## Software Libre: GNU y la licencia GPL.

**GNU** es un proyecto cuyo principal resultado es (además de muchos programas de excelente calidad), haber elaborado una **licencia** (una especie de *contrato*) entre quien crea y quien usa un programa.
La licencia de la que hablamos se llama **GPL** (Licencia Pública General); esta licencia da a los usuarios **cuatro libertades**:
**Libertad 0:** Ejecutar el programa con cualquier propósito (privado, educativo, comercial, etc.)
**Libertad 1:** Estudiar y modificar el programa.
**Libertad 2:** Copiar el programa y compartirlo con cualquiera, gratis o a cambio de dinero.
**Libertad 3:** Mejorar el programa y publicar las mejoras.
Para que haya posibilidad real de ejercer las libertades 1 y 3, el usuario debe tener acceso al **código fuente**, es decir, a las instrucciones tal cual las escribieron los programadores. Es imposible ejercer estos derechos si solamente nos entregan el programa ejecutable compilado (codigo objeto), pero nos ocultan el código fuente.

Logo de GNU

A la vez, se les impone a los usuarios una **obligación:** toda modificación o mejora de un programa con licencia GPL, debe publicarse también bajo licencia GPL. Esta restricción se conoce como **copyleft**. Existe Software Libre con y sin copyleft.
El software que cumple con esas cuatro libertades es conocido como **software libre.** El resto del software se denomina **no libre** o **privativo.**
Estas definiciones no involucran al precio: hay software libre que se obtiene a cambio de dinero, y software no libre que es gratuito.

## El sistema operativo Linux

En 1991, un estudiante de Finlandia, Linus Torvalds, escribió el núcleo de un Sistema Operativo, que luego sería conocido como **Linux**. Torvalds decidió publicarlo **bajo licencia GPL**, lo que permitió que muchos programadores de todo el mundo se interesaran en el proyecto, ampliándolo y mejorándolo. Este fue el motivo de su enorme crecimiento en tan poco tiempo. El núcleo Linux, con una gran cantidad de programas del proyecto GNU, dio como resultado el Sistema Operativo GNU/Linux.
Surgieron las ***distribuciones***: CDs o DVDs que contenían Linux y muchos otros programas GNU que cubren casi todas las necesidades del usuario.
**Ejemplos de distribuciones**: SUSE, Red Hat, Fedora, Debian, Huayra, Ubuntu, Mint, etc.

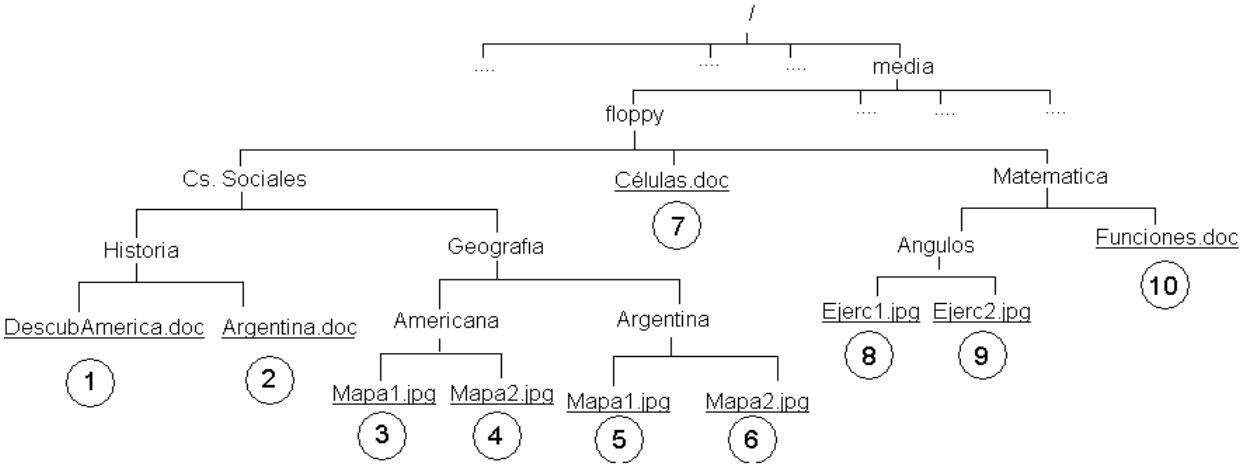
"Tux", Logo de Linux

## El sistema de archivos de GNU/Linux

El sistema de archivos de Linux se caracteriza por representar todos los archivos en un **único árbol**. La **raíz** del árbol es el disco en donde está instalado el sistema, y se representa con el símbolo "**/**". Los otros dispositivos (CDs, DVDs, pendrives, etc.) **se *montan*** en una carpeta del árbol.
**Ejemplo:**
Un disquete se ha montado en la carpeta `/media/floppy`. Esta carpeta se llama ***punto de montaje***.
Hay dos formas de referirse a un archivo. Una de ellas es especificando su ***ruta absoluta***. Para el archivo que tiene el número 4 sería:

```
/media/floppy/CsSociales/Geografia/Americana/Mapa2.jpg
```

Otra forma es especificar la ***ruta relativa***. Por ejemplo, si en este momento me encuentro dentro de la carpeta Geografía, la ruta relativa al archivo que tiene el número 4 sería: `Americana/Mapa2.jpg`




El **símbolo "`..`"** representa a la carpeta que contiene a la carpeta actual (la que está un nivel más arriba en el árbol). Por ejemplo, si estamos en la carpeta Geografia, el símbolo ".." representa a la carpeta CsSociales. Este símbolo puede utilizarse en rutas relativas. Por ejemplo, si estamos en la carpeta "Geografia" y queremos referirnos al archivo que tiene el número 1:

`../Historia/DescubAmerica.doc`

**IMPORTANTE: Recordar que Linux distingue mayúsculas y minúsculas.**

## *Otros símbolos:*

| | |
|---|---|
| . | Carpeta actual. |
| **~** | Carpeta "home" del usuario actual (explicado más abajo). |
| ? | "Comodín" que representa a un solo carácter. |
| * | "Comodín" que representa a cero o más caracteres. |

**Ej:** Si tengo 5 archivos: `arch1.txt - arch2.txt - arch3.txt - archivo.txt - hola.txt`
La expresión `arch?.txt` representa a los tres primeros archivos.
La expresión `a*.txt` representa a todos los archivos que empiecen con "a" y teminen con ".txt" (los cuatro primeros archivos).
La expresión `*.txt` representa a todos los archivos que terminen con ".txt".
La expresión `*` representa a todos los archivos.

## La carpeta "home"

Linux maneja un estricto sistema de permisos con el que trabajaremos en detalle en las próximas clases. Por ahora señalaremos que, salvo que se establezca otra cosa, los usuarios sólo tienen permiso de modificar su carpeta home. Por lo general, para el usuario "fulano" esta carpeta es `/home/fulano`. Como ya se dijo, se simboliza con "**~**".

## Interfaz de usuario.

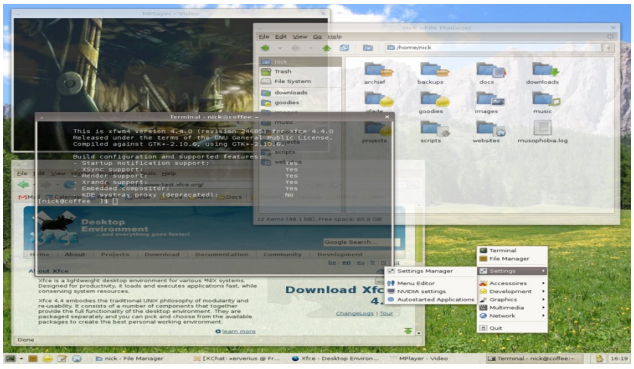
Es el software que permite que el usuario interactúe con la computadora. Las dos más importantes son:

### *Entornos de Escritorio (o interfaz gráfica).*

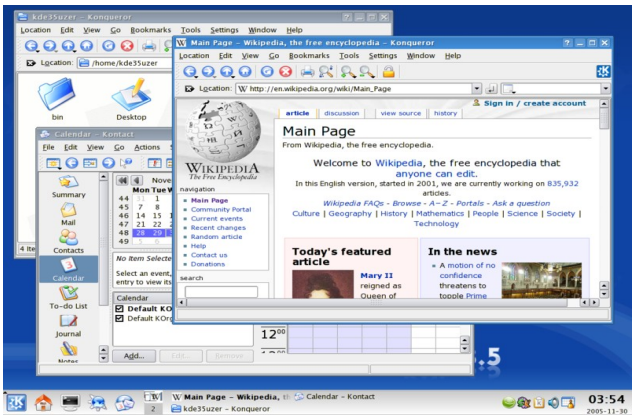
La mayoría de los Sistemas Operativos orientados al usuario final, nos presentan un entorno de Escritorio, lo que hace mucho más sencilla la utilización de la PC.
El Sistema Operativo GNU/Linux originalmente no tenía entorno de escritorio. Distintos equipos de programadores han diseñado entornos de escritorio para GNU/Linux.

Los más difundidos son: GNOME, KDE, Xfce, Unity y Lxde.

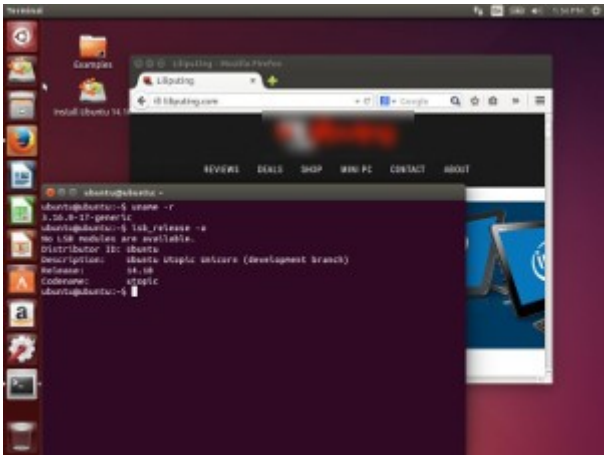
GNOME 3



Xfce



KDE



Unity

### *La línea de comandos.*

Durante el año pasado, han utilizado la *línea de comandos*. Con esta interfaz, las órdenes deben aprenderse de memoria, lo que hace mucho más difícil interactuar de este modo con la PC para la mayoría de las personas, puesto que la curva de aprendizaje es más pronunciada.

Pero los usuarios que son expertos en determinado tipo de tareas, encuentran mucho más conveniente utilizar la línea de comandos, puesto que, si bien requiere un aprendizaje mucho mayor, resulta más eficiente a largo plazo.

Además, el entorno de escritorio consume una cierta cantidad de recursos del equipo. Para ciertas tareas críticas, puede ser necesario destinar *todo* el potencial de la máquina a la tarea en cuestión, por lo que usar un entorno de escritorio sería un desperdicio.

En general, GNU/Linux prevé que las operaciones críticas puedan ser utilizadas sólo desde la línea de comandos. Esto lo vuelve un sistema un poco más difícil de usar, pero muy seguro; ya que ningún malware (virus o similar) puede realizar tareas críticas sin que nos demos cuenta.

La línea de comandos era la única interfaz de usuario en los primeros Sistemas Operativos. Hoy en día sólo se usa en casos específicos, principalmente por parte de los profesionales de IT y no tanto por los usuarios finales.

