COMANDOS BÁSICOS DE LINUX

Comandos para gestionar ficheros y directorios

1.- Introducción

Todos los sistemas operativos estructuran sus objetos en una estructura jerárquica en forma de árbol que contiene información sobre los diferentes objetos (directorio, archivos, links).

Esta estructura permite contener distintos objetos en distintas localizaciones con el mismo nombre.

Un directorio es un objeto destinado a contener otros objetos.

Un archivo es un objeto destinado a contener información.

El origen del sistema de archivos de LINUX se encuentra en el directorio root o “/”

2.- Directorios de Linux

/bin /sbin

Estos directorios contienen programas ejecutables que forman parte del sistema operativo. El directorio /sbin contiene las utilidades del sistema de archivos, particiones e inicio del sistema. El directorio /bin contiene el resto de comandos de la consola y utilidades varias.

/boot

Este directorio contiene la información necesaria para poder arrancar el sistema operativo. Entre otros archivos, aquí se encuentran los núcleos del sistema que se pueden iniciar y la configuración de gestor de arranque.

/etc

Este directorio contiene todos los archivos de configuración de nuestro sistema GNU/LINUX. Este directorio posee distintos subdirectorios que se utilizan para la configuración de los distintos elementos o servicios del sistema operativo.

2.- Directorios de Linux

/dev

Este directorio contiene archivos de dispositivos que permiten la comunicación con los distintos elementos hardware que tengamos instalados en el sistema; discos duros (como /dev/hda o /dev/sda), particiones de discos duros (como /dev/hda3 o /dev/sda6), unidades de CD-ROM (como /dev/scd0), disqueteras (como /dev/fd0), impresoras (como /dev/lp0), puertos serie (como /dev/ttyS0 o /dev/cua0), puerto PS2 (como /dev/psaux), tarjetas de sonido (como /dev/audio), etc.

/lib

Contiene las librerías que son necesarias durante el inicio del sistema operativo. La ventaja de usar librerías reside en que no es necesario integrar su código en los programas que las usan, reduciendo así el tamaño de los ejecutables. Cuando un programa necesita alguna de sus funciones, se carga la librería en memoria y puede ser usada por cualquier otro programa que la necesite, sin necesidad de volver a cargarla en memoria.

2.- Directorios de Linux

/mnt

Este directorio es típico de las distribuciones RedHat, y puede no estar presente en otras distribuciones. Su misión consiste en agrupar en un mismo lugar los puntos de montaje de diversos dispositivos. Este directorio contiene un subdirectorio adicional para cada una de las particiones o dispositivos disponibles en el sistema. Cuando accedemos a estos subdirectorios estamos accediendo a los dispositivos.

/home

Este directorio contiene los directorios personales "home" de todos los usuarios del sistema (menos el root). Los usuarios convencionales únicamente pueden escribir en su directorio "home".

/root

Este es el directorio personal del usuario root o súper usuario. Contiene básicamente la misma información que los directorios personales de los usuarios del sistema, pero orientada única y exclusivamente al usuario root.

2.- Directorios de Linux

/var
Su nombre procede de variable, y esa es la naturaleza de la información que contienen sus subdirectorios y archivos, como colas de impresión (/var/spool/lpd), correo electrónico, o archivos de registro creados por los distintos procesos del sistema.
/usr
Su nombre proviene de user y contiene una réplica de otros directorios del sistema operativo orientados a usuarios en lugar de al propio sistema operativo.
/usr/X11R6
Contiene todos los elementos que componen el entorno gráfico X Windows; binarios (/usr/X11R6/bin), librerías (/usr/X11R6/lib), manuales, etc.
/usr/bin
En este directorio se guardan los binarios o ejecutables de todas las aplicaciones orientadas al usuario.

2.- Directorios de Linux

/usr/src
Este directorio contiene el código fuente del núcleo del sistema GNU/LINUX y aplicaciones instaladas.
/tmp
Este directorio contiene diversos archivos temporales que son usados por los programas del sistema operativo.
/proc
Contiene los archivos de proceso. No son verdaderos archivos sino una forma de acceder a las propiedades de los distintos procesos que se están ejecutando en nuestro sistema. Para cada proceso en marcha existe un subdirectorio /proc/<número de proceso> con información relativa a ese proceso.

3.- I-Nodos

Estructura de datos propia de los sistemas UNIX/Linux.

Contiene información de los objetos del sistema de archivos (archivo regular, directorio, enlaces simbólicos):

-Permisos de usuario

-Fechas última modificación

-Ubicación en el disco (NO el nombre)

Cada inodo queda identificado por un número entero, único dentro del sistema de ficheros, y los directorios recogen una lista de parejas formadas por un número de inodo y nombre identificativo que permite acceder al archivo en cuestión: cada archivo tiene un único inodo, pero puede tener más de un nombre en distintos o incluso en el mismo directorio para facilitar su localización.

4.- Links en Linux

Un link o enlace es un archivo especial que crea un atajo al archivo original situado en cualquier parte del sistema de archivos.

Existen dos tipos de enlaces:

- **Links Simbólicos o Blandos**. Es un pequeño archivo que contiene un puntero al archivo apuntado. Cuando se abre un enlace simbólico, Linux lee el puntero y abre el archivo apuntado.

-Los enlaces simbólicos pueden apuntar a sistemas de archivos diferentes, dispositivos diferentes o incluso a otros ordenadores conectados en red.

-La orden ls –l muestra una “l”en los links e informa a que archivo apunta el link.

-Cuando un archivo que posee un enlace simbólico se borra del sistema, el link no apunta a nada (link “stale”)

4.- Links en Linux

- **Links Hardware o duros**. En un enlace hardware, un mismo objeto (con un único i-nodo) posee dos o más referencias.

Las referencias tienen diferentes nombres pero apuntan al mismo i-nodo, es decir al mismo objeto.

-Los enlaces hardware tienen dos importantes limitaciones:

-1º como comparten i-nodo, el archivo y el enlace tienen que estar en el mismo sistema de archivo.

-2º no pueden apuntar a directorios.

-Son más fáciles de manejar y más versátiles que los enlaces hardware. Por esta razón la mayoría de enlaces en sistemas UNIX son links simbólicos.

5.- Comandos para gestionar ficheros y directorios

- **pwd** (print working directory)

Sintaxis: pwd

La orden pwd muestra la ruta de acceso del directorio actual.

- **cd** (change directory)

Sintaxis: cd directory

La orden cd cambia al directorio especificado en directory.

- **mkdir**(makedirectory)

Sintaxis: mkdir [options] directory

Crea un directorio. El usuario tiene que poseer permisos de escritura en el directorio donde se creara el directorio. Opciones:

-p Crea los directorios intermedios si estos no existen.

5.- Comandos para gestionar ficheros y directorios

- **ls** (list)

Sintaxis:

ls [options] directory

Opciones:

-a : Lista todos los archivos, incluidos los ocultos.

-l : Lista los archivos en formato largo y muestra información detallada sobre ellos.

-R : Lista de forma recursiva los contenidos de los subdirectorios.

-i Muestra el número de i-nodo de cada fichero.

-s : Muestra el tamaño en KiloBytes junto a cada archivo.

-u : Clasifica por fecha y hora del último acceso.

-t : Clasifica por fecha y hora de la última modificación.

5.- Comandos para gestionar ficheros y directorios

- **cp(copy)**

Sintaxis:

- **cp [options] file1 file2** .Copia file1a file2. Si file2 existe y el usuario tiene los permisos apropiados el archivo seráremplazado.
- **cp[options] files directorio** .Copia uno o más archivos en directorio. Si no existe se mostraráun mensaje de error.

Opciones:

-f (force): Fuerza a sobrescribir los archivos existentes en el destino.

-i (interactive) :Pregunta antes de sobrescribir cualquier archivo.

-p : Mantiene toda la información del archivo; propietario, grupo propietario, permisos, hora y fecha. Sin esta opción, el archivo o archivos copiados tendrán la fecha y hora actual, los permisos, propietario y grupo propietario por defecto.

-R (recursive): Si en file1 se especifica un directorio, la opción –r o –R copia toda la jerarquía del directorio en el destino especificado.

-v (verbose): Muestra el nombre de cada archivo mientras se copia.

5.- Comandos para gestionar ficheros y directorios

- **mv (move)**

Sintaxis:

mv [options] source target

Mueve o renombra archivos y directorios. Si target no existe, source es renombrado. Si target existe, será sobrescrito. Si target es un directorio, source será movido dentro de ese directorio.

Opciones:

-f Fuerza a no preguntar si el targete xiste, eliminado los mensajes de advertencia.

-i Fuerza a preguntar antes de mover cualquier archivo.

5.- Comandos para gestionar ficheros y directorios

- **rm(remove**)

Sintaxis:

rm [options] files

Elimina uno o más archivos del sistema. Para eliminar un archivo es imprescindible que el usuario tenga permiso de escritura en el directorio que contiene el archivo, pero no necesita permiso de escritura en el archivo. El comando rm también puede borrar directorios cuando se usan las opciones -r o –R.

Opciones:

-f Fuerza a no preguntar al borrar archivos sin permiso de escritura.

-i Fuerza a preguntar al borrar cada archivo.

-r Si file es un directorio, elimina recursivamente el contenido completo del directorio, incluidos los subdirectorios.

5.- Comandos para gestionar ficheros y directorios

- **rmdir**(removedirectory)

Sintaxis:

rmdir [options] directory

Borra directorios vacíos.

Opciones:

-p Borra los directorios intermedios si estos están vacíos como resultado de la orden.

5.- Comandos para gestionar ficheros y directorios

- **touch**

Sintaxis:

touch [options] files

Cambia la fecha del último acceso o/y modificación de files. Si no se especifica ninguna opción se actualizaran ambas fechas (acceso y modificación).

Opciones:

-a Actualiza únicamente la fecha del último acceso del archivo.

-m Actualiza únicamente la fecha de modificación del archivo.

-t No utiliza la fecha actual, sino el especificado a continuación mediante el formato de [[CC]YY]MMDDhhmm[.ss].

Ejemplos:

Modifica la fecha del último acceso al 12 de enero de 2001 a las 18 horas, 45 minutos.

$ touch –ta 200101121845 file

5.- Comandos para gestionar ficheros y directorios

- **ln(link)**

Sintaxis:

ln [options] file link

ln [options] files directory

Crea enlaces entre archivos. En la primera forma se crea un enlace llamado link que apunta al archivo file. En la segunda forma, se crea un enlace dentro del directorio directory para cada uno de los archivos especificados en files.

Opciones:

-f Fuerza a sobrescribir los enlaces si existen previamente.

-i Pregunta antes de crear cada enlace.

-s Crea un enlace simbólico. Por defecto crea enlaces hardware.

5.- Comandos para gestionar ficheros y directorios

- **chown**

Sintaxis:

chown [options] dueño[:grupo] <ficheros>

Este comando se utiliza para cambiar de dueño y el grupo de un fichero. El dueño del fichero solo lo puede cambiar el usuario root mientras que el grupo ademas de root, lo puede cambiar el propio dueño siempre que pertenezca al nuevo grupo.

Opcion:

- R: en los directorios cambia el dueño y/o el grupo recursivamente.

5.- Comandos para gestionar ficheros y directorios

- **chgrp**

Sintaxis:

chgrp [options] grupo <ficheros>

Se utiliza para cambiar el grupo de un fichero.

Opción:

-R : cambia en los directorios el grupo recursivamente.

6.- Comandos para paginar, visualizar y editar ficheros

- **cat**

Sintaxis: cat [options] <ficheros>

Este comando nos permite concatenar ficheros, es decir, coger varios ficheros de texto y unirlos en uno solo. Por defecto, cat manda los ficheros a la salida estandar del sistema que es la pantalla, por lo que normalmente utilizamos esta orden para visualizar ficheros de texto.

Opciones:

-n : numera todas las lineas.

-b : numera solo las lineas que no estan en blanco.

-E : marca el final de linea con un $

-s : imprime solo la primera de varias lineas en blanco. Sirve para ahorrar espacio en pantalla.

-v : Muestra los caracteres “raros” en el archivo.

6.- Comandos para paginar, visualizar y editar ficheros

- **less**

Sintaxis: *less [options] fichero*

Se usa, este comando, para mostrar texto en la pantalla. Solo muestra el texto del archivo dado, no puedes editar o manipular texto. Para mostrar el archivo desde la linea especificada, introduce el numero de linea seguido de dos puntos (:) . Permite movimiento hacia adelante y hacia atrás en el archivo.

Opciones:

-c limpia la pantalla antes de mostrar.

+n iniciar el archivo desde el numero dado.

:p examina el archivo previo en la lista de linea de comandos.

:d elimina el archivo actual de la lista de archivos.

q sale del programa

/<cadena> realiza búsquedas de la cadena <cadena>

/ repite la ultima busqueda

:<n>b back (retrocede) n paginas

:<n>f forward (avanza) n paginas

6.- Comandos para paginar, visualizar y editar ficheros

- **more**

Sintaxis: *more [options] fichero*

Se usa para mostrar texto en la pantalla del terminal. Solo permite movimiento hacia detrás.

Opciones:

-c limpia la pantalla antes de mostrar.

-e salir inmediatamente despues de escribir la ultima linea del ultimo archivo en la lista de argumentos.

-n especifica cuantas lineas se muestran por pantalla para un archivo dado.

+n inicia el archivo desde el numero dado.

6.- Comandos para paginar, visualizar y editar ficheros

- **Editor vi**

Es el editor estandar de Unix y esta orientado a comandos. Existe una version conocida como vim que permite la edicion de multiples ficheros, edicion automatica para varios lenguajes de programacion, ayuda en linea, selección visual, etc..

Basicamente, vi posee dos modos de interaccion : el de insercion (edicion) y el de comandos. Para pasar al modo comando se pulsa Esc y para pulsar al de insercion se pulsa i.

Hay que tener mucho cuidado al usar vi con los cursores. Estos solo funcionan en el modo de comandos, si estamos en modo insercion debemos acostumbrarnos a no usar los cursores ya que se interpretaran como caracteres y empezaran a aparecernos caracteres extraños en la pantalla.

6.- Comandos para paginar, visualizar y editar ficheros

Algunos comandos utiles de vi (pulsando Esc para pasar al modo comando, que veremos en la ultima linea de la pantalla) son:

comando	accion
dd	Borrar la linea actual
D	Borra desde la posicion actual hasta final de la linea
dG	Borra hasta el final del fichero
u	Deshace el ultimo comando
:q	Sale del editor (si se han hecho modificaciones y no se ha salvado genera un error)
:q!	Sale sin salvar
:w	Salva
:wq	Salva y sale
:x	Salva y sale
<n> <comando>	Ejecuta el comando n veces

6.- Comandos para paginar, visualizar y editar ficheros

- **Comando echo**

Es un comando que nos permite escribir algo directamente en pantalla.

Sintaxis : *echo [options] [cadena]*

Opciones:

-n suprime el carácter de nueva linea final

-e activa la interpretacion de los caracteres especiales precedidos por \

-E desactiva la interpretacion de esas secuencias

7.- Comandos para hacer búsquedas ficheros y patrones

- **Comando grep**

Permite encontrar líneas de texto que contengan un cadena o patrón dentro de uno o varios archivos y las imprime.

Sintaxis: *grep [opciones] patron [archivo]*

Opciones:

-c En lugar de imprimir las líneas que coinciden, muestra el número de líneas que coinciden.
-r busca recursivamente dentro de todos los subdirectorios del directorio actual.
-v nos muestra las líneas que no coinciden con el patrón buscado.
-i ignora la distinción entre mayúsculas y minúsculas.
-n Numera las líneas en la salida.
-o muestra sólo la parte de la línea que coincide con el patrón.
-f ARCHIVO extrae los patrones del archivo que especifiquemos. Los patrones del archivo deben ir uno por línea.
-H nos imprime el nombre del archivo con cada coincidencia.

7.- Comandos para hacer búsquedas ficheros y patrones

-A Muestra el número de líneas de texto que hay después de la línea coincidente.

-a No suprime las lineas de salida con datos binarios, las trata como texto.

-b Mostrar el offset de bytes del archivo de entrada antes de cada linea en la salida.

-h Muestra las líneas coincidentes pero no los nombres de archivo.

-q Mostrar en modo silencioso, no muestra nada.

-V Muestra la versión.

-w Coincidencia en palabras completas únicamente.

* Patrones para la operación de búsqueda.

. Coincidencia de caracteres únicos.

***** Caracter comodín.

^ Empieza por.

$ Termina en.

7.- Comandos para hacer búsquedas ficheros y patrones

Para una mejor visualización de las coincidencias encontradas añada, si no lo tiene ya, la siguiente línea a su bashrc o bash_aliases:

```
alias grep='grep --color=auto'
```

Si no sabe cómo, copie y pegue lo siguiente en su terminal:

```
echo 'alias grep="grep --color=auto"' >> ~/.bashrc; source ~/.bashrc
```

7.- Comandos para hacer búsquedas ficheros y patrones

¿Qué es una expresión regular?

Se puede decir que una expresión regular es un buscador de patrones, esto es, para cualquier patrón que defina (recuerde el ejemplo del email) la expresión regular buscará todas las coincidencias.
Como puntualización habría que añadir que las expresiones regulares buscarán coincidencias por líneas, esto es que la coincidencia de un patrón en el texto tiene que encontrarse en su totalidad en la misma línea.

Ejemplo: grep 'hola que tal' *archivo*

Buscará todos los 'hola que tal' (sin comillas) que aparezcan en el texto en la misma línea, esto es:

Encontrará: hola que tal estás hoy

Pero no encontrará:

hola
que tal estás hoy

7.- Comandos para hacer búsquedas ficheros y patrones

Entrecomillando los patrones

Como norma general entrecomillaremos los patrones que le pasemos a grep usando comillas simples, esto es así debido a que bash realiza sustituciones en determinados caracteres que tienen significado especial para él, son los llamados "wildcards".

Wildcards:
Estos símbolos en concreto tienen significado especial, y bash los sustituye antes de pasárselo al comando:
* -> cero o más caracteres (cualquier carácter).
? -> exactamente un carácter (cualquier carácter).
[abcde]-> exactamente uno de los caracteres entre los corchetes.
[a-o] -> exactamente uno de los caracteres en el rango. -> a, e, i, o en este caso
[!abcde] -> cualquier carácter que no esté entre los corchetes.
[!a-e] -> cualquier carácter que no esté en el rango.
{debian,linux} -> alguna de las palabras que están entre las llaves.

7.- Comandos para hacer búsquedas ficheros y patrones

Comillas simples vs comillas dobles:
Las comillas simples, o comillas fuertes, hacen que cualquier símbolo que esté entrecomillado no tenga significado especial, es decir, todo se envía al comando tal cual se ha puesto.
Las comillas dobles, o comillas débiles, no interpretan las "wildcards", pero sí la expansión de variable, las de comandos y los escapados, es decir, interpreta los símbolos: $ ` \
Ejemplo:
rm * #Borra todo, ya que '*' coincide con todas las cadenas.
rm *.jpeg #Borra todos los ficheros que acaben en '.jpeg'.
rm file?.txt #Borra todos los ficheros que empiecen por 'file', tengan un carácter y acaben en '.txt'
rm "mi fichero“ #Borra un fichero llamado "mi fichero" (sin las comillas), si existen.
rm mi fichero #Borra un fichero llamado "mi" y un fichero llamado "fichero", si existen.

No hay que confundir los patrones de sustitución de bash con expresiones regulares.

7.- Comandos para hacer búsquedas ficheros y patrones

Expresiones regulares

1. 'carácter'

La expresión regular más sencilla que se puede crear es la compuesta por un carácter o por un conjunto de caracteres (palabra).
Ejemplo:
grep 'hola' archivo # Busca 'hola' en archivo
grep 'a' archivo # Busca 'a' en archivo
grep 'bienvenido al tutorial' archivo # Busca 'bienvenido al tutorial' en archivo

2. Barra invertida

En las expresiones regulares, las barras invertidas "\" normalmente quitan el significado especial de algún carácter, aunque algunas veces se lo den, esto puede ser un poco complicado al principio pero se acabará acostumbrando. Como norma general, cualquier carácter con un significado especial que vea en este tutorial estará en la forma en la que tenga el significado especial, para quitarle ese significado no tiene más que quitarle la barra si la tiene o ponérsela si no la tiene. Normalmente a los caracteres precedidos por barra invertida y que sin ella tendrían significado especial, se llaman escapados.

7.- Comandos para hacer búsquedas ficheros y patrones

3. Punto

El carácter punto "." se sustituye en una expresión regular por cualquier carácter, excepto nueva línea.
Ejemplos:

echo 'a b' | grep -o '.‘ encontrará como coincidencias: "a" y "b"

echo 'a b . ;' | grep -o '.‘ encontrará como coincidencias: "a", "b", "." y ";".

echo 'a b . ;' | grep -o '\.‘ sólo encontrará como coincidencia: ".", ya que al escapar el patrón ha hecho que este pierda su significado de cualquier carácter, a tener el significado de: el carácter punto.

7.- Comandos para hacer búsquedas ficheros y patrones

quien=tu

echo 'en un'"$quien"' mi' | grep -o '..‘ encontrará como coincidencias: "en", "un", "tu" y "mi"

echo ';:a una ala uno' | grep -o '..a‘ encontrará como coincidencias: ";:a", "una", "ala", pero no "uno" ya que no coincide con el patrón cualquier palabra de tres caracteres siendo el último la "a"

echo 'vuela miedo fuera' | grep -o '..e.a‘ encontrará como coincidencias: "vuela", "fuera", pero no "miedo" ya que aunque cumple la parte del patrón "..e." no cumple lo de acabar en a.

7.- Comandos para hacer búsquedas ficheros y patrones

4. \<\>

El metacarácter \<\> hace que todos los patrones dentro de él sólo coincidan con palabras enteras, esto es, palabras separadas entre sí por espacios. Para este metacarácter los símbolos de puntuación (.,< etc...) no forman parte de la palabra.

- echo 'Este tutorial de grep me gusta mucho' | grep -o '\<..\>‘ encontrará como coincidencias: "de" y "me", ya que son las únicas palabras enteras que coinciden con el patrón.
- echo 'Ya hemos aprendido algunas cosas' | grep -o '\<....s\>‘ encontrará como coincidencias: "hemos" y "cosas"
- echo 'Ya hemos aprendido algunas cosas' | grep '\<.a\>‘ encontrará como coincidencia: "Ya", sin embargo si no se hubiese puesto \<\> entonces el patrón habría coincidido también con: " a", "na", "sa".

7.- Comandos para hacer búsquedas ficheros y patrones

5. * (Repitiendo patrones)

Todos los metacarácteres anteriores están muy bien si sabe exáctamente cuántos caracteres tiene la palabra que quiere buscar, pero si no se sabe , habrá que usar el metacarácter *, * repetirá el patrón que le precede cero o más veces.

echo '4 454 4554554 455545554 4555455545554 44444' | grep '4*'
grep encontrará como coincidencias todos y cada uno de los números que contengan cuatros.

echo '4 454 4554554 455545554 4555455545554 44444' | grep '.*'
grep encontrará como coincidencias todos, ya que el punto se traduce como cualquier carácter y * (repetido cero o más veces).

echo '.* 44 58.* hhial' | grep '\.*'
grep encontrará como coincidencias: ".*" debido a que los caracteres están escapados.

echo '4 454 4554554 455545554 4555455545554 44444' | grep '\<4*\>'
grep encontrará como coincidencias sólo los números que contengan únicamente cuatros.

7.- Comandos para hacer búsquedas ficheros y patrones

6. Rango de caracteres

Se pueden elegir diferentes caracteres a los cuáles aplicarles algún otro patrón de sustitución.

echo '5895 589435 539853 54849' | grep -o '[0123456789]*'
grep encontraría como coincidencias a todos los conjuntos numéricos.

echo 'hola hhaif aieo uuieo' | grep -o '\<[aeiou]*\>'
grep encontraría como coincidencias a todos los grupos de vocales: "aieo", "uuieo"

echo 'hola hhaif AiEo uu66ieo' | grep -o '\<[A-Za-z]*\>'
grep encontraría como coincidencias todas las palabras que sólo contengan letras alfabéticas (inglesas).

echo 'hola hhAIf AiEo uu66ieo' | grep -o '\<[AIa-z]*\>'
grep encontraría como coincidencias todas las palabras que sólo contengan letras alfabéticas minúsculas (cualquiera) y las letras mayúsculas A e I.

7.- Comandos para hacer búsquedas ficheros y patrones

Existen conjuntos de caracteres POSIX definidos para añadir portabilidad, son los siguientes:

[:alnum:] Alfanuméricos
[:alpha:] Carácteres alfabéticos
[:upper:] Carácteres en mayúscula
[:lower:] Carácteres en minúscula
[:digit:] Dígitos
[:print:] Carácteres imprimibles

Ejemplo:

echo '5895 589435 539853 54849' | grep -o '[[:digit:]]*'
echo 'hola hhaif AiEo uu66ieo' | grep -o '\<[[:alpha:]]*\>'
grep encontraría como coincidencias todas las palabras que sólo contengan letras alfabéticas (inglesas).

7.- Comandos para hacer búsquedas ficheros y patrones

7. Paréntesis \(\)

El metacarácter paréntesis sirve para seleccionar un patrón al que le aplicaremos un metacarácter modificador, por ejemplo *

echo 'holaholaholahola holahola adios' | grep -o '\(hola\)*'
grep encontraría como coincidencias todas las palabras que contengan repeticiones de la palabra hola

echo 'holahelloholahola holahola adios' | grep -o '\<\(hola\)*\>'
grep encontraría como coincidencias todas las palabras que contengan únicamente repeticiones de la palabra hola

7.- Comandos para hacer búsquedas ficheros y patrones

Otra característica interesante es la de poder elegir entre dos patrones distintos usando \| entre los dos patrones:

echo 'ftp://asdfasf.com http://asfasdf.com' | grep -o '\(ftp\|http\)://[a-z]*\.com'
grep encontrará como coincidencias cualquier palabra que empiece con ftp o http y que continúe con un conjunto de letras minúsculas y acabe en .com

echo 'ftp://ulises.hostalia.com/debian http://ftp.es.debian.org/debian/' | grep -o '\(ftp\|http\)://[a-z\.-\/]*'
grep encontrará como coincidencias cualquier palabra que empiece con ftp o http y que continúe con un conjunto de letras minúsculas, puntos, y guiones.

echo 'Yes No Sí No' | grep -o '\(Sí\|Yes\)'
grep encontrará como coincidencias: "Sí" y "Yes".

echo '6hola6 9hola9 5jaja4 5haha5' | grep '\(6\|9\)[[:alpha:]]*\1'
grep encontrará como coincidencias cualquier palabra que empiece con 6 o 9 y que acabe con el mismo número, con letras en medio.

7.- Comandos para hacer búsquedas ficheros y patrones

8. Eligiendo el número de repeticiones \{\}

Con los metacarácteres \{\}, poniendo un número o dos entre las llaves se puede elegir el número de veces que queremos que se repita un patrón.

echo '555 555555 555555 5555555 55 5555' | grep -o '\<5\{6\}\>'
grep encontraría las palabras formadas por exáctamente seis cincos.

echo '555 555555 555555 5555555 55 5555' | grep '\<5\{3,6\}\>'
grep encontrará las palabras formadas por entre tres y seis cincos.

También las podemos mezclar con otros metacarácteres

echo '555 asd#lha sdf 55hi5 555aASasd55 55 5555' | grep -o '.\{4,11\}'
grep encontrará como coincidencias cualquier palabra con entre 4 y 11 caracteres, el resultado puede parecer raro, pero la explicación es sencilla: el metacarácter "." significa cualquier carácter, incluido el espacio, y grep tiende a buscar la palabra más grande que se adapte al patrón.