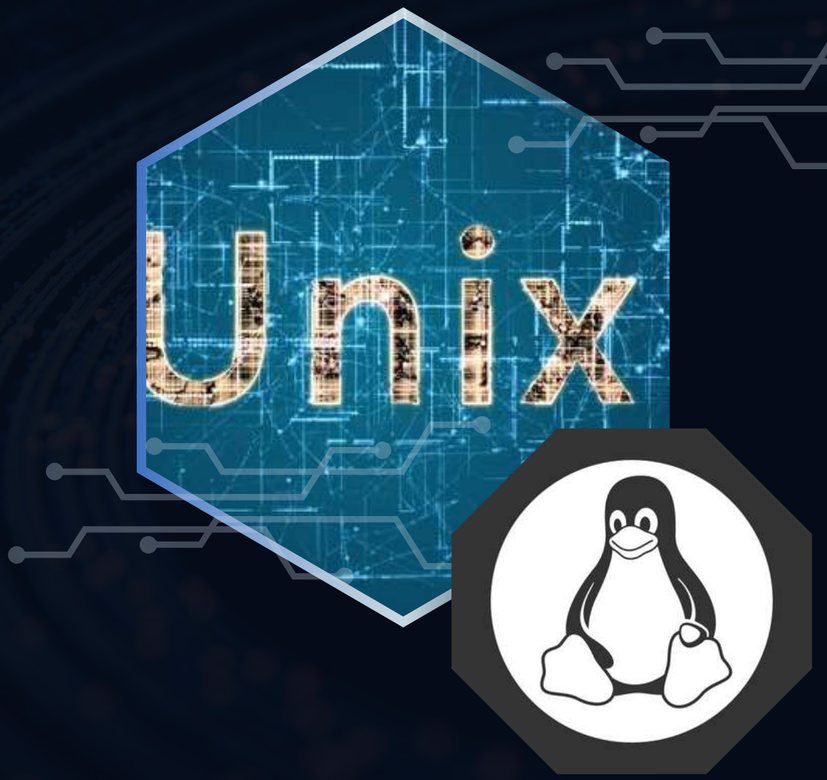
HFC

INTRODUCIÓN A UNIX

DANNA MÁRQUEZ
FERNANDO ROMERO

¿QUÉ ES UNIX?

Familia de sistemas operativos relacionados estrechamente. Sento las bases para familias de sistemas operativos modernos como Linux, BSD e incluso para un temprano macOS.

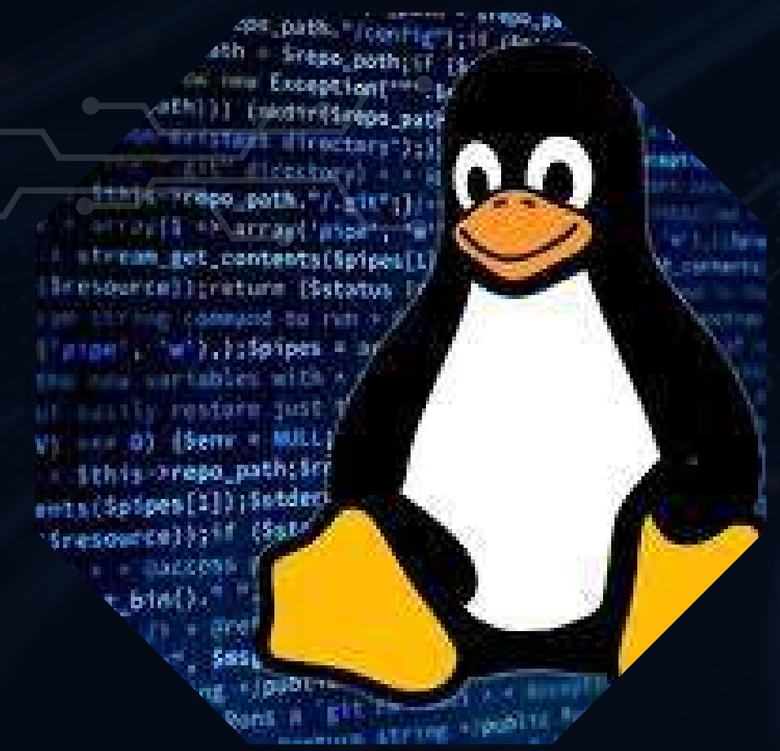


Características clave:

- **Multiusuario**: Permite que varios usuarios accedan y compartan simultáneamente a los recursos del equipo.
- **Multitarea**: Permite que un usuario ejecute más de un programa a la vez.
- **Open Source Code**: El código fuente de las principales variantes de UNIX está disponible para los usuarios y programadores.
- **Utilidades:** variedad de herramientas que trabajan en conjunto con otras que pueden aprovecharse para realizar una gran variedad de tareas.

