GNU/Linux: linea de comandos (bash shell).

Introducción al BASH shell.

Algo que acompaña a todas las distribuciones GNU/LINUX es la terminal de línea de comandos (*Unix shell, bash shell, o simplemente "shell"*) la cual puede ser lanzada en modo de **superusuario** (*root*) o con la identidad con que hayamos iniciado sesión (aunque aún así podemos ejecutar comandos con privilegios de superusuario anteponiendo el comando **sudo** acompañado de su respectiva contraseña) o "escalando" privilegios con el comando **su** (más contraseña).

> xD pic.twitter.com/3m8X1Z4HzS
>
> — Damián (@dam1an) 21 October 2016

Nosotros utilizamos la línea de comandos "bash shell" desde 1990 con al antiquísimo MSDOS y luego con el venerable Netware Novell en 1993 (hoy migrado a Linux) y no tenemos aversión alguna a esta forma de trabajar **pero hemos vistos casos severos de usuarios que se marean y les duele la cabeza al tipear 10 ó 12 comandos por la terminal.**

https://twitter.com/ragrawal804/status/837611344409001984

Desde aquellos años nuestros terminales de BASH shell, los monitores donde aprendimos la computación, tenían un ancho de 80 columnas (80 caracteres o letras), justos lo que cabía de ancho -y todavía es la norma- en una impresora de matriz de puntos para una hoja tamaño carta (hoy A4 según normas ISO). Pero en los años 1980 la cosa no era tan fácil, os traemos a colación este vídeo para que comprobéis que lo que a nosotros nos parecía una norma de maravilla, no fue tan fácil llegar hasta allí:

https://www.youtube.com/watch?v=BJzOErvJwZs

El entorno gráfico es bello, bellísimo pero en muchas oportunidades necesitamos algo que vaya al grano y funcione rápido ***e incluso que funcione de manera remota y de manera segura abriendo incluso comunicaciones diversas***, pero no quiero abrumarlos ni abrumarlas, con que sepan que es una opción disponible es más que suficiente.

Actualizado el domingo 27 de agosto de 2017.

Un interesante reportaje en inglés, de apenas 7 párrafos de introducción, describe muy bien la historia del **bash** y lo podéis leer en este enlace.

https://twitter.com/CommitStrip_fr/status/812001397445185536

Abrir una ventana terminal.

En Ubuntu la combinación de teclas para abrir el "bash shell" es **CTRL+ALT+T** pero cada distribución tiene su método, si es por medio de menú busquen algo como "Accesorios->Terminal" o sino algo parecido en "Herramientas del sistema". Investigando un poco pude conseguir varios atajos de teclado y un "atajo" de ratón {si es que eso existe, si no pues lo cabo de inventar ;-) }.

Si pulsamos la combinación de teclas ALT+F1 nos mostrará el menú principal del sistema (probado en **GNU/LINUX** Ubuntu, Canaima y Debian).

El otro atajo de teclado que me parece más poderoso es la combinación **ALT+F2** la cual nos permite ejecutar casi todo programa instalado en nuestra máquina. Para comprobar esto último probé en varias de mis máquinas virtuales, incluyendo Chromixium cómo abrir el BASH shell y las que nombré en el párrafo anterior.

BASH shell en Ubuntu 14:

BASH shell en Canaima Kukenan:

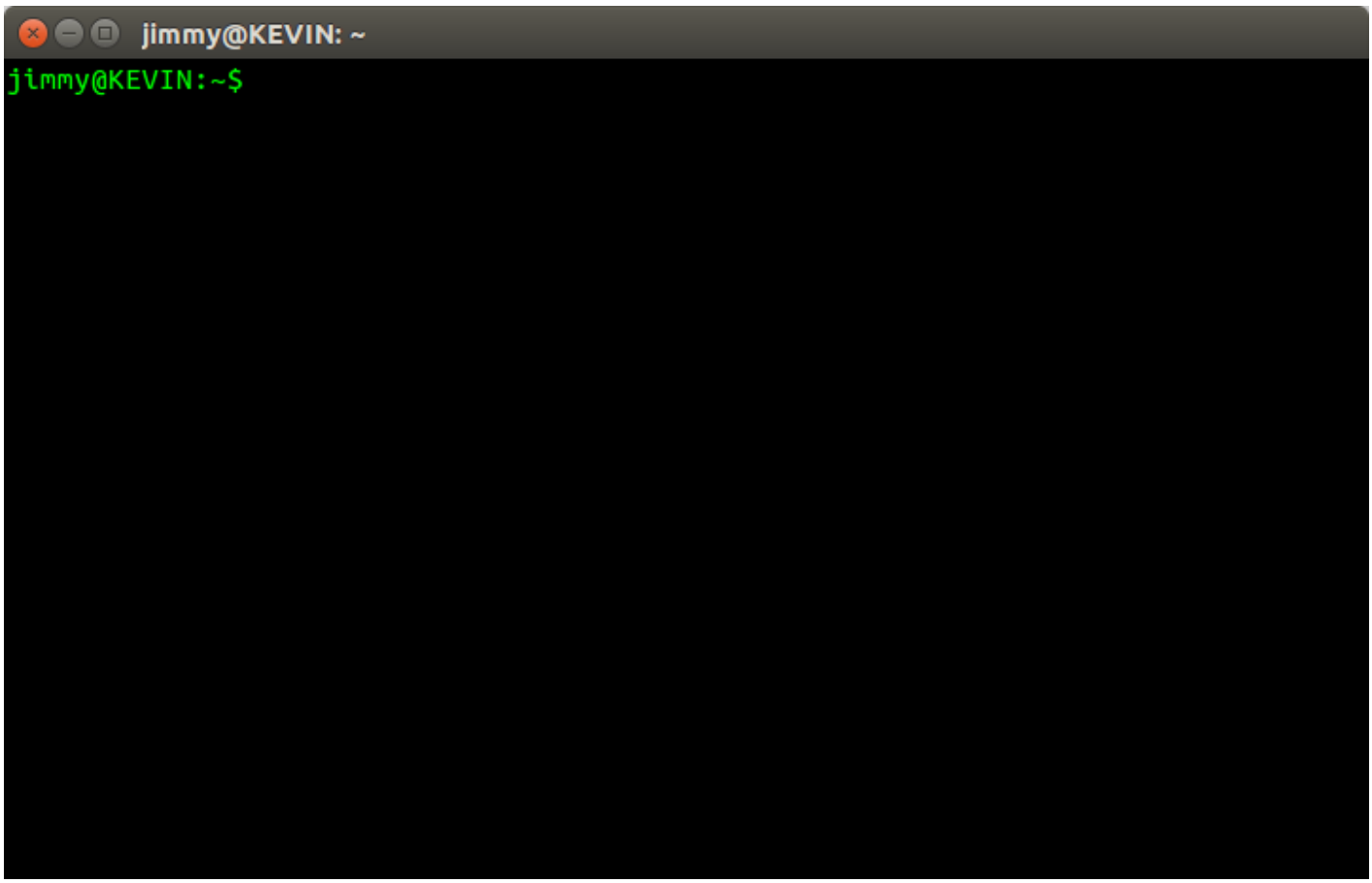
BASH shell en Debian 7:

BASH shell en Chromixium OS 14:

Muy probablemente su cuenta creada en la distribución Linux es *limitada* por razones de seguridad: esto se traduce en que no podrán instalar programas, periféricos como impresoras o incluso a acceder a estadísticas sin ingresar credenciales de **superusuario** (ya ustedes saben que me refiero al usuario **root**, en adelante simplemente lo denominaré en castellano). Es así que iniciaremos la terminal con nuestras credenciales y si necesitaramos hacer algo avanzado pues "escalamos" privilegios.

Una cónsola se verá algo así como la siguiente (yo utilizo verde sobre negro porque los monitores

de la universidad, los antigüos CTR eran de ese color aunque el que yo poseía en mi casa era de un horrible color ámbar):



Hay otra manera de tener acceso a la terminal de linea de comandos: *por medio de una cónsola virtual,* presionando simultáneamente las teclas CONTROL+ALTERNO+F1.

Antes de que probéis la cónsolo virtual os debo advertir que NO funciona para nada el ratón, **solamente por medio del teclado podréis salir de este modo a la cónsola virtual gráfica (generalmente con las teclas CONTROL+ALTERNO+F7).** He de advertiros también que debéis iniciar sesión con usuario y contraseña, sin importar que ya hayas abierto sesión gráfica, es decir, no se heredan las credenciales. *Lo bueno de utilizar la cónsola virtual es que contamos con una GRAN Y AMPLIA pantalla en "modo ASCII",* ideal para editar grandes archivos de texto. Por último debo indicaros que del F1 al F6 son 6 cónsolas virtuales totalmente independientes la una de las otras. Para más detalles en cuanto a personalizar vuestra ventana terminal os aconsejo leer el excelente artículo del Licenciado Pedro Ruíz Hidalgo en Ubunlog.com.

Este viejo no los aburre más con historias así que entramos en materia:

Manejo del cursor con el teclado en bash shell.

El título de este párrafo os parecerá redundante *y lo que paso a explicaros aún más*. El **cursor** siempre ha sido un símbolo "|" (cuando está en modo de inserción) o con un símbolo de bloque "?" cuando está en modo de sobreescritura y sirve para indicarnos dónde estamos tecleando (los monitores vienen con un modo de parapadeo automático y en modo gráfico esto depende del sistema operativo correspondiente).

Cuando se masificó el uso de los "mouses" o **ratones** nos llegó el concepto de puntero del ratón o simplemente **puntero**. Es así como el teclado maneja el movimiento del cursor y el ratón el movimiento del puntero *siendo posible ubicar el cursor con ayuda del puntero a donde esté disponible, generalmente cuando el puntero cambia su forma al famoso icono de "doble te".*

Cuando estamos en la línea de comandos podemos usar las siguientes teclas para manejar nuestro cursor:

- Tecla Flecha izquierda: nos movemos un caracter a la izquierda.
- Tecla Flecha derecha: nos movemos un caracter a la derecha, si no se ha acabado la línea.
- Tecla INTRO o "ENTER", RETORNO o "RETURN" (hay dos en el teclado, tienen diferentes códigos, la historia es larga): indicamos al ordenador que introduzca en memoria la cadena de caracteres que hemos escrito, si hay comando(s) válido(s) los ejecuta, de lo contrario nos indica aproximadamente dónde puede estar el error(es) de nomenclatura.
- Tecla Flecha arriba: repetimos última línea introducida, si hubiera.
- Si estamos seguros acerca de ual es el último comando que introdujimos simplemente tecleamos "!!" y al presionar Intro lo "repetimos" (nos coloca también una línea previa con el comando en sí apra que veamos qué estamos solicitando, por si las dudas -uno de esos raror casos explícitos en GNU/Linux). También podemos introducir "!!" seguido de comandos adicionales, **el detalle es el siguiente: con flecha arriba modificamos la última línea y aís queda grabada en el historial** (más adelante hablaremos del historial) **y con "!!" introduciremos una línea nueva al historial.**
- Tecla Flecha abajo: luego de pulsar Flecha Arriba podremos volver a una línea de comandos nueva "bajando" tantas veces hayamos "subido" con la otra tecla.
- Tecla Inicio o "HOME": nos permite ubicar el cursor a al inicio de la línea que estamos escribiendo.
- Tecla Fin o "END": nos permite ir al final de la línea.

