**Cómo leer y configurar variables de entorno y de shell en Linux**

## [Introducción](https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-linux-es#introduccion)

Cuando interactúa con su servidor a través una sesión de shell, hay muchos elementos de información que el shell compila para determinar su comportamiento y acceso a los recursos. Algunas de estas opciones se incluyen en las opciones de configuración, y otras las determina la entrada del usuario.

Una forma en que el shell realiza un seguimiento de todas estas configuraciones y detalles es en un área que se denomina **entorno**. El entorno es un área que el shell crea cada vez que inicia una sesión que contiene variables que definen las propiedades del sistema.

En esta guía, analizaremos cómo interactuar con el entorno y leer o configurar variables de entorno y de shell de forma interactiva y mediante archivos de configuración.

## [Cómo funcionan el entorno y las variables de entorno](https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-linux-es#como-funcionan-el-entorno-y-las-variables-de-entorno)

Cada vez que se genera una sesión de shell, se lleva a cabo un proceso para recopilar y compilar información que debería estar disponible para el proceso de shell y sus procesos secundarios. Los datos para estas configuraciones se obtienen de diversos archivos y configuraciones en el sistema.

El entorno proporciona un medio a través del que el proceso de shell puede obtener o establecer configuraciones y, a la vez, transmitirlas a sus procesos secundarios.

El entorno se implementa como cadenas que representan pares clave-valor. Si se transmiten múltiples valores, estos suelen separarse mediante símbolos de dos puntos (:). Normalmente, cada par tendrá un aspecto similar a este:

KEY=value1:value2:...

Si el valor contiene un espacio en blanco significativo, se utilizan comillas:

KEY="value with spaces"

Las claves en estos casos son variables. Pueden ser de dos tipos: variables de entorno o variables de shell.

Las **variables de entorno** son variables definidas para el shell actual y heredadas por cualquier shell o proceso secundario. Las variables de entorno se utilizan para transmitir información a procesos que se producen desde el shell.

Las **variables de shell** son variables que se encuentran **exclusivamente dentro del shell** en que se configuraron o definieron. A menudo, se utilizan para realizar un seguimiento de datos efímeros, como el directorio actual de trabajo.

Por costumbre, estos tipos de variables suelen definirse utilizando letras en mayúsculas. Esto ayuda a los usuarios a distinguir las variables de entorno en otros contextos.

[**Impresión de variables de entorno y de shell**](https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-linux-es#impresion-de-variables-de-entorno-y-de-shell)

Cada sesión de shell hace un seguimiento de sus propias variables de entorno y de shell. Podemos acceder a ellas de diferentes formas.

Podemos ver una lista de todas nuestras variables de entorno usando los comandos env o printenv. En su estado predeterminado, deben funcionar exactamente igual

Esto es bastante típico del resultado de printenv y env. La diferencia entre los dos comandos es visible solamente en su funcionalidad más específica. Por ejemplo, con printenv, puede solicitar los valores de variables individuales:

***$ printenv SHELL***

Por otro lado, env le permite modificar el entorno en que se ejecutan los programas transmitiendo un conjunto de definiciones de variables a un comando:

1. $ ***env VAR1="valor" comando a ejecutar opciones***

Dado que, como vimos anteriormente, los procesos secundarios suelen heredar las variables de entorno del proceso principal, esto le da la oportunidad de anular valores o agregar otras variables al proceso secundario.

Como se puede ver en el resultado de nuestro comando printenv, hay algunas variables de entorno configuradas en los archivos y procesos de nuestro sistema sin nuestra intervención.

Se muestran las variables de entorno, pero ¿cómo vemos las variables de shell?

El comando set se puede utilizar para esto. Si escribimos set sin ningún otro parámetro, obtendremos una lista de todas las variables de shell, variables de entorno, variables locales y funciones de shell:

***$ set***

Por lo general, es una lista enorme. Es recomendable canalizarla a un programa de buscapersonas para manejar con mayor facilidad la cantidad de resultados:

***$ set |less***

## [Variables frecuentes de entorno y de shell](https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-linux-es#variables-frecuentes-de-entorno-y-de-shell)

Algunas variables de entorno y de shell son muy útiles y se utilizan con bastante frecuencia. Aquí hay algunas variables frecuentes de entorno con las que se encontrará:

