

Linux

José Ramón Pérez Navarro

Agosto 2019

Origen, campos de aplicación y características principales

Linux es un sistema operativo: un conjunto de programas que le permiten interactuar con su ordenador y ejecutar otros programas. Nació el 28 de diciembre de 1969 en Finlandia. su sistema operativo consiste en varios programas fundamentales que necesita el ordenador para poder comunicar y recibir instrucciones de los usuarios; tales como leer y escribir datos en el disco duro, cintas, e impresoras; controlar el uso de la memoria; y ejecutar otros programas. La parte más importante de un sistema operativo es el núcleo. En un sistema GNU/Linux, Linux es el núcleo. El resto del sistema consiste en otros programas, muchos de los cuales fueron escritos por o para el proyecto GNU. Dado que el núcleo de Linux en sí mismo no forma un sistema operativo funcional, preferimos utilizar el término “GNU/Linux” para referirnos a los sistemas que la mayor parte de las personas llaman de manera informal “Linux”. Los usuarios de Linux tienen

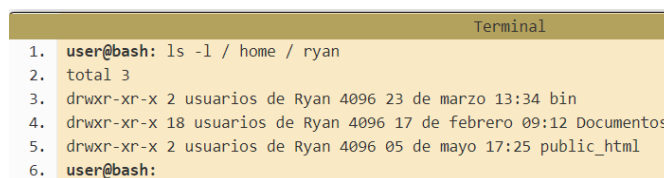
una gran libertad al elegir sus programas. Por ejemplo, un usuario de Linux puede elegir entre docenas de distintos intérpretes de línea de órdenes y entre distintos entornos de escritorio. Es menos probable que un sistema Linux se colapse, además tiene mejor capacidad para ejecutar múltiples programas al mismo tiempo y es más seguro que muchos otros sistemas operativos. Debido a estas ventajas, Linux es el sistema operativo que ha experimentado mayor crecimiento en el mercado de los servidores. Algunas aplicaciones son: procesos de big data, análisis estadístico, visualización y almacenamiento, modelos físicos y matemáticos, programación, etc

La línea de comando

Una línea de comando, o terminal, es una interfaz de texto basada en el sistema. Puede ingresar comandos escribiéndolos en el teclado y recibirá comentarios de manera similar al texto.

La línea de comando generalmente le presenta un aviso. A medida que escribe, se mostrará después de la solicitud. La mayoría de las veces emitirá comandos.

Ejemplo:



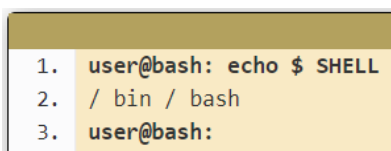
```
Terminal
1. user@bash: ls -l / home / ryan
2. total 3
3. drwxr-xr-x 2 usuarios de Ryan 4096 23 de marzo 13:34 bin
4. drwxr-xr-x 18 usuarios de Ryan 4096 17 de febrero 09:12 Documentos
5. drwxr-xr-x 2 usuarios de Ryan 4096 05 de mayo 17:25 public_html
6. user@bash:
```

Figura 1:

The shell, bash

Dentro de una terminal tienes lo que se conoce como un shell. Esta es una parte del sistema operativo que define cómo se comportará el terminal y qué aspecto tiene después de ejecutar (o ejecutar) comandos por usted. Hay varios shells disponibles, pero el más común se llama bash, que significa Bourne nuevamente.

Ejemplo:



```
1. user@bash: echo $ SHELL
2. / bin / bash
3. user@bash:
```

Figura 2:

o.5cm

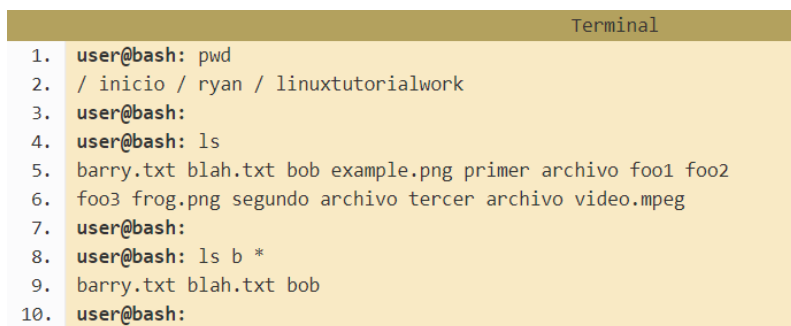
Comodines

Los comodines son un conjunto de bloques de construcción que le permiten crear un patrón que define un conjunto de archivos o directorios.

