HFC

INTRODUCIÓN A UNIX

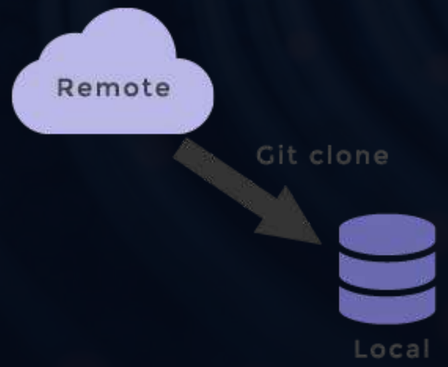
DANNA MÁRQUEZ
FERNANDO ROMERO

INSTALACIONES SENCILLAS DE GITHUB

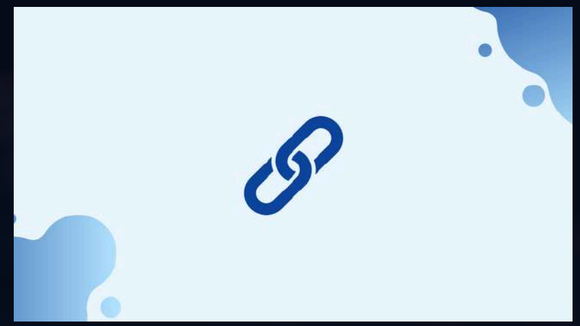
En ciertas ocasiones, disponemos de herramientas alojadas en repositorios remotos relativamente sencillas de configurar si seguimos un procedimiento adecuado, ordenado y pensando en no alterar el orden del sistema.

¿Y cuál es ese procedimiento?

- Lo primero siempre es **clonar el repositorio**.
- Realizar las **configuraciones** que se requieran.
- Crear un **enlace simbólico** del programa principal en algún directorio válido.

