



Te has instalado Linux... ahora, ¿qué?

Luis Daniel Casais
@rajayonin

XXXV Jornadas Técnicas del GUL
Abril 2023
gul.uc3m.es

Tabla de contenidos

Personalización

- Desktop Enviroments
- Emuladores de terminal
- Shells
- Dotfiles

La terminal

- Funcionalidades de la terminal
- Gestión de procesos
- Shell scripts

Comandos

Editores de texto en terminal

Personalizando la terminal

Permisos

Más paquetes

Linux Subsystem for Windows

Ayuda

Secure Shell Hashing

Backups



Transparencias



github.com/rajayonin/linux-now-what

GUI (Graphical User Interface)

Por defecto, Linux cuenta con tres tipos principales de interfaces: dos gráficas, **Lockscreen** (pantalla de bloqueo) y **Desktop Enviroment**; y una de texto, la **TTY/shell**.

Se puede acceder a cualquiera de ellos con las siguientes combinaciones de teclas:

- ▶ CTRL + ALT + F1: Lockscreen
- ▶ CTRL + ALT + F2: Desktop Environment
- ▶ CTRL + ALT + F3: TTY3
- ▶ CTRL + ALT + F4: TTY4

...

Cualquiera de los tres tipos son personalizables e intercambiables, ya que **LINUX ES EL AMO Y SEÑOR DE LA PERSONALIZACIÓN.**



Desktop Enviroments

Es la principal forma (moderna) de interacción con el ordenador. Todo lo que puedes ver y tocar (iconos, ventanas, *toolbars*, *wallpapers*, etc.) es parte del Desktop Enviroment.

Son todos intercambiables y, aunque la forma de hacerlo depende de la *distro* específica, normalmente se pueden cambiar en la *lockscreen*. Algunas *distros* incluso te permiten elegir cual usar al instalar el SO.

Hay dos tipos principales, dependiendo de cómo manejan el uso de las ventanas: **Floating Window Managers** y **Tiling Window Managers**.



Floating Window Managers

La típica interfaz de un Sistema Operativo moderno, con ventanas "flotantes". Intuitivo y fácil de usar.

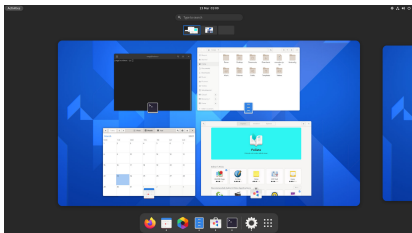


Figure: Gnome 40 Desktop



Figure: KDE Plasma Desktop

Tiling Window Managers

Máximo uso del espacio de la pantalla, 100% del tiempo (automático).
Infinito control y personalización del escritorio.
Todo con *hotkeys* (ratón pa' qué?), pero más difícil de aprender.

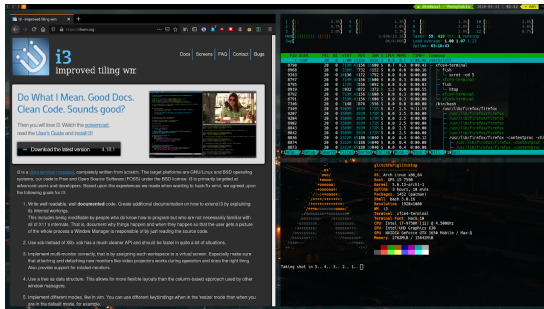
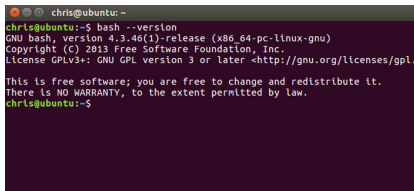


Figure: i3 Window Manager

Shells

Existen distintos programas de terminal, con distintas funcionalidades (configuración, autocompletado, plugins, etc.).