Unix Inicialmente fue desarrollado como un solo sistema operativo por los laboratorios Bell de AT&T en el Instituto Tecnológico de Massachusetts (MIT), a partir del de *Multics*, un sistema operativo que intento innovar sin mucho éxito.

GNU LINUX



GNU/Linux es un sistema operativo de tipo Unix, formado por una colección de muchos programas, mayormente de código libre, recopiladas por *GNU*.

Da nombre al Proyecto GNU anunciado en 1983 por Richard Stallman, el cual tenía como objetivo crear un **software libre.**

Muchos usuarios de ordenadores ejecutan una versión modificada del sistema GNU cada día, sin saberlo. Debido a un particular giro en los acontecimientos, la versión de GNU que se utiliza ampliamente hoy en día es generalmente llamada «Linux», y muchos de sus usuarios no se dan cuenta de que básicamente se trata del sistema GNU, desarrollado por el proyecto GNU.

Linux es el núcleo: un programa que es parte del sistema y cuya función es asignar los recursos de la máquina a los otros programas que el usuario ejecuta.
El núcleo es una parte esencial en un sistema operativo, pero inútil por sí mismo, sólo puede funcionar en el contexto de un sistema operativo completo.

Linux se usa normalmente en combinación con el sistema operativo GNU: el sistema completo es básicamente GNU con la adición de Linux, o GNU/Linux.

¿POR QUÉ ESTUDIAMOS UNIX?

Linux hoy en día es un pilar clave en la formación de especialistas de ciberseguridad por varios aspectos como:

- Es de **código abierto** y es **transparente**, a *Linux* le interesa que lo aprendas.
- Ofrece un **mayor control** y **entendimiento** de nuestro sistema y de nuestra computadora.
- La mayoría de herramientas **educativas** y de **hacking** están disponibles en *Linux*.
- A pesar de que *Windows* domina el mercado de las computadoras personales, *Linux* sigue siendo una opción importante para los servidores y supercomputadoras tan necesarios en las organizaciones modernas.

La **mayor debilidad** y la **fortaleza** de *Linux* reside en su constante evolución debido a que se trata de un proyecto de **código libre**.

CÓDIGO LIBRE

El **código libre o abierto** actualmente es un **movimiento** y una forma de trabajo enfocada a la producción de *software*. Adopta valores y modelos **descentralizados** y **colaborativos** donde la retroalimentación, flexibilidad y autonomía son clave.

La **idea principal** es que sea accesible a todos los que deseen **ver**, **modificar**, **distribuir** y **contribuir** al código. Además, las distribuciones de éste código modificado suelen liberarse bajo este mismo concepto como código libre.

Una de las licencias más relevantes del código libre es **GPL** (*GNU General Public Licence*) que introdujo por primera vez el concepto de **copyleft**, como una protección contra futuros intentos de privatización u apropiación del *software*.

HISTORIA

1969
Ken Thompson, Dennis Ritchie y otros en los Laboratorios Bell de AT&T desarrollan el primer sistema UNIX en una PDP-7.

1971
Primera edición de UNIX lanzada, escrita en ensamblador.

1973
Cuarta edición de UNIX lanzada, reescrita en lenguaje C por Thompson y Ritchie.

1975
Versión 6 de UNIX lanzada y se convierte en la primera versión ampliamente
disponible fuera de los Laboratorios Bell.

1979
UNIX System V (Versión 7) lanzada, la última versión de la familia original UNIX de los Laboratorios Bell.

1982
AT&T comienza a vender UNIX System V. El sistema UNIX se empieza a licenciar comercialmente.
1984:

1984
Creación de la Free Software Foundation (FSF) por Richard Stallman, que influirá en el desarrollo de software libre y sistemas operativos como GNU

