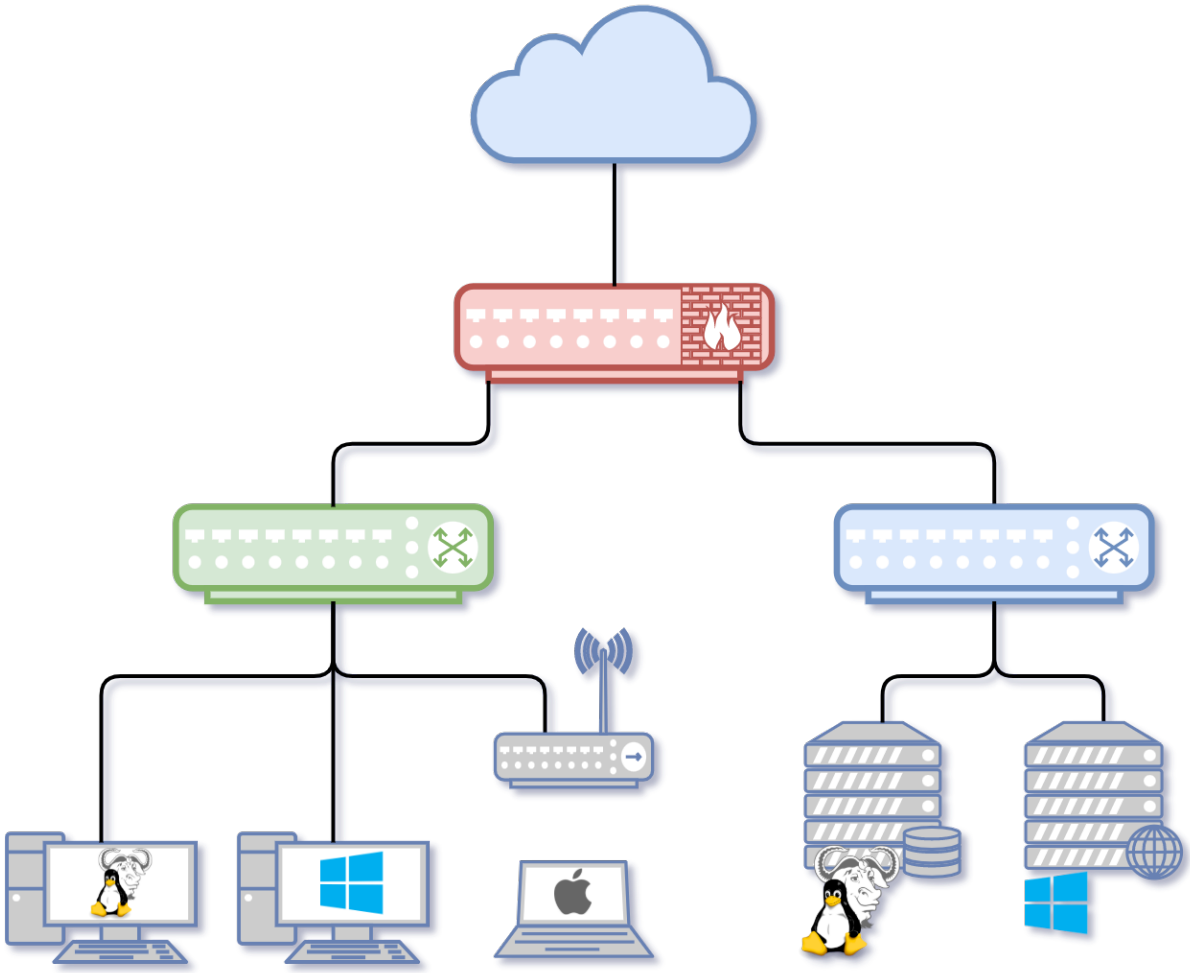
Técnico superior en Desarrollo de Aplicaciones Multiplataforma



Sistemas Informáticos

Rubén Gómez Olivencia

2023-2024

- Github: https://github.com/yuki

Puedes encontrar la última versión de este libro en formato **HTML** en el siguiente link, así como otros libros que he creado. Para descargar el código fuente en formato **Markdown** visita el repositorio en GitHub.

Información

Por favor, ponte en contacto conmigo si encuentras algún fallo, falta de ortografía o quieres mejorar de alguna manera este libro. Gracias.

I Sistemas de numeración y unidades

II Hardware y Software

Sistemas de numeración y unidades

1. Sistemas de numeración

La información que queremos utilizar en un sistema informático debe estar representada de alguna manera que el ordenador pueda entender.

En sistemas orales, o escritos, lo habitual es hacer uso de un idioma concreto mediante un alfabeto conocido. En informática se hace uso de distintos sistemas de numeración para representar tanto números como el resto de información.

1.1. Sistema decimal

El ser humano, desde hace tiempo ha utilizado como sistema para contar el sistema decimal, representado mediante el sistema arábigo. Posiblemente se adoptó este sistema por contar con 10 dedos en las manos.

El sistema numérico decimal está basado en diez símbolos ordenados (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), situados de manera ponderada (cada posición tiene un peso específico), que permiten representar las cantidades deseadas. Debido a que hacemos uso de diez símbolos se dice que utiliza la **base 10**.

Cuando se combina con otros sistemas de numeración, debemos indicar la base en la forma $\mathbf{19}_{(10}$, es decir, poniendo un pequeño "**(10**" a la derecha del número representado la base 10.

La representación de cualquier combinación del sistema decimal se puede representar en forma de potencia, donde la base es 10 y el exponente es la posición en la que se sitúa el símbolo.

Vamos a tomar como ejemplo el siguiente número: **146**. La representación en forma de potencias:

$$146 = 100 + 40 + 6$$

$$146 = 1 \times 100 + 4 \times 10 + 6 \times 1$$

$$146 = 1 \times 10^2 + 4 \times 10^1 + 6 \times 10^0$$

Como se puede comprobar, lo que hemos hecho ha sido coger cada símbolo representado y lo hemos multiplicado por la base (en este caso base 10) y a la base le hemos puesto el exponente de la posición en la que se encuentra. **El símbolo de más a la derecha tiene como exponente el cero**, y hacia la izquierda el exponente se incrementa en uno para cada posición.

1.2. Sistema binario

En informática el sistema binario es el más importante ya que es el sistema que internamente utilizan los circuitos digitales. En este sistema sólo se hace uso de dos símbolos, el "0" y el "1", y por tanto **su base es 2**. Los dos dígitos se denominan **bits** (contracción de **binary digit**).

Para representar que estamos haciendo uso del sistema binario debemos indicar la base al lado del número, por ejemplo: $\mathbf{101001}_{(2}$. Como se puede ver es añadir "**(2**" en pequeño al final del último símbolo.

1.3. Sistema hexadecimal

Esta vez necesitamos dieciséis símbolos ordenados, así que es un sistema de **base 16**. Para la representación se hace uso de los símbolos numéricos que conocemos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) y para representar los siguientes, las letras "A", "B", "C", "D", "E" y "F", de esta manera formamos los 16 símbolos que necesitamos.

Teniendo en cuenta esto, podemos hacer la representación directa de que $\mathbf{A}_{(16} = \mathbf{10}_{(10}$ y que $\mathbf{E}_{(16} = \mathbf{14}_{(10}$

En informática es muy habitual hacer uso del sistema hexadecimal a la hora de trabajar con **bytes** (que es una "palabra" de **8 bits**). Un símbolo hexadecimal se representa como 4 bits, por lo que necesitaríamos 2 símbolos hexadecimales para un byte.

También se usa durante la edición de código en formato de datos, o durante la programación en ensamblador.

Al igual que con los sistemas anteriores, debemos añadir la base cuando estemos utilizando el sistema hexadecimal: $\mathbf{F17A}_{(16}$, $\mathbf{FBE1D}_{(16}$, $\mathbf{1FAB27}_{(16}$

1.4. Sistema octal

En ordenadores antiguos era habitual hacer uso del sistema octal. Hoy día se usa más como sistema intermedio entre binario y hexadecimal.

Esta vez nos basamos en ocho símbolos ordenados (0, 1, 2, 3, 4, 5, 6, 7), que, al combinarlos, permiten representar las cantidades deseadas. Debido a que hacemos uso de ocho símbolos se dice que utiliza la **base 8**.

Para representar la base, debemos añadir "(8" a la derecha del número que hayamos indicado, como por ejemplo: $\mathbf{770}_{(8}$, $\mathbf{175}_{(8}$

1.5. Conversiones entre los distintos sistemas de numeración

Hasta ahora no nos habíamos encontrado con distintos sistemas de numeración, pero ahora que conocemos cuatro de ellos, tenemos que saber que existe la posibilidad de realizar conversiones entre ellos.

Una vez entendidos los distintos sistemas de numeración nos tiene que quedar claro que aunque la representación de los símbolos sea la misma, el número o cantidad representada no es la misma. Por ejemplo:

¡Cuidado!

$$\mathbf{1010}_{(10} \neq \mathbf{1010}_{(2} \neq \mathbf{1010}_{(16} \neq \mathbf{1010}_{(8}$$

A continuación se va a explicar cómo realizar conversiones entre los distintos sistemas de numeración que hemos visto, y a modo de resumen está la tabla de conversiones directa.

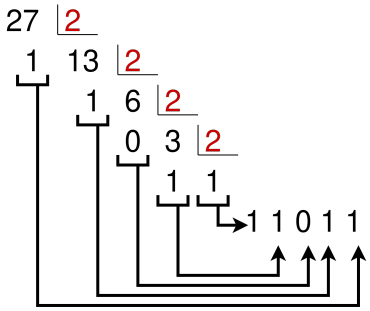
1.5.1. Conversión de decimal a...

La manera más sencilla para realizar las distintas conversiones partiendo de un número decimal es hacer divisiones sucesivas usando la base a la que queremos realizar la conversión.

... binario

Se trata de dividir sucesivamente el número decimal y los sucesivos cocientes entre dos (la base binaria).

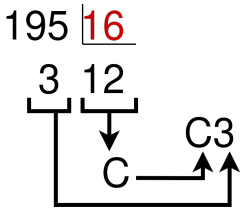
Vamos a utilizar como ejemplo el número decimal $\mathbf{27}_{(10}$:



Los restos los cogemos en orden inverso para obtener la siguiente equivalencia: $\mathbf{27}_{(10} = \mathbf{11011}_{(2}$

... hexadecimal

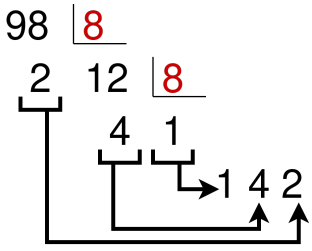
Se trata de dividir sucesivamente el número decimal y los sucesivos cocientes entre 16 (la base hexadecimal). Cuando el cociente o resto sea entre 10 y 15, habrá que cambiarlo por la letra correspondiente.



Los restos los cogemos en orden inverso para obtener la siguiente equivalencia: $\mathbf{195}_{(10} = \mathbf{C3}_{(16}$

... octal

Al igual que los anteriores, hacemos divisiones sucesivas:



Los restos los cogemos en orden inverso para obtener la siguiente equivalencia: $\mathbf{98}_{(10} = \mathbf{142}_{(8}$

1.5.2. Conversión de binario a...

... decimal

El sistema de numeración binario es un sistema posicional donde cada dígito binario (bit) tiene un valor basado en su posición relativa al **LSB** (*Least Significant Bit* = bit menos significativo, que es el que está más a la derecha y que tiene el menor valor).

Cualquier número binario puede convertirse a su equivalente decimal multiplicando cada bit por la base (2) y usando como exponente la posición (siendo 0 el exponente del bit de más a la derecha). Para ilustrarlo, cojamos como ejemplo el número binario $11011_{(2}$:

$$11011_{(2}$$

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$16 + 8 + 0 + 2 + 1 = 27_{(10}$$

Nótese que el procedimiento consiste en determinar los valores (es decir, las potencias de 2) de cada posición de bit que contenga un 1 y luego sumarlos.

Nótese también que el **MSB** (*Most Significant Bit* = bit más significativo, **el que está más a la izquierda**, el que tiene mayor valor) tiene un valor de 2^4 a pesar de que es el quinto bit. Esto se debe a que el **LSB** (*Least Significant Bit*, el bit menos significativo, el que está a la derecha) es el primer bit y tiene un valor de 2^0.

... octal

Para convertir un número binario a octal **se agrupan los dígitos de 3 en 3 empezando desde el lado derecho** hacia la izquierda, sustituyendo cada trío de dígitos binarios por su equivalente en octal.

Si en el lado izquierdo quedase algún bit "suelto" (sin formar un grupo de 3), se pueden poner "0" a la izquierda.