Aquí está el conjunto básico de comodines:

1. `*` - representa cero o más caracteres
2. `?` - representa un solo personaje
3. `[]` - representa un rango de caracteres

Ejemplo:



```
Terminal
1. user@bash: pwd
2. / inicio / ryan / linuxtutorialwork
3. user@bash:
4. user@bash: ls
5. barry.txt blah.txt bob example.png primer archivo foo1 foo2
6. foo3 frog.png segundo archivo tercer archivo video.mpeg
7. user@bash:
8. user@bash: ls b *
9. barry.txt blah.txt bob
10. user@bash:
```

Figura 3:

Permisos

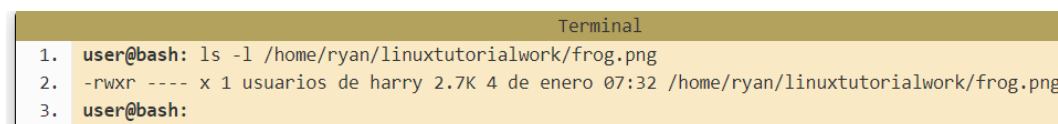
Los permisos de Linux dictan 3 cosas que puede hacer con un archivo, leer, escribir y ejecutar. Se hace referencia a ellos en Linux con una sola letra cada uno.

1. (r) leer: puede ver el contenido del archivo.
2. (w) escribir: puede cambiar el contenido del archivo.
3. (x) ejecutar: puede ejecutar o ejecutar el archivo si es un programa o script.

Para cada archivo definimos 3 conjuntos de personas para quienes podemos especificar permisos.

1. propietario : una sola persona propietaria del archivo. (normalmente la persona que creó el archivo, pero ciertos usuarios pueden otorgarle la propiedad a otra persona)
2. grupo : cada archivo pertenece a un solo grupo.
3. otros : todos los demás que no están en el grupo o el propietario.

Ver permisos, ejemplo:



```
Terminal
1. user@bash: ls -l /home/ryan/linuxtutorialwork/frog.png
2. -rwxr---- x 1 usuarios de harry 2.7K 4 de enero 07:32 /home/ryan/linuxtutorialwork/frog.png
3. user@bash:
```

Figura 4:

Para cambiar los permisos en un archivo o directorio, usamos un comando llamado chmod.

Configuración de permisos abreviados:

Con 3 permisos y cada uno activado o desactivado, tenemos 8 combinaciones posibles (2^3). Ahora también podemos representar nuestros números usando binarios que solo tienen 2 símbolos (0 y 1). El mapeo de octal a binario está en la tabla a continuación.

Grep y expresiones regulares

Las expresiones regulares son similares a los comodines. Nos permiten crear un patrón. Sin embargo, son un poco más poderosos. Los Re generalmente se usan para identificar y manipular datos específicos.

egrep es un programa que buscará un conjunto de datos dado e imprimirá cada línea que contenga un patrón dado.

Expresiones regulares

Octal	Binario
0 0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4 4	1 0 0
5 5	1 0 1
6 6	1 1 0
7 7	1 1 1

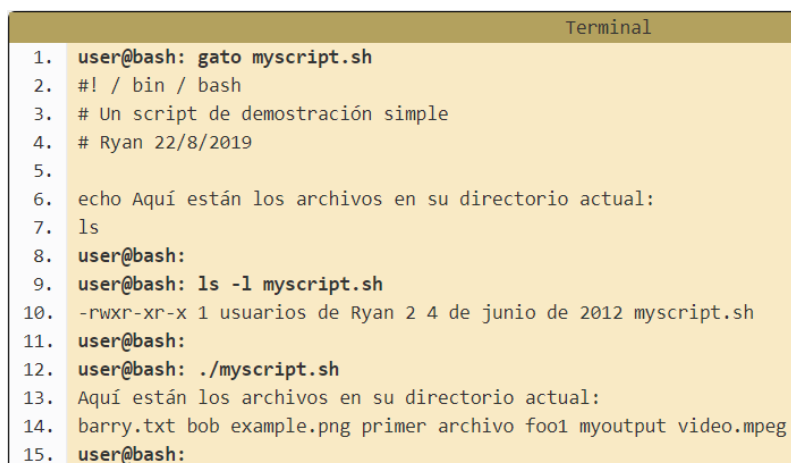
Figura 5:

1. . (punto): un solo carácter.
2. ? - el carácter anterior coincide solo 0 o 1 veces.
3. * - el carácter anterior coincide con 0 o más veces.
4. + : el carácter anterior coincide 1 o más veces.
5. n : el carácter anterior coincide exactamente n veces.
6. n, m : el carácter anterior coincide al menos n veces y no más de m veces.
7. [agd] : el personaje es uno de los incluidos entre corchetes.
8. [^agd] : *el personaje es uno de los incluidos entre corchetes*.
9. [cf]: el guión dentro de los corchetes funciona como un rango. En este caso significa las letras c, d, e o f.
10. () : nos permite agrupar varios caracteres para que se comporten como uno solo.
11. El: (símbolo de tubería): la operación lógica OR.
12. : coincide con el comienzo de la línea.

Bash scripting

Una secuencia de comandos Bash en términos informáticos es similar a una secuencia de comandos en términos teatrales. Es un documento que indica qué decir y qué hacer. Aquí, en lugar de que una persona lea y actúe el guión, la computadora lo lee y ejecuta.

Un script Bash nos permite definir una serie de acciones que la computadora realizará sin que tengamos que ingresar los comandos nosotros mismos. Ejemplo:



```
Terminal
1. user@bash: gato myscrip.sh
2. #! / bin / bash
3. # Un script de demostración simple
4. # Ryan 22/8/2019
5.
6. echo Aquí están los archivos en su directorio actual:
7. ls
8. user@bash:
9. user@bash: ls -l myscrip.sh
10. -rwxr-xr-x 1 usuarios de Ryan 2 4 de junio de 2012 myscrip.sh
11. user@bash:
12. user@bash: ./myscrip.sh
13. Aquí están los archivos en su directorio actual:
14. barry.txt bob example.png primer archivo foo1 myoutput video.mpeg
15. user@bash:
```

Figura 6:

Tubería y redireccionamiento

Cada programa que ejecutamos en la línea de comando tiene automáticamente tres flujos de datos conectados.

1. STDIN (0): entrada estándar (datos introducidos en el programa)
2. STDOUT (1) - Salida estándar (datos impresos por el programa, por defecto en el terminal)
3. STDERR (2): error estándar (para mensajes de error, también predeterminados para el terminal)

Redireccionando un archivo

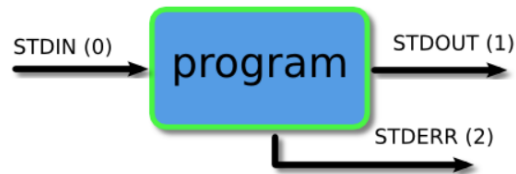


Figura 7:

El operador mayor que (`>`) indica a la línea de comando que deseamos que la salida del programa (o lo que sea que envíe a `STDOUT`) se guarde en un archivo en lugar de imprimirse en la pantalla. Veamos un ejemplo:

	Terminal
1.	<code>user@bash: ls</code>
2.	<code>barry.txt bob example.png primer archivo foo1 video.mpeg</code>
3.	<code>user@bash: ls> myoutput</code>
4.	<code>user@bash: ls</code>
5.	<code>barry.txt bob example.png primer archivo foo1 myoutput video.mpeg</code>
6.	<code>user@bash: gato mi salida</code>
7.	<code>barry.txt</code>
8.	<code>mover</code>
9.	<code>ejemplo.png</code>
10.	<code>primer archivo</code>
11.	<code>foo1</code>
12.	<code>myoutput</code>
13.	<code>video.mpeg</code>
14.	<code>user@bash:</code>

Figura 8:

Redirigir desde un archivo

Si usamos el operador menor que (`<`), entonces podemos enviar datos de la otra manera. Leeremos los datos del archivo y los alimentaremos al programa a través de su flujo `STDIN`.