1991
Linus Torvalds comienza el desarrollo del kernel de sistema operativo **Linux**.

1992
GNU integra el Kernel **Linux** a su proyecto como la única pieza faltante de su sistema operativo

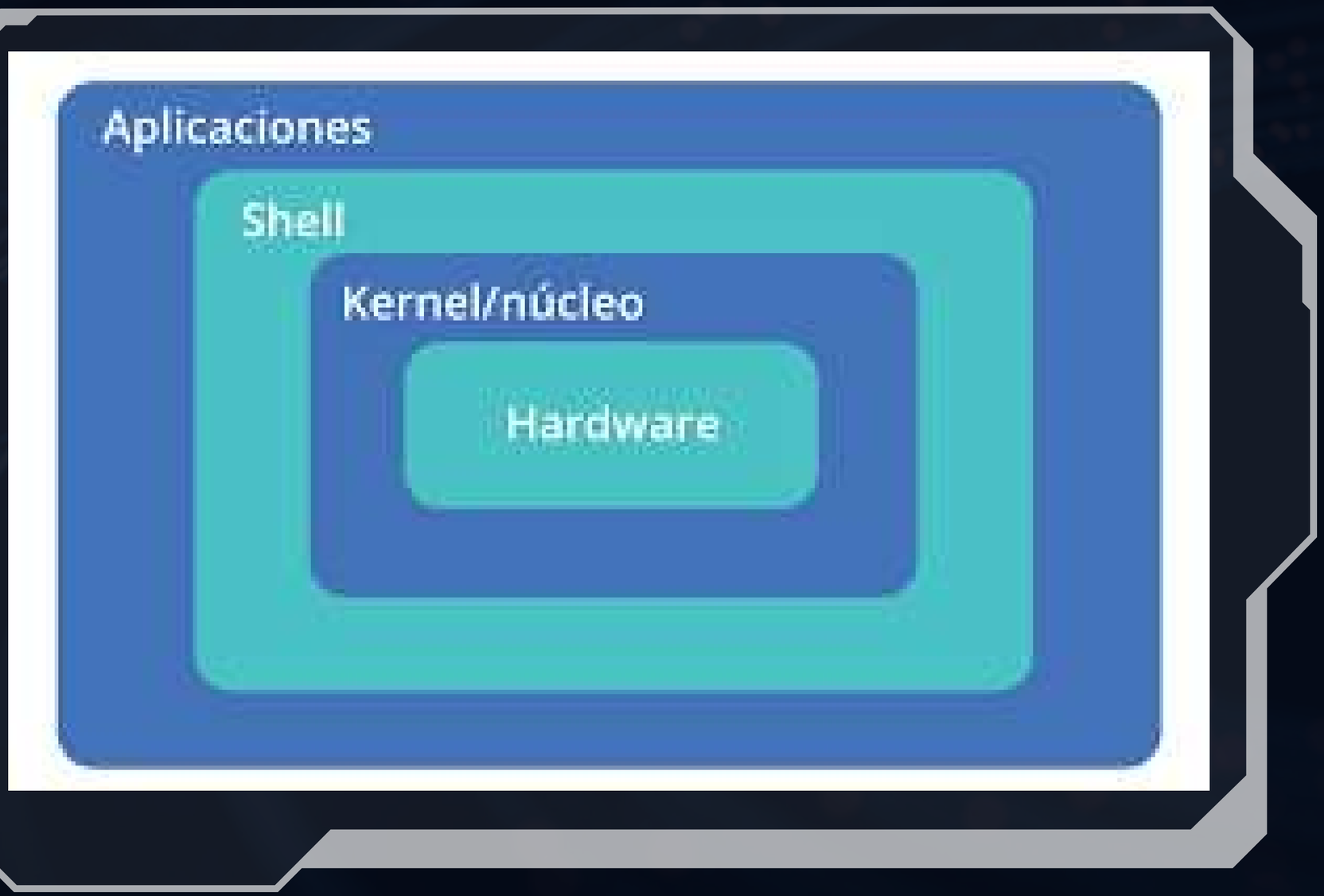
SISTEMA OPERATIVO

Es un *software* **fundamental** en una computadora con el objetivo principal de administrar tanto *hardware* como *software*.
Por una parte, actúa como un intermediario entre programas básicos y componentes del *hardware*, sentando base para todas las interacciones más complejas del sistema.
Por otra parte, se compone de todas las utilidades y funcionalidades construidos sobre este

También, es el encargado de ciertas funcionalidades básicas como:

- **Administración de procesos**
- **Gestión de memoria y almacenamiento**
- **Gestión de Periféricos**
- **Seguridad y permisos**
- **Interacción con un usuario**

CAPAS DE UN SISTEMA OPERATIVO



KERNEL

El **kernel** es el **núcleo** o **corazón** del sistema operativo, principalmente encargado de intermediar de forma directa entre el *Hardware* de la computadora y el *Software* de más bajo nivel.

De esta manera, administra y organiza las funcionalidades más básicas del sistema y sienta las bases para los procesos más complejos.

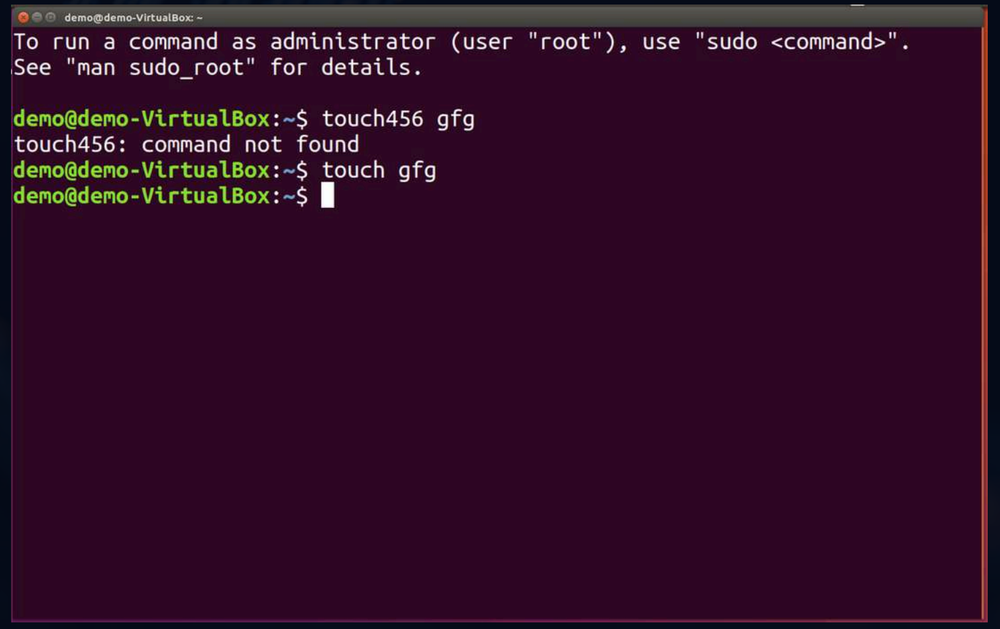
Sin el **kernel** una computadora no es más que una mezcla de programas almacenados en los distintos componentes incapaces de desempañar su función.

SHELL

Interactuar con el **Kernel**, no es precisamente sencillo, recordemos que su objetivo es administrar los recursos físicos de la computadora, por lo que ser intuitivo o amigable para un usuario pasa a segundo plano.

Para esto, se desarrollo el **Shell** que sirve como un intermediario entre procesos complejos como aplicaciones de usuario y el **Kernel**, facilitando la interacción con el sistema en general.

Pa

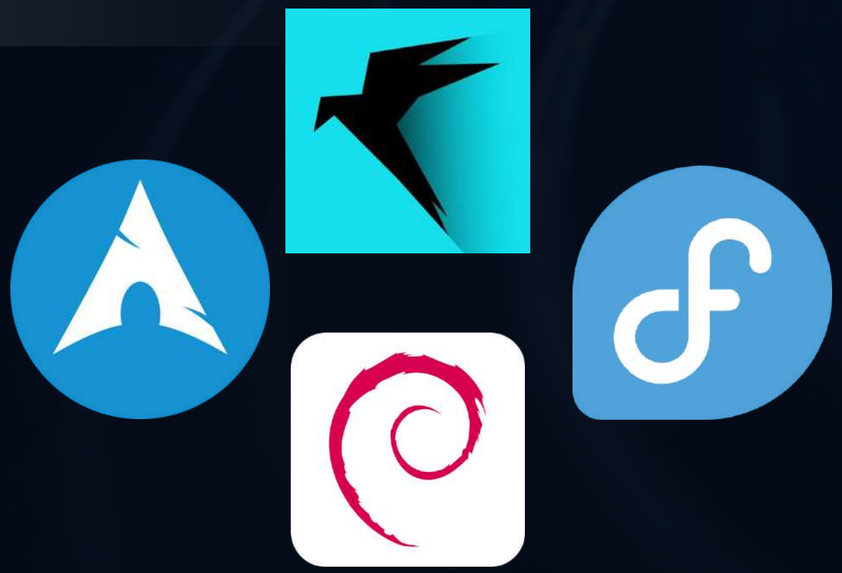


PROGRAMAS Y UTILIDADES

Son las aplicaciones y utilidades que los usuarios pueden ejecutar en el sistema. Estos programas pueden ser de varios tipos, como:

1. **Utilidades del sistema:** Relacionadas con la administración o el mantenimiento. Algunos ejemplos comunes incluyen: ls, grep, ps y top, que son comandos básicos que nos ayudan con la administración del sistema.
2. **Aplicaciones:** Son programas diseñados para que los usuarios realicen tareas específicas. Por ejemplo, los editores de texto (nano, vim, vss), navegadores web, compiladores y depuradores, entre otras.
3. **Scripts y programas personalizados:** Los usuarios pueden crear sus propios scripts o programas para automatizar tareas o ejecutar procesos personalizados.
4. **Aplicaciones multimedia.**

Estos programas permiten a los usuarios aprovechar al máximo las capacidades del sistema operativo, facilitando tanto tareas cotidianas como funciones avanzadas.



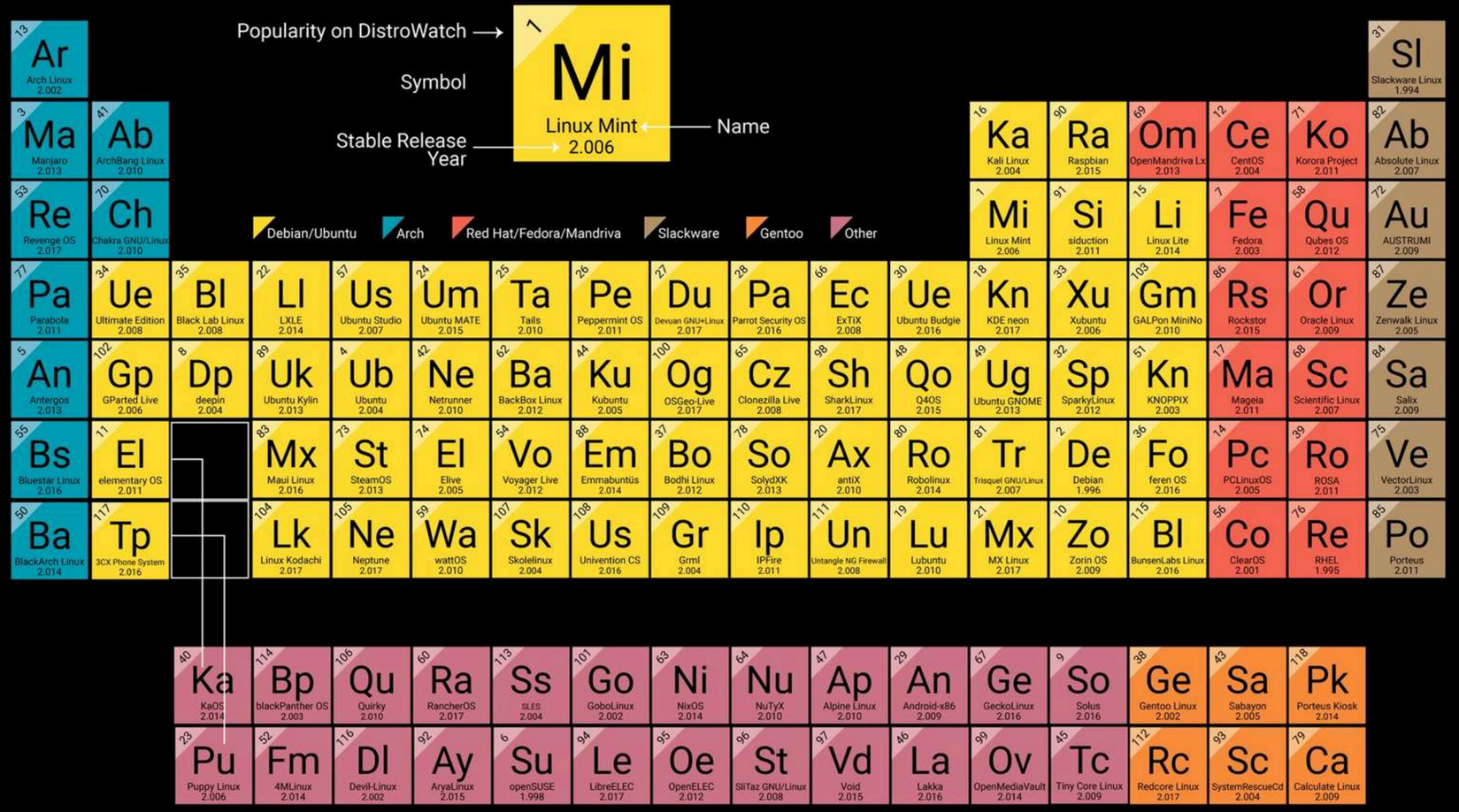
DISTROS

Una **distro** *Linux* es una **ramificación** del sistema original *GNU/Linux* que extiende el catálogo de *software* disponible, se enfoca a cierto público, necesidad o idea y posee ciertas características según la ideología que persigue.

Algunos de los ejemplos más reconocidos de distros son **Debian**, **Ubuntu**, **Fedora**, **Arch Linux**, **OpenSUSE**, etc.

Algunas buscan brindar un sistema operativo más **estable**, otros más **novedoso**, otros más **amigable**, etc.