* SHELL: Describe el shell que interpretará cualquier comando que ingrese. En la mayoría de los casos, será bash de forma predeterminada, pero se pueden establecer otros valores si prefiere otras opciones.
* TERM: Especifica el tipo de terminal a emular cuando se ejecuta el shell. Se pueden emular diferentes terminales de hardware para diferentes requisitos de funcionamiento. Sin embargo, generalmente no tendrá que preocuparse por esto.
* USER: El usuario que inició sesión actualmente.
* PWD: El directorio actual de trabajo.
* OLDPWD: El directorio anterior de trabajo. Esto se mantiene en el shell para volver a su directorio anterior ejecutando cd -.
* LS\_COLORS: Define los códigos de color que se utilizan para añadir de forma opcional un resultado de color al comando ls. Esto se utiliza para distinguir diferentes tipos de archivos y proporcionar más información al usuario de forma rápida.
* MAIL: La ruta al buzón del usuario actual.
* PATH: Una lista de directorios que el sistema comprobará cuando busque comandos. Cuando un usuario escriba un comando, el sistema comprobará los directorios en este orden para el ejecutable.
* LANG: Las configuraciones actuales de idioma y localización, incluida la codificación de caracteres.
* HOME: El directorio principal del usuario actual.

Además de estas variables de entorno, algunas variables de shell que aparecerán a menudo son:

* BASHOPTS: La lista de opciones que se utilizaron cuando se ejecutó bash. Esto puede ser útil para averiguar si el entorno de shell funcionará de la forma que usted desea.
* BASH\_VERSION: La versión de bash que se está ejecutando, en un formato que los usuarios puedan leer.
* BASH\_VERSINFO: La versión de bash, en un resultado que un equipo pueda leer.
* COLUMNS: El número de ancho de columnas que se utilizan para generar el resultado en la pantalla.
* DIRSTACK: La pila de directorios que están disponibles con los comandos pushd y popd.
* HISTFILESIZE: La cantidad de líneas del historial de comandos que se almancenan en un archivo.
* HISTSIZE: La cantidad de líneas del historial de comandos que se permiten en la memoria.
* HOSTNAME: El nombre de host de la computadora en este momento.
* IFS: El separador de campo interno para separar las entradas en la línea de comandos. De forma predeterminada, es un espacio.
* PS1: La definición de la entrada del comando principal. Esto se utiliza para definir cómo se ve la entrada de la línea de comandos cuando se inicia una sesión de shell. PS2 se utiliza para declarar las entradas de la línea de comandos secundarias para cuando un comando se extiende a varias líneas.
* SHELLOPTS: Opciones de shell que se pueden configurar con la opción set.
* UID: El UID del usuario actual.

## Configuración de variables de entorno y de shell

Para entender mejor la diferencia entre las variables de entorno y de shell, y para introducir la sintaxis para configurar estas variables, haremos una pequeña demostración.

### Creación de variables de shell

Empezaremos definiendo una variable de shell en nuestra sesión actual. Esto es fácil de lograr; solo necesitamos especificar un nombre y un valor. Seguiremos la norma de mantener el nombre de la variable en mayúsculas y lo configuraremos en una cadena simple.

***$TEST\_VAR='Hello World!'***

Aquí utilizamos comillas, ya que el valor de nuestra variable contiene un espacio. Además, usamos comillas simples porque el signo de exclamación es un carácter especial en el shell de bash que generalmente se expande al historial de bash si no se escapa o se introduce en comillas simples.

Ahora, tenemos una variable de shell. Esta variable está disponible en nuestra sesión actual, pero no se transmitirá a los procesos secundarios.

Podemos ver esto realizando una búsqueda de nuestra nueva variable en el resultado de set

***$set | grep TEST\_VAR***

TEST\_VAR='Hello World!'

Podemos verificar que esta no es una variable de entorno intentando lo mismo con printenv

***$ printenv | grep TEST\_VAR***

No debería aparecer ningún resultado.

Aprovecharemos esto como una oportunidad para demostrar una forma de acceder al valor de cualquier variable de entorno o de shell.

***echo $TEST\_VAR***

Como se puede ver, se hace referencia al valor de una variable al precederla con un signo $. El shell interpreta que esto significa que debe sustituir el valor de la variable cuando se encuentre con casos como este.