Cogemos como ejemplo el número binario $1100101001001_{(2}$ para pasarlo a octal, haremos:

$$001\ \ 100\ \ 101\ \ 001\ \ 001_{(2} = 14511_{(8}$$

... hexadecimal

Similar al caso anterior, pero en este caso **la agrupación que se realiza debe de ser de 4 en 4 bits**. Si usamos el mismo ejemplo anterior $1100101001001_{(2}$:

$$0001\ \ 1001\ \ 0100\ \ 1001_{(2} = 1949_{(16}$$

1.5.3. Conversión de hexadecimal a...

... binario

Para pasar de hexadecimal a binario convertiremos cada símbolo hexadecimal a **4 bits**.

$$\mathrm{F17A}_{(16} = 1111\ \ 0001\ \ 0111\ \ 1010_{(2}$$

$$\mathrm{1A4F}_{(16} = 0001\ \ 1010\ \ 0100\ \ 111_{(2}$$

... decimal

Al igual que hemos hecho con las conversiones previas a decimal, se podría realizar haciendo potencias de 16, pero se entiende que es más complicado de realizar.

Por lo tanto, **la manera más sencilla es pasar primero a binario** como acabamos de ver **y posteriormente convertir ese binario a decimal** como hemos visto previamente.

... octal

Pasar primero a binario y después a octal.

1.5.4. Conversión de octal a...

... binario

Cada dígito en octal se convierte en su representación **en 3 bits**:

$$\mathbf{167}_{(8} = \mathbf{001\ \ 110\ \ 111}_{(2}$$

$$\mathbf{253}_{(8} = \mathbf{010\ \ 101\ \ 011}_{(2}$$

Los ceros de la izquierda se podrían quitar, ya que no alteran el valor.

... decimal

Se puede realizar de dos maneras. La primera es hacer uso de potencias de 8 (similar al paso de pasar de binario a decimal, pero cambiando la base):

$$\mathbf{157}_{(8} = \mathbf{1} \times \mathbf{8}^{\mathbf{2}} + \mathbf{5} \times \mathbf{8}^{\mathbf{1}} + \mathbf{7} \times \mathbf{8}^{\mathbf{0}} =$$

$$\mathbf{1} \times \mathbf{64} + \mathbf{5} \times \mathbf{8} + \mathbf{7} \times \mathbf{1} =$$

$$\mathbf{64} + \mathbf{40} + \mathbf{7} = \mathbf{111}_{(10}$$

$$\mathbf{157}_{(8} = \mathbf{111}_{(10}$$

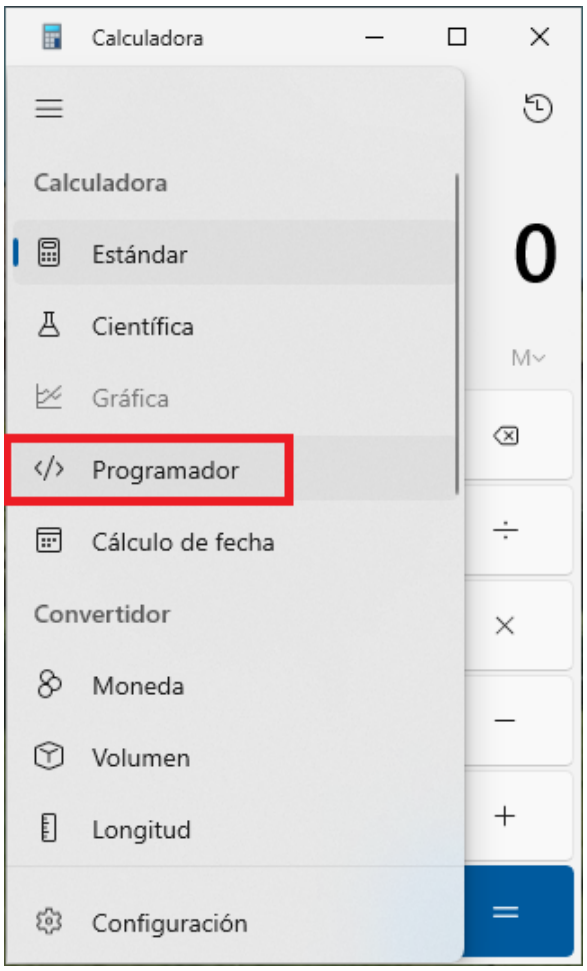
Con números grandes puede ser un poco complicado calcular las potencias de 8, por lo que **la alternativa es pasarlo primero a binario** como hemos visto, **y después pasarlo de binario a decimal**.

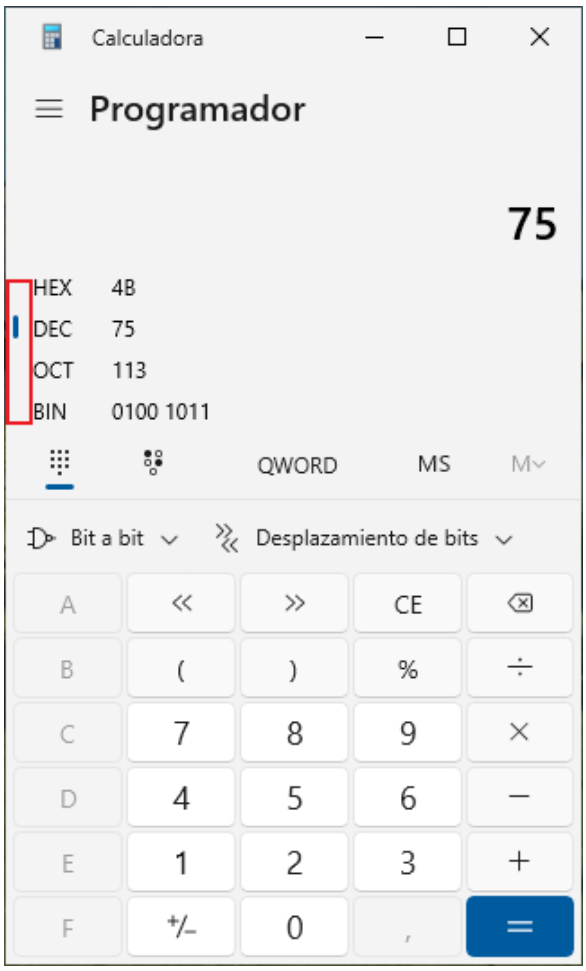
... hexadecimal

La manera más sencilla es realizar la conversión primero a binario tal como hemos visto, y posteriormente pasar el número binario a hexadecimal como se ha visto previamente.

1.6. Comprobar conversiones

Podemos hacer uso de la calculadora del sistema Windows para comprobar si estamos realizando de manera correcta las conversiones. El problema es que por defecto sólo hace uso del sistema decimal. Para poder utilizar los sistemas de numeración que hemos aprendido, es necesario utilizar la versión “Programador”.





Una vez está en modo "Programador", nos debemos fijar qué sistema de numeración está elegido. Al escribir cualquier número, en el resto de opciones veremos las conversiones automáticamente.

2. Unidades de Información digital

A la hora de almacenar información digital es importante conocer el tamaño del objeto que queremos almacenar y el espacio libre del lugar en el que lo queremos guardar.

En decimal estamos acostumbrados a contar usando el Sistema Internacional de Unidades, y dependiendo de la magnitud que queramos medir, haremos uso de unos prefijos establecidos. Por ejemplo, para medir la distancia usamos el **metro**, y por tanto nos quedaría:

- **Sin prefijo** : metro, una unidad.
- **deca**-metro: 10^1 metros
- **hecto**-metro: 10^2 metros
- **kilo**-metro: 10^3 metros
- **mega**-metro: 10^6 metros
- **giga**-metro: 10^9 metros
- ...

En decimal hacemos uso de la **base 10**, y por tanto con cada prefijo lo que estamos haciendo es modificar el exponente que indica la cantidad. Para unidades pequeñas el exponente varía de uno en uno, pero a partir de cierta cantidad (**kilo-**), la cantidad cambia multiplicando por 1000 (10^3).

2.1. Sistema binario

En informática la información se guarda en formato binario, y la unidad más pequeña es el **bit**, que es un acrónimo de ***bi**nary digi**t***. Cada bit es una única unidad que sólo permite dos estados: **0** o **1**. A la hora de representarlo se hace uso de la letra **b** minúscula, por lo tanto, **10b** son 10bits.

2.1.1. Múltiplos

Al igual que en decimal, a medida que aumentamos la cantidad, se hace uso de prefijos para facilitar el conocer de qué cantidad estamos hablando.

A continuación se expone una tabla de las medidas más utilizadas:

Nombre	Símbolo	Cantidad
Bit	**b**	$2^0 = 1$
Nibble		4b
Byte	**B**	8b
Kibibyte	**KiB**	$2^{10} = 1024$ B
Mebibyte	**MiB**	$2^{20} = 1024$ KiB
Gibibyte	**GiB**	$2^{30} = 1024$ MiB
Tebibyte	**TiB**	$2^{40} = 1024$ GiB
Pebibyte	**PiB**	$2^{50} = 1024$ TiB
Exbibyte	**EiB**	$2^{60} = 1024$ PiB

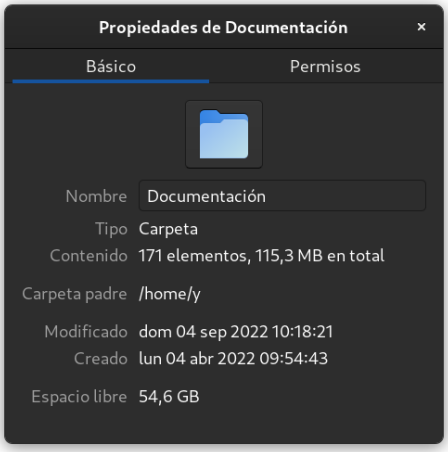
Aunque esta es la manera correcta de nombrar las unidades cuando hablamos en términos informáticos, lo habitual es que se haga uso de los prefijos del sistema decimal.

2.2. Usos

A la hora de utilizar las medidas vistas previamente hay que diferenciar qué queremos medir, ya que no se hará siempre igual.

- **Almacenamiento**: A la hora de querer expresar una cantidad de almacenamiento (para un disco

duro, un *pendrive* USB, RAM, ...) se hace uso del **Byte** y de sus múltiplos usando los prefijos vistos previamente.



- **Transmisión**: Cuando hablamos de tasa de transferencia de datos se hace uso del término "tasa de bits" (en inglés ***bitrate***), que indican el número de bits que se transmiten por unidad de tiempo. Hoy día se suele medir en kbps (o kb/s, kilobits por segundo), Mbps (Mb/s, o megabit por segundo), ... Para convertirlo a Bytes por segundo habría que dividirlo por "8".

¡Atención!

La transmisión de datos se expresa en "bits por segundo"

II

Hardware y Software

1. Sistemas Informáticos

1.1. Introducción

La **informática** es un área de la ciencia que abarca distintas disciplinas teóricas (como la creación de algoritmos, teoría de computación, teoría de la información, ...) y disciplinas prácticas (diseño de hardware, implementación de software). Normalmente llamamos informática al uso, almacenado o procesado de datos e información en formato digital.

Un **sistema informático** es aquel que nos permite almacenar y procesar datos en formato digital, para convertirlos en información. Normalmente un sistema informático lo asociamos a los ordenadores (computadoras), que contienen distintos componentes que podemos utilizar en nuestro día a día.

1.2. Breve historia de la computación/informática

Los ordenadores que estamos acostumbrados a utilizar, y que creemos que es lo que conocemos como informática, no es más que una evolución de un conjunto de ideas y avances que ha habido a lo largo de la historia como: la lógica, el álgebra, la mecánica, la electrónica, creación de materiales, ...

Es por eso que no podemos ceñirnos a la evolución de la informática como algo que ha ocurrido en las últimas décadas, si no que podemos remontarnos a varios siglos atrás. Lo que viene a continuación es un pequeño resumen de un listado más largo que aparece en la wikipedia.

1623 Primera calculadora mecánica.

1666 Se crea la primera calculadora mediante ruedas y engranajes.

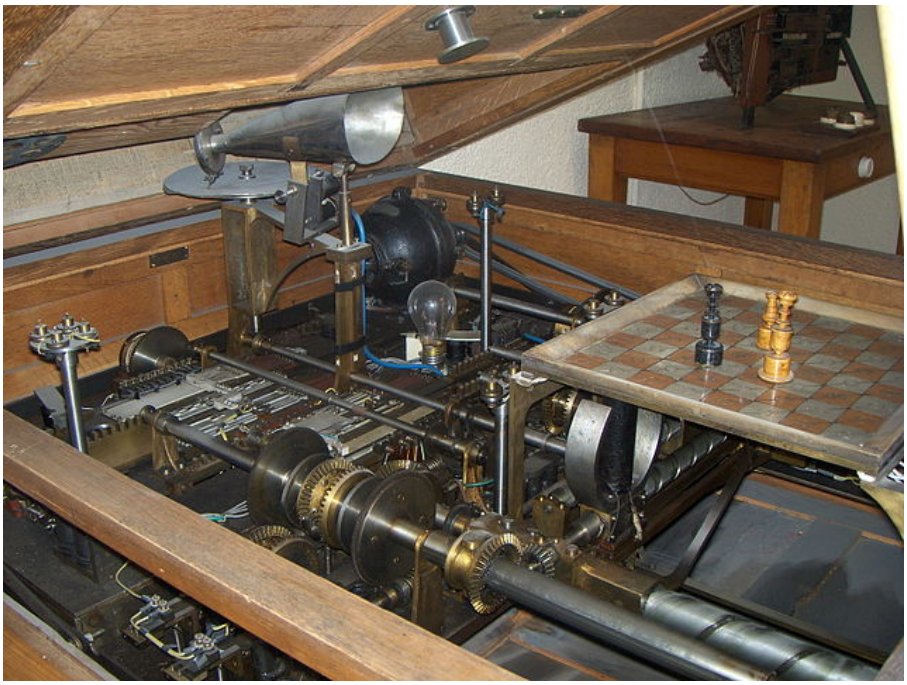
1801 Mediante el uso de tarjetas perforadas se controla el mecanismo de una máquina de tejer para realizar dibujos y diseños. Vídeo.

1837 Charles Babbage describe la máquina analítica. Es el diseño de un computador moderno de propósito general.

1843 Ada Augusta Lovelace sugirió la idea de que las tarjetas perforadas se adaptaran de manera que causaran que el motor de Babbage repitiera ciertas operaciones. Debido a esta sugerencia algunos consideran a Lady Lovelace la **primera programadora**.

1854 George Boole publica su Álgebra de Boole. A causa del desarrollo del álgebra de Boole, Boole es considerado por muchos como el **padre de la teoría de la informática**.

1912 Leonardo Torres Quevedo construye un Autómata capaz de jugar finales de ajedrez (torre y rey contra rey) que llamó "El Ajedrecista".



El ajedrecista. Fuente: Wikipedia

1919 Los inventores estadounidenses W. H. Eccles y F. W. Jordan desarrollan el primer circuito multivibrador o biestable (en léxico electrónico *flip-flop*). El flip-flop permitió diseñar Circuitos electrónicos que podían tener dos estados estables, alternativamente, pudiendo representar así el "**0**" como un estado y el otro con un "**1**". Esto formó la base del almacenamiento y proceso del **bit binario**, estructura que utilizan las actuales computadoras.

1924 Walther Bothe construye una puerta lógica **AND** para usarla en experimentos físicos, por lo cual recibió el premio Nobel de física en 1954.

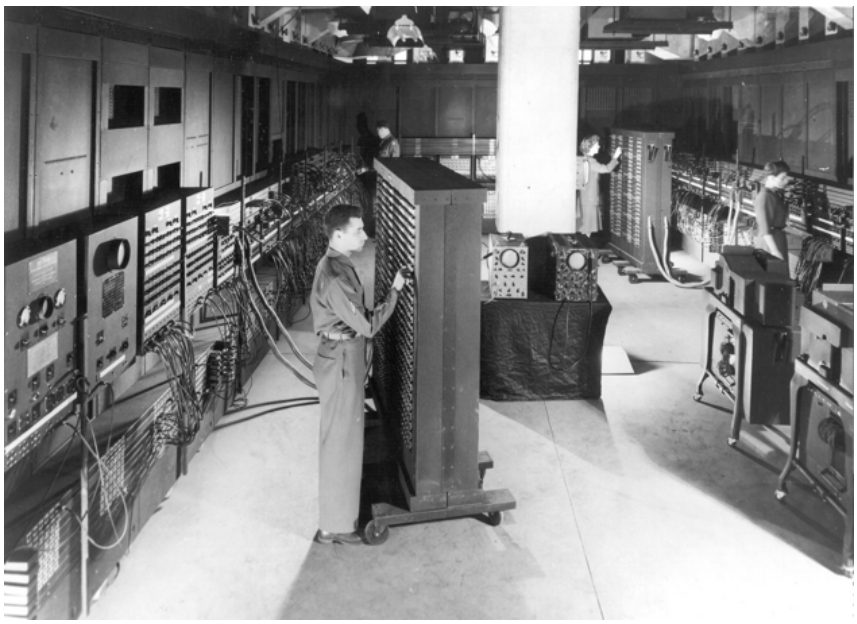
1936 Alan Turing describe la máquina de Turing, la cual formaliza el concepto de algoritmo.

1938 Konrad Zuse completa la primera computadora electromecánica, aunque no 100 % operativa, la **Z1**.

1944 En Inglaterra se construyeron los ordenadores Colossus (Colossus Mark I y Colossus Mark 2), con el objetivo de descifrar las comunicaciones de los alemanes durante la Segunda guerra mundial.

1945 John Von Neumann escribe el "First Draf of a report on the EDVAC" una página del primer documento donde se describe el diseño lógico de una computadora utilizando el concepto de programa almacenado (stored-program). Hoy día conocido como **Arquitectura de von Neumann**.

1946 En la Universidad de Pensilvania se construye la ENIAC (Electronic Numerical Integrator And Calculator), que fue la primera computadora electrónica de propósito general



ENIAC. Fuente: Wikipedia

1951 El Sistema A-0 fue inventado por Grace Murray Hopper. Fue el **primer compilador** desarrollado para una computadora electrónica.

1958 Comienza la segunda generación de computadoras, caracterizados por usar **circuitos transistorizados** en vez de válvulas al vacío.

A partir de la década de los 60 se aceleran los avances, y cada año se crean nuevos sistemas que permiten evolucionar lo ya conocido.

1964 La aparición del IBM 360 marca el comienzo de la tercera generación de computadoras. Comienzan las placas de circuitos integrados.

1969 Se publica el primer borrador de lo que será conocido como ARPANET (el precursor de la actual Internet).

1970 Intel crea la primera memoria dinámica RAM.

1971 Intel presenta el primer **procesador comercial** y a la vez el primer chip microprocesador, el Intel 4004.

1971 Ray Tomlinson crea el primer programa para enviar correo electrónico.

1972 Se decide reescribir Unix, pero esta vez utilizando lenguaje C.

1974 Se crea el sistema Ethernet para enlazar a través de un cable único a las computadoras de una LAN.

1981 Se lanza al mercado el **IBM PC**, que se convertiría en un éxito comercial, marcaría una revolución en el campo de la computación personal y definiría nuevos estándares.

1981 Se termina de definir el protocolo **TCP/IP**. Lo utilizamos actualmente para navegar por Internet.

1983 Richard Stallman anuncia públicamente el proyecto GNU, con el objetivo de crear el primer sistema operativo libre de tipo Unix.

1986 El lenguaje **SQL** es estandarizado por ANSI.

1990 Tim Berners-Lee idea el hipertexto para crear el World Wide Web (www) una nueva manera de interactuar con Internet.

1991 Linus Torvalds comienza a desarrollar Linux, el **kernel** (o núcleo) de un sistema operativo compatible con Unix.

1991 Comienza a popularizarse la programación orientada a objetos.

1995 Aparece la primera versión de MySQL.

1995 Se inicia el desarrollo del servidor Apache.

1997 EL IEEE crea la primera estándar WLAN y la llamaron 802.11. El primer protocolo para WiFi.

Se podrían añadir más momentos importantes hasta el día de hoy, pero como ya se ha dicho previamente, se ha elegido sólo un pequeño resumen.

1.3. Componentes de un sistema informático

Podemos diferenciar distintos componentes dentro de un sistema informático:

- **Hardware**: Es todo lo que forma parte del ordenador, que **puede ser tocado físicamente**. Es decir: teclado, ratón, monitor, placa base, procesador, memoria, disco duro, cables, etc. Es la "maquinaria" necesaria utilizada para el tratamiento automático de la información.
- **Software**: Es el elemento lógico, es todo aquello que es "intangible". Es el **conjunto de programas y datos** que permiten manejar el hardware, controlando y coordinando su funcionamiento para que realice las tareas deseadas.

2. Hardware

Como ya se ha comentado previamente, el **hardware** es todo lo que forma parte del ordenador, que **puede ser tocado físicamente**. Dentro de un ordenador vamos a poder diferenciar distintos componentes que cumplirán una función distinta que detallaremos más adelante.

Es posible que ya conozcamos alguno de estos componentes, pero debemos conocer el origen y cómo surge la arquitectura de los ordenadores modernos.

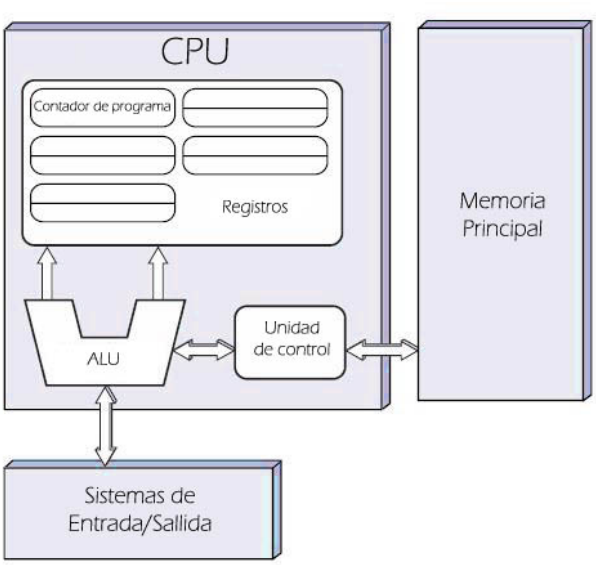
2.1. Arquitectura Von Neumann

Las primeras computadoras electromecánicas eran diseñadas para un único propósito, estaban "diseñadas" para realizar una tarea. Un caso conocido puede ser **Bombe**, una máquina electromecánica capaz de descifrar los sistemas criptográficos nazis de Enigma. The Imitation Game

Algunas se podían "reprogramar", pero a base de recablear distintos componentes tras un estudio de lo que se quería realizar. Podía tomar hasta tres semanas preparar un programa de ENIAC y conseguir que funcionara.

El concepto de máquinas de computación universal y el uso de programa almacenado ya existía a nivel teórico desde mediados de la década de 1930 (escrito por Alan Turing).

El matemático y físico John von Neumann, junto con otros compañeros, **describe en 1945 un diseño para una arquitectura de computadoras** en el que se describren los siguientes componentes que se interrelacionan entre sí a través del bus del sistema que actúa como canal de comunicación entre ellos:



Arquitectura Von Neumann. Fuente: wikipedia

- **Unidad Central de Proceso** (**CPU**, por sus iniciales en inglés), que a su vez, contiene:
  - **Unidad Aritmético Lógica** (**ALU** en inglés): Es un circuito digital que realiza operaciones aritméticas (suma, resta, multiplicaciones,…) y operaciones lógicas (AND, OR, X-OR,…) entre los valores de los argumentos (uno o dos).
  - **Registros del procesador**: Memoria de alta velocidad y poca capacidad integrada en la CPU para almacenar datos utilizados durante la ejecución de un programa:
    - Contador de programa
    - Acumulador
    - Registro de instrucciones
  - **Unidad de control**: Su función es buscar las instrucciones en la memoria principal,

decodificarlas y ejecutarlas, empleando para ello la unidad de proceso.

- La **memoria principal**: Sistema donde se almacenan las instrucciones y los datos del programa que se está ejecutando en ese instante, dividida en celdas que se identifican por medio de una única dirección.
- Los **sistemas de Entrada/Salida**: Realizan la transferencia de información entre periféricos de entrada y/o salida para extender las capacidades del equipo.

Hoy en día, los ordenadores han evolucionado, pero la arquitectura sigue siendo la misma, aunque más compleja.

Ejercicio

Podemos ver una simulación de la Arquitectura Von Neumann aquí

2.2. Componentes básicos

Un ordenador moderno se puede distinguir de distintos componentes, los cuales cumplen una función específica. Así mismo, también pueden contar con subcomponentes integrados que son necesarios para cumplir su cometido final.

A continuación se van a detallar los componentes necesarios de un ordenador moderno.

2.2.1. Placa base

La placa base (conocida en inglés como *motherboard*) es una tarjeta de circuito impreso que tiene elementos electrónicos (resistencias, condensadores, reguladores …) a la que se conectan el resto de componentes que forman el ordenador. Es por eso que se puede considerar como la parte fundamental a la hora de montar un ordenador, ya que sin ella, el resto de componentes no se podrán comunicar entre sí.