Periodic Table of Linux Distros

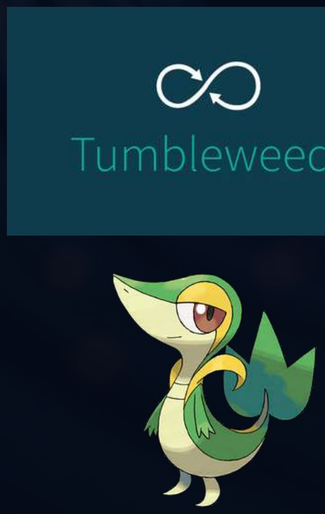
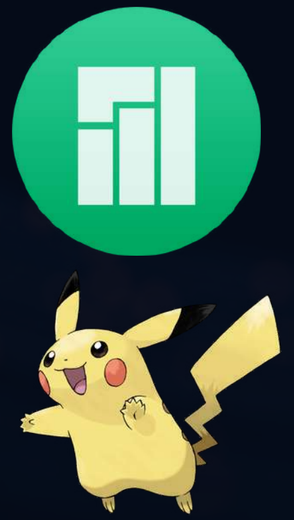


ESCOGER DISTRO







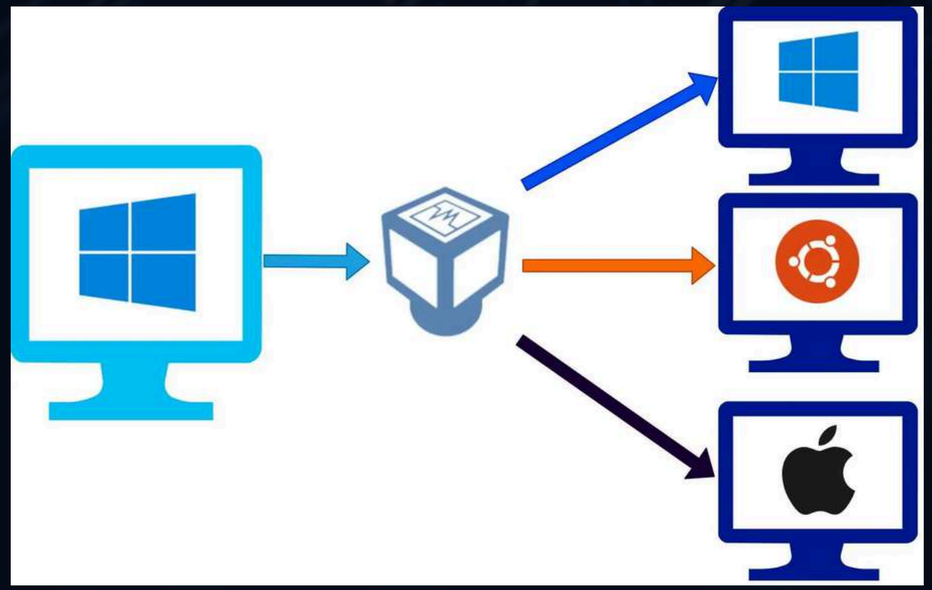





MÁQUINAS VIRTUALES

Son aplicaciones avanzadas que permiten la **emulación** de un sistema operativo completo dentro de nuestro sistema operativo nativo, a partir de componentes de *Hardware* virtuales y simulados a partir de los reales. Permiten un alto grado de configuración y personalización sobre estos componentes virtuales, para adecuar el sistema emulado a nuestras necesidades.

Tienen muchisimas aplicaciones, pero particularmente nos enfocaremos en la simulación de distros *Linux* sin arriesgar la integridad de nuestro sistema.



VMWARE PRO

GIT

Git es un sistema de control de versiones para proyectos grandes que requieren de colaboración, sincronización y orden durante todo el flujo de trabajo.

Esto es posible gracias a su sistema de **commits**, **ramas** o **branches** y a sitios de repositorios remotos como *Github* o *Gitlab*.

Algunos conceptos importantes a tener en cuanto son:

- **Stage**: Cambios aún por confirmar
- **Commits**: Cambios ya confirmados
- **Ramas**: Son formas de organizar distintos flujos de desarrollo sin que se estorben unos entre otros para, idealmente, fusionar los cambios en una **rama** principal.
- **Forks**: Ramificaciones de proyectos ya realizados