Ahora tenemos una variable de shell. No debería transmitirse a ningún proceso secundario. Podemos generar un *nuevo* shell de bash desde el actual para demostrar:

***$bash***

***$echo $TEST\_VAR***

Si escribimos bash para generar un shell secundario y, luego, intentamos acceder al contenido de la variable, no se obtendrá ningún resultado. Eso es lo que esperábamos.

Vuelva a nuestro shell original escribiendo exit:

## [Creación de variables de entorno](https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-linux-es#creacion-de-variables-de-entorno)

Ahora, vamos a convertir nuestra variable de shell en una variable de entorno. Podemos hacer esto *exportando* la variable. El comando para hacerlo se denomina correctamente:

$ export TEST\_VAR

Esto convertirá nuestra variable en una variable de entorno. Podemos verificar que el cambio se realizó revisando nuevamente nuestra lista de variables de entorno:

***$ printenv | grep TEST\_VAR***

Esta vez, aparecerá nuestra variable. Intentemos nuestro experimento de nuevo con nuestro shell secundario:

***$ bash***

***$ echo $TEST\_VAR***

 Nuestro shell secundario recibió la variable establecida por el shell principal. Antes de salir de este shell secundario, intentemos exportar otra variable. Podemos configurar variables de entorno en un solo paso de esta manera:

***$ export NEW\_VAR="Testing export"***

Esta es una prueba de que se exportó como variable de entorno:

***$ printenv | grep NEW\_VAR***

NEW\_VAR=Testing export

Ahora, regresemos a nuestro shell original:

***$ exit***

***echo $NEW\_VAR***

No se muestra ningún resultado.

Esto se debe a que las variables de entorno solo se transmiten a procesos secundarios. No hay una forma incorporada de configurar variables de entorno del shell principal. Eso es bueno en la mayoría de casos y evita que los programas afecten al entorno de funcionamiento desde el que fueron invocados.

La variable NEW\_VAR se configuró como una variable de entorno en nuestro shell secundario. Esta variable estaría disponible para sí misma y para cualquiera de **sus** shells y procesos secundarios. Cuando regresemos a nuestro shell principal, el entorno se eliminará.

## [Degradación y anulación de variables](https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-linux-es#degradacion-y-anulacion-de-variables)

Todavía tenemos nuestra variable TEST\_VAR definida como una variable de entorno. Podemos volver a convertirla en una variable de shell escribiendo:

***$ export -n TEST\_VAR***

***$ printenv | grep TEST\_VAR***

Sin embargo, aún es una variable de shell:

***$ set | grep TEST\_VAR***

***TEST\_VAR='Hello World!'***

Si queremos anular por completo una variable, de shell o de entorno, podemos hacerlo con el comando unset:

***$ unset TEST\_VAR***

Podemos verificar que ya no está configurada:

***$ echo $TEST\_VAR***

No se genera ningún resultado porque la variable se anuló.

# [Configuración de variables de entorno al iniciar sesión](https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-linux-es#configuracion-de-variables-de-entorno-al-iniciar-sesion)

Ya mencionamos que muchos programas utilizan variables de entorno para decidir los detalles de cómo operar. No queremos tener que configurar variables importantes cada vez que iniciemos una nueva sesión de shell, y ya vimos cuántas variables ya están configuradas al iniciar sesión, entonces ¿cómo creamos y definimos las variables automáticamente?

En realidad, este es un problema más complejo de lo que parece inicialmente debido a los numerosos archivos de configuración que el shell de bash lee dependiendo de cómo se inicie.

## [Diferencia entre las sesiones de shell interactivas, no interactivas, con inicio de sesión y sin inicio de sesión](https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-linux-es#diferencia-entre-las-sesiones-de-shell-interactivas-no-interactivas-con-inicio-de-sesion-y-sin-inicio-de-sesion)

El shell de bash lee diferentes archivos de configuración dependiendo de cómo se inicia la sesión.

Una distinción entre las diferentes sesiones es si el shell se genera como una sesión **con inicio de sesión** o **sin inicio de sesión**.

Un shell **con inicio de sesión** es una sesión de shell que comienza cuando se autentica al usuario. Si inicia sesión en una sesión de terminal o mediante SSH y realiza una autenticación, la sesión de shell se configurará como un shell con inicio de sesión.

Si inicia una nueva sesión de shell desde la sesión autenticada, como hicimos invocando al comando bash desde el terminal, se inicia una sesión de shell **sin inicio de sesión**. No se solicitaron sus detalles de autenticación cuando inició su shell secundario.