Más adelante veremos combinaciones de teclas (tecla Control+otra tecla, por ejemplo). Esta sección la finalizamos con otro detalle: muchas veces necesitamos escribir una linea de comando muy larga y aunque la ventana gráfica que contiene la terminal la podemos redimensionar con el puntero (arrastrando) e incluso podemos escoger una fuente más pequeña para tener más espacio de escritura de caracteres, *nuestro monitor tiene un ancho finito.* Para estos casos

utilizamos para concatenar el símbolo de barra invertida que le indica al ordenador de que a pesar de que presionemos la tecla Intro no la va a procesar si le colocamos el "\"

Generalmente usamos esta última opción con comandos que hay que pasarles parámetros:

Como veís en la imagen anterior, introducimos el comando, luego un espacio y una barra invertida, presionamos Intro y a continuación escribimos el primer parámetro acompañado al final de otro espacio y otra barra invertida; debemos tener cuidado al finalizar con todos los parámetros NO introducir la barra invertida y presionamos Intro de nuevo: así el ordenador procesa como una sola línea. Al finalizar de procesar el comando deseado y cuando volvemos a tener entrada de teclado, podemos presionar la tecla Flecha Arriba y veremos la línea completa que introdujimos previamente.

Otros atajos de teclado para el BASH shell.

Nosotros mostramos predilección por las teclas que son evidentes en nuestro teclado: **Inicio (Home)** , **Fin (End), AvPag (PgDown),** etcétera PERO hay unas combinaciones que son muy útiles. Dichas combinaciones implican las "teclas muertas" o "teclas pasivas" (CONTROL, ALTERNO, INVERSO) presionadas primero y sostenidas mientras se presiona otra tecla. Acá tenemos una cuenta en Twitter que se dedica a recordarnos cada cierto tiempo el uso agresivo de la línea de comandos, acá unos cuantos que consideramos muy útiles:

https://twitter.com/climagic/status/849267092209770497

https://twitter.com/climagic/status/849272281620721671

https://twitter.com/climagic/status/849353192731275264

https://twitter.com/climagic/status/849366031365746688

https://twitter.com/climagic/status/849627337377370113

https://twitter.com/climagic/status/849328017998577666

Manejo del historial de comandos introducidos en el BASH shell.

Antes de comenzar a introducir comandos os digo de antemano que todos ellos quedan registrados en memoria (así apaguemos nuestro ordenador) y solo necesitaremos escribir el comando **history** para revisar el historial del "bash shell". Como dicho listado puede llegar a ser

muy largo, podemos presionar la tecla CONTROL+R y escribir una palabra clave y nos buscará rápidamente en cual linea (y nos ubicará en esa línea, presionando de nuevo CTRL+R podemos "navegar" tal como indicamos anteriormente). Por último, si deseamos borrar el historial solo debemos escribir "**history -c" .**

Combinaciones de teclas útiles.

https://twitter.com/climagic/status/849252995002978306

Control+R es apenas una de las combinaciones, tal como describimos en la sección anterior, acá otros comandos adicionales:

https://twitter.com/climagic/status/849340857249652737

https://twitter.com/climagic/status/849645995747094530

Referencia de Comandos en Linux y su BASH shell.

Tomado de la traducción hecha por Alex Barrios (alexbariv at gmail.com) -cuenta Twitter @alexbariv (en desuso, cuenta actual @alexertech, página web donde imparte cursos: http://www.alexertech.com/)- **quien a su vez lo tomó de** Jacob Peddicort **en su blog "**Unix/Linux Command Cheat Sheet**" bajo licencia "**Creative Commons**" con la cual les puedo ofrecer esta entrada reconociendo la autoría ("appropiate credit").** En este enlace alterno de mi propio servidor suministro dicho documento traducido en formato pdf, ya ustedes saben cómo es el internet y su dinámica en el alojamiento de páginas web.

Actualizado el martes 15 de agosto de 2017.

Encontramos una útil lista abreviada de comandos realizada por Peteris Krumins y **compartida** por el señor Fran en su cuenta Twitter, haciendo alusión a sus talleres GNU/Linux que imparte ¡enhorabuena!:

https://twitter.com/0fjrm0/status/896661658881359872

Comandos del sistema operativo.

Comando "date".

Comenzamos por conocer nuestro entorno, nuestro ordenador (o un ordenador remoto, luego veremos cómo). El primer comando es "**date**" el cual nos devuelve la hora y fecha de nuestro equipo. A pesar de ser uno de los comandos más sencillos no os dejéis engañar por GNU/Linux: para saber cuales otras opciones tiene podemos escribir "**man**" + nombre de comando. Al invocar esta ayuda sobre comando, podéis subir y bajar con las teclas de dirección del teclado, así como avanzar o retroceder páginas con las teclas correspondientes (también podéis usar la rueda del ratón, si es que está disponible). Si queréis leer ayuda sobre el comando "**man**" en sí mismo allí presionaréis la tecla "h" -nemotécnico para "help"- y para salir del mismo usaremos la tecla "q" -nemotécnico para "quit"-.

Es así que podemos describir que el comando "**date**" puede ser utilizado tanto para saber la hora y fecha del ordenador *así como para fijarlo* (se necesitan permisos de administrador o "root") , así como también para conocer hora y fecha de otras ciudades del mundo. A medida que toméis experiencia disfrutaréis del comando "**date**" e incluso podréis usarlo como herramienta para calcular fechas y horas en formatos personalizados (muy útil para cuando querramos introducir valores de tiempo en bases de datos). *Paciencia, "Roma no se construyó en un solo* día". Mirad la siguiente animación en formato gif sobre unas pocas llamadas al comando de marras:

Comando "clear".

Muchas veces necesitaremos "limpiar la pizarra" (recordad vuestros tiempos en la escuela y liceo donde nuestros profesores escribía con gis o tiza), para ello solo escribiremos "**clear**" y la tecla introducir o "enter". Acá observamos que la rueda del ratón es útil para ir rápidamente hacia lo que acabamos de "borrar" (o usamos las barras de desplazamiento de la venta terminal, si es que estamos en modo gráfico).

Comando "cal".

Ya sabemos visualizar o fijar la fecha actual ahora veremos cómo visualizar el calendario del mes actual (con la fecha actual resaltada): simplemente introducimos "**cal**" y presionamos introducir o "enter" -de ahora en adelante obviaremos el indicar que se debe presionar la tecla INTRO para ejecutar los comandos-. Otra cosa que os preguntaréis para qué utilizar este comando si podemos ver la bandeja del sistema con tan solo mover nuestra vista a la bandeja del sistema, *pero recordad que en modo TTY tendremos absolutamente toda la pantalla para la línea de comandos o tal vez estemos trabajando con un servidor al cual no dispone de entorno gráfico -lo cual es más común de lo que se cree-.*

Como ya es costumbre no nos quedaremos con esa descripción tan simple del comando, sino que ahondaremos en unos cuantos más:

- Si queremos ver el mes de diciembre del año en curso: "**cal -m 12**", donde "12" es el número de mes deseado.
- Si queremos ver el mes anterior, el actual y el próximo: "**cal -3**".
- Si queremos ver el mes actual y los próximos 5 meses: "**cal -A 5**" ("A" es nemotécnico para "after").
- Si por el contrario queremos ver los últimos 5 meses y el actual: "**cal -B 5**" ("B" es nemotécnico para "before").
- **El comando "ncal" permite ver el calendario en base a los días de la semana y acepta los parámetros anteriores,** además de otros específicos para "**ncal**" como por ejemplo el día de comienzo de semana: "**-M**" para lunes -"monday"- y "**-S**" para domingo -"sunday"-.

Comando "uptime".

Permite conocer la hora actual, el tiempo que ha estado encendida nuestra computadora, el número de usuarios conectados (cada ventana TTY que hayamos iniciado sesión o cada conexión SSH cuenta como usuario) y el promedio de carga de trabajo que presenta el equipo los últimos 1, 5 y 15 minutos. Con la opción "**-p**" (nemotécnico para "**--pretty**") simplemente veremos el tiempo que ha estado ejecutando el sistema operativo.

Comando "w".

Pues sí, ¡un comando de una sola letra! Este comando complementa a "**uptime**" ya que muestra la misma información *pero con un reporte detallado de los usuarios conectados actualmente:*

- **USER**: Nombre de usuario.
- **TTY**: Ventana *TeleTYpe* en uso (recordad que van del 1 al 6 en texto y 7 en gráfico).
- **FROM**: El nombre del equipo donde está conectado (dirección IP), de ser remoto.
- **LOGIN@**: Hora cuando inició sesión.
- **IDLE**: Tiempo transcurrido -en minutos- desde que inició sesión.
- **JCPU**: Tiempo acumulado -en minutos- de las tareas adjuntas a cada terminal.
- **PCPU**: Tiempo utilizado -en minutos- de la tarea que indica en **WHAT.**
- **WHAT**: muestra la orden que inició la ventana terminal o la última ejecutada.

Los argumentos que podemos pasarle al comando "**w**" son los siguientes, a saber:

- "**-h**" o "**--no-header**": Muestra solamente los usuarios conectados (omite la información que lista el comando "**uptime**").
- "**-s**" o "**--short**": no muestra las columnas **LOGIN@**, **JCPU** ni **PCPU**.
- "**-f**" o "**--from**": mo muestra de donde están conectados los usuarios (dependiendo de la distribución Linux utilizada, puede ser que no sea posible desactivar esta opción).
- "**-i**" o "**--ip-addr**": muestra la dirección IP en vez de nombre de equipo.

- "**-o**" o "**--old-style**": imprime espacios en blanco si el tiempo transcurrido es menor a un minuto.
- "**nombre_de_usuario"**: muestra las conexiones abiertas para un usuario específico; se pueden solicitar varios usuarios al mismo tiempo si se solicitan los nombres separados por comas.

Comando "whoami".

Acá por fin vemos un comando sumamente simple: "**whoami**" nos devuelve el nombre con que iniciamos sesión. Solamente tiene dos argumentos: "**-help**" muestra la ayuda que nos indica que el otro argumento es "**--version**" que visualiza la versión que tenemos instalada. *Tal vez nos parezca un comando tonto pero si lo usamos en un* ***comando de procesos por lotes*** *-"bash script"- podremos usar el comando "tubería" -"|"- para pasar esta variable a otros procesos más complejos.* Por ejemplo, podremos crear un guión que comprima los documentos del usuario cada cierto tiempo con **crontab** y colocarle el nombre del usuario, acompañado con fecha y hora, al nombre del archivo comprimido y guardarlo en una ubicación remota.

Comando "finger".

Un comando que NO viene instalado por defecto en Ubuntu y debemos descargarlo con "**sudo apt install finger**". Sirve para obtener el usuario, el nombre de usuario, directorio de datos personales, última conexión, último correo leído por dicho usuario e incluso la agenda almacenada. *Un uso muy útil es para verificar si existe un usuario en particular, ya que dicho comando devuelve si existe o no en el equipo.* Por demás está decir que con el comando "**chfn**" podemos cambiar nuestros datos personales para que pueda ser visualizado por otros usuarios del sistema.

Comando "uname".

Con este comando podremos ver el sistema operativo utilizado, **Linux**. Si queremos ver toda la información del ordenador usaremos el argumento "**-a**" o "**--all**" o podremos ver valores específicos (de nuevo, *por ejemplo,* podremos realizar un guión que nos devuelva la versión Linux utilizada para instalar un programa por nosotros escrito). Los argumentos detallados son:

- "**-s**" o "**--kernel-name**": nombre del kernel instalado.
- "**-n**" o "**--nodename**": nombre del ordenador para indetificarse en la red de área local.
- "**-r**" o "**--kernel-release**": subversión del kernel instalado.
- "**-v**" o "**--kernel-version**": versión del kernel instalado.
- "**-p**" o "**--processor**": tipo de procesador.
- "**-i**" o "**--hardware-platform**": plataforma del procesador.
- "**-o**" o "**--operating-system**": sistema operativo instalado.

Manejo de archivos con bash shell.

En la figura arriba mostrada aparece el siguiente prefijo de manera constante:

`jimmy@KEVIN:~$ (cursor titilando)`

Esto me indica que estoy conectado con la cuenta de usuario *jimmy* en la máquina *KEVIN* seguido de **":~$"**: todo esto es el **indicador** o **prompt** y al aparecer como el signo de pesos indica que somos un usuario sin privilegios (ya comentado en un párrafo anterior) y cuando tenga el símbolo almohadilla o numeral estaremos en modo de **superusuario**. El símbolo de la virgulilla nos indica que estamos en nuestro directorio raíz del sistema.

En este otro ejemplo trabajamos con una máquina con **Canaima 4.1 Kukenan** y como pueden observar el nombre del usuario coincide con el nombre de la máquina y estamos con derechos limitados, ver signo de pesos:

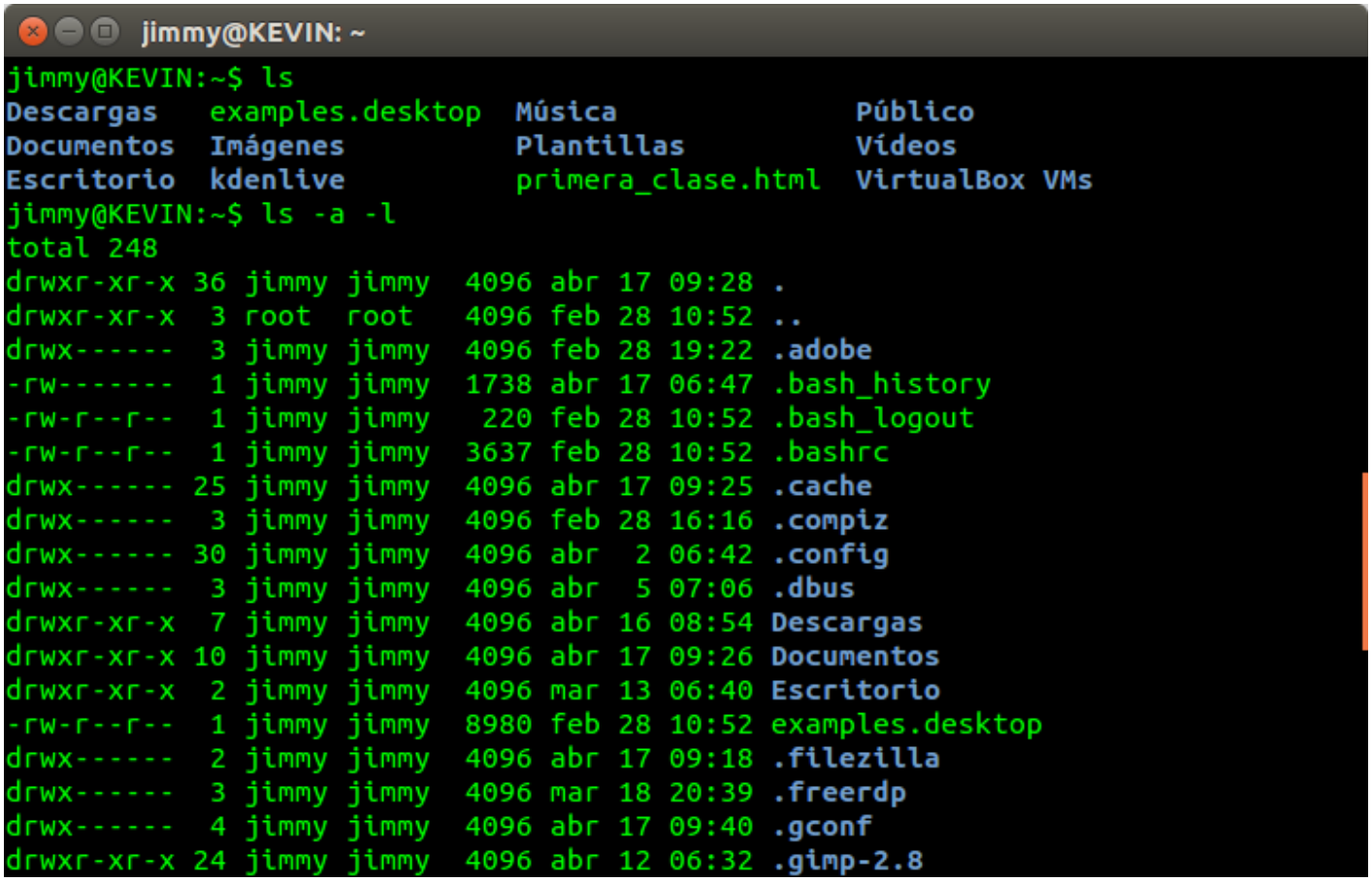
Volviendo a nuestro ejemplo original comenzamos por ver los comandos para el manejo de archivos por medio del BASH shell.

