

Línea de comandos y comandos principales

***Arquitectura y Sistemas Operativos.
Tecnatura Superior en Programación.
UTN-FRA***

Autores: *Prof. Martín Isusi Seff*

Revisores: *Prof. Marcos Pablo Russo*

Versión: 1



Esta obra está bajo una [Licencia Creative Commons Atribución-CompartirIgual 4.0 Internacional](https://creativecommons.org/licenses/by-sa/4.0/).

Entendiendo la línea de comandos

La línea de comandos, en GNU/Linux, Unix, OSX o Windows, es una terminal que permite al usuario ejecutar instrucciones en la computadora. La línea de comandos puede parecer complicada al principio, pero es una de las herramientas más potentes que tiene a mano cualquier desarrollador o administrador de sistemas.

Si bien la mayoría de los usuarios acostumbra a usar una interfaz gráfica para realizar cualquier acción, mediante la línea de comandos, en gran medida, se pueden realizar las mismas acciones.

Dependiendo del sistema operativo, la línea de comandos tendrá más o menos importancia. En Microsoft Windows, la línea de comandos prácticamente no es utilizada ya que el propio sistema operativo se construye bajo una arquitectura basada en "ventanas". Su base es una interfaz gráfica. Por otro lado, en GNU/Linux (cualquiera sea la distribución), la línea de comandos es quizás la herramienta más importante. Si bien hoy en día cualquier distribución orientada al usuario se instalará con una interfaz gráfica, la misma no hace más que ejecutar los comandos que se el usuario ejecutaría en la terminal de línea de comandos.

La principal ventaja de la línea de comandos radica en la facilidad que nos da a la hora de automatizar procesos extensos o repetitivos. En estos casos se podría, por ejemplo, crear un *script*¹, que ejecute paso a paso los comandos que de otra manera se ejecutarían a mano.

Existen distintos intérpretes de línea de comandos, bash, csh, cmd, etc.

Cada comando que ejecutamos en la terminal de línea de comandos no forma parte del intérprete en sí, sino que cada comando no es más que una aplicación almacenada en un directorio específico.

Tal como cualquier aplicación, los sistemas operativos tienen variables. Las variables, llamadas "variables de entorno", contienen información necesaria durante el funcionamiento del sistema. Desde la terminal de línea de comandos podemos ver el valor de las variables, así como también modificar el mismo. Cuando hablamos de terminal de línea de comandos, es importante mencionar particularmente una de las tantas variables de entorno, la variable PATH. Esta variable contiene rutas a directorios o enlaces. Estos directorios contienen los ejecutables de los comandos que ejecutamos desde la terminal línea de comandos. Si desde la terminal ejecuto el comando *ls*, el archivo ejecutable del mismo deberá estar dentro de alguno de los directorios especificados en la variable de entorno PATH.

¿Cómo se ve la terminal línea de comandos?

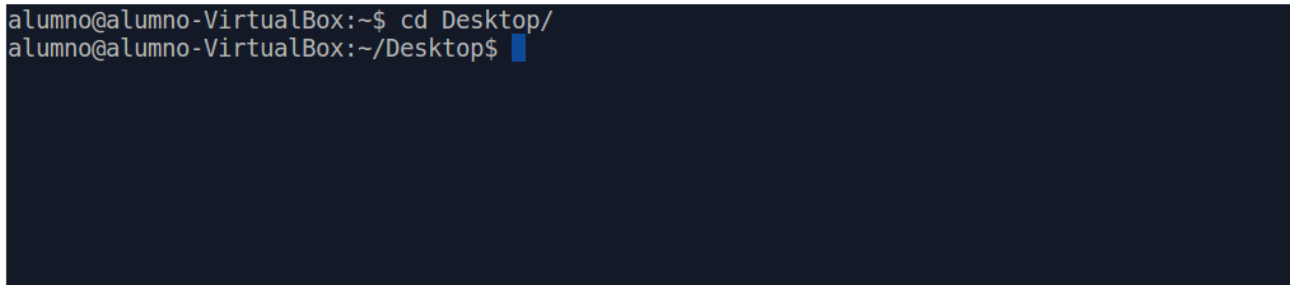
La terminal de línea de comandos, en cualquier sistema operativo, no es más que una pantalla (o una ventana si se abre desde una interfaz gráfica) que imprime texto. La terminal de línea

¹ Llamamos script al código fuente de una aplicación, sea el lenguaje que sea. En el contexto de un sistema operativo nos referimos a *script* al código que contiene comandos a ser ejecutados por el sistema operativo.

de comandos no muestra gráficos.