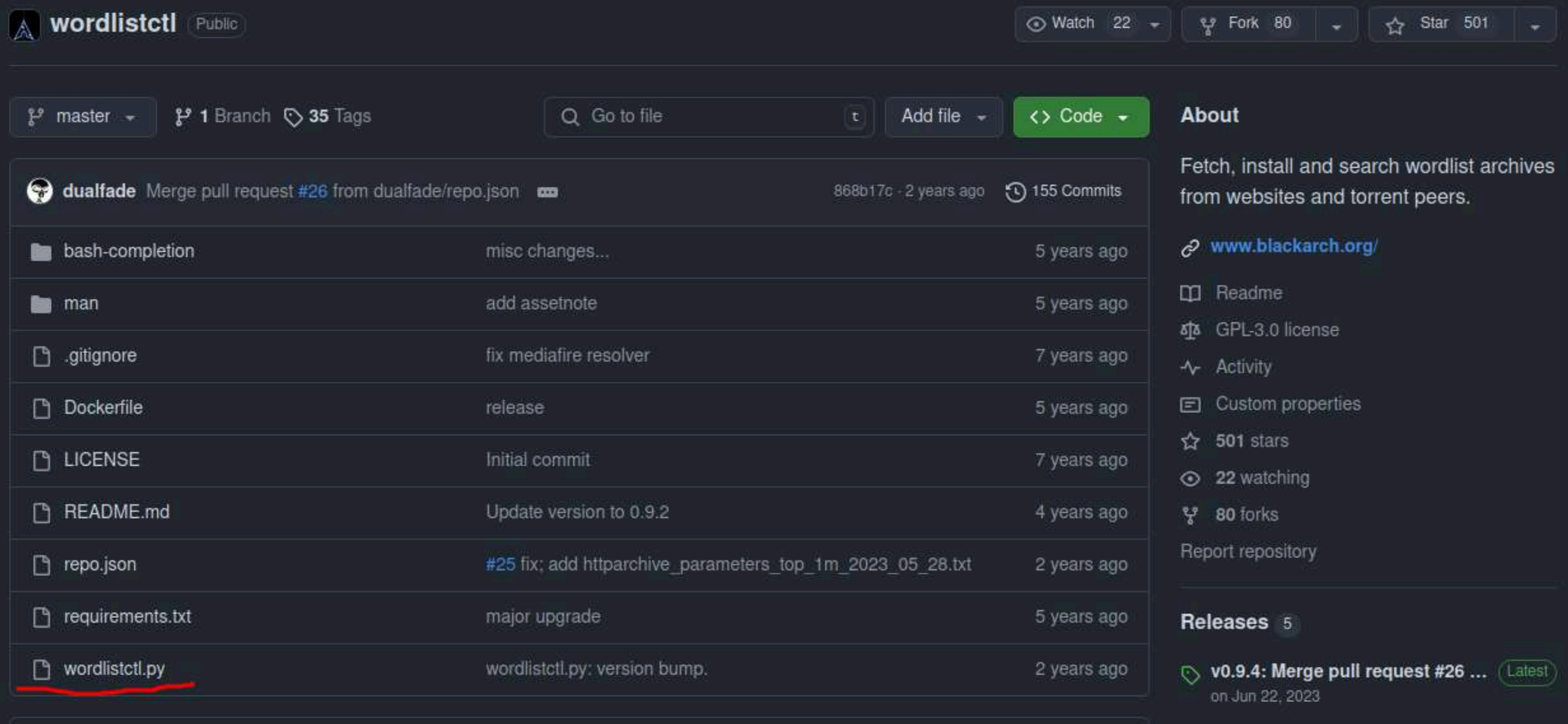


sudo ln -sf /ruta/archivo /ruta/enlace



WORDLISTCTL

Wordlistctl es un *Script* de *Python* desarrollado originalmente para la distribución *BlackArch* que nos permite la descarga y gestión de **diccionarios** para su uso en cualquier procedimiento relacionado a la fuerza bruta.



INSTALACIONES DE FUENTE

Cuando la situación es desesperada, nuestra última opción es la **compilación del código fuente**, una de las formas más engorrosas de instalar alguna utilidad, pues debemos preocuparnos nosotros mismos por las dependencias, librerías y demás requisitos de compilación de forma **manual**.

¿Cuál es ese procedimiento?

- En este caso, iniciamos por identificar las dependencias o librerías requeridas y las instalamos.
- Posteriormente clonamos o descargamos el código fuente
- Con todo en regla, iniciamos la **configuración**, **compilación** e **instalación** del software.
- Verificamos la integridad de la instalación

En este tipo de instalaciones la **lectura de la documentación** es clave, pues es un proceso poco estable.

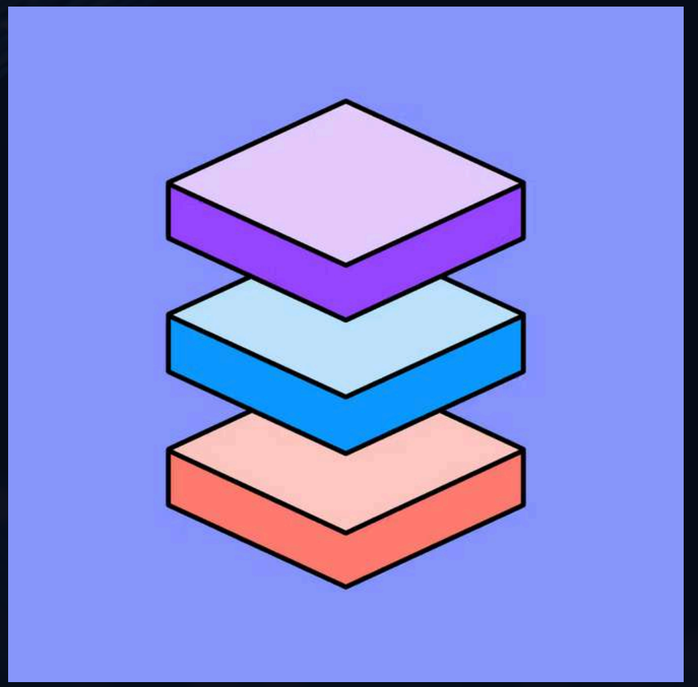
Comandos

- ./configure
- make
- sudo make install

CMATRIX

JERARQUÍA DE INSTALACIONES

- **Instalación del repositorio (*APT*, *DNF*, PACMAN)**
- **Instalación de algún paquete alternativo (*pipx*, *flatkpak*, appimage, *snap*)**
- **Instalación por un *script***
- **Instalación sencilla de *git***
- **Instalación de fuente**



PROCESOS

Un proceso es un **programa en ejecución**. Cuando ejecutas una aplicación en Linux, el sistema crea un proceso asociado que administra los recursos necesarios para su ejecución.
Cada proceso tiene un ID de proceso (PID) único y puede estar en diferentes estados (ejecución, suspendido, detenido, etc.).

Nos ayuda a:

- Gestión eficiente de recursos: Linux asigna CPU, memoria y otros recursos a los procesos de manera eficiente.
- Permiten la multitarea: Varios procesos pueden ejecutarse simultáneamente.
- Facilitan la estabilidad del sistema: Los procesos están aislados, evitando que un fallo afecte todo el sistema.
- Ejecutan servicios esenciales: Como la red, la gestión de archivos y la seguridad del sistema.

PROCESOS

Tipo de proceso	Descripción	Ejemplo
Interactivo	Iniciado por el usuario, necesita entrada manua	nano, firefox, htop
Daemon	Se ejecuta sin interacción de usuario en segundo plano	sshd, cron, ngix
Hijo	Creado a pairtir de otro proceso	bash iniciando ls
Zombie	Proceso finalizado, pero aun listado	
Huerfano	Proceso padre ha terminado antes de él	

PROCESOS

- **ps:** El comando ps (Process Status) muestra información sobre los procesos en ejecución.
  - ps-aux:
    - a → Muestra procesos de todos los usuarios.
    - u → Muestra información detallada (usuario, memoria, CPU, etc.).
    - x → Muestra procesos sin terminal asociada.

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.1  0.3 163432  8476 ?        Ss   10:00   0:01 systemd
user       2345  2.0  1.2 310000 25000 pts/0    R+   10:01   0:02 firefox
```

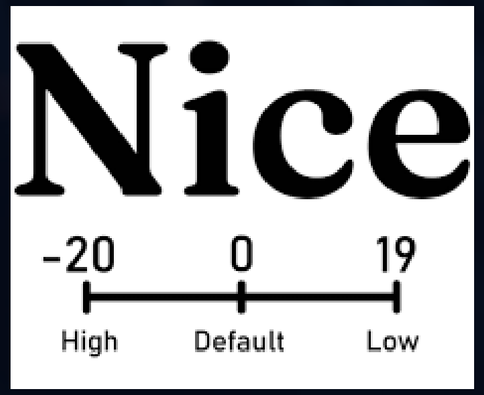
PROCESOS

Los procesos tienen una prioridad que determina cuánta CPU recibirán en comparación con otros procesos en ejecución. Los comandos **nice y renice** permiten ajustar esta prioridad.

El valor "**nice**" (también llamado niceness) indica qué tan "amable" (nice) es un proceso con respecto a los demás. Un valor más alto significa que el proceso es más "amable" y deja más CPU a otros procesos. Un valor más bajo indica que el proceso quiere más CPU.

El comando **renice** permite modificar la prioridad de un proceso que ya está ejecutándose.

top: Muestra información en vivo sobre los procesos del sistema: Es una herramienta fundamental para gestionar rendimiento, identificar procesos que consumen muchos recursos y solucionar problemas del sistema.



PROCESOS

DETENER PROCESOS:

ctrl+c: termina procesos
ctrl+z: pausa proceso
bg: reanuda en segundo plano
fg: reaunuda en primer plano.
kill: envía una señal a un proceso para detenerlo
kilall: mata todos los procesos que coincidan con un nombre específico
oña

DAEMONS

Son procesos que se ejecutan en segundo plano en un sistema Linux/UNIX y realizan tareas sin intervención del usuario.

Características principales:

- No tienen una interfaz gráfica ni interactúan directamente con el usuario.
- Se inician automáticamente al arrancar el sistema.
- Generalmente terminan con la letra d (daemon), como sshd, cron, httpd.
- Se administran con comandos como systemctl, service o init.d.

DAEMONS

- sshd (Secure Shell Daemon): Permite conectarse de manera segura a un servidor a través del protocolo SSH (Secure Shell).
- cron (Planificación de Tareas): Permite ejecutar comandos o scripts en horarios específicos sin intervención humana.
- syslogd sistema: Captura y almacena logs de aplicaciones, sistemas y errores en archivos como /var/log/syslog.
- httpd: Permite que un servidor responda a peticiones web (como Apache o Nginx).
- auditd: Sirve para auditar accesos, modificaciones de archivos y acciones sospechosas en el sistema.

SYSTEMCTL

Systemctl es una utilidad incorporada en sistemas *Linux* para administrar los procesos **daemon** y servicios corriendo en segundo plano, con apoyo de instrucciones como **status**, **enable**, **disable**, **start**, **stop**, etc.