Son extremadamente fáciles de intercambiar (`chsh`), y aún más rápido de arrancar una u otra (son programas: eg. `bash`).



```
chris@ubuntu: ~  
chris@ubuntu:~$ bash --version  
GNU bash, version 4.3.46(1)-release (x86_64-pc-linux-gnu)  
Copyright (C) 2013 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.  
This is free software; you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
chris@ubuntu:~$
```

Figure: Bash



```
-zsh  
dracula-powerlevel10k main system 12:13:31 PM  
colorls --tree  
├── INSTALL.md  
├── LICENSE  
├── README.md  
├── files/  
└── screenshot.png  
dracula-powerlevel10k main system 12:13:32 PM
```

Figure: Z-Shell (con `powerlevel10k`)

Dotfiles

En Linux la mayoría de aplicaciones **guardan su configuración en archivos de texto plano**, ya sea en `/etc/` o en `/home/<user>`, y suelen llamarse `".<program>rc"`, de ahí su nombre de *"resource files"* o *"dotfiles"*.

Ésto hace que crear, modificar, y compartir configuraciones sea muy sencillo y poderoso (*scripting*, repositorios, ...).

```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
    *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000
```

Figure: `.bashrc`

La terminal

La principal funcionalidad de la shell/TTY/terminal es ejecutar programas. Los programas más genéricos y usados para interactuar con el SO se suelen llamar "comandos".

Por defecto busca los programas en `/usr/bin/`, pero también puedes especificar el archivo con su *path*¹.

Suele consistir de un *prompt*, que indica el estado, y un cursor para escribir los comandos (ejecutados con el **Enter**).

El prompt suele tener el formato `<user>@<host>:<cwd><$#>`, consistente del nombre del usuario, el nombre de la máquina (*host*), el directorio actual (*Current Working Directory*), y un caracter indicando el modo: `$` para usuario y `#` para superusuario.

¹El archivo tiene que ser ejecutable.

Sintaxis general

La sintaxis general de cualquier comando es:

```
<cmd> [-<f>] [--<flag>] [<parameters>] [<arguments>]
```

Eg.: `tar -rvf ../stuff.tar --exclude-from exclude_file .`

- **Flags:** Son opciones del comando. Dependiendo de la opción pueden implicar la presencia de un parámetro. Pueden ser escritas con una letra (`-<f>`) o una palabra (`--<flag>`), normalmente equivalentes (e.g: `-h` y `--help`). Normalmente puedes juntar varios *flags* de una letra: `-<f><g><h>`.
- **Parámetros:** Expanden las opciones del comando.
- **Argumentos:** Los *inputs* del comando. Suelen ser archivos.

Funcionalidades de la terminal (I)

La shell también trae una serie de funcionalidades extra:

- ▶ **Concatenaciones:** Sirven para concatenar comandos:

`<cmd1> <cc> <cmd2>`

- ▶ `;` ejecuta **siempre** el segundo comando cuando termina el primero (THEN).
- ▶ `&&` ejecuta el segundo comando sólo **si el primero ha funcionado** (retorna 0) (AND).
- ▶ `||` ejecuta el segundo sólo **si el primero ha fallado** (OR).

- ▶ **Pipes:** Conectan la salida de un comando (stdout) a la entrada (stdin) del siguiente comando.

Útil para enlazar comandos: `<cmd1> | <cmd2>`

- ▶ **Redirecciones:** Redirigen la entrada ó salida de un comando a un archivo: `<cmd> <rr> <file>`.

- ▶ `>` redirige la salida de un comando a un archivo (sobrescribe).
- ▶ `>>` concatena la salida de un comando a un archivo (no sobrescribe).
- ▶ `<` redirige los contenidos de un archivo a la entrada del comando.