Cuando se inicia la terminal de línea de comandos, la misma se “ubicará” dentro de un directorio determinado (algo similar a lo que sucede cuando se abre un explorador de archivos gráfico) y quedará “esperando” que el usuario escriba y ejecute un comando.

El comando [quizás] más importante, es el comando **cd**. Este comando es el utilizado para cambiar de ubicación la línea de comandos. El comando *cd*, se utiliza escribiendo *cd ruta*, donde *ruta* es la *ruta relativa* o *ruta absoluta* a donde nos queremos ubicar.



```
alumno@alumno-VirtualBox:~$ cd Desktop/
alumno@alumno-VirtualBox:~/Desktop$
```

Figura 1.

En la Figura 1 se ve una terminal de línea de comandos. En la primera línea de la misma vemos la palabra 'alumno'. La misma corresponde al nombre del usuario con el que se inició la sesión. Luego de '@' se encuentra el nombre del equipo. Seguido de ':' se encuentra la ruta en la que está ubicada la terminal. Aquí vemos que está ubicada en '~'. Esto quiere decir que la terminal está ubicada en la carpeta personal del usuario 'alumno'. El símbolo '\$' indica que se trata de un usuario no administrador. Y, por último, tenemos 'cd Desktop', que es el comando que se ejecutó. Con ese comando, la terminal se ubicará en el directorio 'Desktop' que está dentro de '~'. Nótese cómo en la segunda línea, luego de ':' ahora se muestra '~/Desktop'.

Rutas o paths

Antes de continuar con los comandos principales, es necesario mencionar el concepto de *ruta* o *path*. Una *ruta*, es la dirección de un archivo (o directorio) dentro del sistema de archivos. Cuando trabajamos en la terminal de línea de comandos, las *rutas* pueden representarse de dos maneras: *rutas absolutas* o *rutas relativas*.

Rutas absolutas

Las rutas absolutas son rutas que se escriben desde la raíz del sistema de archivos. Tomemos como ejemplo la siguiente estructura de directorios:

```
/
|
|__ Directorio 1
|   |
|   |__ Directorio 2
|   |
|   |__ Directorio 3
```

En la estructura tenemos un directorio raíz llamado / (en GNU/Linux, siempre el directorio raíz

es /. En Windows, por ejemplo, el directorio raíz es aquel conocido como C:/). Dentro del directorio raíz se encuentra **Directorio 1**, y dentro del mismo se encuentran **Directorio 2** y **Directorio 3**.

La ruta absoluta de **Directorio 3** sería: **/Directorio 1/Directorio 3**. Allí vemos que está la raíz (/), el Directorio 1 y luego, seguido por / (en este caso no representa la raíz), se encuentra el Directorio 3.

Rutas relativas

Las rutas relativas son aquellas que no se indican desde el directorio raíz, sino que se escriben desde la "posición actual". Supongamos que abrimos una terminal de línea de comandos que se encuentra ubicada en **Directorio 1**. La ruta absoluta de **Directorio 3** sería la mencionada previamente, pero la ruta absoluta sería simplemente escribir **Directorio 3**. ¿Por qué? Porque **Directorio 3** se encuentra dentro de la posición donde estamos ubicados. Analicemos el siguiente caso:

```

/
|
|__ Directorio 1
   |
   |__ Directorio 2
   |
   |__ Directorio 3
      |
      |__ Directorio 4

```

Ahora, dentro de **Directorio 3**, se encuentra **Directorio 4**. Si nos encontramos con la terminal dentro de **Directorio 1**, ¿Cuál sería la ruta relativa hacia **Directorio 4**? Primero tenemos que ver que **Directorio 4**, está dentro de **Directorio 3**. Para ir desde nuestra supuesta ubicación hacia **Directorio 3**, simplemente tenemos que escribir el nombre del mismo. Por último, como **Directorio 4** está dentro de **Directorio 3** escribiríamos lo siguiente: **Directorio 3/Directorio 4**.

Comando ls

El comando **ls** nos permite poder ver los contenidos de los directorios. Al este comando pueden aplicarse las siguientes opciones:

-l	Muestra los archivos en forma de columna.
-a	Muestra todos los archivos (incluyendo los ocultos)
-R	Listado recursivo de todos los archivos y subdirectorios.
-i	Muestra el número de inodo.
-h	Muestra el espacio ocupado del archivo en MB, Kbyte, etc.
-t	Ordena por la fecha y hora de modificación.

Ejemplos:

```
alumno@alumno-VirtualBox:~$ ls -l
total 44
drwxrwxr-x 4 alumno alumno 4096 ago 11 13:53 Datos
drwxrwxr-x 6 alumno alumno 4096 ago 11 13:52 Datos_Generales
drwxr-xr-x 2 alumno alumno 4096 ago 9 14:18 Desktop
drwxr-xr-x 2 alumno alumno 4096 ago 11 13:25 Documents
drwxr-xr-x 2 alumno alumno 4096 ago 9 14:18 Downloads
drwxr-xr-x 2 alumno alumno 4096 ago 9 14:18 Music
drwxrwxr-x 3 alumno alumno 4096 ago 11 13:50 NuevoDirectorio
drwxr-xr-x 2 alumno alumno 4096 ago 9 14:18 Pictures
drwxr-xr-x 2 alumno alumno 4096 ago 9 14:18 Public
drwxr-xr-x 2 alumno alumno 4096 ago 9 14:18 Templates
drwxr-xr-x 2 alumno alumno 4096 ago 9 14:18 Videos
alumno@alumno-VirtualBox:~$
```

Figura 2

Obtener información de los comandos

Gracias a una serie de comandos podemos obtener información de los archivos de configuración, y de los comandos que ejecutamos, también tendremos otros comandos que nos permite realizar búsquedas de manuales. El comando que utilizaremos es `apropos`. Cada manual contiene una pequeña descripción, este comando nos permite buscar dentro de esas descripciones. Las opciones que podemos usar con este comando son las siguientes:

-e, --extract	Cada palabra clave se comparará de forma exacta con los nombres de páginas y las descripciones.
-d, --debug	Imprime información de depuración.
-r, --regex	Interpreta cada palabra clave como una expresión regular. Este comportamiento es el predeterminado. Cada palabra clave se comparará con los nombres de las páginas y las descripciones.
-w, --wildcard	Idem a la opción -r, --regex , aunque además permite la utilización de comodines.
-M ruta, --manpath=ruta	Especifica rutas separadas por "dos puntos" de la ubicación de los manuales. Por defecto apropos utiliza la variable de entorno \$MANPATH en todo caso que esté vacía la variable de entorno, utiliza como ruta para los manuales la variable \$PATH
-h, --help	Ayuda

Ejemplos:

```
alumno@alumno-VirtualBox:~$ apropos cd
apt-cdrom (8) - APT CD-ROM management utility
cd-create-profile (1) - Color Manager Profile Creation Tool
cd-fix-profile (1) - Color Manager Testing Tool
cd-it8 (1) - Color Manager Testing Tool
hex2hcd (1) - firmware converter
hipercdecode (1) - Decode a HIPERC stream into human readable form.
rsyncd.conf (5) - configuration file for rsync in daemon mode
sbigtopgm (1) - convert an SBIG CCDOPS file into a portable graymap
systemd-timesyncd (8) - Network Time Synchronization
systemd-timesyncd.service (8) - Network Time Synchronization
tcdrain (3) - get and set terminal attributes, line control, get and set baud rate
timesyncd.conf (5) - Network Time Synchronization configuration files
timesyncd.conf.d (5) - Network Time Synchronization configuration files
xfburn (1) - Simple CD/DVD burning tool
alumno@alumno-VirtualBox:~$
```

Figura 3

Comando *man*

Con el comando *man* podemos ver los manuales de los programas y los archivos de configuración. La siguiente tabla muestra los números de sección del manual y los tipos de páginas que contienen.

1	Programas ejecutables y guiones del intérprete de comandos.
2	Llamadas del sistema (funciones servidas por el núcleo).
3	Llamadas de la biblioteca (funciones contenidas en las bibliotecas del sistema).
4	Ficheros especiales (generalmente en /dev).
5	Formato de ficheros y convenios (por ejemplos, /etc/passwd).
6	Juegos.
7	Paquetes de macros y convenios.
8	Órdenes de administración del sistema (generalmente solo son para root)
9	Rutinas del núcleo.