Redireccionando `STDERR`

`STDERR` es la secuencia número 2 y podemos usar estos números para identificar las secuencias. Si colocamos un número antes del operador `>`, redirigirá esa secuencia (si no usamos un número, como lo hemos estado haciendo hasta ahora, el valor predeterminado es la secuencia 1).

```
Terminal
1. user@bash: wc -l myoutput
2. 8 myoutput
3. user@bash: wc -l <myoutput
4. 8
5. user@bash:
```

Figura 9:

```
Terminal
1. user@bash: ls -l video.mpg blah.foo
2. ls: no se puede acceder a blah.foo: no existe tal archivo o directorio
3. -rwxr - r-- 1 usuarios de ryan 6 16 de mayo 09:14 video.mpg
4. user@bash: ls -l video.mpg blah.foo 2> errors.txt
5. -rwxr - r-- 1 usuarios de ryan 6 16 de mayo 09:14 video.mpg
6. user@bash: cat errors.txt
7. ls: no se puede acceder a blah.foo: no existe tal archivo o directorio
8. user@bash:
```

Figura 10:

Tuberia

Es un mecanismo para enviar datos de un programa a otro. Lo que este operador hace es alimentar la salida del programa a la izquierda como entrada al programa a la derecha. En la figura 11, enumeraremos solo los primeros 3 archivos en el directorio.

En la figura 12 Canalizamos la salida a la cola para obtener solo el tercer archivo:


```
Terminal
1. user@bash: ls
2. barry.txt bob example.png primer archivo foo1 myoutput video.mpeg
3. user@bash: ls | cabeza -3
4. barry.txt
5. mover
6. ejemplo.png
7. user@bash:
```

Figura 11:

```
Terminal
1. user@bash: ls | cabeza -3 | cola -1
2. ejemplo.png
3. user@bash:
```

Figura 12:

Manipulación de archivos

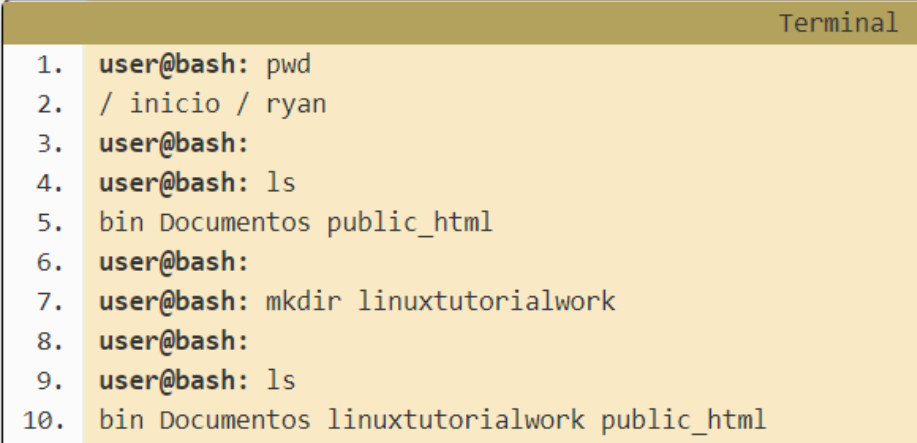
Hacer un directorio

Crear un directorio es bastante fácil. El comando que buscamos es `mkdir`, que es la abreviatura de `Make Directory`.

```
mkdir [opciones] <Directorio>
```

Figura 13:

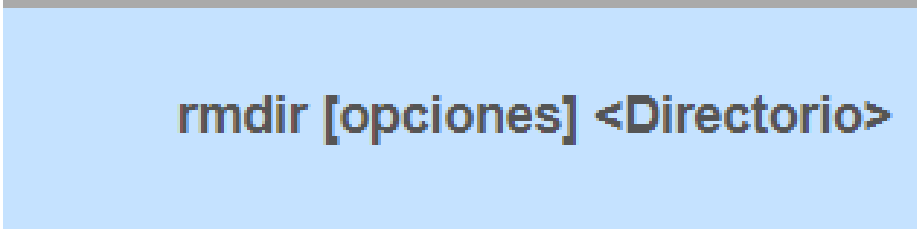
En su forma más básica, podemos ejecutar `mkdir` suministrando solo un directorio y lo creará para nosotros.



```
Terminal
1. user@bash: pwd
2. / inicio / ryan
3. user@bash:
4. user@bash: ls
5. bin Documentos public_html
6. user@bash:
7. user@bash: mkdir linuxtutorialwork
8. user@bash:
9. user@bash: ls
10. bin Documentos linuxtutorialwork public_html
```

Figura 14:

Para eliminar un directorio



```
rmdir [opciones] <Directorio>
```

Figura 15:

Para crear un archivo en blanco

```
toque [opciones] <nombre de archivo>
```

Figura 16:

Para copiar un archivo o directorio

```
cp [opciones] <source> <destination>
```

Figura 17:

Mover un archivo o directorio

```
mv [opciones] <source> <destination>
```

Figura 18:

Para eliminar un archivo

```
rm [opciones] <archivo>
```

Figura 19: