Los que trabajan con servidores **Linux** o **Unix** saben bien que trabajar por **terminal** o **consola** es un recurso imprescindible para un buen informático, a la vez que supone un ahorro gigante de tiempo, por el tema de poder administrar remotamente un sistema lejano físicamente.



En realidad no serán **10 comandos**, sino 10 **grupos** o **familias** de comandos que -junto a otras herramientas útiles como podrían ser [vim](http://www.emezeta.com/articulos/manual-para-aprender-a-utilizar-vim) o **netcat**- nos permitirán administrar sistemas en **modo extremo** y nos harán la vida en terminal mucho más cómoda.

Antes de seguir, debemos saber utilizar correctamente las **redirecciones** de salida (*símbolos mayor y menor que*) y las **tuberías** (*pipes*).

Las **redirecciones** sirven para guardar en un nuevo fichero la salida del comando ls -lh >fich\_vacio.txt o para añadir al final de un fichero existente ls -lh >>fich\_existente.txt.

Las **tuberías** sirven para enlazar la salida de un comando con la entrada de otro, y así aplicar el comando posterior a la salida del primer comando. Por ejemplo, ls -lh | grep "hola".

**1. Movimiento de directorios**

Comandos básicos como **cd** (*cambiar de directorio*), **mkdir** (*crear directorio*), **rmdir** (*eliminar directorio vacío*) son muy necesarios en entorno de terminales.

Unos comandos que pueden ayudarnos para movernos más rápido son **pwd**, que nos muestra la **ruta completa** del directorio donde estamos, y los comandos **pushd** y **popd**.



Como se puede ver en la imagen, **pushd** y **popd** son dos comandos que guardan en una pila el directorio actual donde estamos trabajando. Así, podemos guardar la ruta actual con **pushd .**, cambiar de ruta y realizar varias operaciones, y volver a la ruta anterior, haciendo un **popd**.

Esto en principio puede parecer engorroso, pero en trabajos con rutas largas, se agradece mucho.

**2. Copias y conexiones remotas**

Todos sabemos utilizar el **cp** (*copiar*) o el **mv** (*mover*) para trasladar archivos. Pero también existe el **scp** (*copia mediante SSH*).

El comando **ssh** sirve para conectar a otro sistema **Linux** o **Unix**, que tenga un servidor ssh activo, y así poder ejecutar ordenes en él (*tener una terminal abierta en ese equipo remoto*). Funciona con la sintaxis ssh usuario@ip, por ejemplo:

ssh admin@122.41.51.63

Por otra parte, el comando **scp** realiza una copia de un fichero desde nuestro sistema a un sistema remoto (*o viceversa*). Veamos un ejemplo:

scp fichero.zip admin@122.41.51.63:/home/admin/files

* **fichero.zip** es el fichero local que queremos copiar.
* **admin** es la cuenta de usuario en el sistema remoto.
* **122.41.51.63** es el sistema remoto.
* **/home/admin/files** es el directorio remoto donde vamos a copiar.

Con respecto al mundo HTTP, también esta presente en el mundo de las terminales. Existen varios navegadores de texto como **lynx**, **links**, **w3m** o **elinks**.

Por su parte, **wget** es un excelente comando para descargar ficheros de internet, y **curl** (*muy util para programadores*) un comando con el que obtener el código fuente de una URL.

**3. Compresión y empaquetado**

En prácticamente todas las distribuciones de linux, existen varios paquetes de **compresión** y **empaquetado**.

Así, en cualquier momento podemos comprimir nuestros datos para pasarlos a un pendrive, copiarlos por la red (*ver* ***scp*** *en apartado anterior*), enviarlos por correo o incluso hacer copias de seguridad, de forma más sencilla y compacta.

Los comandos **compress**, **gzip** y **bzip2** (*con sus respectivos* ***uncompress****,* ***gunzip*** *y* ***bunzip2*** *para descomprimir*) son los más famosos.

No obstante, lo que más se suele utilizar es el comando **tar**, que no es más que un **empaquetador** de archivos:

tar -czvf paquete.tar.gz \*

Donde:

|  |  |
| --- | --- |
| **Flags** | **Descripción** |
| **-c** | Crea un paquete. |
| **-z** | Comprime con gzip. |
| **-v** | Verbose mode, muestra información. |
| **-f** | Indica el nombre del fichero, **paquete.tar.gz** en el ejemplo. |
| **\*** | Lista de ficheros a incluir en el paquete, en este caso todos los ficheros de la ruta actual, incluyendo subdirectorios (*comodin*). |

Otro comando poco conocido y bastante útil para programadores es **gzexe**, que viene a ser el equivalente de unix de programas como [LZExe](http://fabrice.bellard.free.fr/lzexe.html) (*DOS*) o [ASPack](http://www.aspack.com/) (*windows*).

Se trata de un comando para **comprimir ficheros ejecutables**, con la característica de que reducen su tamaño pero continua funcionando, sin necesidad de volverlo a descomprimir.

Algo, sin duda, muy útil en unidades donde no tenemos demasiado espacio en disco.

**4. Filtrado de salida**

Al comando **grep** se le podría dedicar un artículo completo, de hecho hay [libros](http://www.amazon.com/grep-Pocket-Reference-OReilly/dp/0596153600) enteros que hablan sobre el.

Se trata de un comando que **filtra** lineas que coincidan con un patrón dado. Por ejemplo:

cat empleados.txt | grep "Garcia"

Sabiendo que **cat** muestra el contenido de un fichero, con esta linea conseguiremos ver los empleados que se apelliden **Garcia**.