Comando "ls".

Se define como "listar directorio" pero yo agregaría también que archivos también. Así como en una oficina hay escritorios con gavetas y documentos almacenados en ellas, en nuestros discos duros (=escritorios) cada gaveta (=directorio) guardamos hojas (=archivos) y estas hojas pueden tener diferentes tamaños, colores y usos (=extensiones de archivos). *Esa es la abstracción más fácil para comprender, de manera física y humana, el sistema de archivos.* Al introducir **ls** y presionar la tecla **intro(ducir)** o "**enter**" (dependiendo de cuál tipo de teclado tengan ustedes)

En seguida notamos los colores que destacan los diferentes elementos (vuelvo y repito, estoy utilizando **Ubuntu 14.04 LTS *Trusty Tahr***, muy probablemente el **GNU/Linux** que vosotros utilicéis sea diferente). Coloreado en verde son los archivos y en azul los directorios (nosotros los hombres distinguimos sólo un puñado de colores así que me disculpan la falta de precisión y/o daltonismo).

Los archivos contienen *atributos* que le indican al sistema operativo la fecha de creación, de modificación, quién es el propietario y a cual grupo de usuarios pertenece (ya les comenté que existe el **root** y su cuenta de usuario ¿recuerdan?) así como también si es de lectura, escritura y hasta si es oculto para los usuarios. En **GNU/Linux** ocurre que los archivos ocultos simplemente llevan un punto delante de su descripción (por favor, los que sepan la historia de esta "costumbre" pro favor comente a pie de entrada). Al introducir "**ls -a -l**" en el BASH shell obtendremos el siguiente resultado (la lista es larga, la captura de pantalla NO abarca todos los archivos y directorios listados):



Como pueden obervar, podemos notar que siguen apreciendo "Descargas", "Documentos", "Escritorio" entre otros *pero además los archivos y directorios ocultos que comienzan con un punto en su nombre.* También notamos que aparecen unos arriba de otros, a modo de columna **o simplemente en modo lista (de allí el parámetro "-l" del comando).** Este modo de lista, de izquierda a derecha -como es el castellano-:

- Tipo de archivo ("filetype"): el primer caracter si es "d" es un directorio ("directory"), de lo contrario es un archivo "-".
- Permisos de archivo ("file permissions"): los siguientes 9 caracteres (más adelante veremos qué significan al agruparlos en tríos de caracteres).
- Enlaces ("links"): indica en qué nivel se encuentra, cuántas carpetas "por encima" tiene cada archivo o cada directorio. Para comprender estos volvamos al ejemplo de nuestra oficina en el mundo real: dentro de cada gaveta puedo tener hojas sueltas y además carpetas que agrupan folios; es como una estructura jerárquica en la que si queremos leer un documento debemos abrir la gaveta y luego abrir la carpeta: *ésos son los niveles a los cuales se refiere.*
- Propietario ("user"): el usuario quien creó el archivo (o heredó, veremos más adelante).
- Grupo propietario ("group user"): el grupo de usuario al cual pertenece el archivo o carpeta.
- Tamaño en bytes que ocupa en disco.

- Fecha de la última modificación.
- Nombre del archivo o directorio.

Como podrán deducir, el parámetro "**-a**" del comando **ls** permite mostrar los archivos y carpetas ocultas; también es válido "juntar" parámetros que sean compatibles y escribir simplemente "**ls -la**" o "**ls -al**", intenten y comprueben ustedes que produce los mismos resultados.

Si queréis ver una lista completa de los parámetros de cualquier comando sólo tendréis que escribir "**man *comando***" (en este caso "**man ls**") y obtendréis toda la información.

Comando "pwd".

Si en algún momento se nos olvida en cual carpeta estamos trabajando simplemente utilizamos el comando **pwd,** esto es especialmente útil si personalizamos nuestro indicador o "prompt" (*también podemos usar este comando para pasarlo como variable en archivos de procesos por lotes si queremos, por ejemplo, recorrer de manera recursiva el árbol de directorios para buscar un archivo: "**pwd**" nos indicará en cual directorio se encuentra*).

Comandos "df", "du" y "free".

- Con el comando "**df**" prodremos obtener la cantidad de espacio utilizado en nuestro disco duro.
- Con el comando "**du**" nos dará la información de los archivos en una carpeta pero será una lista extensa.
- Con "**free**" podremos observar la cantidad de memoria RAM y SWAP y con los parámetros **-b** lo podremos ver expresadas en bytes, con **-k** en kilobytes y así sucesivamente: **-m**, **-g** y si queremos subar ambas memorias incluiremos el parámetro **-t.**

https://twitter.com/climagic/status/844560946823598081

Comandos "whereis" y "wich".

- Con el comando "**whereis**" podremos saber dónde se encuentra instalada determinada aplicación, si está instalada en neustro ordenador, ejemplo "**whereis pinta**".
- Con el comando "**which**" podremos saber dónde se ejecutará una aplicación, ejemplo: "**which sh**".

Comando "cat".

Muchas veces necesitamos "mostrar por pantalla" el contenido de uno o varios archivos de texto:

pues bien a esto se le llama "standard ouput" o salida normal del BASH shell. El comando cat es capaz enviar a la salida normal todo lo que le envíen, incluso si se ejecuta sin parámetros todo lo que escribamos -y presionamos intro- lo observaremos por pantalla.

