La Shell de Unix y de Linux

La shell de Unix es el término usado en informática para referirse al intérprete de comandos de los sistemas operativos basados en Unix y similares, como **GNU/Linux**, y que es su interfaz de usuario tradicional. Mediante las instrucciones que aporta el intérprete, el usuario puede comunicarse con el núcleo y por extensión, ejecutar dichas órdenes, así como herramientas que le permiten controlar el funcionamiento de la computadora. Por ello, en inglés se le denominó así, shell, que puede ser traducido como «cáscara», porque es la envoltura visible del sistema informático.

Los comandos que aportan los intérpretes, pueden usarse a modo de guion si se escriben en ficheros ejecutables denominados **shell-scripts**, de este modo, cuando el usuario necesita hacer uso de varios comandos o combinados de comandos con herramientas, escribe en un fichero de texto, marcado como ejecutable, las operaciones que posteriormente, línea por línea, el intérprete traducirá al núcleo para que las realice. Sin ser un shell estrictamente un lenguaje de programación, al proceso de crear scripts de shell se le denomina **programación shell o en inglés, shell programming o shell scripting.**

Los usuarios de Unix y similares, pueden elegir entre distintos shells (programa que se debería ejecutar cuando inician la sesión, véase bash, ash, csh, Zsh, ksh, tcsh). Las interfaces de usuario gráficas para Unix, como son GNOME, KDE y Xfce pueden ser llamadas shells visuales o shells gráficas. Por sí mismo, el término shell es asociado usualmente con la línea de comandos. En Unix, cualquier programa puede ser un shell de usuario. Los usuarios que desean utilizar una sintaxis diferente para redactar comandos, pueden especificar un intérprete diferente como su shell de usuario.

El sistema de ficheros ***ext4*** es la última versión de la familia de sistemas de ficheros ***ext y son los más utilizados por las distribuciones Linux***. Sus principales ventajas radican en su eficiencia (menor uso de CPU, mejoras en la velocidad de lectura y escritura) y en la ampliación de los límites de tamaño de los ficheros, ahora de hasta 16TB.

Cuando **instalamos un sistema operativo Linux**, se establecen los siguientes directorios:

Vamos a ver ahora como nos podemos mover por la Shell de Linux usando diferentes comandos:

Cuando arrancas la consola en tu ordenador con tu distribución de Linux, siempre aparece del siguiente modo:

[usuario@linuxbox ~]$

- usuario: hace mención al usuario logado y linuxbox especifica el nombre de la máquina.

- Cuando indica (~) quiere decir que el usuario ahora mismo se encuentra en su directorio /home/usuario.

- Cabe tener en cuenta que el signo dólar ($) cambia a # cuando nos logamos con permisos de administrador. Para ello, se puede usar **su -**, o **sudo -s** o sudo **su –**.

1. Listar, visualizar y desplazarte por las diferentes carpetas o directorios

Para obtener un listado completo de los directorios que cuelgan de raíz puedes usar el comando **cd** para situarte en el directorio raíz y con **ls** podrás visualizar todos los archivos y carpetas contenidos en él.

$ cd /

$ ls

También puedes jugar un poco con el comando **ls**, añadiendo ciertos parámetros para obtener listados más detallados. Una opción muy útil, por ejemplo, es **ls -l**, con la que obtendrás los diferentes directorios en forma de lista, junto con **los permisos de lectura, escritura y ejecución asociados a cada una de ellos.**

El comando **pwd** te indica la ruta completa del directorio de trabajo en el que se encuentra tu usuario. Su función es meramente informativa, peor muy útil en ciertas ocasiones, como, por ejemplo, conocer el nombre del directorio de trabajo actual.

$ pwd

El comando **cd** te permite cambiar de directorio de trabajo. Sería el equivalente a ingresar o entrar en la carpeta, pero desde la consola. Básicamente requiere indicar el nombre del directorio en el que deseas moverte. Acepta **rutas absolutas y relativas.**

$ cd /home/usuario/Documentos

El comando de arriba te llevará al directorio Documentos dentro de la carpeta personal del usuario llamado usuario. En este caso he utilizado una ruta absoluta, empezando por el directorio **raíz /**, e indicando el camino completo hasta situarme a Documentos.