2.2.1.1. Formatos de placa base

Las placas base deben tener un tamaño compatible con las cajas en las que van a ir montadas, y es por eso que hay distintos tamaños estandarizados. Cada uno de estos tamaños determinan dónde van a ir montados algunos de los componentes y conectores, así como los agujeros donde irán los tornillos de sujeción a la caja.

Si queremos profundizar más en los distintos formatos, la Wikipedia cuenta con una sección en la que se comparan los distintos tamaños.

2.2.1.2. Conectores de la placa base

Como ya hemos indicado, a la placa base se conectan el resto de componentes que forman el ordenador, y es por eso que va a tener distintos conectores:

- **Zócalo del microprocesador**: También llamado ***socket***. Es donde se conecta el microprocesador sin tener que soldarlo a la placa, y de esta manera puede ser sustituido. El número de conexiones que

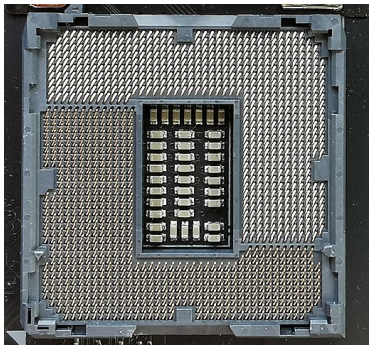
conectan la placa base al microprocesador ha ido aumentando a medida que ha ido evolucionando la tecnología, siendo hoy día de hasta 1700 conectores.

Dependiendo del tipo de procesador, y el modelo, el socket variará en número de contactos y el tipo de los mismos. Existen distintas maneras de interconexión:

- **PGA**: De *ping grid array*, o matriz de rejilla de pines. El procesador cuenta con unos pines en formato perpendicular que se conectan al socket donde estarán unos agujeros.

  En la imagen se puede ver un Socket AM4 con tecnología PGA que tiene 1331 contactos.

- **LGA**: De *land grid array*, o matriz de contactos en rejilla. En este caso el procesador **no cuenta** con pines, sino que es una matriz de contactos chapados en oro. Esta rejilla de contactos hacen contacto con el zócalo de la placa base que es la que cuenta con unos pequeños pines flexibles.



- **BGA**: De *ball grid array*, o matriz de rejilla de bolas. El procesador cuenta con unas bolas de estaño que al calentarse se sueldan a la placa base. Hoy en día se utiliza en componentes de tamaño reducido, como en móviles, los chips de memoria en los módulos de RAM, …

- **Conectores de alimentación**: La placa base tendrá distintos conectores provenientes de la fuente de alimentación con diferentes voltajes, para así proveer de alimentación a los componentes conectados a ella.

- **Ranuras de memoria RAM**: Hoy en día es habitual contar con varias ranuras donde conectar las memoria RAM. Más adelante hablaremos en profundidad sobre la memoria RAM.

- **Chipset**: Es un conjunto de chips, o circuitos electrónicos, que gestionan la comunicación entre los distintos componentes que forman el ordenador, y que están conectados en la placa base. Hoy en día se suelen dividir en dos partes:

  - **Northbridge**: O puente norte. Controla el tráfico de los componentes que trabajan a más alta frecuencia. Comunica el microprocesador, la memoria RAM y la GPU (la ranura PCI express).

  - **Southbridge**: O puente sur, comunica los periféricos, los dispositivos de almacenamiento, puertos de entrada/salida como USB, ethernet, …

Información

Hoy día el northbridge está integrado en el propio procesador y en algunos casos partes del Southbridge también.

- **Ranuras de expansión**: Vamos a identificar estas ranuras como las más modernas **PCIexpress**. Son un bus de comunicación de datos de alta velocidad que es usado principalmente para conectar tarjetas gráficas.

  Es cierto que se pueden conectar otro tipo de tarjetas de expansión, como capturadoras de vídeo, tarjetas de red, controladoras RAID, …

  Hoy en día existen conectores (**M.2**) donde poder conectar discos duros que tienen un tipo de conector especial.

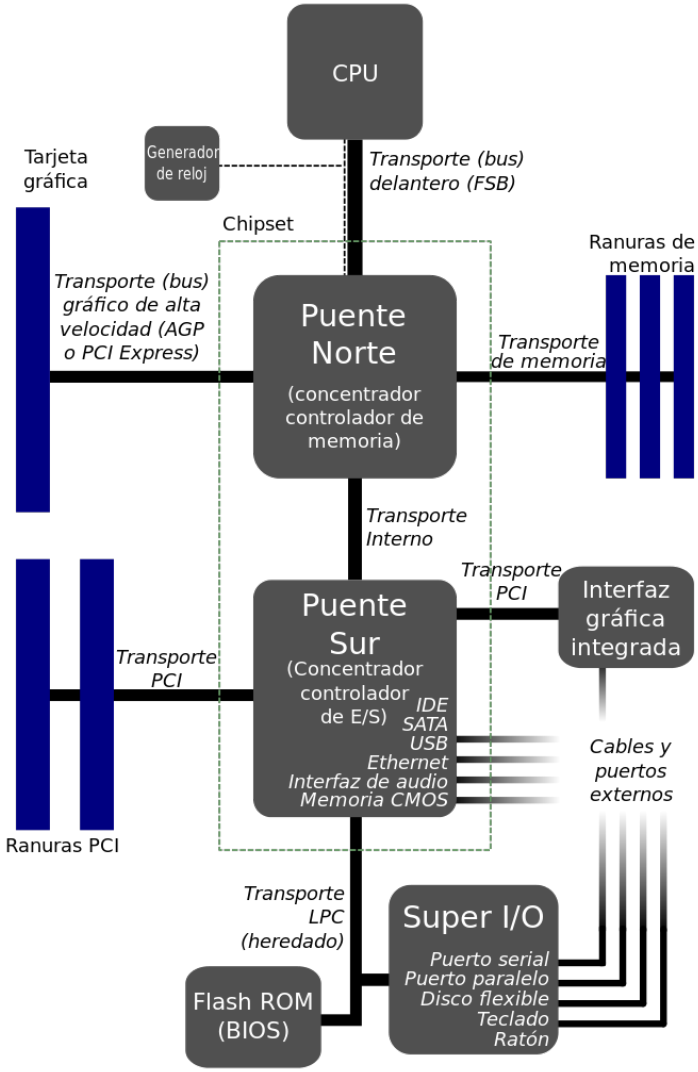
- **Otros conectores de entrada/salida**: En la placa base existen otros muchos conectores de entrada y salida, que cumplirán distintas funciones dependiendo del tipo de conector, función y/o protocolo de comunicación que utilicen.

  Algunos de estos conectores tendrán conector exterior (al que se podrá conectar directamente el dispositivo), mientras que otros necesitarán de un adaptador (como sucede hoy día con conectores extra USB o el conector "serie"). Algunos ejemplos son:

  - **USB**: Donde poder conectar distintos dispositivos como teclados, ratón, pendrives, mandos de juegos, impresoras, … El USB (**Universal Serial Bus**) es un estándar de comunicación de periféricos hoy día. En las placas actuales también hay conectores de tipo **USB-C**.
  - Conectores de **pantalla** como **VGA**, **HDMI** o **DisplayPort**. Dependiendo de lo moderna que sea la placa base, contará con uno o varios de estos conectores.
  - **Red**: Hoy día el conector RJ45 es el estándar, que dependiendo de la versión ethernet, nos dará al menos 1Gbit de transmisión. Dependiendo del modelo de placa base también puede tener conectores para realizar conexiones a redes inalámbricas.
  - **Audio**: Tanto de entrada como de salida. Normalmente se hace uso de conectores de tipo **jack**, pero también puede haber conectores de salida digital.
  - **Pila**: Las placa base cuentan con una pila para mantener la alimentación para guardar la información de la RAM-CMOS, que es una pequeña memoria que usa la BIOS durante el arranque del sistema.
  - **Conectores para ventiladores**: Para regular la temperatura del microprocesador y del interior de la caja, la placa tiene varios conectores que se conectarán a distintos ventiladores que se regularán en intensidad.
  - **Otros conectores**: Existen otros conectores para otros puertos que hoy en día no se usan tanto (serie, paralelo, …) y también los conectores para encender el ordenador, realizar el reset,

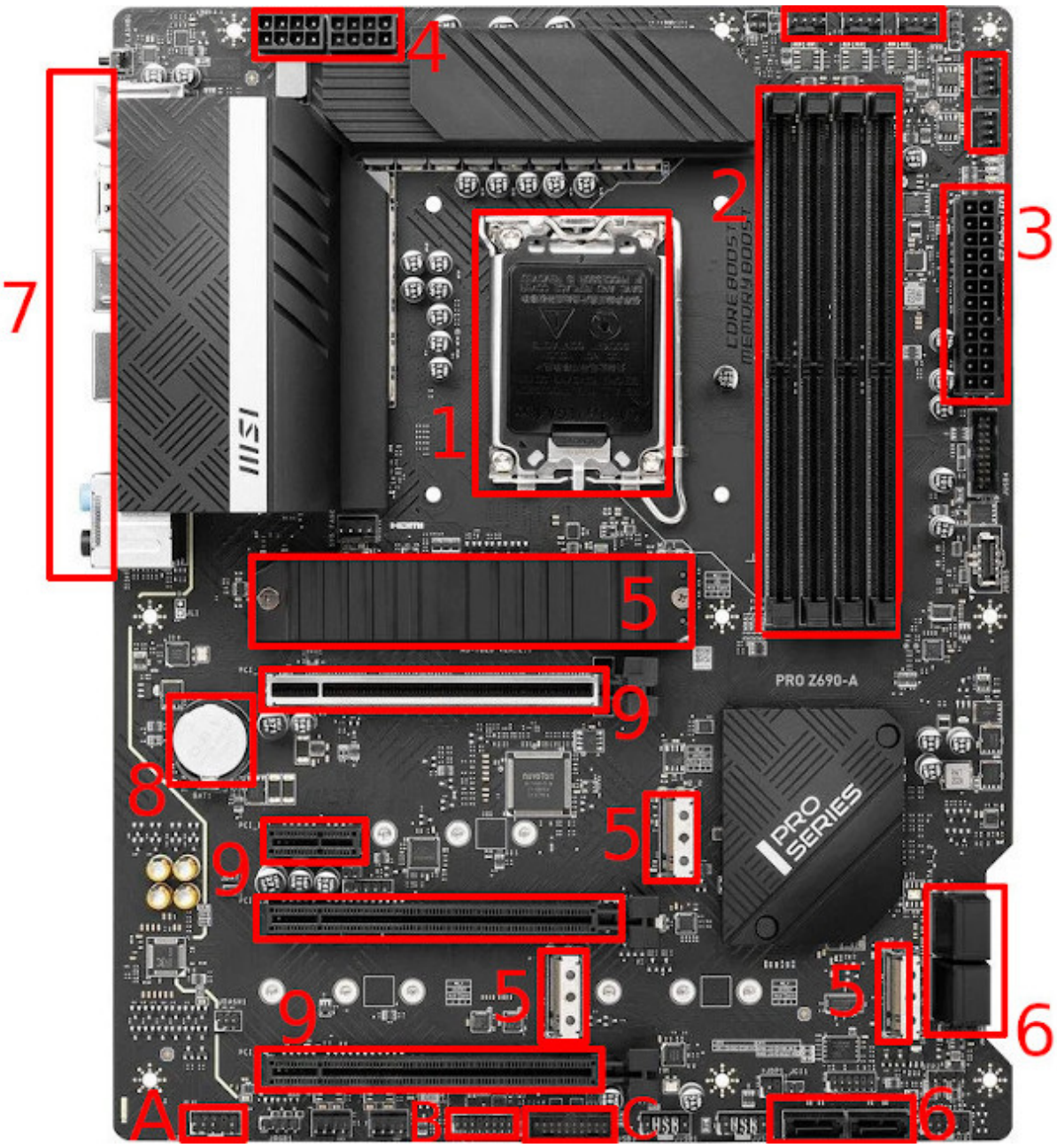
comprobar el funcionamiento del disco…

A continuación un diagrama simplificado de una placa base. Fuente: Wikipedia.



2.2.1.3. Ejemplo de placa base

A continuación se van a diferenciar los componentes vistos anteriormente en una placa base real, utilizada para crear un equipo de escritorio moderno:



Placa Base MSI PRO Z690-A. Manual

1. Zócalo (socket) del procesador.
2. Ranuras para la memoria RAM.
3. ATX de alimentación.
4. Conectores de alimentación extra necesarios por la CPU.
5. M.2 para discos duros.
6. SATA para discos duros.
7. Conectores exteriores (se verán a continuación)
8. Pila.
9. Ranuras PCIexpress de distintas velocidades.