Con el comando *man*, podemos usar las siguientes opciones:

-d, --debug	Imprime información de depuración.
-t comando	Formatea la página del manual del comando en postscript.
-k comando	Ídem a apropos .
-f comando	Busca las páginas del manual referenciadas por <i>comando</i> , e imprime la descripción de las que encuentra. Equivalente al comando whatis .
-u, --update	Los cachés de los índices de las bases de datos son actualizados sobre

	la marcha, es decir, no es necesario que mandb se ejecute periódicamente para mantener la consistencia.
-M ruta --manpath=ruta	Especifica rutas separadas por "dos puntos" de la ubicación de los manuales. Por defecto man utiliza la variable de entorno \$MANPATH en todo caso que esté vacía la variable de entorno, utiliza como ruta para los manuales la variable \$PATH .
-P página --pager=página	La <i>página</i> es el tamaño, esa sentencia sobrescribe la variable de entorno \$PAGER .
-d, --debug	Imprime información de depuración.

Comando *info*

Este comando supera la información del comando *man*, y nos permite navegar por medio de los enlaces como si fuera una página web. Las opciones que podemos usar con el comando *info* son las siguientes:

-k, --aprops	Buscar cadena en todos los índices de todos los manuales.
-d --directory=DIR	Agrega un directorio <i>DIR</i>
-f --file=ARCHIVO	Se especifica un archivo determinado <i>ARCHIVO</i>
-M ruta --manpath=ruta	Especifica rutas separadas por "dos puntos" de la ubicación de los manuales. Por defecto man utiliza la variable de entorno \$MANPATH en todo caso que esté vacía la variable de entorno, utiliza como ruta para los manuales la variable \$PATH .
-h, --help	Muestra ayuda del comando.

Comando *whatis*

El comando *whatis* nos da una pequeña descripción de un comando y las secciones de *man* que podemos consultar. Las opciones que podemos usar con el comando *whatis* son las siguientes:

-d, --debug	Imprime información de depuración.
-r, --regex	Interpreta cada palabra clave como una expresión regular. Este comportamiento es el predeterminado. Cada palabra clave se comparará con los nombres de las páginas y las descripciones.
-w, --wildcard	Idem a la opción -r, --regex , aunque además permite la utilización de

	comodines.
-M ruta, --manpath=ruta	Especifica rutas separadas por "dos puntos" de la ubicación de los manuales. Por defecto apropos utiliza la variable de entorno \$MANPATH en todo caso que esté vacía la variable de entorno, utiliza como ruta para los manuales la variable \$PATH
-h, --help	Ayuda

Creación de un archivo vacío o cambio de fecha y hora: *touch*

Este comando nos sirve para dos motivos: Si un archivo no existe este comando lo creará con el nombre especificado y con la fecha del momento. Por otro lado, si el archivo existe, le cambiará la fecha y hora. Las opciones que podemos utilizar con el comando *touch* son:

-a	Cambia solamente el tiempo de acceso.
-d string --date=string	Se cambia a la fecha que se indica en <i>string</i> .
-m	Cambia solamente la hora.

Ejemplo:

```
alumno@alumno-VirtualBox:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
alumno@alumno-VirtualBox:~$ touch nuevo_archivo
alumno@alumno-VirtualBox:~$ ls
Desktop Documents Downloads Music nuevo_archivo Pictures Public Templates Videos
alumno@alumno-VirtualBox:~$
```

Figura 4

Copia de archivos: *cp*

El comando *cp* nos permite copiar un archivo o directorio completo. Para usarlo escribimos:

cp [opciones] origen destino

Las opciones que podemos usar con este comando son:

-a	Es la combinación de -d (mantiene los enlaces) y -R (recursivo).
-i	Interactivo. Preguntará por cada archivo que se está copiando.
-p	Mantiene los permisos, dueño y grupo de los archivos/directorios.
-v	Mostrará información de los archivos/directorios que se están copiando (verbose).