La instrucción **cd** la puedes utilizar siempre que quieras volver a situarte al directorio principal de usuario, que en este caso sería en /home/usuario. Muy interesante siempre que queramos volver al punto de partida (ojo, no confundir eso con ir al directorio raíz, que sería el directorio /)

$ cd

Situados ahora en /home/usuario, si queremos ir al directorio Documentos, usando la ruta relativa sería:

$ cd Documentos

Cuando se dice relativa significa que se indica la ruta relativa a la posición en la que me encuentro en ese momento. (/home/usuario)

Siguiendo con esta nueva instrucción:

$ cd ..

Con esta instrucción subes un directorio. Si te encontrabas en /home/usuario/Documentos, ahora te encuentras en /home/usuario.

Y con esta instrucción:

$ cd ../..

Saltas dos directorios hacia arriba, situándote en el directorio raíz.

Hasta aquí, tienes algunos usos simples para moverte a través de las diferentes carpetas. A continuación, y teniendo claro lo anterior, podemos pasar a aprender a listar archivos y directorios.

podrás listar los diferentes archivos y directorios de la carpeta de trabajo en la que te encuentres. El comando acepta multitud de opciones, algunas de las cuales te mostraré a continuación.

2. Listar directorios

El comando **ls** es el uso más simple del comando ls. Si no le indicas ninguna opción, te enumerará todos los archivos y directorios que se encuentran en la carpeta de trabajo actual, sin tener en cuenta archivos ocultos.

$ ls -a

Con esta opción, el comando te mostrará, en forma de lista, todo el contenido que se encuentre dentro del directorio de trabajo, incluyendo, además, archivos y carpetas ocultos.

$ ls -l

Esta opción es similar al primer caso, pero muestra el contenido en forma de lista e incluye información referente a cada elemento. Se usa muchísimo y es especialmente útil a la hora de conocer el propietario y los permisos de cada fichero.

Estas son sólo algunas de las muchísimas posibilidades de las que disponemos para nombrar o listar el contenido de un directorio, desde la terminal de Linux. Existen muchas opciones más, las cuales puedes explorar en todo momento haciendo uso del comando **man ls**.

El comando **find** es muy similar en su función básica a **ls**, ya que de entrada sirve para listar todo el contenido de un directorio. La diferencia es que, aplicando filtros, te puede servir para buscar archivos de forma más precisa.

$ find

La sentencia más básica te listará todo el contenido del directorio de trabajo actual de forma recursiva. La diferencia respecto a **ls** es justamente que **find** no se limita a mostrar los archivos y directorios de primer nivel, sino que también te mostrará el contenido de estos, y así recursivamente hasta recorrer todos los niveles hacía abajo.

$ find ./Documentos

Con esta opción, **find** te listará todo el contenido del directorio Documentos (dentro del directorio de trabajo actual) también de forma recursiva, recorriendo todos los niveles hacía abajo.

$ find ./Documentos -name archivo.txt

Si quieres empezar a establecer filtros por nombre, puedes añadir el parámetro **-name**. En este ejemplo, estamos intentando localizar un archivo concreto dentro de Documentos que su nombre corresponda a archivo.txt.

$ find ./Documentos -name *.pdf

Incluso puedes hacer filtros más concretos gracias al uso de comodines. En el caso de arriba, por ejemplo, estamos buscando en la carpeta Documentos todos los archivos que con la extensión **.pdf**, al igual que puedes hacerlo con cualquier otro tipo de extensión.

El comando **locate** es una alternativa útil a **find** la hora de localizar archivos o directorios que no recuerdas donde tienes. Aquí tiene algunos ejemplos que te pueden ser de gran utilidad:

$ locate archivo1.txt

En este caso tienes un claro ejemplo de cómo realizar una búsqueda simple del archivo archivo1.txt directamente por su nombre. Es útil solo si sabes el nombre exacto del elemento que estás buscando.

3. Crear, borrar, copiar y mover archivos y directorios

En esta parte conocerás algunos comandos necesarios a la hora de realizar acciones tales como: crear un nuevo directorio, copiar un archivo y pegarlo en otra ubicación, mover ficheros de una ubicación en otra, etc.