A. Audio

B. Conectores frontales.

C. USB 3.0

Los conectores exteriores de esta placa tienen el siguiente aspecto:

Placa Base MSI PRO Z690-A. Manual

De izquierda a derecha, y de arriba abajo:

- Pulsador para actualizar la BIOS.
- Conector PS2 y USB para actualizar la BIOS.
- DisplayPort y HDMI
- Usb 2.0 y 3.2
- Conector LAN, USB y USB-C
- Conectores de audio

2.2.2. BIOS/UEFI

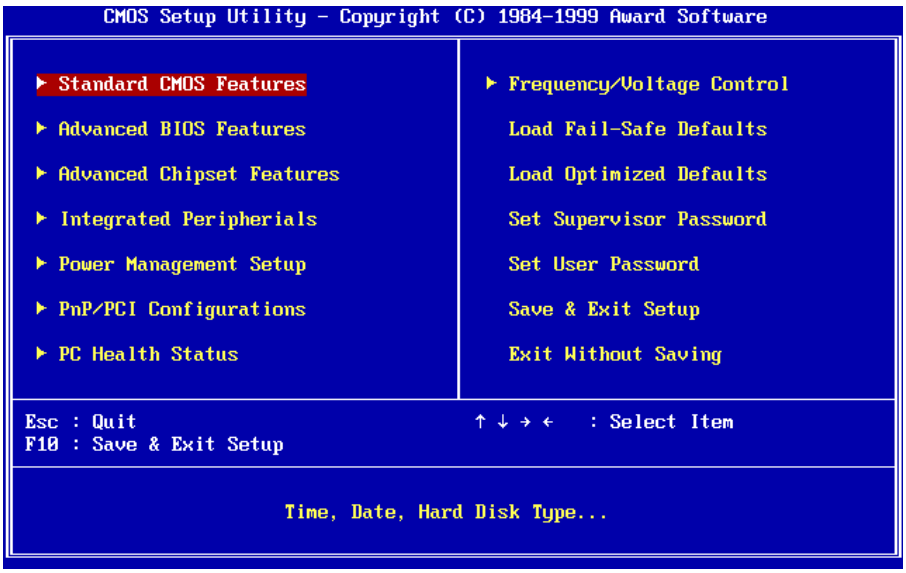
La BIOS/UEFI es un interfaz de firmware que está incorporado en un chip en la placa base.

La función principal es la de iniciar el ordenador, realizar una comprobación del *hardware* del sistema y se encarga de arrancar el gestor de arranque.

Se ha unificado en esta sección BIOS y UEFI ya que cumplen de manera similar la misma función, pero la segunda es una evolución de la primera.

2.2.2.1. BIOS

El sistema básico de entrada-salida (del inglés *Basic Input/Output System*, o BIOS) lo creó IBM para sus ordenadores "**Personal Computer**". Posteriormente se obtuvo por ingeniería inversa las funciones que realizaba tratando de buscar equipos que fueran compatibles (denominados "PC-compatible") y de esta manera convirtiéndose en un estándar de facto.



Interfaz BIOS. Fuente wikipedia

A través de este interfaz se podían configurar algunos aspectos del hardware como las interrupciones de teclado que utilizaban los sistemas operativos antiguos (como MS-DOS), direcciones, el orden del sistema de arranque, …

2.2.2.2. UEFI

La ***Unified Extensible Firmware Interface*** (UEFI o «interfaz unificada de firmware extensible») es una especificación pública que define un interfaz entre el sistema operativo y el firmware de la plataforma.

Se puede considerar una evolución de la BIOS que tiene las siguientes características:

- Permite arrancar desde particiones de más de 2TB gracias a eliminar las limitaciones del MBR (*master boot record*).
- Diseño modular y extensible.
- Retro-compatible con BIOS.
- Interfaz gráfica más amigable con el usuario.

Interfaz UEFI placa Asus.

2.2.3. Procesador

El procesador (o microprocesador) es la unidad central de proceso (CPU) de la arquitectura Von Neumann, y es el circuito integrado más complejo que tiene el ordenador. Se puede considerar el "cerebro".

Es el encargado de ejecutar todos los programas y operaciones que realizamos, pero sólo sabe ejecutar instrucciones en lenguaje máquina (código binario).

El microprocesador se conecta a la placa base a través del socket, y encima de él se añade un sistema de refrigeración para disipar el calor que genera durante su funcionamiento.

2.2.3.1. Características

A la hora de determinar las características principales que cuenta un procesador podemos destacar las siguientes:

- **Frecuencia del reloj**: Es la cantidad de veces que los transistores internos del procesador pueden conmutar eléctricamente (abrir y cerrar el flujo de corriente eléctrica). Hoy en día se mide en GHz (giga hercios), donde 1GHz es mil millones de ciclos por segundo.

  Normalmente se confunde con el número de operaciones o instrucciones que puede ejecutar en un segundo, pero eso no es del todo correcto.

  Tampoco determina que cuanta mayor frecuencia el procesador vaya a ser mejor que otro con menor frecuencia (ya existían procesadores a 4GHz hace años).

- **Bus de direcciones**: Este tamaño determinará la cantidad máxima de memoria que podemos direccionar. Con 32 bits se pueden direccionar 2^{32}, es decir, 4GB. Mientras que con 64 bits en los procesadores modernos llegamos hasta los 16 exabytes de memoria RAM (2^{64}).

- **Bus de datos**: Es el dato más grande que es capaz de manejar en una única instrucción.

- **Memoria caché**: Es una memoria que se encuentra internamente dentro del procesador, que es mucho más rápida, pero también de mucho menor tamaño, si la comparamos con la memoria RAM. Los procesadores actuales cuentan con unos pocos MB de tamaño dependiendo del nivel de caché. Ejemplo el Intel Core i5-12400 con 7.5MB de L2 caché.

- **Voltaje**: Para que el procesador funcione necesita ser alimentado con frecuencia eléctrica. Normalmente a mayor voltaje se consigue mayor frecuencia de reloj.

- **Número de cores**: Los microprocesadores actuales no cuentan con una única CPU interna (como era habitual hasta el año 2006 más o menos), ya que pueden contar con varias, denominadas "***cores***".

  Actualmente también se pueden diferenciar en el tipo de *core* que tiene, ya que algunos están diseñados para más eficiencia mientras que otros para mayor carga de trabajo.

- **Multihilo/Hyperthreading**: Consiste en simular dos procesadores lógicos dentro de un único procesador físico. Permite ejecutar programas que están preparados, y eso permite una mejora en

el rendimiento.

Existen otro tipo de características más técnicas, pero que también son importantes de conocer. A nivel de cómo está diseñado el procesador, lo que se denomina la **arquitectura interna** también contamos con diferencias. Hoy día podemos encontrar dos arquitecturas diferenciadas:

- **CISC**: Del inglés *Complex Instruction Set Computer* (conjunto de instrucciones complejas), es un conjunto de instrucciones que se caracteriza por ser muy amplio y permitir operaciones complejas entre operandos situados en la memoria o en los registros internos.

  Los CISC pertenecen a la primera corriente de construcción de procesadores. Pertenecen a esta arquitectura la mayoría de los procesadores actuales de ordenadores personales: AMD, X86_64.

- **RISC**: Del inglés *Reduced Instruction Set Computer* (computador con conjunto de instrucciones reducido) es una filosofía de diseño de CPU para computadora que está a favor de conjuntos de instrucciones pequeñas y simples que toman menor tiempo para ejecutarse.

  Hoy en día podemos encontrar esta arquitectura sobre todo en ARM que se utiliza en procesadores de móviles como los A15 de Apple (pero también en los procesadores de escritorio M1 y M2), Qualcomm Snapdragon, …

2.2.3.2. Rendimiento

Dadas todas las características que hemos visto previamente, no podemos determinar si un procesador es mejor a otro sólo mirando sus características y determinando que "cuanto más mejor". Por ejemplo:

- **Intel Pentium 4**: 2.80GHz de frecuencia de reloj
- **Intel Core 2 Duo E8200**: 2.66GHz de frecuencia de reloj
- **Intel i5-11400**: Frecuencia de reloj base 2.60GHz.
- Comparativa entre los dos primeros
- Comparativa entre los dos últimos: 1 y 2.

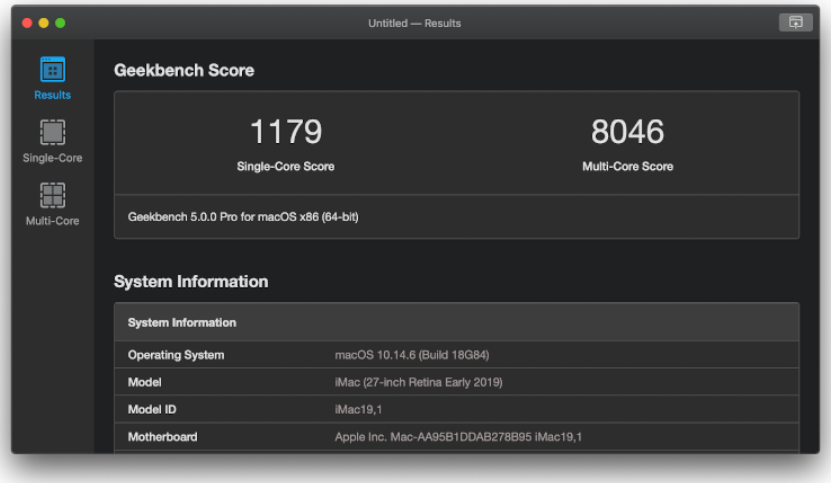
¡Cuidado!

No nos podemos quedar con la idea de que "cuanto más mejor" cuando nos referimos a cantidades en las características del procesador

Es por eso que existen las **pruebas de rendimiento**, también conocido en inglés como **benchmark**.

Estas pruebas de rendimiento se ejecutan a través de un programa que va a ejecutar un conjunto de operaciones (que siempre serán las mismas) y determinará el tiempo llevado a cabo, y junto con otras especificaciones terminará dando una puntuación al resultado obtenido.

De esta manera, si utilizamos el mismo programa de benchmark en dos procesadores distintos, obtendremos puntuaciones distintas. Por ejemplo, Geekbench es un programa multiplataforma muy popular hoy día:



2.2.4. Sistema de refrigeración

Debido a que el procesador genera calor durante su funcionamiento, y que esto repercute en su funcionamiento, se debe de mantener a una temperatura acorde. Es por ello que debemos hacer uso de un sistema de refrigeración.

El sistema de refrigeración cuenta con dos partes:

- **Disipador**: Está en contacto con el procesador y por el principio cero de la termodinámica le traspasa el calor.
- **Sistema de reducción de temperatura**: Trata de reducir el calor que recibe el disipador para que el procesador se enfríe. Podemos diferenciar los siguientes sistemas:
  - **Por aire**: Usando ventiladores.
  - **De agua autocontenida y aire**: Es un circuito cerrado de agua que pasa a través de un radiador refrigerado por ventiladores. Venden el sistema cerrado, por lo que no hay que hacer nada con el líquido.
  - **Refrigeración líquida**: Se utilizan componentes especiales para realizar el contacto con la CPU (y la GPU), y se debe hacer un circuito cerrado por el que pasará el líquido refrigerante, y una bomba que moverá el líquido.

Disipador con ventilador y refrigeración líquida autocontenida.

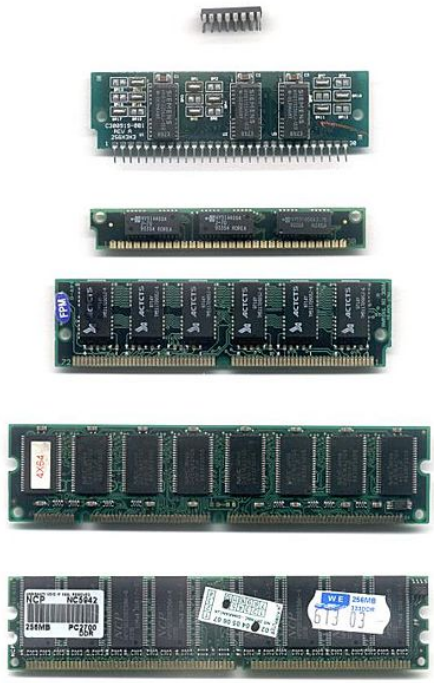
2.2.5. Memoria RAM

La memoria de acceso aleatorio (en inglés *Random Access Memory*) es una memoria a corto plazo para almacenar los programas que están siendo ejecutados.

Cuando un programa se ejecuta se cargan todas sus instrucciones en RAM, así como todos los datos que va a manipular.

Es una memoria **volátil**, esto quiere decir que cuando deja de recibir electricidad, se pierde la información, por ejemplo al apagarse el ordenador o al reiniciarlo.