## Algunos comandos

### *ls*

Sirve para ver el contenido de una carpeta. Si se usa solo, muestra todo el contenido. Puede limitarse la lista, por ejemplo:

`ls a*` muestra una lista de todos los archivos cuyo nombre empiece con a.
`ls *.txt` muestra todos los archivos cuyo nombre termine con .txt

## *mkdir*

La orden mkdir sirve para crear carpetas (directorios). Se utiliza así:

```
mkdir nombreDeCarpeta
```

Es importante no dejar espacios en el nombre de la carpeta, una orden como

`mkdir nombre de carpeta` **[Incorrecto]**

crearía tres carpetas: una llamada "nombre", otra llamada "de" y otra llamada "carpeta". Si se quiere crear una carpeta cuyo nombre tenga espacios en blanco se deben usar las comillas:

`mkdir "nombre de carpeta"` **[Correcto]**

Este detalle del uso de comillas es válido también para los comandos que siguen.

## *cd*

Para cambiar de carpeta se utiliza la orden cd:

```
cd nombreDeCarpeta
```

Nuevamente, podemos usar la ruta absoluta o la relativa.
Ejemplo (ver el árbol de la página 2): Si estamos en la carpeta "Americana" y queremos ir a la carpeta "Matematica":

`cd ../../../Matematica` o bien `cd /media/floppy/Matematica`

## *cp*

El comando cp sirve para copiar. Se utiliza así:

```
cp origen destino
```

El origen es el archivo que se va a copiar. El destino es la carpeta donde se copiará. Tanto el origen como el destino se pueden especificar con su *ruta relativa* o *absoluta*.
**Ejemplo:** Con la figura ya presentada, supongamos que estamos en la carpeta "Americana" y que queremos copiar el archivo 4 a la carpeta "Historia". Las siguientes expresiones son equivalentes:
1) Con la ruta relativa: `cp Mapa2.jpg ../../Historia`
2) Con la ruta absoluta:

```
cp /media/floppy/Geografia/Americana/Mapa2.jpg /media/floppy/Historia
```

3) También se pueden combinar: cp Mapa2.jpg /media/floppy/Historia

El comando cp solamente copia archivos, no carpetas. Si queremos copiar una carpeta (con todo su contenido), debemos usar `cp **-r** origen destino`

## *rm*

Para eliminar archivos se utiliza la orden rm:

```
rm nombreDelArchivo
```

"nombreDelArchivo" es la ruta (absoluta o relativa) al archivo que queremos borrar.
Ejemplo:
Si estamos en la carpeta "Americana" y queremos borrar el archivo "Mapa1.jpg", deberíamos escribir:

`rm Mapa1.jpg` o bien
`rm /media/floppy/Geografia/Americana/Mapa1.jpg`

El comando rm elimina definitivamente los archivos, no los envía a la papelera; por este motivo debe utilizarse cuidadosamente.
Al igual que cp, este comando no elimina carpetas. Para eliminar una carpeta (con todo su contenido), debemos escribir `rm **-r** nombreDeLaCarpeta`

### *rmdir*

El comando rmdir elimina carpetas:

```
rmdir nombreDeCarpeta
```

De nuevo, "NombreDeCarpeta" es la ruta (absoluta o relativa) a la carpeta que queremos borrar.
La carpeta a eliminar debe estar vacía, caso contrario no se puede eliminar, y nos dará un error ("not empty", "no vacía").

### *touch*

Es un comando que sirve para crear un archivo vacío. Su sintaxis es:

```
touch nombreDelArchivo
```

### *cat*

El comando cat muestra por pantalla el contenido de un archivos:

```
cat nombreDeArchivo
```

Donde "NombreDeArchivo" es la ruta (absoluta o relativa) al archivo cuyo contenido queremos mostrar.

### *echo*

Sirve para mostrar un mensaje por pantalla. Por ejemplo:

```
echo "Buenas noches"
```

mostrará por pantalla el mensaje "Buenas noches".

## *"Entubamiento" o "pipe" ( | )*

El *entubamiento* (o *pipe*) consiste en hacer que la salida de un comando se convierta en la entrada de un segundo comando. Para ello se utiliza la barra vertical: |
Por ejemplo, si la salida del comando `who` es:

```
juan   tty8         2015-04-28 13:44 (:0)
ana    pts/2        2015-04-28 15:24 (:0.0)
pedro  pts/3        2015-04-28 15:30 (:0.0)
```

La salida del comando `who | grep juan` será

```
juan   tty8         2015-04-28 13:44 (:0)
```

puesto que la salida de `who` sirve como entrada a `grep`, que busca las líneas que contengan la cadena `juan`.
La salida del comando `who | grep an` será

```
juan   tty8         2015-04-28 13:44 (:0)
ana    pts/2        2015-04-28 15:24 (:0.0)
```

pues esas dos líneas contienen la cadena `an`. La salida del comando `who | grep an | wc -l` será

```
2
```

ya que la salida del comando `grep` es "entubada" al comando `wc -l`. Como la salida del comando grep consta de dos líneas, se muestra este número.
Nota: el comando `grep` no figura en este apunte, pero ha sido trabajado en clases de práctica.

### *tee*

Es un comando que nos permite redirigir la salida hacia un archivo y, al mismo tiempo, mostrar esa salida por pantalla. Se utiliza "entubando" la salida de otro comando para que `tee` lo reciba como entrada. Por ejemplo, si queremos que el comando `ls` se muestre por pantalla *y al mismo tiempo* se guarde en un archivo de texto:

```
ls | tee archivo.txt
```

## *Redirección de salida hacia un archivo (> y >>)*

En las próximas clases, y hasta que abordemos el uso de editores más completos, utilizaremos el comando echo, *redirigiendo la salida* hacia otros archivos. Ejemplo:

```
echo "Buenas noches" > saludo.txt
```

hará que el contenido del archivo "saludo.txt" sea "Buenas noches". Si el archivo existía, su contenido anterior se pierde. Si no existía, será creado. El comando

```
echo "Buenas noches" >> saludo.txt      (nótese el doble signo: >>)
```

**agregará** al final del archivo "saludo.txt" la línea "Buenas noches". Si el archivo existía, su contenido anterior se conserva, agregándose la nueva línea. Si no existía, será creado.

## *pwd, clear, history* y *exit*

Estos cuatro comandos realizan acciones muy específicas:

- El comando "pwd" indica la ubicación actual. Simplemente escribimos
  `pwd` y pulsamos Enter.
- El comando "clear" limpia la pantalla. Simplemente escribimos
  `clear` y pulsamos Enter.
- El comando "exit" sale del terminal. Escribimos
  `exit` y pulsamos Enter.
- El comando "history" muestra una lista de las últimos comandos que se han escrito en la terminal. Escribimos
  `history` y pulsamos Enter.

Los comandos en el historial aparecen numerados. Si, por ejemplo, queremos volver a ejecutar el comando número 150 de nuestro historial, basta con escribir `!150` y pulsar Enter.
Para recorrer el historial, se puede pulsar repetidamente la flecha hacia arriba, hasta que aparezca el comando deseado. Si lo único que queremos es ejecutar el último comando, podemos escribir `!!`
Por ejemplo, si ejecutamos el comando `ls`, y luego queremos ejecutar `ls -l`, podemos escribir `!! -l`

## *Función autocompletar*

Con la tecla de tabulador, podemos activar la función autocompletar. Por ejemplo, si escribimos `hist` y pulsamos el tabulador, autocompletará a `history`, pues no hay ningún otro comando que comience con `hist`.
Si el comando a autocompletar es ambiguo, no aparecerá nada en pantalla. Pulsando nuevamente la tecla de tabulación, nos mostrará una lista de las posibles opciones. Por ejemplo, si escribimos `mkdi` y pulsamos el tabulador dos veces, nos mostrará las dos opciones posibles: `mkdir` y `mkdiskimage`.