El comando **mkdir** te permitirá crear un directorio con el nombre y la ruta que especifiques. Si no le indicas ninguna ruta, por defecto, te creará la carpeta dentro del directorio de trabajo en el que te encuentres. A continuación, tienes algunos ejemplos sencillos.

$ mkdir /home/usuario1/directorio1

En el caso de arria, **mkdir** te creará el directorio de nombre directorio1, en la ruta que le hayas especificado, en este caso dentro de la carpeta principal de usuario.

$ mkdir directorio2

Con esta sintaxis, el comando te creará una carpeta de nombre directorio2 dentro del directorio de trabajo en la que te encuentres (recuerda utilizar pwd para saber dónde estás).

Estos son las dos principales maneras de crear carpetas en Linux desde la consola. Asimismo, si quieres profundizar más en el uso de este comando, puedas explorar otras muchas opciones a través del comando **man mkdir**.

El comando **rmdir** te permite eliminar el directorio que le especifiques. Para poder utilizar este comando, el directorio a borrar debe estar vacío. A continuación, tienes un par de ejemplos.

$ rmdir /home/usuario1/directorio1

En este caso, **rmdir** borrará el directorio de nombre directorio1, que se encuentra en la ruta especificada, en este caso dentro de la carpeta de usuario.

$ rmdir directorio2

En este otro ejemplo, el **rmdir** eliminará el directorio de nombre directorio2, el cual debe encontrarse dentro de la carpeta en el que te encuentres. De lo contrario, indicará que el directorio no existe. Se está utilizando una ruta relativa.

El comando **rm** te permite eliminar archivos sueltos y directorios que no se encuentren vacíos. A continuación, tienes algunos de los usos principales del comando.

$ rm /home/usuario1/archivo1.txt

En este caso, **rm** te borrará el archivo de texto archivo1.txt, que se encuentra en la ruta especificada, para este caso dentro de la carpeta de usuario. Aquí se está utilizando una ruta absoluta.

$ rm -r /home/usuario1/directorio1

Con esta opción, **rm** borrará el directorio directorio1 de forma recursiva. Esto significa, incluyendo todos los archivos y subdirectorios que se encuentren dentro de él (pidiéndote, eso si, confirmación para cada archivo).

rm -rf /home/usuario1/directorio1

Si te quieres saltar el paso de tener que confirmar archivo por archivo que realmente deseas borrarlo, con este comando borrarás todo el contenido del directorio sin advertencias.

Eso si, hay que tener cuidado con **rm**, puesto que dependiendo de cómo lo uses, puede dar cabida a situaciones como esta. Se trata de ser consciente de cómo funciona y de los parámetros que estas introduciendo en cada momento.

Usando el comando **cp**, serás capaz de copiar archivos y directorios, así como ubicarlos en otras rutas. A continuación, tienes un par de ejemplos de cómo se puede utilizar.

$ cp archivo1.txt archivo2.txt

Este es posiblemente el uso más simple de **cp**. Con esta forma, crearás una copia del archivo archivo1.txt la cual se guardará con el nombre archivo2.txt. En este caso, el archivo de partida debe encontrarse dentro del directorio de trabajo en el que estés.

$ cp /home/usuario1/archivo1.txt /tmp/archivo2.txt

Como en todos los casos, puedes explorar muchas más opciones tecleando **man cp** en la consola.

El comando **mv** te servirá para mover archivos desde la consola. Sería lo equivalente a arrastrar un archivo desde una ubicación a otra. La sintaxis es muy sencilla, solamente debes especificar la ubicación de inicio, incluyendo el nombre del archivo, y la ubicación de destino. También puedes modificar el nombre del archivo en su ubicación de destino.

mv /home/usuario1/Descargas/archivo1.txt /home/usuario1/Documentos/archivo1.txt

En este ejemplo de arriba estamos moviendo el archivo de nombre archivo1.txt desde la carpeta Descargas hacía la carpeta Documentos. Para ello hemos utilizado rutas absolutas.

mv Descargas/archivo1.txt Documentos/archivo1.txt

En este otro ejemplo he hecho exactamente lo mismo, pero utilizando una ruta relativa, suponiendo que nos encontramos en la carpeta de usuario dentro de la Home.