Funcionalidades de la terminal (II)

- ▶ **Sudo:** Ejecuta el siguiente comando como *root*: `sudo <cmd>`.
- ▶ **Contraseñas:** No se muestra *nada* en pantalla (se puede cambiar en `/etc/sudoers` ²).
- ▶ **Copiar y pegar:** Se usan `Ctrl-Shift-C` y `Ctrl-Shift-V`.
- ▶ **Cancelar comandos:** Cancela el comando que estás ejecutando actualmente con `Ctrl-C`.
- ▶ **Cerrar sesión:** Cierra la sesión de terminal actual con `Ctrl-D` (útil para SSH).
- ▶ **Limpiar pantalla:** Con `Ctrl-L`, ó `clear`.
- ▶ **Borrar línea:** Borra el comando que estás escribiendo con `Ctrl-U`.
- ▶ **Autocompletado:** Con el `Tab`. Si hay más de una opción posible, tienes que darle dos veces³.

²Más info en howtogeek.com/194010.

³Como norma general, aunque depende de la shell.



Funcionalidades de la terminal (III)

La mayoría de shells cuentan con un historial de comandos, normalmente guardado en `/home/<user>/.<shell>_history` (eg. `.bash_history`)⁴.

Viene con un tamaño máximo definido en la configuración de la shell, y se guarda entre sesiones del mismo usuario.

- ▶ Puedes navegar el historial con las flechas del teclado (arriba y abajo).
- ▶ Con `Ctrl-R` puedes hacer una búsqueda dentro historial. Sigue pulsando `Ctrl-R` para seguir buscando hacia atrás, y pulsa `Esc` para salir.
- ▶ Puedes repetir el último comando con `!!` y el comando `n` veces anterior con `!-<n>`.

⁴También puedes ver el historial con `history`.

Gestión de procesos

Los procesos en Linux pueden estar en ejecutándose en *foreground*, bloqueando la terminal actual, en *background*, sin bloquearla ⁵; o en suspensión, parado y quietecito.

- ▶ Puedes lanzar un proceso en *background* con `<cmd> &`.
- ▶ Para suspender un proceso ejecutándose en *foreground*, usa `Ctrl-Z`.

Ambos retornarán un número de proceso en *background*, e.g. `[1]`.

- ▶ Para traer al *foreground* el proceso n^6 basta con usar `%<n>^7`. Para el último proceso, basta con usar `%`.
- ▶ Para evitar que un proceso termine cuando se cierra la sesión, usa `nohup <cmd>` (recomendable usar junto con `&`).
- ▶ Puedes programar la ejecución de un comando (o shell script) con `at <time> <cmd>`.

⁵Un proceso en *background* sigue pudiendo sacar mensajes por pantalla y, aunque puedas seguir usando la terminal, se vuelve difícil (prueba `ping 8.8.8.8 &`).

⁶Procesos tanto en *background* como en suspensión.

⁷Equivalente a `fg <n>`.

Shell scripts

Linux te permite crear *scripts* para terminal, muy útiles para hacer tareas repetitivas, y en general para la automatización.

Cuentan con un "lenguaje de programación" propio, con variables, funciones, sentencias de control, y comentarios; aparte de todas las funcionalidades de la terminal.

El "lenguaje" más usado es Bash Script.

- ▶ Se suelen guardar con la extensión `.sh` y en la primera línea se suele poner `#!/bin/bash`.
- ▶ Cada línea se trata como un comando y se ejecuta en la terminal actual.
- ▶ Los comentarios usan `#`.
- ▶ Es recomendable ejecutarlos con `bash <script.sh>`, aunque se pueden ejecutar el archivo directamente (`<script.sh>`⁸).

⁸Recuerda que hay que especificar el *path* completo, eg. `./<script.sh>`.

Comandos: Básicos

Los comandos básicos son:

- ▶ `ls`: Muestra los contenidos del directorio (`-lah` para mostrar toda la info).
- ▶ `cd <dir>`: Moverse al directorio `dir`.
- ▶ `mv <origin> <destination>`: Mover archivo `origin` al *path* `destination` (`-r` para carpetas).
- ▶ `cp <origin> <destination>`: Mover archivo `origin` al *path* `destination` (`-r` para carpetas).
- ▶ `rm <file>`: Eliminar el archivo `file` (`-r` para carpetas).
- ▶ `mkdir <dir>`: Crea el directorio `dir`.



Comandos: Generales¹⁰

- ▶ `who`: Muestra los usuarios *loggeados* actualmente en el *host*.
- ▶ `echo <msg>`: Imprime texto por pantalla.
- ▶ `find <dir> -iname <file>`: Busca archivos en un directorio (y subdirectorios).
- ▶ `which <cmd>`⁹: Busca el *path* donde está guardado un ejecutable.
- ▶ `script`: *Loggea* la terminal y lo guarda en un archivo (usa `exit` para parar).
- ▶ `df -h`: Muestra información del espacio usado en disco.
- ▶ `du -h <dir>`: Muestra información del espacio usado en un directorio (y subdirectorios).
- ▶ `cksum <file>`: Calcula el *checksum* de un archivo.
- ▶ `grep <str> <file>`: Busca patrones en archivos. También útil con *pipes* para buscar en *outputs* de comandos: `<cmd> | grep <str>`.

⁹whereis da más información, pero es más lento.

¹⁰Echadle un vistazo a las *coreutils*.

Comandos: Trabajando con archivos (I)

- ▶ `touch <file>`: Crea un archivo vacío.
- ▶ `cat <file>`: Imprime por pantalla los contenidos de un archivo.
- ▶ `xxd <file>`: Muestra el archivo en hexadecimal.
- ▶ `wc <file>`: Muestra el número de líneas, palabras, y bytes de un archivo.
- ▶ `less <file>`: Visor de archivos con funcionalidades extra (mejor que `cat` para archivos grandes). Pulsa `q` para salir.
- ▶ `head <file>`: Muestra las primeras líneas de un archivo (`head -n <n> <file>` para mostrar n líneas).
- ▶ `tail <file>`: Muestra las últimas líneas de un archivo (`tail -n <n> <file>` para mostrar n líneas).
- ▶ `sort <file>`: Ordena el archivo alfanuméricamente y lo saca por pantalla.
- ▶ `uniq <file>`: Elimina las líneas repetidas de un archivo y lo saca por pantalla.



Comandos: Trabajando con archivos (II)

- ▶ `tac`¹¹ `<file>`: Da la vuelta a un archivo (línea a línea) y lo saca por pantalla (rev para byte a byte).
- ▶ `diff`¹² `<file1>` `<file2>`: Muestra las diferencias entre dos archivos.
- ▶ `cmp` `<file1>` `<file2>`: Comprueba si dos archivos son iguales.
- ▶ `cut -c <n>- <file>`: Corta los *n* primeros caracteres de cada línea (`-<n>` para los *n* últimos).
- ▶ `expand` `<file>`: Convierte *tabs* a espacios.
- ▶ `unexpand` `<file>`: Convierte espacios a *tabs*.
- ▶ `sed/awk`: Permiten editar archivos desde comandos. Muy poderosos para *scripting* (especialmente `awk`).
- ▶ `jq/yq/xmlint`: Similares a `sed`, pero para formatos específicos: JSON, yaml, XML...

¹¹"tac" es "cat" al revés.

¹²`comm` es similar pero más flexible.

Comandos: Procesos y sesiones

- ▶ `ps`: Muestra los procesos activos.
- ▶ `top`¹³: Muestra información de procesos y recursos en tiempo real.
- ▶ `service <service>`¹⁴ `stop`: Para un servicio en ejecución. También puedes ver su estado (`status`) o reiniciarlo (`restart`).
- ▶ `kill <pid>`: Mata a un proceso dado su PID (Process ID) (`-9` para forzar la muerte).
- ▶ `killall <name>`: Mata todos los procesos asociados a un nombre.

- ▶ `shutdown now`: Apaga el ordenador.
- ▶ `restart now`: Reinicia el ordenador.
- ▶ `logout`: Cierra la sesión (equivalente a `Ctrl-D`).

¹³El Administrador de tareas de terminal.

¹⁴Los servicios son los scripts que están en `/etc/init.d/`.

Comandos: Network

- ▶ `ping <dir>`¹⁵: Hace *pings* continuos a la dirección indicada.
- ▶ `traceroute <dir>`: Muestra la ruta hasta la dirección indicada.
- ▶ `wget <dir>`: Obtiene los archivos de una dirección web (HTTP/HTTPS y FTP).
- ▶ `curl`: Herramienta que permite enviar y recibir *requests*, archivos, etc. Soporta muchos protocolos.
- ▶ `ifconfig`: Muestra la configuración de red. Éste comando también permite modificar la configuración.

¹⁵Puedes especificar la dirección IP (eg. 8.8.8.8) o un *hostname* (eg. google.com).

Comandos: Sistema de ficheros

- ▶ `lsblk`: Muestra los discos y particiones del sistema (`-f` para ver más info como el UUID).
- ▶ `mount <partition>16 <path>`: Monta la partición en el directorio indicado¹⁷.
- ▶ `umount <partition>`: Desmonta la partición.
- ▶ `fdisk18`: Utilidad para administrar las particiones del disco.

¹⁶Normalmente se encuentran en `/dev/`, eg. `/dev/sda`.

¹⁷Para montar de forma permanente, hay que modificar `/etc/fstab`.

¹⁸Una alternativa gráfica es `gparted`.

Comandos: Archivos comprimidos

- ▶ `zip <archive.zip> <files>`: Comprime los archivos. Usa `-r` para directorios, y `-x <files>` para excluir archivos.
- ▶ `unzip <archive.zip> [-d <path>]`: Descomprime el archivo al *path* especificado (por defecto, el actual).
- ▶ `tar -tvf <archive.tar>`: Lista los contenidos del archivo comprimido.
- ▶ `tar -cvf <archive.tar> <files>`: Usa `-r` para directorios, y `--exclude-from <exclude-file>` para excluir los archivos definidos en el *exclude file*. Para comprimir Tar Gz, usa `-z`.
- ▶ `tar -xvf <archive.tar> [-C <path>]`: Descomprime el archivo al *path* especificado (por defecto, el actual). También puedes descomprimir archivos específicos: `tar -xvf <archive.tar> <files>`.



Comandos: Otros

- ▶ `lscpu`: Muestra información de la CPU.
- ▶ `neofetch`¹⁹: Muestra información del sistema.
- ▶ `tmux`²⁰: Multiplexador de terminal. Permite tener varias "ventanas" en la misma terminal.
- ▶ `tree [-L <n>] <dir>`: Muestra un diagrama de árbol de todos los archivos y subdirectorios del directorio especificado, con nivel de profundidad máximo *n*.

¹⁹github.com/dylananraps/neofetch

²⁰github.com/tmux/tmux



Comandos: Comandos chorra

Porque no todo debe ser útil.

- ▶ `cowsay <msg>`: Muestra el mensaje como si lo dijera una vaca.
- ▶ `fortune`: Una fuente casi infinita de sabiduría.
- ▶ `sl`: Animación de un trenecito que se activa cada vez que escribes `ls` mal.
- ▶ `lolcat`: Un cat arcoíris.
- ▶ `bastet`: El Tetris bien hecho.
- ▶ `cmatrix -a`: Conviértete en hacker.
- ▶ `:(){:|:&};;::` Fork bomb.
- ▶ `rm -fr /:` Elimina el idioma francés del sistema²¹.

²¹Obviamente no hace eso.