```
● lothric
    State: running
    Units: 320 loaded (incl. loaded aliases)
     Jobs: 0 queued
   Failed: 0 units
    Since: Wed 2025-02-12 14:03:05 CST; 48min ago
  systemd: 257.2-2-arch
   CGroup: /
           ├─init.scope
           │ └─1 /sbin/init
           └─system.slice
             ├─NetworkManager.service
             │ └─458 /usr/bin/NetworkManager --no-daemon
             ├─containerd.service
             │ └─550 /usr/bin/containerd
             ├─dbus-broker.service
             │ ├─455 /usr/bin/dbus-broker-launch --scope system --audit
             │ └─457 dbus-broker --log 4 --controller 9 --machine-id 63bb19f193e0428b84ac8ea6af246ce9 --max-bytes 536870912 --max-fds 4096 --max-matches 131
             ├─docker.service
             │ └─590 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
             ├─ly.service
             │ └─549 /usr/bin/ly-dm
             ├─polkit.service
             │ └─890 /usr/lib/polkit-1/polkitd --no-debug --log-level=notice
             ├─rtkit-daemon.service
```

PROTOCOLOS

Un protocolo es un conjunto de reglas que definen cómo se comunican los dispositivos en una red. Los protocolos nos permiten transferir datos de manera segura y eficiente, autenticar usuarios y cifrar información y establecer conexiones remotas.

- SSH (Secure Shell) → Acceso remoto seguro.
- OpenVPN → Conexiones seguras con VPN.
- HTTP/HTTPS → Transferencia de páginas web.
- FTP/SFTP → Transferencia de archivos.

SSH

SSH es un protocolo de comunicación entre dispositivos que permite interactuar de forma remota, mediante una terminal, con el otro dispositivo.

Es altamente útil para la **administración remota** de servidores y demás dispositivos, además soporta cifrado para una comunicación segura similar a otros protocolos como *HTTPS*, *FTPS* y demás, funcionalidad que carecía su predecesor *Telnet*.

También incluye un mecanismo de autenticación ya sea por la **contraseña** del usuario en el servidor o por un **archivo llave** (*identity file*) que pueda autenticarnos directamente, o también solicitar una contraseña.

SSH

- ssh usuario@dominio -p 1000
- ssh usuario@ip
- ssh usuario@dominio -i archivo_llave
- ssh usuario@ip -i archivo_llave

OPENVPN

OpenVPN es un protocolo y software de código abierto que permite crear redes privadas virtuales (VPNs) seguras a través de Internet. Utiliza **cifrado fuerte** para proteger los datos, lo que lo hace ideal para:

- Conectarse de manera segura a redes privadas desde cualquier parte del mundo.
- Evitar censura y restricciones geográficas.
- Proteger la privacidad y anonimato en línea.

OVERTHEWIRE

OVERTHEWIRE

Aclaraciones

- Es un servidor compartido disponible en Internet, para todos
- No tendrán permisos para escribir en $HOME
- Se les permite escribir en archivos dentro de /tmp pero no podrán listar el contenido de la carpeta
- Por cuidar la experiencia de los demás usuarios, se recomienda que los nombres de archivos temporales **no sean predecibles**.

Comandos

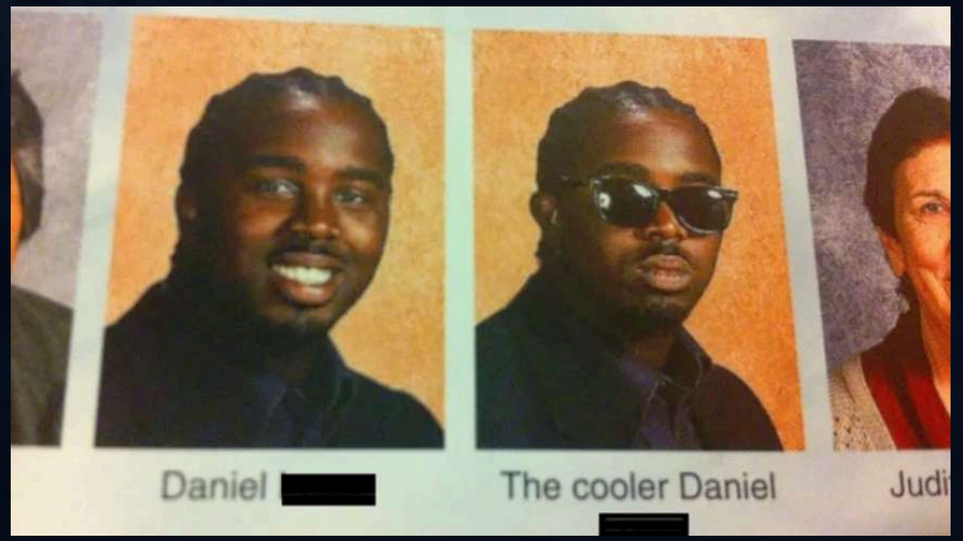
- **mktemp**: Crea un archivo temporal y devuelve la ruta absoluta
- **mktemp -d**: Crea un directorio temporal y devuelve la ruta

NC / NCAT

Tanto **nc** como **ncat** son utilidades *Linux* que permiten una interacción sencilla entre dispositivos y puertos
TCP/UDP a través de la red.

nc fue la herramienta típica durante mucho tiempo y viene ya instalada en muchas distribuciones, pero a medida que fue quedandose corta ante la tecnología moderna y los mecanismos de cifrado ampliamente utilizados, fue la utilidad **ncat** la que ofreció una solución moderna a estas carencias.

nc **ncat**



En resumidas cuentas, nos permite **abrir puertos** de nuestro dispositivo, **conectarnos a puertos** de otro dispositivo, **enviar información**, contenidos de archivos, **recibirlos**, etc.

CONTINUARÁ...