En este punto cabe recordar os que en GNU/Liux *todo es un archivo* incluso los procesos del sistema operativo, por ejemplo los siguientes co mando nos devolverán datos interesantes de nuestro procesador y memoria e n nuestro ordenador: cat /proc/cpuinfo cat /proc/meminfo

Como funciona la entrada y salida normal en GNU/Linux BASH.

Haremos aquí un pequeño paréntesis para poneros un ejemplo sencillo sobre cómo aprovechar la salida normal "output standard" del comando "**cat**" y para ello nombraremos al artículo publicado en Totaki.com (desconocemos su autoría) y donde explican rapidamente cómo funcionan los ciclos:

```
for j in `cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_freq
uencies` do echo "Frecuencia: "$j" Hz" done
```

En la línea número uno observamos todo lo que está entrecomillado: el comando "**cat**" lee un archivo que contiene las frecuencias disponibles del CPU en nuestro ordenador (en este año 2017 es común y frecuente que tengan dos o más núcleos como en nuestro caso que tenemos cuatro) y dicha salida es almacenada en una variable *implícita* hacia un ciclo evaluado en una variable j y que separa los elementos por medio del **Separador Interno de Campos (Internal Field Separator)** -espacio, tabulador, nueva línea- imprimiendo por pantalla todos y cada uno de los elementos.

De igual manera en otro artículo publicado en DigitalOcean sobre como instalar nuestro servidor web Ngix se hace eco de un truco para conocer nuestra dirección IP local, la asignada por nuestro enrutador de área local o bien la que nos asigna nuestro proveedor de internet cuando estamos conectados directamente al modem respectivo. Para ello podemos ejecutar una ventana terminal donde escribimos la siguiente orden:

```
ip addr show eth0
```

(si usáis un ordenador portátil conectado por inalámbrico, cambiad a **eth0** por **wlan0**)

Si la ejecutamos en nuestra ventana terminal notaremos que es mucha más información que la que necesitamos (os invito a practicar directamente los comandos en vuestra computadora a medida que vayáis leyendo nuestro artículo) así que notamos que hay una palabra clave "**inet**" sobre la cual nos apoyaremos para extraer lo quenos interesa, así que ahora le agregamos el símbolo de tubería "|" acompañado de otro comando llamado "**grep**" el cual nos permite imprimir *líneas* de acuerdo un patrón -en este caso la palabra clave "**inet**"-:

```
ip addr show eth0 | grep inet
```

Como veréis ya estamos más cerca de nuestro cometido. Para seguir adelante usaremos la metodología de estudio de la Facultad de Ingeniería de la Universidad de Carabobo (*donde nos quitaron a nosotros la costumbre de propinar coces a diestra y sisniestra*) y os presentamos el resto de los comandos necesarios los cuales dejamos a vuestro estudio posterior *-metodología de estudio: ésta es la letra A, la B, la C... ahora recitadme el abecedario completo aunque NO os lo hayamos enseñado aún-:*

```
ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.*$//'
```

Preludio a los atajos de teclado.

Como un avance a la siguiente sección podemos indicaros que no es necesario escribir el nombre completo de las rutas y/o archivos: con escribir dos o tres letras y presionar la tecla tabulador nuestro shell completará automáticamente *siempre y cuando el nombre del archivo o carpeta sea único con las letras que introduzcamos.* Es por ello que si no "completa" entonces volvemos a presionar la tecla tabulador y nos mostrará una lista de carpetas o archivos que comiencen con las letras solicitadas. Acá podéis ver parte del proceso:

Volviendo a los atajos de teclado en bash shell.

Es sumamente útil sólo usar el teclado para trabajar en nuestra terminal pero la primera frustración que tendremos es la muy famosa acción "copiar y pegar", una técnica que nació en las antiguas imprentas para aligerar el trabajo con los tipografías de plomo. La historia del por qué no funciona en el bash las combinaciones **CRTL+C** y **CTRL+V** es larga y está bien explicada (en inglés) en este enlace. Para hacer corta esta entrada sólamente les enseño que lo que debemos hacer es usar **MAYÚ+CTRL+C** (SHIFT+CTRL+C) para "copiar" y **MAYÚ+CTRL+V** para "pegar".

Hasta aquí no representa mayor sacrificio de nuestra parte, lo podemos aprender rápidamente, sólo es cuestión de pulsar sólo una tecla más. ¿Recuerdan que les dije al principio sobre un "atajo

de ratón"? Pues bien para seleccionar, aunque ustedes no lo crean, ***sólamente se puede seleccionar con el*** puntero del ratón, y luego usar las combinaciones de teclas descritas. Hoy en día los ratones generalmente traen tres botones: izquierdo, central (rueda) y derecho. Pulsando la rueda sobre un texto seleccionado COPIARÁ Y PEGARÁ AUTOMÁTICAMENTE EN DONDE TENGAMOS EL CURSOR. ¿Cómo puede sernos útil dicho comportamiento? Será cuestión primero de practicar cómo funciona para luego encontrarle uso. 8-) Aquí les dejo un video explicativo:

Otras combinaciones de teclas útiles:

- **CTRL+FLECHA IZQ** o **CTRL+FLECHA DER** va de palabra en palabra, al principio de cada una, en la dirección correspondiente.
- **TECLA "INICIO"** y **"FIN"** nos ubica al principio o al final de la línea que estemos introduciendo en el **prompt**.

Actualizado el lunes 29 de febrero de 2016, año bisiesto.