Se denominan "de acceso aleatorio" porque se puede leer o escribir en cualquier posición tardando lo mismo, no siendo necesario seguir un orden para acceder.



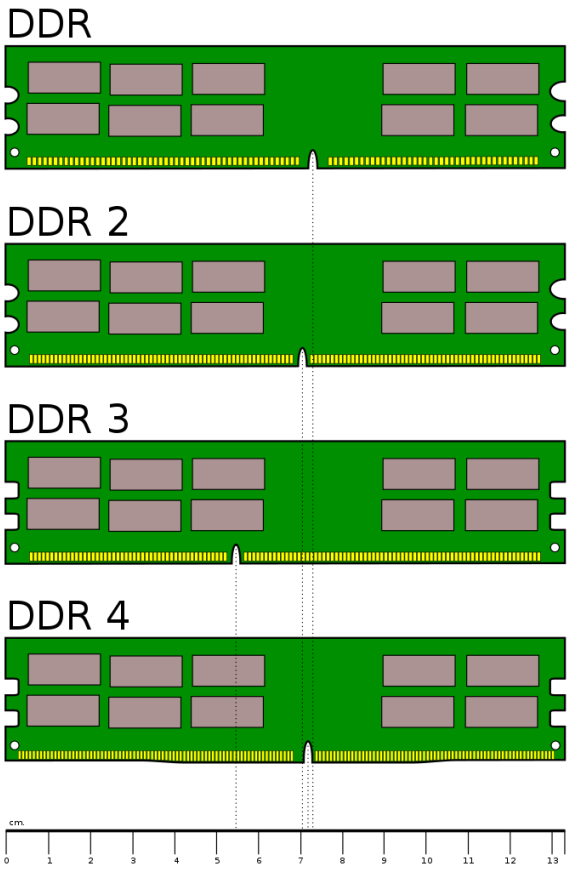
fuente: Wikipedia

A lo largo de la historia la memoria RAM ha tomado distintas formas:

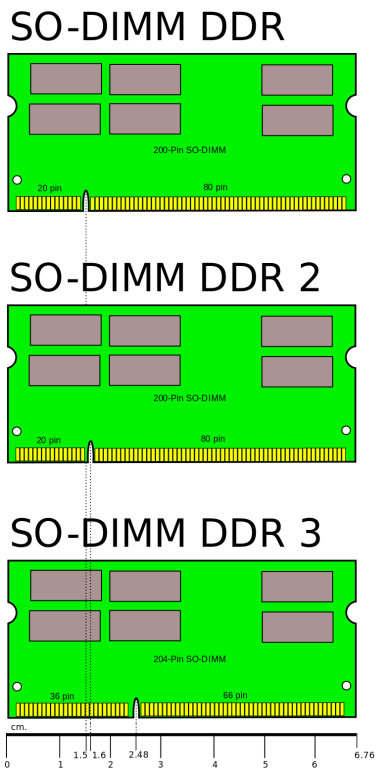
- Antes de los circuitos integrados era una matriz metálica que funcionaba por electromagnetismo. Foto.
- Con la llegada de los circuitos integrados se instalaban en la placa soldandolos o sobre pequeños zócalos.
- Para hacer el sistema modular, se pasó al formato SIPP (*Single In-line Pin Package*). En una sola tarjeta se integraban varios módulos de memoria, pero los pines eran frágiles.
- Como evolución llegó el formato SIMM (*single In-line Memory Module*), que en lugar de pines tenía contactos en ambas caras del módulo.
- Hoy en día hacemos uso del formato DIMM (*Dual In-line Memory Module*) y su versión reducida SO-DIMM utilizada en portátiles.

Hoy en día hacemos uso de memoria dinámica de acceso aleatorio que tiene un interfaz síncrona (SDRAM) y que tienen la capacidad de transferir simultáneamente datos por dos canales distintos en un mismo ciclo de reloj (DDR, de *double data rate*).

A continuación se pueden diferenciar cómo ha variado el formato físico.



Fuente: Wikipedia



Fuente: Wikipedia

En la Wikipedia podemos ver una tabla con la evolución desde DDR hasta DDR5, con todos los datos técnicos como: voltaje utilizado, número de pines, ancho de banda en MB/s…

2.2.6. Dispositivos de almacenamiento de datos

Los dispositivo de almacenamiento de datos nos permiten leer o grabar datos de forma temporal o permanente. Existen distintos tipos de dispositivos que se pueden diferenciar por formato, tamaño, tecnología, tipo de acceso, …

Si diferenciamos por el tipo de tecnología utilizada para realizar el almacenamiento podemos distinguir:

- **Dispositivos magnéticos**: Se utilizan las propiedades magnéticas de materiales para realizar el almacenamiento de datos digitales sobre el soporte de datos. En este apartado podemos poner como ejemplo:
  - Unidades de cinta magnética: No sólo utilizadas para almacenar datos en informática, también se ha utilizado en formato casete para la música.
  - Disquete: O floppy disk, es un formato formado por una fina lamina circular dentro de una caja de plástico. Los tamaños más utilizados fueron de 8”, 5 ¼ƶ 3½”.
  - Discos duros: Luego profundizaremos sobre ellos.
- **Dispositivos ópticos**: Es un tipo de unidad de disco que utiliza un láser para realizar la lectura y escritura de datos. Los formatos más habituales en informática han sido los CDs, DVDs y Blu-Ray.

- **Unidad de estado sólido**: Conocidos como SSD (*solid state drive*), hacen uso de memoria flash para almacenar datos de manera persistente.

Si diferenciamos por el acceso a los datos podemos diferenciar por:

- **Acceso secuencial**: Para realizar la lectura del dato que nos interesa debemos leer registro a registro desde el inicio hasta llegar al dato que deseamos encontrar.
- **Acceso aleatorio**: Para realizar la lectura de un dato concreto, podemos acceder de manera directa, sin tener que pasar por el resto de datos.

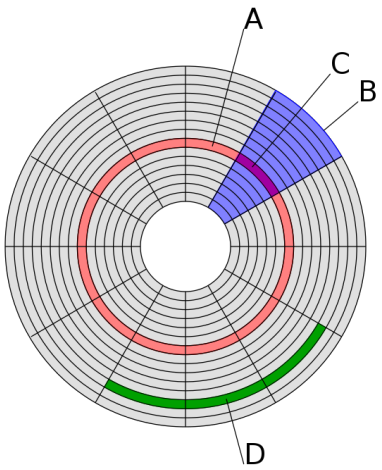
Vamos a centrarnos en los denominados “discos duros” y que son más utilizados a día de hoy:

2.2.6.1. Discos duros HDD

Las unidades de discos duros (también conocidos como HDD, de *hard disk drive*) emplean un sistema de grabación magnética para almacenar y recuperar archivos digitales.

Están compuestos por varios **platos** de aluminio que giran todos a la vez sobre el mismo eje y que son recorridos por unos **cabezales** que están montados sobre unos brazos que recorren la superficie.

Estos cabezales son los encargados de magnetizar la superficie del plato al realizar las escrituras o leyendo la superficie para determinar cuál es el estado magnético y de esta manera conocer los datos guardados.



Fuente: Wikipedia

A la hora de guardar la información en los platos se sigue la estructura de la imagen superior, donde:

A. Es una pista del disco.
B. Es un sector geométrico.
C. Es un sector de una pista.
D. Es un grupo de sectores.

Si tenemos en cuenta las características de un HDD, podemos destacar:

- **Tiempo medio de acceso**: tiempo medio que tarda el cabezal en situarse en la pista y el sector deseado.
- **Tiempo de lectura/escritura**: tiempo medio que tarda el disco en leer o escribir nueva información: Depende de la cantidad de información que se quiere leer o escribir, el tamaño de bloque, el número de cabezales, el tiempo por vuelta y la cantidad de sectores por pista.
- **Velocidad de rotación**: Es la velocidad de giro de los platos. Por norma general a mayor velocidad de rotación, más alta será la transferencia de datos, pero también el ruido generado y el calor producido. Se mide en RPM (revoluciones por minuto). Dependiendo del tipo de disco puede variar entre 5.400RPM en equipos portátiles a 15.000RPM para servidores.

Debido a que los discos duros utilizan partes mecánicas hay que tener cuidado al transportarlos (aunque estén parados) y con el movimiento, ya que un golpe puede romper algún componente interno.

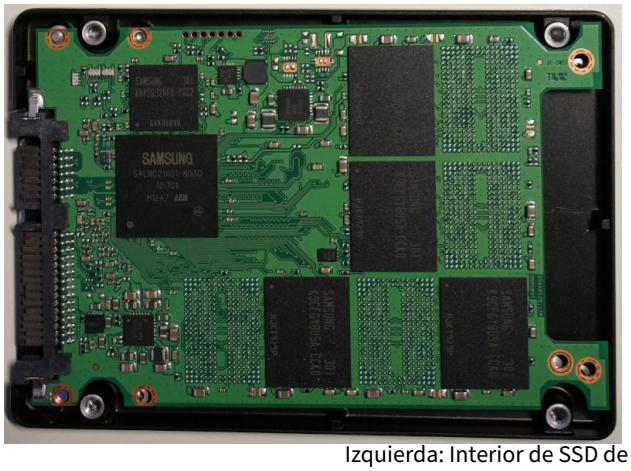
¡Cuidado!

Los discos duros HDD son propensos a golpes, debido a los componentes móviles internos

2.2.6.2. SSD

Conocidos como *solid state drive*, hace uso de memorias flash para el almacenamiento de datos en lugar de platos, y debido a que no tiene componentes móviles, son menos propensos a daños por golpes.

Debido a la mejora en la tecnología de guardado de datos, y por no poseer partes móviles, no generan ruido, son más ligeros, el tiempo de acceso a los datos es menor, y todo ello hace que la transferencia de datos aumente en comparación con los HDD.



Izquierda: Interior de SSD de 2,5". Derecha: SSD conector m.2

Hoy en día la manera más habitual de usar este tipo de unidades es con el factor de forma de 2,5" o en conocido como mSATA o m.2.

2.2.6.3. NVMe

La especificación de interfaz de controlador de host de memoria no volátil (NVMHCIS, en inglés *non-volatile memory host controller interface specification*) que está conectado a través del bus PCI Express (PCIe).

Normalmente se llama NVMe para abreviar.

Este tipo de dispositivos, al igual que el anterior, hacen uso de tecnología FLASH para el almacenamiento de datos. Debido a que están conectados al bus PCI Express, y que la especificación de acceso se creó desde cero (para aprovechar la tecnología moderna de memorias FLASH, el paralelismo de las CPUs…), consiguen un rendimiento muy superior a las generaciones anteriores.

Izquierda: NVMe en formato tarjeta PCIe. Derecha: NVMe con conector m.2

Las primeras unidades tenían un formato de tarjeta de expansión que se conectaba directamente a la ranura PCIexpress, mientras que hoy día existen los conectores M.2 para instalarlos.

2.2.6.4. Comparativa HDD, SSD y NVMe

En la siguiente tabla se puede comparar algunas características básicas de los distintos tipos de unidades de almacenamiento vistas.

	HDD	SSD	SSD (M.2)	NVMe (PCIe 3.0)	NVMe (PCIe 4.0)
Conector	SATA	SATA	M.2	M.2	M.2
Velocidad Lectura	150MB/s	560 MB/s	560 MB/s	3500 MB/s	7000 MB/s
Velocidad Escritura	120MB/s	510 MB/s	520 MB/s	3000 MB/s	5300 MB/s
Precio por TB	Bajo	Medio	Medio	Alto	Alto

Hay que tener en cuenta que las velocidades dependen de la tecnología de la unidad y también de la conexión utilizada. Son velocidades aproximadas, y por tanto habría que ver las especificaciones técnicas de cada dispositivo antes de comprarlo.

Información

Con las velocidades de lectura y escritura suelen indicar si es secuencial o aleatoria. En lecturas y escrituras aleatorias la velocidad es más baja.

También existen pruebas de rendimiento para sistemas de almacenamiento, por lo que es importante informarse bien antes de elegir uno.

2.2.7. Fuente de alimentación

La fuente de alimentación en un ordenador es el componente que convierte la corriente alterna a varias corrientes continuas ya que el ordenador hace uso de diferentes voltajes.

A la hora de elegir una fuente de alimentación debemos tener en cuenta:

- **Potencia**: Se mide en vatios (W, de *watts*), y tendremos que tener en cuenta el consumo de los distintos componentes que tienen nuestro ordenador.
- **Factor de forma**: En los ordenadores de sobremesa hoy en día el formato es ATX, pero podemos elegir dependiendo del tipo de conexiones:
  - **Cableado completo**: La fuente de alimentación cuenta con todo el cableado completo.
  - **Semi-modular**: Algunos de los cables que son necesarios se pueden poner y quitar, dependiendo de las necesidades que tengamos.
  - **Full-modular**: Todos los cables se pueden poner y quitar, lo que facilita la instalación de la fuente de alimentación y el orden dentro de la caja durante el montaje

Fuentes de alimentación con cableado completo, semi-modular y full-modular.