Comandos: Comandos chorra

```
-----  
/ You have the capacity to learn from \  
\ mistakes. You'll learn a lot today. /  
-----  
  \  
  \  
    .-.-.  
    |o_o |  
    |:~/ |  
  //      \  
 (|        |)  
 /' \_    _/ \  
 \---) = (---/
```

Figure: `fortune | cowsay -f tux | lolcat`

Mejores comandos

La comunidad FOSS y Linux crea reemplazos para los comandos clásicos que mejoran sus funcionalidades.

Algunos ejemplos son trash²² (rm), locate²³ (find), htop²⁴ (top), exa²⁵ (ls), o bat²⁶ (cat).

```
1  [|||||] 16.0% Tasks: 88, 168 thr, 2 running
2  [||] 4.0% Load average: 1.17 2.31 2.42
3  [||] 1.7% Uptime: 1 day, 05:11:34
Mem[|||||] 1007/74 MB
Swp[ ] 0/0 MB

PID USER PPID PRI DIRT DTS DWR S CPU% MEM% IOW% Command
16499 ioe 20 0 10000 3390 73396 5 17.7 4.4 325.71 cinnamon --replace
16157 ioe 20 0 3807 7436 37616 5 3.9 1.0 162.78 /usr/bin/X :0 -audi
18419 ioe 20 0 2784 2316 2888 0 1.3 0.0 0:00.29 htop
17369 ioe 20 0 5819 27724 21688 5 1.3 0.4 0:13.16 gnome-terminal
17391 ioe 20 0 13639 2939 47748 5 0.7 3.9 0:11:10 simplescreenrecorde
18077 ioe 20 0 13639 2939 47748 5 0.7 3.9 2:12.44 simplescreenrecorde
18079 ioe 20 0 13639 2939 47748 5 0.7 3.9 0:02.26 simplescreenrecorde
18421 ioe 20 0 4439 22788 18048 5 0.0 0.3 0:00.25 gnome-screenshot --
18069 ioe 20 0 13639 2939 47748 5 0.0 3.9 0:05.40 simplescreenrecorde
16591 ioe 20 0 19049 3297 73396 5 0.0 4.4 0:07.82 cinnamon --replace
18078 ioe 20 0 13639 2939 47748 5 0.0 3.9 0:39.42 simplescreenrecorde
18331 ioe 20 0 19049 3297 73396 5 0.0 4.4 0:00.02 cinnamon --replace
16438 ioe 20 0 8869 40912 17292 5 0.0 0.2 0:01.20 /usr/lib/xdp_64-lin
1 ioe 20 0 1884 244 2552 5 0.0 0.1 0:01.79 /sbin/init
F1|F2|F3|F4|F5|F6|F7|F8|F9|F10|
```

Figure: htop

```
bat src/index.html

File src/index.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>a changed title</title>
6 <link rel="stylesheet" href="style.css">
7 </head>
8 <body>
9 + New lines that have been
10 + added since last commit.
11 </body>
12 </html>
```

Figure: bat

²²github.com/andreafrancia/trash-cli

²³Parte de mlocate.

²⁴htop.dev

²⁵github.com/ogham/exa

²⁶github.com/sharkdp/bat

Editores de texto en terminal

Muy útiles cuando no dispones de GUI o para ver y modificar archivos rápidamente desde la propia terminal.

Tienden a ser mucho más eficientes en el uso de recursos, y a proporcionar muchas más funcionalidades, personalización, y lenguajes de *scripting*. Van desde los más simples e intuitivos (nano o micro²⁷), a los más avanzados (vim/neovim²⁸ o emacs).

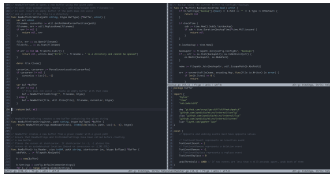


Figure: Micro

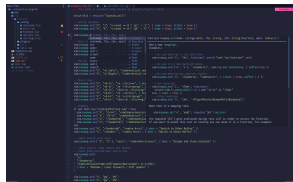


Figure: Neovim con LazyVim

²⁷ github.com/zyedidia/micro

²⁸ Recomiendo LazyVim.