A esto se le une la potencia de poder utilizar [expresiones regulares](http://www.emezeta.com/articulos/introduccion-a-las-expresiones-regulares) con **grep**, además de muchos parámetros que facilitan la búsqueda:

|  |  |
| --- | --- |
| **Flags** | **Descripción** |
| **-i** | Busca tanto **Garcia** como **GARCIA** como **garcia**. |
| **-c** | En lugar de las lineas que contienen **García**, devuelve el número de ellas que se encontraron. |
| **-E** | Activa el uso de expresiones regulares (*también* ***egrep***). |
| **-B 6** | Muestra las anteriores 6 lineas de donde encontró un **García**. |
| **-A 6** | Idem, con las 6 lineas siguientes. |

Existen muchísimas más opciones. Un comando mucho más potente (*y también mucho más complejo*) es **awk**. Pero eso lo dejaremos para otro artículo.

**5. Comandos de manipulación de texto**

De mis preferidos. Combinados con las tuberías (*pipes*) pueden ser armas totalmente útiles.

|  |  |
| --- | --- |
| **Comando** | **Descripción** |
| **cut** | Corta fragmentos de lineas siguiendo determinados patrones (*por ejemplo, un delimitador*). |
| **head -X** | Muestra sólo las X primeras lineas. |
| **tail -X** | Muestra sólo las X últimas lineas. |
| **sort** | Ordena los resultados alfabéticamente (*útil para ordenar numéricamente, usar flag* ***-n***). |
| **uniq** | Elimina las líneas repetidas (*especialmente útil después de un* ***sort***) |
| **rev** | Muestra las líneas en orden inverso (*horizontalmente*). |
| **tac** | Muestra las líneas en orden inverso (*verticalmente*) |
| **strings** | Muestra sólo carácteres textuales (*útil para buscar texto en binarios*). |
| **expr** | Realiza evaluaciones de expresiones (*cálculos, operaciones, métodos...*). |
| **wc** | Cuenta el número de líneas, carácteres o palabras. |

Así podemos utilizar combinaciones de este tipo:

tail -50 log | cut -d" " -f1 | sort | uniq -c | sort -n

Secuencia de comandos que:

* Muestra las últimas 50 lineas del log del servidor web (*access\_log*).
* Corta la primera fila, delimitada por espacios (*IP*).
* La ordena alfabéticamente y le aplica el **uniq -c** (*elimina repetidos y añade el número de veces que encontró la linea*).
* Ordena las lineas numéricamente (*última la IP más repetida*).

Un comando también especialmente útil es **tr** (*traducir*), que cambia carácteres por otros. Por ejemplo:

# echo "Hola amigo mio" | tr 'oi' 'aa'
Hala amaga maa

**6. Seguridad y permisos**

Uno de los clásicos que un buen **administrador de sistemas** no puede olvidar en entornos Unix o Linux, es la gestión de permisos.

Con [chmod](http://es.wikipedia.org/wiki/Chmod) podemos cambiar los permisos de lectura, escritura y ejecución de uno o varios ficheros.



Por otra parte, con **chown** y **chgrp** podremos cambiar el usuario propietario de un fichero y el grupo al que pertenece ese fichero respectivamente.

**7. Comandos de administrador**

Existen varios comandos para mantener monitorizado el sistema, veamos algunos de ellos.

El comando **fdisk** permite ver y hacer cambios en las particiones del disco duro (*un fdisk -l nos muestra todas las existentes, incluso pendrives o unidades usb*).

Con respecto a la memoria del PC o el consumo de recursos, disponemos del comando **free -m**, que muestra la memoria utilizada y libre, y de **uptime**, que nos muestra el tiempo sin reiniciar el equipo, los usuarios conectados a el, y la carga del sistema en el último minuto, últimos 5 minutos y últimos 15 minutos.

# uptime
13:44:58 up 166 days, 10:07, 1 user, load average: 0.13, 0.15, 0.16

Volviendo a las unidades de disco, con **df** (*disk free*) se puede ver cuanto espacio tenemos disponible en nuestras unidades, mientras que con **du** se hace la operación inversa (*cuanto tenemos ocupado en la ruta actual*). Con **fsck** podemos realizar chequeos para comprobar el estado de una partición.

**8. Conversión de codificación**

He dejado un apartado sólo para este gran comando. Se llama **iconv** y se trata de un conversor de codificaciones, tremendamente útil para cuando tenemos problemas de formateado con diferentes codificaciones como **utf8** y las diferentes **isos**.

Veamos un ejemplo:

# cat texto.txt
Â¡Buenos dÃ­as! Â¿PodrÃ­a hacerlo tambiÃ©n? Â¡Muchas gracias!
# cat texto.txt | iconv -f utf8 -t latin1
¡Buenos días! ¿Podría hacerlo también? ¡Muchas gracias!

**9. Procesos y gestión**

Otro de los clásicos de administración, es mantener un sistema limpio de procesos que consuman demasiada memoria o tiempo de CPU.

Con el comando **ps aux**, podremos ver todos los procesos que están ejecutándose en la máquina.

Cada proceso tiene un PID (*número de proceso*). Sabiendo este PID, podemos matar el proceso (*terminar su ejecución*) que nos moleste haciendo un kill -9 1234, o simplemente con el comando **pidof** conocerlo.

También podemos utilizar **killall firefox**, para matar por el nombre del proceso en lugar del PID.

**10. Comandos de red**

Existen múltitud de comandos de red, los más utilizados son **ping**, con el cuál podemos ver si una determinada IP está activa, mandándole paquetes (*siempre y cuando no esté configurada para ignorarlos*) o el **traceroute** el cuál además te indica los nodos por donde pasa el paquete antes de llegar a su destino.

Además, existen **nslookup** o **dig**, que sirven para resolver IPs a su DNS inverso y viceversa y **netstat -a**, el cuál te devuelve las **conexiones activas** en el sistema actual.