2.2.8. GPU/Tarjeta gráfica

Hoy en día es habitual contar con una tarjeta gráfica en los ordenadores personales, cuando el desempeño de su función va a requerir realizar grandes procesamientos de gráficos como: juegos, edición de vídeo, edición fotográfica, uso de dibujo asistido por ordenador, …

Podemos diferenciar:

- **Gráficos integrados**: Hoy en día los procesadores pueden contar con una unidad de procesamiento gráfico interna, que para el desempeño del uso del ordenador y tareas livianas (ver vídeos, juegos antiguos o con pocas necesidades) puede ser suficiente.

Para confirmar si nuestro procesador tiene o no, deberíamos mirar las especificaciones técnicas del mismo (por ejemplo: Intel i7-14700KF no cuenta con procesador gráfico mientras que el i7-14700 sí).

- **Gráficos dedicados**: Este es el caso de las denominadas tarjetas gráficas, que van conectadas a una ranura PCI-express. **Nos vamos a centrar en este tipo de tarjetas**.

Las tarjetas gráficas actualmente se instalan en la ranura PCI-Express de mayor velocidad de la placa base y suele contar con los siguientes componentes:

- **Unidad de procesamiento gráfico**: O **GPU**, es un procesador como la CPU pero diseñado para el procesamiento gráfico. Su finalidad es la de realizar operaciones con vectores, triángulos, texturas, … lo más rápido posible.

  Las tarjetas gráficas también sirven para realizar codificación/decodificación de vídeo por hardware, lo que disminuye el tiempo en comparación a realizar esa compresión a través de la CPU.

  Actualmente la tecnología puntera trata de simular la luz de la manera más real posible haciendo uso del denominado raytracing.

- **VRAM**: O memoria gráfica, son los chips que almacena y transporta información hacia la tarjeta gráfica. En el caso de las tarjetas gráficas dedicadas, cuentan con sus propios chips en la tarjeta, mientras que cuando hablamos de gráficos integrados suele ser RAM que se reserva para el uso de gráficos.

- **Conectores de salida**: Para poder realizar la conexión entre la tarjeta y los monitores que tengamos conectados. Hoy en día lo más habitual es tener conectores HDMI y DisplayPort.

Si nuestro procesador cuenta con una gráfica integrada y aparte tenemos una gráfica dedicada, dependiendo del uso que queramos darle al ordenador, quizá sea conveniente desactivar la integrada a través de la UEFI.

Información

Si tenemos gráfica integrada y dedicada, quizá nos interese desactivar la integrada a través de UEFI

Las compañías que crean las tarjetas gráficas también han creado **SDK** (*Software Development Kits*, o kits de desarrollo de software), como Nvidia CUDA, para poder realizar computación paralela y así aprovechar la potencia de cálculo para proyectos de *machine learning*, simulaciones científicas, cálculo de proteínas, secuencias de ADN, …

Información

Podemos usar el procesamiento de la tarjeta gráfica para ayudar a la ciencia usando proyectos como Folding@Home gracias a la computación distribuida

2.2.9. Conectores más importantes

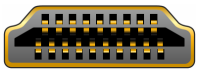
Aunque ya hemos visto de manera generalizada algunos tipos de conexiones que tiene la placa base, vamos a profundizar en este apartado separándolos por secciones.

2.2.9.1. Conectores gráficos

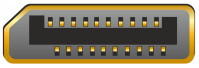
Al igual que el resto de componentes, los conectores para dispositivos gráficos (pantallas) han sufrido una evolución, y aunque alguno de ellos tiene muchos años, hoy día se sigue utilizando.

El conector **VGA** es un conector analógico que sólo envía la señal gráfica al dispositivo conectado. Hoy día, aunque se puede considerar obsoleto a nivel tecnológico, sigue estando presente en servidores y en proyectores de gama baja, ya que ofrece la suficiente calidad gráfica.

El conector **DVI** era el sucesor del conector anterior, y podía ser retrocompatible con VGA, aunque la idea es que este conector permitía tener señales digitales. Había diferentes tipos de conectores, dependiendo del tipo de señal que transportaba. Para portátiles también existió una versión "mini" y otra "micro", que era más delgada.



El conector **HDMI** hoy en día es un estándar muy utilizado, sobre todo en televisiones, ya que permite enviar la señal gráfica y audio. Aunque el conector se mantiene igual, existen distintas revisiones que permiten mayor transmisión de datos (para tecnologías nuevas como HDR, audio con más canales, …)

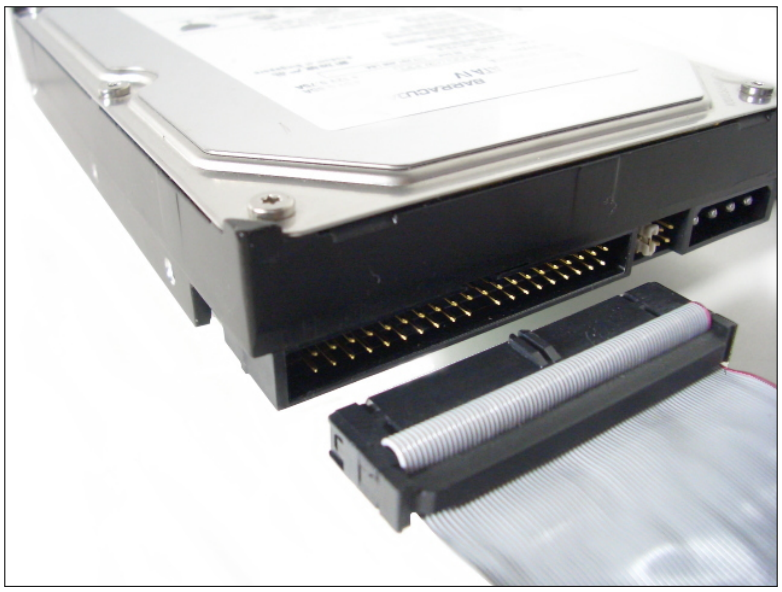


DisplayPort es un interfaz digital desarrollado por la Asociación de Estándares Electrónicos de Vídeo (VESA). Es libre de licencias, y opcionalmente permite la transmisión de audio y datos (por ejemplo USB).

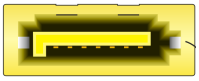
2.2.9.2. Conectores de dispositivos de almacenamiento

Para dispositivos de almacenamiento, como discos duros, CD-ROMs… ha habido varios tipos de conectores que es importante conocer.

Parallel-ATA, o IDE, era un conector que se utilizaba en discos duros y lectores de CD-ROM, con el que a través de un único cable podían conectarse dos dispositivos. Debido a esto, los dispositivos tenían un "*jumper*" que identificaba si era el "maestro" o "esclavo".



Disco duro, cable IDE y jumper. Fuente: wikipedia



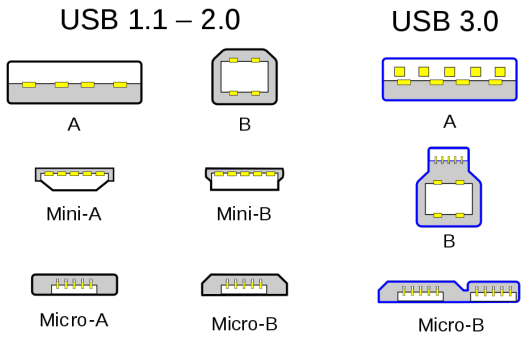
Serial-ATA, o SATA, es la evolución del conector anterior. Es un conector más pequeño pero que permite más velocidad. Ha habido distintas versiones (siendo todas retrocompatibles), siendo la última SATA 3 (subversión 3.5), que admite hasta 600MB/s.

M.2, es el nuevo conector que incluyen los nuevos discos duros de estado sólido **NVMe**.

2.2.9.3. USB

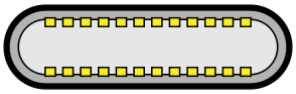
El USB (*Universal Serial Bus*) es un estándar que define los cables, conectores y protocolos que más se utiliza hoy en día para conectar ordenadores y una infinidad de tipos de dispositivos.

Aunque se creó a mediados de los 90, su conector más utilizado (el tipo-A) apenas ha variado (buscando ser retrocompatible), pero sí su velocidad.



Distintos conectores USB. Fuente: wikipedia

Para la nueva especificación USB-C trajo consigo un nuevo conector que es reversible (se puede conectar en ambas direcciones), con idea de reemplazar todos los conectores anteriores. La idea es que ambos dispositivos (anfitrión y huésped) se hace uso del mismo conector y el cable que los una llegue a ser universal, teniendo en cuenta la especificación que utilice.



USB-C. Fuente: Wikipedia

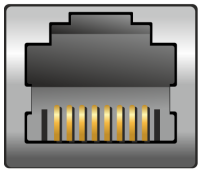
Las placas bases tienen varios conectores USB typo-A soldados para poder realizar conexiones, pero también tiene conexiones en la placa para poder tener más USB (por ejemplo, gracias a los que vienen en la caja).

Estos conectores externos tienen un cable que se conectan a unos pines en la placa base, que dependiendo del protocolo, tendrán una forma u otra (por eso es importante ver el manual de la placa base).

Conector USB-2 y USB-3 en placa base (pines)

2.2.9.4. Conexiones de red

Para que nuestro ordenador se pueda conectar a una red, las placas base ya tienen incorporado al menos un conector para ello.



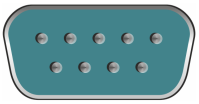
El conector **RJ-45** es el utilizado en redes de ordenadores que contiene cuatro pares de cables de cobre para realizar la transmisión de datos a través del protocolo **ethernet**, que veremos más adelante.

El conector **SMA** se utiliza en algunos tipos de antenas WiFi desmontables que nos podemos encontrar en algunos routers, placas base, tarjetas PCI… Es un conector enroscable y fácilmente desmontable.

2.2.9.5. Otros conectores

PS2 era el conector utilizado para realizar la conexión de teclados y ratones antes de la llegada del USB. Normalmente venía con dos colores, violeta para el teclado y verde para el ratón, ya que aunque el conector era el mismo, el teclado requiere en ambos lados un colector abierto para permitir la comunicación bidireccional.

El **jack** de 3,5mm es el conector más utilizado para audio analógico desde hace muchos años en el ordenador, a pesar de que su aparición (en distinto tamaño) es del año 1878. Hoy en día las placas base tienen distintos conectores para introducir estos jacks dependiendo de si es para altavoces, micrófono, sonido envolvente…



El conector **RS232** (también conocido como "puerto serie"), es un interfaz que permite el intercambio de datos binarios entre dos equipos. Originalmente para mandar información a un terminal de datos, y posteriormente muy utilizado en switches. Aunque hoy en día las placas base no tienen el conector externo, suelen tener los pines de conexión para añadir un adaptador.

Desde que los ordenadores se hicieron populares en la década de los 80 hasta ahora ha habido muchos otros tipos de conectores que han llegado a los ordenadores de consumo personal.

También ha habido otros muchos tipos de conectores que se han quedado en el ámbito más profesional (transceiver SFP, conectores SAS para discos duros, …) por lo que es imposible abarcarlos a todos.

2.2.9.6. Conector, protocolo y cables: errores frecuentes

Hemos visto distintos conectores que durante años su forma física no ha variado, buscando ser retrocompatible con versiones anteriores, pero que la transmisión de velocidad sí se ha ido incrementando a lo largo de los años.

Algunos ejemplos:

- PCI
- USB
- SATA
- HDMI
- DisplayPort

Es por eso que es importante entender y comprender que la forma del conector hoy en día no nos tiene por qué indicar la velocidad de transmisión máxima que acepta el dispositivo conectado, y por eso deberemos ir a las especificaciones técnicas de la placa base o el dispositivo concreto.

Por otro lado, **con los cables sucede lo mismo**. Debemos confirmar y asegurar que los cables que utilizamos van a ser capaz de transmitir la velocidad máxima que tanto dispositivo como placa base aceptan.

¡Cuidado!

Es importante conocer las especificaciones técnicas de cada componente y cable que usemos, para no realizar ningún cuello de botella.

2.2.10. Caja del ordenador

La caja del ordenador, o chasis, es la estructura metálica donde se introducen (de manera ordenada, y anclando mediante tornillos) los distintos componentes que hemos visto hasta ahora.

Caja con ordenador instalado. Fuente: wikipedia

Existen distintos tipos de caja, normalmente variando el tamaño, por lo que es importante adecuar la caja a los componentes que queramos albergar dentro.