Por Twitter seguimos la cuenta @Ubunlog donde se dedican de lleno a publicar temas sobre GNU/Linux Ubuntu pero sin dejar de lado asuntos que son comunes a todas las "distro's". En esta caso en particular tuvimos la oportunidad de leer el artículo "Los 5 comandos que todo usuario de Linux debería conocer" y, con toda humildad, leemos todas las noticias que podemos para mantenernos actualizados. Una opción que NO conocíamos es el comando **gksudo** el cual nos permite, con derechos de **superusuario**, "correr" una aplicación gráfica, como por ejemplo **gedit**. Generalmente utilizamos nano para editar con la linea de comandos, pero a veces los ojos agradecen un entorno más agradable, y como la consigna es *hagamos más con menos trabajo y/o con más agrado* pues publicamos esta enmienda a nuestra entrada en el blog. Aquí tenéis el mensaje "tuit" publicado por nuestros "tutores" del otro lado del "charco":

> Los 5 comandos que todo usuario de Linux debería conocer - https://t.co/32XM18zH6v pic.twitter.com/4hXOmsHEUe
>
> — Ubunlog (@Ubunlog) February 28, 2016

Es así que si deseamos "lanzar" gedit como superusuario pues tecleamos lo siguiente seguido de la tecla INTRO ("ENTER") en el BASH shell:

```
gksudo gedit
```

Y si tecleamos solamente el comando **gksudo** pues se nos muestra una ventana gráfica *para que podamos seleccionar el usuario que necesiteis utilizar:*

> GNU/Linux: linea de comandos (shell). https://t.co/awfGUF1O41 (actualizado #gksudo gracias a @Ubunlog) pic.twitter.com/ovfjJyUYp4
>
> — ks7000 (@ks7000) February 29, 2016

Actualizado el viernes 29 de septiembre de 2017.

¿Cuales son los comandos que más utilizamos?

Para responder esta pregunta, la cual nos puede sorprender por su resultado, utilizamos el siguiente truco en este hilo creado por Brian P. Hogan (más abajo nuestro resultado, agregad el vuestro a al hilo en Twitter por favor):

```
history | awk '{print $2;}' | sort | uniq -c | sort -rn | head -n 10
```

https://twitter.com/ks7000/status/913864698990383105

Enlaces relacionados al BASH shell.

Enlaces relacionados (en castellano):

- "22 trucos para ser un ninja de Ubuntu" en Genbeta.
- "Usando la línea de comandos" por Marcelo Horacio Fortino.
- "Crear tus propios scripts usando bash" por Pedro Ruíz Hidalgo en Ubunlog.com
- "Cómo usar funciones en Bash" por Pedro Ruíz Hidalgo en Ubunlog.com
- "Bucles for en BASH" en Totaki.com.
- "Aprende a realizar el cálculo de la letra del DNI mediante un script Bash" por Pedro Ruiz

Hidalgo en Ubunlog.com
- "Personaliza tu terminal en Ubuntu" por Pedro Ruíz Hidalgo en Ubunlog.com
- "Matar procesos y obtener información del sistema desde la terminal" por Damián Amoedo.
- «Probamos Bash, la integración de Ubuntu Linux en Windows 10» por Rubén Velasco.
- «Listado de 400 comandos útiles para Gnu/Linux».

Cuentas en Twitter dedicadas a la terminal (y otras cositas adicionales):

Tweets by sudosu_blog

Enlaces relacionados (en inglés) con el BASH shell:

- «The Ultimate A To Z List of Linux Commands | Linux Command Line Reference»
- "BASH Programming - Introduction HOW-TO" by Mike G mikkey at dynamo.com.ar
- "Bash History: Display Date And Time For Each Command" by Vivek Gite.
- "The w Command" at The Linux Information Project.
- "Old School Linux Solutions: finger" By Jack Wallen.
- "Linux and Unix which command".
- "How To Install Nginx on Ubuntu 16.04" by Justin Ellingwood.
- "How To Write and Use Custom Shell Functions and Libraries" by Aaron Kili.
- "How to Search for Files from the Linux Command Line" by Jack Wallen.
- "10 Tools To Add Some Spice To Your UNIX Shell Scripts" by Vivek Gite.
- "Deep Hardware Discovery With lshw and lsusb on Linux" by Carla Schröder.
- "Hardware Lister (lshw)" at ezIX.
- "How to run sudo command without a password on a Linux or Unix" by Vivek Gite.
- «How to Use the Command Line for Apple macOS and Linux» by Tania Rascia.
- «Bash prompt tips and tricks» by Dave Neary.
- «Learn just enough Linux to get things done».

Some funny facts about command terminal:

https://twitter.com/DennisCode/status/873225437798834176

https://twitter.com/amy_bat/status/844047339153215488

https://twitter.com/CommitStrip/status/836638321514328064

Enlaces relacionados en Twitter:

Tweets about #shell from:@ks7000

!function(d,s,id){var js,fjs=d.getElementsByTagName(s)[0],p=/^http:/.test(d.location)?'http':'https';if(!d.getElementById(id)){js=d.createElement(s);js.id=id;js.src=p+"://platform.twitter.com/widgets.js";fjs.parentNode.insertBefore(js,fjs);}}(document,"script","twitter-wjs");

Tweets about #bash from:ks7000

!function(d,s,id){var js,fjs=d.getElementsByTagName(s)[0],p=/^http:/.test(d.location)?'http':'https';if(!d.getElementById(id)){js=d.createElement(s);js.id=id;js.src=p+"://platform.twitter.com/widgets.js";fjs.parentNode.insertBefore(js,fjs);}}(document,"script","twitter-wjs");

Mensajes puntuales, "tuits" útiles

https://twitter.com/hoyd/status/970007880551288834