Personalizando la terminal

Dependiendo de la shell, habrá más o menos configuraciones, pero las básicas son:

- ▶ **Alias:** Puedes crear alias (distintos nombres) para comandos, ya sea para acortar el nombre, o cambiarlo, o crear diferentes comandos nuevos. Ten en cuenta que un alias puede contener parámetros, o varios comandos concatenados, y pueden sobrescribirse unos a otros. Puedes crear alias permanentes añadiendo `alias <alias>="<cmd>"` al archivo de configuración de la shell (eg. `.bashrc`). Puedes ver todos los alias activos con `alias`.
Para evitar que un comando ejecute un alias, usa `\<cmd>`.
- ▶ **Prompt:** Es totalmente personalizable, normalmente se guarda en la variable `PS1` del archivo de configuración de la shell.
- ▶ **Startup:** Puedes ejecutar comandos al arrancar la terminal añadiéndolos directamente en el archivo de configuración de la shell.

Permisos (I)

Linux implementa el control de acceso a archivos y directorios mediante el sistema UGO (User, Group, Other), y cada archivo o directorio pertenece a un usuario y a un grupo.

Cada una de éstas entidades puede tener los siguientes permisos sobre el archivo/directorio:

- ▶ *r* - *read*: Permite ver el archivo en el directorio y leer sus contenidos. En el caso de directorios, permite ver sus contenidos.
- ▶ *w* - *write*: Permite modificar y eliminar el archivo. En el caso de directorios, permite crear y eliminar archivos.
- ▶ *x* - *execute*: Permite ejecutar el archivo. En el caso de directorios, permite entrar en él.

Permisos(II)

Con `ls -l` puedes ver los permisos de los archivos, entre otras cosas.

Los permisos tienen el formato `<d><rwx><rwx><rwx> <owner>
<group>`, donde el primer caracter indica si es un archivo o un directorio (d), los tres siguientes indican los permisos del propietario (U), luego los del grupo (G), y finalmente los del resto de usuarios (O).
Un guión (-) indica que ese permiso no está concedido.

Puedes añadir y quitar permisos con `chmod [<ugo>]<+-><rwx>[, ...]
<file/dir>`, eg. `chmod u+x,-rw file`, o usando el modelo octal²⁹.

²⁹Ejemplo en multacom.com/faq/password_protection/file_permissions.htm.

Más paquetes

Algunos gestores de paquetes tienen falta de paquetes o paquetes desactualizados, por lo que a veces es necesario usar alternativas:

- ▶ **Gestores de paquetes alternativos:** Suelen tener las versiones más actualizadas de los paquetes, y son agnósticos. El problema suele ser el rendimiento. Los más usados son snapd³⁰, flatpak³¹, y Nix³².
- ▶ **Appimages³³:** Ejecutables autocontenidos (.Appimage) que corren sobre cualquier Linux*, ...pero no se actualizan.
- ▶ **Compilar:** Siempre puedes compilar tus propios programas desde el código fuente, si es FOSS...

También existen *wrappers* (interfaces) para gestores de paquetes, recomendando encarecidamente Nala³⁴ para APT.

³⁰Bastante basura, pero aquí lo tienes: snapcraft.io.

³¹flatpak.org

³²nixos.org

³³appimage.org

³⁴gitlab.com/volian/nala

LSW (Linux Subsystem for Windows)⁴⁰

Aunque parezca increíble, es posible ejecutar programas de Windows en Linux:

- ▶ **Wine**³⁵: Capa de compatibilidad, permitiendo ejecutar (ciertos) programas de Windows en Linux.
- ▶ **Lutris**³⁶: Plataforma (llevada por la comunidad) para instalar y ejecutar videojuegos en Linux.
- ▶ **Proton (Steam)**³⁷: Steam trae incorporado Proton, un *fork* de Wine enfocado a videojuegos³⁸.
- ▶ **Heroic Games Launcher**³⁹: *Launcher* de Epic Games y GOG para Linux (incorpora Wine).