Ejemplos:


```
alumno@alumno-VirtualBox:~$ ls
Desktop Documents Downloads Music nuevo_archivo Pictures Public Templates Videos
alumno@alumno-VirtualBox:~$ cp nuevo_archivo nuevo_archivo_2
alumno@alumno-VirtualBox:~$ ls
Desktop Downloads nuevo_archivo Pictures Templates
Documents Music nuevo_archivo_2 Public Videos
alumno@alumno-VirtualBox:~$
```

Figura 5

Para especificar el origen y destino de la copia, es posible utilizar comodines como ***** o **?**.

```
alumno@alumno-VirtualBox:~$ ls
Desktop Downloads nuevo_archivo Pictures Templates
Documents Music nuevo_archivo_2 Public Videos
alumno@alumno-VirtualBox:~$ cp nuevo_* Documents/
alumno@alumno-VirtualBox:~$ ls Documents/
nuevo_archivo nuevo_archivo_2
alumno@alumno-VirtualBox:~$
```

Figura 6

Mostrar contenido de archivos: *more*, *less* y *cat*

Tanto el comando **more** y **less** nos muestra el contenido del archivo y solo podemos bajar de a una línea por vez (con la tecla enter) o avanzar por página (con la tecla space). El comando **less** debe ser instalado como paquete apt, usando el comando **apt-get install less**. Además, el comando **less** nos permite realizar búsquedas dentro del archivo. También con las teclas (pgdown) podemos bajar a la página siguiente y con la tecla (pgup) podemos volver a la página anterior.

El comando **cat**, cumple la misma función que los comandos **more** o **less**, con la diferencia que el contenido completo se muestra en la pantalla.

Ejemplo:

```
alumno@alumno-VirtualBox:~$ ls
Desktop Documents Downloads Music Pictures Public saludo.txt Templates Videos
alumno@alumno-VirtualBox:~$ cat saludo.txt
Hola mundo!
alumno@alumno-VirtualBox:~$
```

Figura 7

Redirección: **>**, **>>**, **<**

A través de la redirección, podemos tomar la salida de un programa y enviarla automáticamente a un archivo. Este proceso lo maneja la propia shell, en un lugar del

programa. La redirección se divide en tres clases: salida a un archivo, añadir al final de un archivo, o envió de un archivo como entrada. Para recoger la salida de un programa en un archivo, finalice la línea del comando con el símbolo mayor que (>) y el nombre del archivo en el cual quiere guardar la salida redirigida.

```
alumno@alumno-VirtualBox:~$ ls
Desktop Documents Downloads Music Pictures Public saludo.txt Templates Videos
alumno@alumno-VirtualBox:~$ ls >> listaDeArchivo.txt
alumno@alumno-VirtualBox:~$ ls
Desktop Downloads Music Public Templates
Documents listaDeArchivo.txt Pictures saludo.txt Videos
alumno@alumno-VirtualBox:~$ cat listaDeArchivo.txt
Desktop
Documents
Downloads
listaDeArchivo.txt
Music
Pictures
Public
saludo.txt
Templates
Videos
alumno@alumno-VirtualBox:~$
```

Figura 8

Utilización de pipes (|)

Los pipes son un mecanismo por el cual la salida de un programa se puede enviar como entrada de otros programas. Los programas individuales se pueden encadenar juntos para convertirse en unas herramientas extremadamente potentes.

Ejemplo:

```
alumno@alumno-VirtualBox:~$ ls -la /etc/ | more
total 1120
drwxr-xr-x 128 root root 12288 ago 11 12:51 .
drwxr-xr-x 24 root root 4096 ago 11 12:51 ..
drwxr-xr-x 3 root root 4096 abr 20 2016 acpi
-rw-r--r-- 1 root root 3028 abr 20 2016 adduser.conf
drwxr-xr-x 2 root root 4096 ago 11 12:48 alternatives
-rw-r--r-- 1 root root 401 dic 28 2014 anacrontab
drwxr-xr-x 6 root root 4096 abr 20 2016 apm
drwxr-xr-x 3 root root 4096 ago 11 12:50 apparmor
drwxr-xr-x 8 root root 4096 ago 11 12:51 apparmor.d
drwxr-xr-x 5 root root 4096 ago 11 12:50 appport
-rw-r--r-- 1 root root 389 abr 18 2016 appstream.conf
drwxr-xr-x 6 root root 4096 ago 9 14:10 apt
drwxr-xr-x 2 root root 4096 abr 20 2016 at-spi2
drwxr-xr-x 3 root root 4096 abr 20 2016 avahi
-rw-r--r-- 1 root root 2188 ago 31 2015 bash.bashrc
-rw-r--r-- 1 root root 45 ago 12 2015 bash_completion
drwxr-xr-x 2 root root 4096 ago 11 12:50 bash_completion.d
-rw-r--r-- 1 root root 367 ene 27 2016 bindresvport.blacklist
drwxr-xr-x 2 root root 4096 abr 12 2016 binfmt.d
drwxr-xr-x 2 root root 4096 abr 20 2016 bluetooth
-rw-r--r-- 1 root root 33 abr 20 2016 brlapi.key
drwxr-xr-x 7 root root 4096 abr 20 2016 brltty
-rw-r--r-- 1 root root 23444 abr 11 2016 brltty.conf
drwxr-xr-x 3 root root 4096 abr 20 2016 ca-certificates
-rw-r--r-- 1 root root 7788 abr 20 2016 ca-certificates.conf
drwxr-xr-x 2 root root 4096 abr 20 2016 calendar
drwxr-s--- 2 root dip 4096 abr 20 2016 chatscripts
drwxr-xr-x 2 root root 4096 abr 20 2016 console-setup
drwxr-xr-x 2 root root 4096 abr 20 2016 cron.d
```

Figura 9

En el caso que se quiera concatenar comandos, se pueden aplicar dos opciones: Separar los comandos con el carácter `;` o con `&&`. Si se utiliza `;`, no importa si el primer comando resulta en un error, el segundo de cualquier manera se ejecutará. En caso de que se separen los comandos con `&&`, si el primer comando termina en error, el segundo no se ejecutará.

Ejemplo:

```
alumno@alumno-VirtualBox:~$ ls /noExiste && ls
ls: cannot access '/noExiste': No such file or directory
alumno@alumno-VirtualBox:~$ ls /noExiste ; ls
ls: cannot access '/noExiste': No such file or directory
Desktop Downloads Music Public Templates
Documents listaDeArchivo.txt Pictures saludo.txt Videos
alumno@alumno-VirtualBox:~$
```

Figura 10

Eliminar archivos o directorios: **rm**

El comando **rm** nos permite borrar tanto archivos como directorios completos. Las opciones que se pueden utilizar son:

-i	Preguntará si se desea borrar el archivo o no.
-f	En forma forzada borra sin importar si contiene o no archivos.
-r i -R	Borra de forma recursiva tanto archivos como directorios.

-v	Muestra qué se está borrando (verbose).
-----------	---

Si como root (es decir super usuario) ejecutamos el comando **rm -rf /** nos borrara casi todo el sistema operativo, haciendo que el sistema deje de funcionar.

Mover archivos/directorios o renombrarlos: **mv**

El comando **mv** nos permite tanto mover como renombrar archivos o directorios. Las opciones que podemos utilizar son:

-i	Preguntará si se desea borrar el archivo o no.
-f	En forma forzada borra sin importar si contiene o no archivos.
--backup	Crearé un backup del archivo antes de ser movido.
-v	Muestra qué se está borrando (verbose).

Creación de un directorio: **mkdir**

Utilizamos el comando **mkdir** cuando necesitamos crear un nuevo directorio. Las opciones que podemos utilizar son:

-p	Crearé tanto el directorio padre como los subdirectorios.
-m --mode=MODE	Permite crear un directorio/subdirectorio con permisos específicos.
-v	Muestra qué es lo que se está creando (verbose).

Ejemplo²:

```
alumno@alumno-VirtualBox:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
alumno@alumno-VirtualBox:~$ mkdir -p NuevoDirectorio/OtroDirectorio
alumno@alumno-VirtualBox:~$ ls
Desktop Documents Downloads Music NuevoDirectorio Pictures Public Templates Videos
alumno@alumno-VirtualBox:~$ tree NuevoDirectorio/
NuevoDirectorio/
├── OtroDirectorio

1 directory, 0 files
alumno@alumno-VirtualBox:~$
```

Figura 11

En el siguiente ejemplo, se crean los directorios "Archivos", "Documentos", "Fotos" e "Imágenes" dentro del directorio "Datos_Generales":

² En el ejemplo se utilizará también el comando **tree**. El mismo no forma parte de los comandos básicos de GNU/Linux y debe ser instalado aparte. El mismo muestra la estructura jerárquica del comando que indiquemos.

```
alumno@alumno-VirtualBox:~$ mkdir -p Datos_Generales/{Archivos,Documentos,Fotos,Imagenes}
alumno@alumno-VirtualBox:~$ tree Datos_Generales/
Datos_Generales/
├── Archivos
├── Documentos
├── Fotos
└── Imagenes

4 directories, 0 files
alumno@alumno-VirtualBox:~$
```