INSTALACIÓN DE LINUX

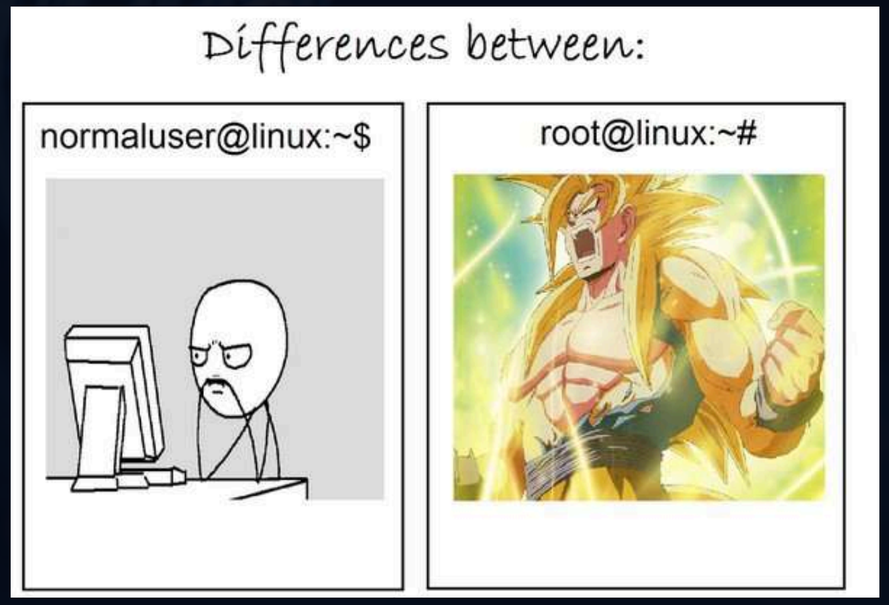
Particiones

Para el correcto funcionamiento del sistema operativo, *Linux* requiere dividir ciertas secciones del disco duro para propósitos específicos.

En un esquema de particionado convencional, se requiere una partición de arranque (*boot*), la partición raíz, y a veces de la partición *swap*.

Usuarios

Durante la instalación se requiere que definas tanto las credenciales para el superusuario del dispositivo, como las credenciales de tu cuenta propia.



INSTALACIÓN DE LINUX

Servidores mirror

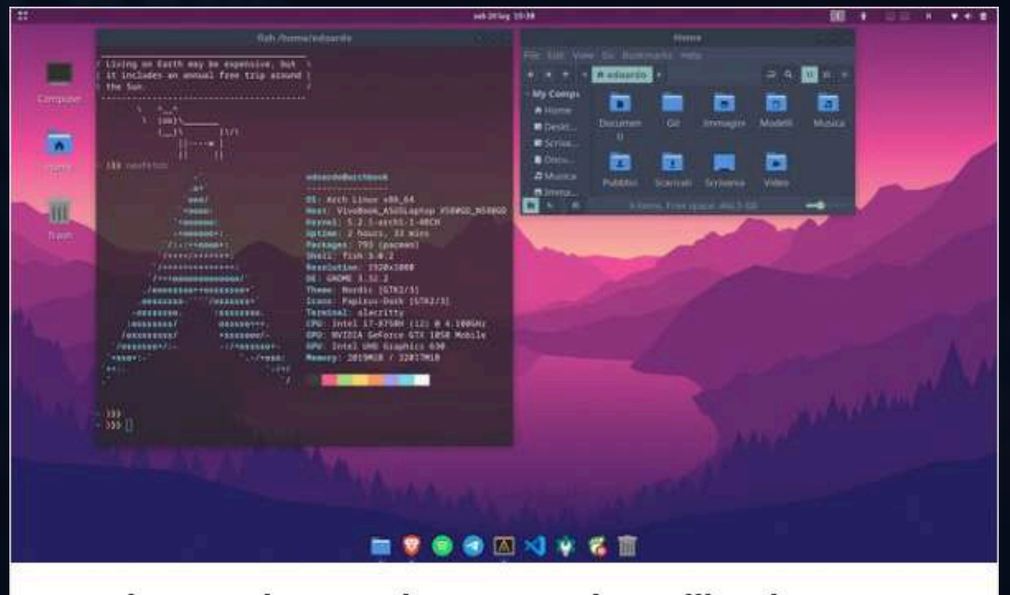
Este tipo de servidores almacenan copias de paquetes, archivos de configuración y demás datos necesarios para la correcta instalación del sistema operativo.

Con su apoyo, es posible distribuir la carga y mantener la disponibilidad de ésta información.

Entornos de escritorio

El comando instala un paquete de software desde el repositorio en el sistema. Durante la instalación, también se instalan automáticamente las dependencias necesarias.
Por ejemplo:

sudo apt install nombre-del-paquete



INSTALACIÓN

COMANDOS BÁSICOS

- **whoami**: Muestra el nombre del usuario actual.
- **hostname**: Muestra el nombre de la máquina en la red.
- **date**: Muestra la fecha y hora actuales.
- **time**: Mide cuánto tiempo tarda un comando en ejecutarse.
- **pwd**: Muestra el directorio actual (ruta absoluta).
- **echo**: Muestra texto o variables en la terminal.

- **ls**: Lista archivos y directorios en el directorio actual.
- **cd**: Cambia de directorio:
- **Ruta absoluta**: Especifica la ruta completa desde la raíz (/).
- **Ruta relativa**: Navega desde el directorio actual.

- **cat**: Muestra el contenido de un archivo en la terminal.
- **more / less**: Muestra contenido de archivos paginado (más cómodo que cat para archivos largos).

- **touch**: Crea un archivo vacío o actualiza la fecha de modificación de un archivo existente.
- **cp:** Copia archivos o directorios.
- **rm**: Elimina archivos o directorios.
- **mv**: Mueve o renombra archivos o directorios.
- **file**: Muestra el tipo de un archivo.
- **zip / unzip**: Comprime (zip) y descomprime (unzip) archivos en formato ZIP.

- **man**: Muestra el manual de usuario para un comando específico.

EJERCICIO

INSTALACIONES

¿QUÉ ES UN REPOSITORIO?

Un repositorio es un lugar (generalmente un servidor) donde se almacena software organizado en forma de **paquetes**. Estos repositorios son mantenidos por los desarrolladores de una distribución específica de Linux (como Ubuntu, Fedora o Debian), o a veces por la misma comunidad, y contienen programas, bibliotecas y demás *Software* compatible con la distro.

Los repositorios en conjunto con los **gestores de paquetes** (*APT, DNF, PACMAN, ZYPPER*) permiten que los usuarios instalen, actualicen, eliminen y gestionen software de manera sencilla utilizando la línea de comandos.

Search

La instrucción search se usa para buscar software disponible en los repositorios. Permite al usuario encontrar programas, bibliotecas u otras herramientas según palabras clave.
Por ejemplo:

```
sudo apt search nombre-del-paquete
```

Install

La intrucción install descarga un paquete de software desde el repositorio en el sistema, lo desempaqueta y lo instala en el sistema.
Durante esta instalación, también se instalan automáticamente las dependencias necesarias.
Por ejemplo:

```
sudo apt install nombre-del-paquete
```

REMOVE, UPGRADE Y PAQUETES ALTERNATIVOS

Remove

El comando remove elimina un paquete instalado del sistema. Sin embargo, este comando generalmente no elimina los archivos de configuración asociados con el paquete.
Por ejemplo:

```
sudo apt remove nombre-del-paquete
```

Para eliminar también los archivos de configuración, se puede usar:

```
sudo apt purge nombre-del-paquete
```

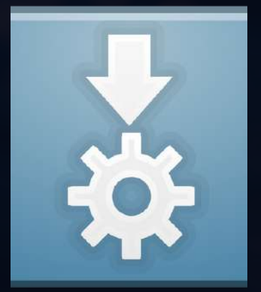
Upgrade

El comando upgrade actualiza los paquetes instalados en el sistema a sus versiones más recientes disponibles en los repositorios configurados.
Por ejemplo:

```
sudo apt upgrade
```

Paquetes alternativos

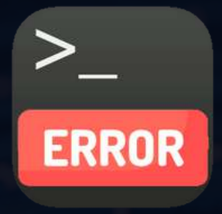
No todo el *Software* que deseamos estará disponible en los repositorios, ya sea por temas de compatibilidad, por que así lo desea el autor o por la misma filosofía de la distro.
Para ciertos casos, podemos optar por plataformas de paquetes alternativos como lo son **Flatpak**, **AppImage**, **Snap**, **Pip**, **Cargo**, etc.



PIP, PIPX PIPENV, LIBRERÍAS

Pip

Actualmente, pip se encuentra cada vez más en desuso debido a que las distribuciones *Linux* más importantes han optado por administrar las instalaciones mediante el gestor de paquetes, tanto por temas de compatibilidad como de seguridad.



Entonces, ¿qué alternativas tenemos?

En caso de requerir tanto paquetes como librerías de *Python*, tenemos una serie de alternativas aceptadas que podemos utilizar según se adecuen a nuestras necesidades.

- Paquetes **python-***, **python3-***, **etc**: Estas son las librerías ofrecidas por el gestor de paquetes del sistema, ofreciendo una instalación segura y sencilla.
- **PIPX**: Esta es la alternativa más similar al pip convencional, gestionando de manera automática un ambiente virtual, pero pensado únicamente para los programas disponibles. **No sirve para instalar librerías**.
- **PIPENV**: Esta alternativa crea un entorno virtual, que podemos inicializar y utilizar pip de manera segura dentro de este. Ofrece una forma más flexible de utilizar pip sin arriesgar nuestro sistema, además dentro del ambiente si es posible instalar las librerías requeridas.

CONTINUARÁ...