³⁵[winehq.org](https://www.winehq.org)

³⁶lutris.net

³⁷store.steampowered.com

³⁸Lista de juegos compatibles en protondb.com.

³⁹heroicgameslauncher.com

⁴⁰Sí, me he inventado el término.



Ayuda

Hay diferentes formas de obtener ayuda en Linux, sin tirar de Google (o DuckDuckGo):

- ▶ **Manual:** Cada programa instalado trae consigo una página del manual explicando T-O-D-O. Aparte de eso, el manual de Linux trae información sobre el propio Linux (configuración, etc.) y sobre el lenguaje de programación C.
Basta con usar `man <término>`.
- ▶ `--help`: La mayoría de comandos y programas tienen un *flag* de ayuda, donde explican brevemente el uso, la sintaxis y las distintas opciones y parámetros.
- ▶ `whatis <cmd>`: Muestra en una línea lo que hace el comando.
- ▶ **tldr pages**⁴¹: Páginas de manual más simples y hechas por la comunidad.

⁴¹tldr.sh

SSH (Secure Shell Hashing)

Permite conectarte a un ordenador a través de una terminal remota, o mover archivos, todo de forma segura.

- ▶ Para conectarse basta con usar `ssh <user>@<host>`.
- ▶ Para enviar un archivo usa `scp <file> <user>@<host>:<path>`.
- ▶ Para descargar un archivo usa `scp <user>@<host>:<file> <path>`.

Infinitamente usado para acceder a servidores⁴², ya que normalmente no cuentan con GUI.

⁴²El Laboratorio del Departamento de Informática de la UC3M proporciona Guernika:
`ssh a0<NIA-sin-el-100>@guernika.lab.inf.uc3m.es`.

Backups

Hacer copias de seguridad de datos en Linux es relativamente sencillo.

Quitando Git⁴³, Rclone⁴⁴ te permite hacerlo muy fácilmente e incluso sincronizarlo con Google Drive⁴⁵.

```
2019/12/15 10:33:05 INFO :
Transferred:    948.457k / 948.457 kBytes, 100%, 39.902 kBytes/s, ET
A 0s
Errors:         0
Checks:         0 / 0, -
Transferred:    59 / 60, 98%
Elapsed time:   23.7s
Transferring:
*              054 - Bad to Bone.odt:100% /17.339k, 1.444k
/s, 0s

2019/12/15 10:33:05 INFO : 054 - Bad to Bone.odt: Copied (new)
2019/12/15 10:33:05 INFO :
Transferred:    948.457k / 948.457 kBytes, 100%, 39.526 kBytes/s, ET
A 0s
Errors:         0
Checks:         0 / 0, -
Transferred:    60 / 60, 100%
Elapsed time:   23.9s
```

Figure: Rclone

⁴³Tutorial aquí: youtube.com/watch?v=jvsneGS00Tw.

⁴⁴rclone.org

⁴⁵howtogeek.com/451262



¿Preguntas, reclamos, improperios?



Más información

- ▶ It's FOSS
- ▶ Arch Wiki
- ▶ Stack Overflow y Stack Exchange
- ▶ Linux Handbook y Linuxize
- ▶ Tutorialspoint — Linux for Beginners
- ▶ A. Calderón — Introducción a Unix/Linux
- ▶ J. Pons — aprendolinux
- ▶ L. D. Casais — rajayonin's Vim cheatsheet
- ▶ GUL — Linux en 90' para no desesperarse en las prácticas
- ▶ GUL — Linux 404: Introducción a GNU/Linux
- ▶ GUL — Formas de instalarse Linux
- ▶ info@gul.uc3m.es — @guluc3m



Transparencias



github.com/rajayonin/linux-now-what