Figura 12

En el siguiente ejemplo se crean los directorios "Datos1" y "Datos2" dentro del directorio "Datos". Además, dentro de los directorios "Datos1" y "Datos2", se crea un directorio "Datos_Generales":

```
alumno@alumno-VirtualBox:~$ mkdir -p Datos/{Datos1,Datos2}/Datos_Generales
alumno@alumno-VirtualBox:~$ tree Datos
Datos
├── Datos1
│   └── Datos_Generales
└── Datos2
    └── Datos_Generales

4 directories, 0 files
alumno@alumno-VirtualBox:~$
```

Figura 13

Remover un directorio: **rmdir**

Utilizamos **rmdir** para eliminar directorios que estén vacíos. La única opción que podemos utilizar con **rmdir** es **-p**, que eliminará también la carpeta superior (parent).

Borrar la pantalla

Para borrar la pantalla, utilizaremos el comando **clear**. En algunos casos es posible utilizar la combinación de teclas **ctrl+l**.

Expresiones regulares

Las expresiones regulares son generalmente cadenas de caracteres, que se utilizan para representar patrones de texto. Cada carácter en una expresión regular puede representar tanto un carácter específico como un conjunto de caracteres.

Estas expresiones son muy utilizadas, ya que permiten, por ejemplo, representar patrones para realizar búsquedas, validar datos que ingrese el usuario, etc. En este curso, utilizaremos las expresiones regulares para realizar búsquedas en archivos, eliminar archivos que comiencen con un determinado patrón, y algunos usos más.

A continuación, se muestra un cuadro con las expresiones regulares, y qué representan las mismas:

?	El elemento precedente es opcional y debe coincidir al menos una vez.
*	El elemento precedente debe coincidir cero o más veces.
{n}	El elemento precedente debe coincidir exactamente n veces.
+	El elemento precedente debe coincidir una o más veces.
{,m}	El elemento precedente es opcional y debe coincidir al menos m veces.
{n,m}	El elemento precedente debe coincidir al menos n veces, pero no más de m veces.
Pablo	Busca la cadena Pablo dentro del texto.
^	La expresión posterior debe ser el inicio de una línea.
\$	La expresión precedente debe ser el final de una línea.
[mn]	El elemento debe ser m o n
[^mn]	El elemento no debe ser ni m ni n
.	El elemento puede ser cualquier carácter.
\	Carácter de escape. Por ejemplo, el símbolo . indica cualquier carácter, si aplicamos \. Hacemos referencia al carácter .
[a-z]	Representa cualquier carácter de la a a la z minúscula.
\t	Representa el carácter TAB .
\r	Representa el carácter "retorno de carro".
\n	Representa el carácter "nueva línea".
\a	Representa una "campana" o "beep". Carácter que produce un sonido cuando se imprime.
\e	Representa la tecla escape .
\f	Representa un salto de página.
\v	Representa un tabulador vertical.
\x	Se utiliza para representar caracteres ASCII o ANSI si se conoce su código. De esta forma, si se busca el símbolo de derechos de autos y la fuente en la que se busca utiliza el conjunto de caracteres Latin-1, es posible encontrarlo utilizando \xA9 .
\u	Se utiliza para representar caracteres Unicode si se conoce su código. \u00A2 representa el símbolo de centavos. No todos los motores de expresiones regulares soportan Unicode.
\d	Representa cualquier dígito del 0 al 9.
\w	Representa cualquier carácter alfanumérico.
\s	Representa un espacio en blanco.
\D	Representa cualquier carácter que no sea un dígito del 0 al 9.
\W	Representa cualquier carácter no alfanumérico.

\S	Representa cualquier carácter que no sea un espacio en blanco.
\A	Representa el inicio de la cadena. No un carácter, sino una posición.
\Z	Representa el final de la cadena. No un carácter, sino la posición.
\b	Marca el inicio y el final de una palabra.
\B	Marca la posición entre dos caracteres alfanuméricos o dos no-alfanuméricos.

Búsqueda en archivos utilizando expresiones regulares

El comando **grep** toma una expresión regular de la línea de comandos, lee la entrada estándar o una lista de archivos, e imprime las líneas que contengan coincidencias para la expresión regular.