Otra distinción que se puede hacer es si una sesión de shell es interactiva o no interactiva.

Una sesión de shell **interactiva** es una sesión de shell que se conecta a un terminal. Una sesión de shell **no interactiva** es una que no se conecta a una sesión de terminal.

Cada sesión de shell se clasifica como una sesión con inicio de sesión o sin inicio de sesión y como interactiva o no interactiva.

Una sesión normal que comienza con SSH suele ser un shell con inicio de sesión interactivo. Normalmente, una secuencia de comandos ejecutada desde la línea de comandos se ejecuta en un shell no interactivo y sin inicio de sesión. Una sesión de terminal puede ser cualquier combinación de estas dos propiedades.

Si una sesión de shell se clasifica como un shell con inicio de sesión o un shell sin inicio de sesión, tiene repercusiones sobre qué archivos se leen para iniciar la sesión de shell.

Una sesión iniciada como sesión con inicio de sesión leerá primero los detalles de la configuración del archivo /etc/profile. Luego, buscará el primer archivo de configuración de shell con inicio de sesión en el directorio principal del usuario para obtener detalles de configuración específicos del usuario.

Lee el primer archivo que puede encontrar con ~/.bash\_profile, ~/.bash\_login y ~/.profile, y no lee ningún otro archivo.

Por el contrario, una sesión definida como shell sin inicio de sesión leerá /etc/bash.bashrc y, luego, el archivo ~/.bashrc específico del usuario para crear su entorno.

Los shells no interactivos leen la variable de entorno llamada BASH\_ENV y leen el archivo especificado para definir el nuevo entorno.

## [Implementación de variables de entorno](https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-linux-es#implementacion-de-variables-de-entorno)

Como se puede ver, hay diversos archivos que generalmente necesitaríamos revisar para realizar nuestra configuración.

Eso ofrece gran flexibilidad que puede ser útil en situaciones específicas en que queremos ciertas configuraciones en un shell con inicio de sesión y otras configuraciones en un shell sin inicio de sesión. Sin embargo, la mayoría de las veces preferiremos las mismas configuraciones en ambas situaciones.

Por suerte, la mayoría de las distribuciones de Linux configura los archivos de configuración con inicio de sesión para obtener los archivos de configuración sin inicio de sesión. Eso significa que puede definir variables de entorno que quiera que sean de ambos tipos de sesión dentro de los archivos de configuración sin inicio de sesión. Luego, se leerán en ambos casos.

Por lo general, configuraremos variables de entorno específicas de usuario y, normalmente, preferiremos que nuestras configuraciones estén disponibles en shells con inicio de sesión y sin inicio de sesión. Eso significa que el lugar para definir estas variables está en el archivo ~/.bashrc.

Abra este archivo ahora:

***$ nano ~/.bashrc***

Lo más probable es que ya contenga bastantes datos. Muchas de las definiciones aquí son para configurar opciones de bash, que no están relacionadas con variables de entorno. Puede configurar variables de entorno de la misma manera en que lo haría en la línea de comandos:

***$ export VARNAME=value***

Cualquier nueva variable de entorno se puede añadir en cualquier lugar del archivo ~/.bashrc, siempre que no se coloque en el medio de otro comando o para crear un bucle. Luego, podemos guardar y cerrar el archivo. La próxima vez que inicie una sesión de shell, su declaración de variable de entorno se leerá y pasará al entorno de shell. Puede forzar su sesión actual a leer el archivo escribiendo lo siguiente:

***source ~/.bashrc***

Si necesita configurar variables para todo el sistema, es recomendable añadirlas a /etc/profile, /etc/bash.bashrc o /etc/environment.

# [Conclusión](https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-linux-es" \l "conclusion)

Las variables de entorno y de shell siempre están presentes en las sesiones de shell y pueden ser muy útiles. Son una forma interesante para que un proceso principal establezca los detalles de la configuración para sus procesos secundarios y son una forma de configurar opciones fuera de los archivos.

Eso tiene muchas ventajas en situaciones específicas. Por ejemplo, algunos mecanismos de implementación utilizan las variables de entorno para configurar la información de autenticación. Eso es útil porque no requiere guardarlos en archivos que terceros puedan ver.

Hay muchos otros casos más triviales, pero también más frecuentes, en que deberá leer o alterar el entorno de su sistema. Estas herramientas y técnicas le darán una buena base para realizar estos cambios y usarlos correctamente.