$PATH:$HOME/bin
export PATH
```

Lines that begin with a # are comments and are not read by the shell. These are there for human readability. The first interesting thing occurs on the fourth line, with the following code:
Las líneas que comienzan con # son comentarios y el shell no las lee. Estos están ahí para la legibilidad humana. Lo primero interesante ocurre en la cuarta línea, con el siguiente código:

```
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi
```

This is called an if compound command, which we will cover fully when we get to shell scripting in Part IV, but for now, here is a translation:
Esto se llama un comando compuesto if, que cubriremos completamente cuando lleguemos a las secuencias de comandos de shell en la Parte IV, pero por ahora, aquí hay una traducción:

```
If the file "~/.bashrc" exists, then
        read the "~/.bashrc" file.
```

We can see that this bit of code is how a login shell gets the contents of .bashrc. The next thing in our startup file has to do with the PATH variable.
Podemos ver que este fragmento de código es la forma en que un shell de inicio de sesión obtiene el contenido de .bashrc. Lo siguiente en nuestro archivo de inicio tiene que ver con la variable PATH.

Ever wonder how the shell knows where to find commands when we enter them on the command line? For example, when we enter ls , the shell does not search the entire computer to find /bin/ls (the full pathname of the ls command); rather, it searches a list of directories that are contained in the PATH variable.
¿Alguna vez se preguntó cómo el shell sabe dónde encontrar los comandos cuando los ingresamos en la línea de comandos? Por ejemplo, cuando ingresamos ls, el shell no busca en toda la computadora para encontrar / bin / ls (la ruta completa del comando ls); más bien, busca en una lista de directorios que están contenidos en la variable PATH.

The PATH variable is often (but not always, depending on the distribution) set by the /etc/profile startup file with this code:
La variable PATH es a menudo (pero no siempre, dependiendo de la distribución) establecida por el archivo de inicio / etc / profile con este código:

```
PATH=$PATH:$HOME/bin
```

PATH is modified to add the directory $HOME/bin to the end of the list.
PATH se modifica para agregar el directorio $ HOME / bin al final de la lista.

This is an example of parameter expansion, which we touched on in Chapter 7. To demonstrate how this works, try the following:
Este es un ejemplo de expansión de parámetros, que mencionamos en el Capítulo 7. Para demostrar cómo funciona, intente lo siguiente:

```
[me@linuxbox~]$ foo="This is some "
[me@linuxbox~]$ echo $foo
This is some
[me@linuxbox~]$ foo=$foo"text."
[me@linuxbox~]$ echo $foo
This is some text.
```

Using this technique, we can append text to the end of a variable’s contents.
Con esta técnica, podemos agregar texto al final del contenido de una variable.

By adding the string $HOME/bin to the end of the PATH variable’s contents, the directory $HOME/bin is added to the list of directories searched when a command is entered. This means that when we want to create a directory within our home directory for storing our own private programs, the shell is ready to accommodate us. All we have to do is call it bin , and we’re ready to go.
Al agregar la cadena $ HOME / bin al final del contenido de la variable PATH, el directorio $ HOME / bin se agrega a la lista de directorios buscados cuando se ingresa un comando. Esto significa que cuando queremos crear un directorio dentro de nuestro directorio de inicio para almacenar nuestros propios programas privados, el shell está listo para acomodarnos. Todo lo que tenemos que hacer es llamarlo papelera y estamos listos para comenzar.

> **Note**
> **Nota**

> Many distributions provide this PATH setting by default. Debian-based distributions, such as Ubuntu, test for the existence of the ~/bin directory at login and dynamically add it to the PATH variable if the directory is found.
> Muchas distribuciones proporcionan esta configuración PATH de forma predeterminada. Las distribuciones basadas en Debian, como Ubuntu, prueban la existencia del directorio ~ / bin al iniciar sesión y lo agregan dinámicamente a la variable PATH si se encuentra el directorio.

Lastly, we have this:
Por último, tenemos esto:

```
export PATH
```

The export command tells the shell to make the contents of PATH available to child processes of this shell.
El comando de exportación le dice al shell que haga que el contenido de PATH esté disponible para los procesos secundarios de este shell.

Modifying the Environment
Modificar el medio ambiente

Because we know where the startup files are and what they contain, we can modify them to customize our environment.
Como sabemos dónde están los archivos de inicio y qué contienen, podemos modificarlos para personalizar nuestro entorno.

Which Files Should We Modify?
¿Qué archivos debemos modificar?

As a general rule, to add directories to your PATH or define additional environment variables, place those changes in .bash_profile (or the equivalent,according to your distribution; for example, Ubuntu uses .profile). For everything else, place the changes in .bashrc.
Como regla general, para agregar directorios a su PATH o definir variables de entorno adicionales, coloque esos cambios en .bash_profile (o el equivalente, según su distribución; por ejemplo, Ubuntu usa .profile). Para todo lo demás, coloque los cambios en .bashrc.

> # Note
> # Nota
>
> Unless you are the system administrator and need to change the defaults for all users of the system, restrict your modifications to the files in your home directory. It is certainly possible to change the files in /etc such as profile, and in many cases it would be sensible to do so, but for now, let's play it safe.
> A menos que sea el administrador del sistema y necesite cambiar los valores predeterminados para todos los usuarios del sistema, restrinja sus modificaciones a los archivos en su directorio personal.

> Ciertamente es posible cambiar los archivos en / etc, como el perfil, y en muchos casos sería sensato hacerlo, pero por ahora, vayamos a lo seguro.

Text Editors
Editores de texto

To edit (that is, modify) the shell's startup files, as well as most of the other configuration files on the system, we use a program called a text editor. A text editor is a program that is, in some ways, like a word processor in that it allows us to edit the words on the screen with a moving cursor. It differs from a word processor by only supporting pure text and often contains features designed for writing programs. Text editors are the central tool used by software developers to write code and by system administrators to manage the configuration files that control the system.
Para editar (es decir, modificar) los archivos de inicio del shell, así como la mayoría de los demás archivos de configuración del sistema, usamos un programa llamado editor de texto. Un editor de texto es un programa que, de alguna manera, es como un procesador de texto en el sentido de que nos permite editar las palabras en la pantalla con un cursor en movimiento. Se diferencia de un procesador de texto porque solo admite texto puro y, a menudo, contiene funciones diseñadas para programas de escritura. Los editores de texto son la herramienta central utilizada por los desarrolladores de software para escribir código y por los administradores del sistema para administrar los archivos de configuración que controlan el sistema.

A lot of different text editors are available for Linux; most systems have several installed. Why so many different ones? Because programmers like writing them and because programmers use them extensively, they write editors to express their own desires as to how they should work.
Hay muchos editores de texto diferentes disponibles para Linux; la mayoría de los sistemas tienen varios instalados. ¿Por qué tantos diferentes? Debido a que a los programadores les gusta escribirlos y debido a que los programadores los usan ampliamente, escriben editores para expresar sus propios deseos sobre cómo deberían funcionar.

Text editors fall into two basic categories: graphical and text-based. GNOME and KDE both include some popular graphical editors. GNOME ships with an editor called gedit , which is usually called "Text Editor" in the GNOME menu. KDE usually ships with three, which are (in order of increasing complexity) kedit , kwrite , and kate .
Los editores de texto se dividen en dos categorías básicas: gráficos y basados en texto. Tanto GNOME como KDE incluyen algunos editores gráficos populares. GNOME viene con un editor llamado gedit, que generalmente se llama "Editor de texto" en el menú GNOME. KDE generalmente viene con tres, que son (en orden de complejidad creciente) kedit, kwrite y kate.

There are many text-based editors. The popular ones we'll encounter are nano , vi , and emacs . The nano editor is a simple, easy-to-use editor designed as a replacement for the pico editor supplied with the PINE email suite. The vi editor (which on most Linux systems has been replaced by a program named vim , which is short for "vi improved") is the traditional editor for Unix-like systems. It will be the subject of Chapter 12. The emacs editor was originally written by Richard Stallman. It is a gigantic, all-purpose, does everything programming environment. While readily available, it is seldom installed on most Linux systems by default.
Hay muchos editores basados en texto. Los más populares que encontraremos son nano, vi y emacs. El editor nano es un editor simple y fácil de usar diseñado como reemplazo del editor pico suministrado con el paquete de correo electrónico PINE. El editor vi (que en la mayoría de los sistemas Linux ha sido

reemplazado por un programa llamado vim, que es la abreviatura de “vi mejorado”) es el editor tradicional para sistemas similares a Unix. Será el tema del Capítulo 12. El editor de emacs fue escrito originalmente por Richard Stallman. Es un entorno de programación gigantesco, multiusos, que hace todo. Si bien está disponible, rara vez se instala en la mayoría de los sistemas Linux de forma predeterminada.

Using a Text Editor
Usando un editor de texto

Text editors can be invoked from the command line by typing the name of the editor followed by the name of the file you want to edit. If the file does not already exist, the editor will assume that we want to create a new file.
Los editores de texto se pueden invocar desde la línea de comandos escribiendo el nombre del editor seguido del nombre del archivo que desea editar. Si el archivo aún no existe, el editor asumirá que queremos crear un nuevo archivo.

Here is an example using gedit:
Aquí hay un ejemplo usando gedit:

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

This command will start the gedit text editor and load the file named some_file, if it exists.
Este comando iniciará el editor de texto gedit y cargará el archivo llamado some_file, si existe.

Graphical text editors are pretty self-explanatory, so we won’t cover them here. Instead, we will concentrate on our first text-based text editor, nano . Let’s fire up nano and edit the .bashrc file. But before we do that, let’s practice some “safe computing.” Whenever we edit an important configuration file, it is always a good idea to create a backup copy of the file first. This protects us in case we mess up the file while editing. To create a backup of the .bashrc file, do this:
Los editores de texto gráfico se explican por sí mismos, por lo que no los cubriremos aquí. En cambio, nos concentraremos en nuestro primer editor de texto basado en texto, nano. Iniciemos nano y editemos el archivo .bashrc. Pero antes de hacer eso, practiquemos algo de "informática segura". Siempre que editemos un archivo de configuración importante, siempre es una buena idea crear primero una copia de seguridad del archivo. Esto nos protege en caso de que estropeemos el archivo durante la edición. Para crear una copia de seguridad del archivo .bashrc, haga lo siguiente:

```
[me@linuxbox ~]$ cp .bashrc .bashrc.bak
```

It doesn’t matter what we call the backup file; just pick an understandable name. The extensions .bak, .sav, .old, and .orig are all popular ways of indicating a backup file. Oh, and remember that cp will overwrite existing files silently.
No importa cómo llamemos al archivo de respaldo; simplemente elija un nombre comprensible. Las extensiones .bak, .sav, .old y .orig son todas formas populares de indicar un archivo de respaldo. Ah, y recuerde que cp sobrescribirá los archivos existentes de forma silenciosa.

Now that we have a backup file, we’ll start the editor.
Ahora que tenemos un archivo de respaldo, iniciaremos el editor.

```
[me@linuxbox ~]$ nano .bashrc
```

Once nano starts, we'll get a screen like this:
Una vez que nano se inicia, obtendremos una pantalla como esta:

```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000
                              [ 137 líneas leídas ]
^G Ver ayuda ^O Guardar   ^W Buscar    ^K Cortar Tex^J Justificar^C
Posición
^X Salir     ^R Leer fich.^\ Reemplazar^U Pegar     ^T Ortografía^_ Ir a
línea
```

Note
Nota

> If your system does not have nano installed, you may use a graphical editor instead.
> Si su sistema no tiene nano instalado, puede usar un editor gráfico en su lugar.

The screen consists of a header at the top, the text of the file being edited in the middle, and a menu of commands at the bottom. Because nano was designed to replace the text editor supplied with an email client, it is rather short on editing features.
La pantalla consta de un encabezado en la parte superior, el texto del archivo que se está editando en el medio y un menú de comandos en la parte inferior. Debido a que nano fue diseñado para reemplazar el editor de texto provisto con un cliente de correo electrónico, es bastante corto en funciones de edición.

The first command you should learn in any text editor is how to exit the program. In the case of nano , you press ctrl -X to exit. This is indicated in the menu at the bottom of the screen. The notation ^X means ctrl -X. This is a common notation for control characters used by many programs.
El primer comando que debe aprender en cualquier editor de texto es cómo salir del programa. En el caso de nano, presione ctrl -X para salir. Esto se indica en el menú en la parte inferior de la pantalla. La notación

^ X significa ctrl -X. Esta es una notación común para los caracteres de control que utilizan muchos programas.

The second command we need to know is how to save our work. With nano it's ctrl -O. With this knowledge, we're ready to do some editing. Using the down arrow key and/or the page down key, move the cursor to the end of the file and then add the following lines to the .bashrc file:
El segundo comando que debemos saber es cómo guardar nuestro trabajo. Con nano es ctrl -O. Con este conocimiento, estamos listos para editar. Con la tecla de flecha hacia abajo y / o la tecla de página hacia abajo, mueva el cursor al final del archivo y luego agregue las siguientes líneas al archivo .bashrc:

```
umask 0002
export HISTCONTROL=ignoredups
export HISTSIZE=1000
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
```

> **Note**
> **Nota**

> Your distribution may already include some of these, but duplicates won't hurt anything.
> Es posible que su distribución ya incluya algunos de estos, pero los duplicados no afectarán nada.

Table 11-4 details the meaning of our additions.
La Tabla 11-4 detalla el significado de nuestras adiciones.

> Table 11-4: Additions to Our .bashrc
> Tabla 11-4: Adiciones a nuestro .bashrc

Line	Meaning
umask 0002	Sets the umask to solve the problem with the shared directories we discussed in Chapter 9. Establece la umask para resolver el problema con los directorios compartidos que discutimos en el Capítulo 9.
export HISTCONTROL=ignoredups	Causes the shell's history recording feature to ignore a command if the same command was just recorded. Hace que la función de registro del historial del shell ignore un comando si el mismo comando se acaba de grabar.
export HISTSIZE=1000	Increases the size of the command history from the usual default of 500 lines to 1,000 lines. Aumenta el tamaño del historial de comandos del valor predeterminado habitual de 500 líneas a 1000 líneas.

Line	Meaning
alias l.='ls -d .* --color=auto'	Creates a new command called l. , which displays all directory entries that begin with a dot. Crea un nuevo comando llamado l. , que muestra todas las entradas del directorio que comienzan con un punto.
alias ll='ls -l --color=auto'	Creates a new command called ll , which displays a long-format directory listing. Crea un nuevo comando llamado ll, que muestra una lista de directorios de formato largo.

As we can see, many of our additions are not intuitively obvious, so it would be a good idea to add some comments to our .bashrc file to help explain things to the humans. Using the editor, change our additions to look like this:
Como podemos ver, muchas de nuestras adiciones no son intuitivamente obvias, por lo que sería una buena idea agregar algunos comentarios a nuestro archivo .bashrc para ayudar a explicar las cosas a los humanos. Usando el editor, cambie nuestras adiciones para que se vean así:

```
# Change umask to make directory sharing easier
umask 0002
# Ignore duplicates in command history and increase
# history size to 1000 lines
export HISTCONTROL=ignoredups
export HISTSIZE=1000
# Add some helpful aliases
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
```

Ah, much better! With our changes complete, press ctrl -O to save our modified .bashrc file, and press ctrl -X to exit nano.
¡Ah, mucho mejor! Con nuestros cambios completos, presione ctrl -O para guardar nuestro archivo .bashrc modificado, y presione ctrl -X para salir de nano.

Why Comments Are Important
Por qué los comentarios son importantes

Whenever you modify configuration files, it's a good idea to add some comments to document your changes. Sure, you'll probably remember what you changed tomorrow, but what about six months from now? Do yourself a favor and add some comments. While you're at it, it's not a bad idea to keep a log of what changes you make.
Siempre que modifique archivos de configuración, es una buena idea agregar algunos comentarios para documentar sus cambios. Seguro, probablemente recordarás lo que cambiaste mañana, pero ¿qué tal dentro de seis meses? Hágase un favor y agregue algunos comentarios. Mientras lo hace, no es mala idea llevar un registro de los cambios que realiza.

> Shell scripts and bash startup files use a # symbol to begin a comment. Other configuration files may use other symbols. Most configuration files will have comments. Use them as a guide.
> Los scripts de shell y los archivos de inicio de bash usan un símbolo # para comenzar un comentario. Otros archivos de configuración pueden utilizar otros símbolos. La mayoría de los archivos de configuración tendrán comentarios. Úselos como guía.

> You will often see lines in configuration files that are commented out to prevent them from being used by the affected program. This is done to give the reader suggestions for possible configuration choices or examples of correct configuration syntax. For example, the .bashrc file of Ubuntu 18.04 contains these lines:
> A menudo verá líneas en los archivos de configuración que están comentadas para evitar que sean utilizadas por el programa afectado. Esto se hace para ofrecer al lector sugerencias sobre posibles opciones de configuración o ejemplos de sintaxis de configuración correcta. Por ejemplo, el archivo .bashrc de Ubuntu 18.04 contiene estas líneas:

```
# some more ls aliases
#alias ll='ls -l'
#alias la='ls -A'
#alias l='ls -CF'
```

The last three lines are valid alias definitions that have been commented out. If you remove the leading # symbols from these three lines, a technique called uncommenting, you will activate the aliases. Conversely, if you add a # symbol to the beginning of a line, you can deactivate a configuration line while preserving the information it contains.
Las últimas tres líneas son definiciones de alias válidas que se han comentado. Si elimina los símbolos # iniciales de estas tres líneas, una técnica llamada descomentar, activará los alias. Por el contrario, si agrega un símbolo # al principio de una línea, puede desactivar una línea de configuración conservando la información que contiene.

Activating Our Changes
Activando Nuestros Cambios

The changes we have made to our .bashrc will not take effect until we close our terminal session and start a new one because the .bashrc file is read only at the beginning of a session. However, we can force bash to reread the modified.
Los cambios que hemos realizado en nuestro .bashrc no tendrán efecto hasta que cerremos nuestra sesión de terminal e iniciemos una nueva porque el archivo .bashrc es de solo lectura al comienzo de una sesión. Sin embargo, podemos obligar a bash a volver a leer el archivo.

bashrc file with the following command:
archivo bashrc con el siguiente comando:

```
[me@linuxbox ~]$ source ~/.bashrc
```

After doing this, we should be able to see the effect of our changes. Try one of the new aliases.
Después de hacer esto, deberíamos poder ver el efecto de nuestros cambios. Pruebe uno de los nuevos

alias.

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

Summing Up
Resumen

In this chapter, we learned an essential skill: editing configuration files with a text editor. Moving forward, as we read man pages for commands, take note of the environment variables that commands support. There may be a gem or two. In later chapters, we will learn about shell functions, a powerful feature that you can also include in the bash startup files to add to your arsenal of custom commands.
En este capítulo, aprendimos una habilidad esencial: editar archivos de configuración con un editor de texto. En el futuro, a medida que leemos las páginas de manual de los comandos, tome nota de las variables de entorno que admiten los comandos. Puede haber una joya o dos. En capítulos posteriores, aprenderemos sobre las funciones de shell, una característica poderosa que también puede incluir en los archivos de inicio de bash para agregar a su arsenal de comandos personalizados.