Las opciones disponibles del comando **grep** son:

-c	Modificar la salida normal del programa. En lugar de imprimir por salida estándar las líneas coincidentes, imprime la cantidad de líneas que coincidieron en cada archivo.
-e PATRÓN	Usar PATRÓN como patrón de búsqueda.
-f ARCHIVO	Obtener los patrones del ARCHIVO .
-H	Imprime el nombre del archivo con cada coincidencia.
-r	Busca recursivamente dentro de todos los subdirectorios del directorio actual.
-i	Busca mayúsculas o minúsculas.
-w	Buscar la palabra completa.
-v	Buscar lo contrario.
-l	Muestra solo los archivos que contengan la palabra buscada.
-c	Cuenta la cantidad de ocurrencias.
-E	Utiliza expresiones regulares extendidas. En vez de usar grep , en este caso se podría usar egrep .
-o	Muestra solamente o que queremos buscar en la línea.

Modificar archivos utilizando el comando sed

El comando **sed** permite, de una manera simple, modificar un archivo, sustituyendo líneas y reemplazándolas por otras. Para poder indicar qué líneas sustituir, se utilizan, también, expresiones regulares.

Las opciones que se pueden utilizar con el comando **sed** son:

-c	Modificar la salida normal del programa. En lugar de imprimir por salida estándar las líneas coincidentes, imprime la cantidad de líneas que coincidieron en cada archivo.
-----------	--

-n --quiet --silence	Suprime la muestra automática del espacio de patrones.
-e order --expresion=order	Agrega <i>order</i> a la lista de órdenes a ejecutar.
-f archivo --file=archivo	Agrega el contenido del fichero <i>archivo</i> a la lista de órdenes a ejecutar.
--posix	Desactiva todas las extensiones de GNU
-r --regex-extended	Utiliza expresiones regulares extendidas en la expresión.
-s --separate	Considera los archivos como separados en lugar de un solo flujo continuo.
-u --unbuffered	Carga cantidades mínimas de datos de los archivos de entrada y vacía los almacenamientos temporales de salida con más frecuencia.
--help	
--version	

Ejemplo. Comando *grep* para obtener mails de un archivo.

1. Buscamos un comienzo de palabra **\b** que contenga uno o más caracteres **+** dentro del conjunto **A-Z, 0-9, ., _ , %** y **-**, que son los caracteres admitidos para un nombre de usuario. Si unimos todas estas condiciones, la expresión regular que obtenemos es: **\b[A-Z0-9._%-]+**
2. En una dirección de email, luego del nombre de usuario vendrá el carácter **@**, por lo que nuestra expresión regular tomará la siguiente forma: **\b[A-Z0-9._%-]+@**
3. Luego del carácter **@**, tendremos el dominio. El dominio se forma por uno o más caracteres de la **A-Z, 0-9, .** y **-**. Por lo tanto nuestra expresión ahora será **\b[A-Z0-9._%-]+@[A-Z0-9.-]+**
4. Por último, luego del dominio tendremos un **punto (.)**, seguido por **2, 3 o 4 caracteres de la A a la Z**. Esto último, además, debe ser el **final de una palabra**. La expresión final será: **\b[A-Z0-9._%-]+@[A-Z0-9.-]+\.[A-Z]{2,4}**

Para el ejemplo, crearemos un archivo llamado mail.txt y le agregaremos dos líneas. En cada una de ellas habrá una dirección de mail.

```
alumno@alumno-VirtualBox:~$ echo "El mail de Paul es paul@hotmail.com" > mail.txt
alumno@alumno-VirtualBox:~$ echo "El mail de Ringo es ringo@hotmail.com" >> mail.txt
alumno@alumno-VirtualBox:~$ egrep -oi '\b[A-Z0-9._%-]+@[A-Z0-9.-]+\.[A-Z]{2,4}' mail.txt
paul@hotmail.com
ringo@hotmail.com
alumno@alumno-VirtualBox:~$
```


Tanto el comando *sed* como el comando *grep*, pueden utilizarse sobre la salida de otro comando. Para poder hacer eso, debemos utilizar el carácter `|` y redirigir la salida del otro comando hacia *grep* o *sed*. Por ejemplo, podríamos utilizar el comando *echo* para imprimir un número, y esa salida reemplazarla por un string distinto.

```
alumno@alumno-VirtualBox:~$ echo "123" | sed 'y/[123]/[456]/'  
456
```