¡Atención!

Cuidado con comprar una caja demasiado pequeña y que luego la placa base, o la anchura de la tarjeta gráfica no entre

Los servidores cuentan con unas cajas de tamaño estandarizados en altura, denominado "Unidad de Rack" (*rack unit* en inglés, o simplemente "U"), cuya unidad equivale a 44.45 milímetros. De esta manera los servidores contarán con una altura fijada para poder ser instalados en un bastidor *rack*.

Servidor Dell 1U de altura

2.3. Arranque de un ordenador

Una vez el hardware está instalado, es momento de entender cómo funciona el sistema de arranque de nuestro ordenador hasta llegar al Sistema Operativo.

Toda la secuencia de arranque se puede dividir en distintas etapas que vamos a ver a continuación:

1. Se acciona el botón de encendido del equipo (o arranca el equipo tras un reinicio).
2. Se carga la BIOS/UEFI y se comienza a ejecutar.
3. Se realiza el *Power-On Self-Test* (POST), que es una secuencia que comprueba el estado de los componentes hardware. En caso de que algún componente no parezca estar correcto, la placa base emitirá sonidos. **Si este paso falla, no continuará el proceso**.

   Información

   Es habitual que si algún componente falla, la placa base emita uno o varios tonos (si tiene un pequeño altavoz) de distinta duración

   - Comprobación del procesador
   - Se comprueba el estado de la RAM y la cantidad instalada
   - Comprueba el estado de la memoria de vídeo.
   - Inicializa los sistemas de acceso a dispositivos de almacenamiento (IDE, Serial-ATA, NVMe…).
4. La BIOS/UEFI comprueba el número de discos duros existentes. Se comprueba la tabla de particiones del disco duro indicado como primario para el arranque.
5. Se ejecuta el gestor de arranque de la tabla de particiones marcada como arrancable.
6. El gestor de arranque prepara todo lo que necesita el Sistema Operativo para funcionar, lo carga y le transfiere la ejecución a él.

3. Particionado y sistemas de ficheros

Anteriormente hemos visto que los sistemas de almacenamiento (más conocidos como discos duros) son dispositivos que pueden ser de distintos tipos, capacidades, tamaños...

A la hora de usar un disco duro en nuestro Sistema Operativo, tenemos que tener en cuenta al menos dos cosas:

- **Tipo de particionado**.
- **Sistema de ficheros**.

Dependiendo del Sistema Operativo, y el modo que elijamos durante la instalación, nos dará más opciones o menos a la hora de elegir, modificar o personalizar algunas de estas opciones.

Para tomar las decisiones correctas, deberíamos conocer, al menos, los siguientes detalles:

- Número de discos duros instalados en el equipo.
- El tipo de cada uno (mecánico, SSD, NVMe...).
- Tamaño de los mismos.
- Función que va a realizar el equipo.

De esta manera, podremos realizar un análisis previo de cómo queremos realizar la instalación.

3.1. Particionado MBR y GPT

Los discos duros se dividen en lo que se llaman particiones. Es una manera de realizar divisiones lógicas del espacio, que actúan de forma independiente entre sí.

El símil del particionado de disco duro es un armario: tenemos una cantidad de hueco posible, que decidimos "particionar" añadiendo estanterías tanto horizontales como verticales, donde almacenar distintos tipos de ropa, de manera independiente, en cada uno de esos compartimentos.

A la hora de crear la tabla de particiones de un disco duro podemos elegir entre los siguientes tipos:

- **MBR**: De *Master Boot Record*, o también conocido como **tabla de particiones DOS**.
- **GPT**: De *GUID Partition Table*, propuesta por la especificación EFI, más moderna que la anterior.

En la siguiente tabla se pueden identificar algunas de las diferencias más destacadas entre ambos sistemas, y tal como se puede ver, las ventajas que se incorporan en GPT hacen que el sistema sea más robusto:

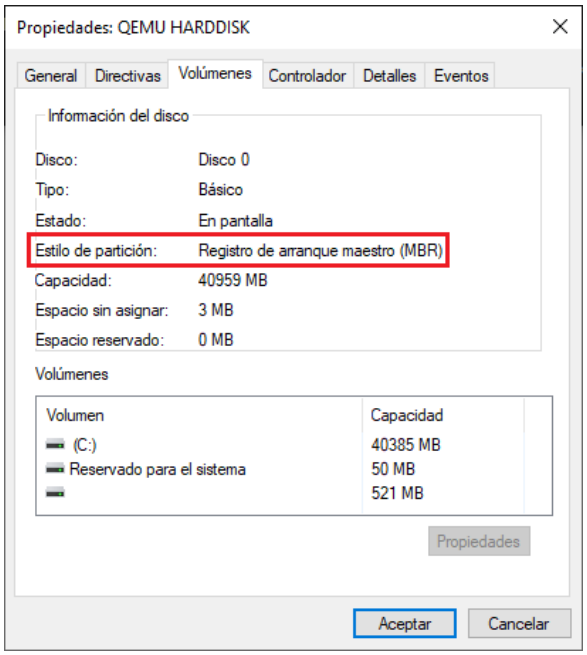
	MBR	GPT
Tamaño máximo de partición	2 TB	18 Exabytes
Nº de particiones primarias	4	Ilimitado (Windows reconoce 128)
Tabla de particiones	Al inicio	Al inicio y al final (backup)
ID de la partición	Se almacena en la partición	Identificador único de GUID
Soporte de arranque múltiple	Débil	Las entradas del gestor de arranque están en una partición separada

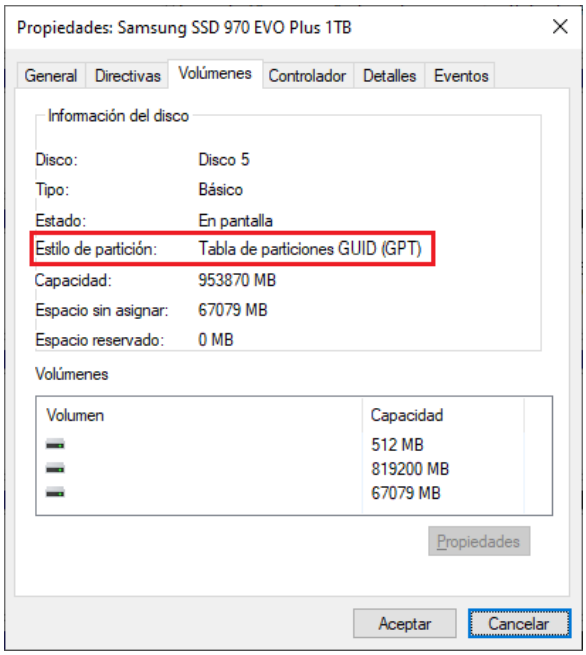
Para comprobar cuál es el sistema de particiones de nuestro disco duro lo podemos hacer desde:

- En **Windows**: Desde el administrador de discos duros.

- En **GNU/Linux**: Con GParted, fdisk, ...

A continuación se pueden ver capturas de pantalla de distintos discos en un equipo Windows.





3.2. RAID

RAID (en inglés *redundant array of independent disks*) es un grupo o matriz redundante de discos independientes que hace referencia a un sistema para crear una unidad de almacenamiento lógica utilizando varias unidades de almacenamiento. De esta manera, **el sistema operativo sólo ve un único almacenamiento lógico, que "por detrás" utiliza varios**.

Dependiendo del tipo de grupo creado, los datos se distribuirán o se replicarán entre las unidades que forman el grupo.

A la hora de crear un sistema RAID, podemos hacerlo de dos maneras:

- **RAID por hardware**: Existe un hardware especializado que se encarga de realizar las operaciones del sistema RAID, así como de crearlo, gestionarlo, hacer las operaciones de paridad necesarias... Podemos diferenciar dos sub-tipos:
  - **Placa Base**: Algunas placas base tienen un controlador especializado que se encarga de la creación del RAID, que se puede gestionar desde el sistema UEFI.
  - **Tarjeta controladora especializada**: Este sistema es el **más profesional**. Los discos duros en lugar de ir conectados a la placa base, se conectan a esta tarjeta (normalmente a través de unos cables especiales que se conectan por SATA al disco duro y mediante conectores especiales a la tarjeta). Hay veces que cuentan con una batería propia (para controlar la escritura en caso de que el servidor se quede sin alimentación) y con un menú propio de configuración.
- **RAID por software**: El Sistema Operativo se encarga de crear y gestionar el sistema RAID. En caso de necesitar realizar paridad, la CPU se encargará de ello, quitando recursos a otros procesos.

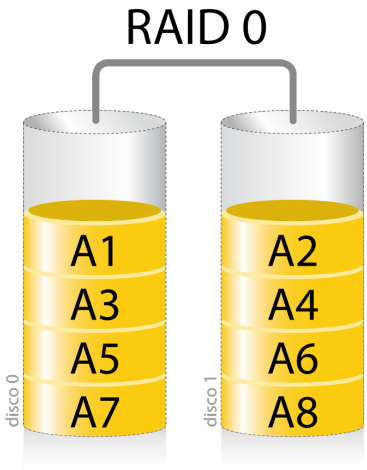
Tal como se puede ver en la Wikipedia, existen distintos tipos de RAID, que también pueden combinarse entre sí, pero a continuación se explicarán los más estándares.

3.2.1. RAID 0

Un sistema RAID 0 distribuye los datos de manera equitativa entre dos o más discos, sin hacer uso de información de paridad. RAID 0 **no habilita redundancia**, por lo que **si uno de los discos duros del sistema se estropea, se perderán todos los datos**.

Espacio total = sumatorio del espacio de los discos.

Con este sistema se puede conseguir una mayor velocidad de escritura, ya que los datos se escriben de manera simultánea en todos los discos.



Fuente: Wikipedia

¡Cuidado!

No se aconseja utilizar RAID 0 en sistemas con datos importantes.

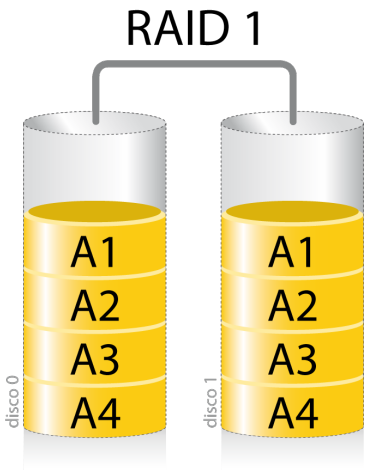
3.2.2. RAID 1

El RAID 1, también conocido como RAID espejo, crea una copia exacta de los datos en dos o más sistemas de almacenamiento.

Resulta útil cuando queremos tener seguridad a la hora de preservar los datos a pesar de no aprovechar la capacidad total del conjunto de los discos. **Si uno de los discos falla, los datos no se pierden**.

El RAID 1 sólo puede ser tan grande como el más pequeño de los discos.

No se mejora el rendimiento en escritura, pero en la lectura el tiempo se reduce al poder usar varios discos de forma simultánea.



Fuente: Wikipedia

3.2.3. RAID 5

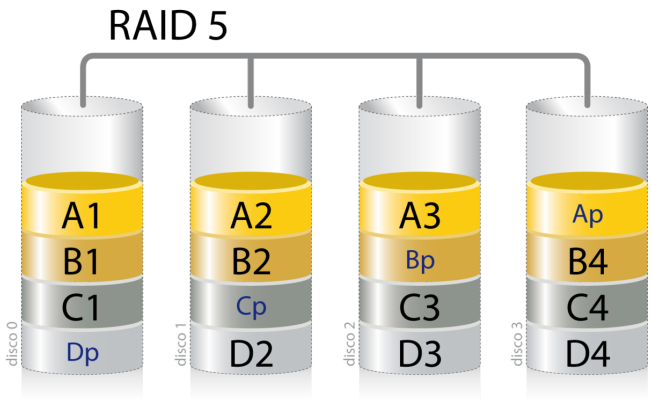
RAID 5 es una división de los datos a nivel de bloques que distribuye la información entre el conjunto de discos que lo forman, y añadiendo **paridad**. La paridad se utiliza para poder corregir errores o generar los datos en caso de pérdida de uno de los discos.

Para crear un sistema RAID 5 se necesitan al menos 3 discos duros, quedando uno de ellos “inutilizado” en

lo que se refiere a guardar información.

El volumen total = (n-1)*tamaño de disco más pequeña, donde "n" es el número de discos que forman el RAID 5. Si tenemos 3 discos de 2TB, el tamaño total será = (3-1)*2TB = 4TB.

En caso de rotura de un disco duro, los datos se mantienen, pero el sistema RAID 5 no soportaría la rotura de un segundo disco duro. Es por ello que cuando sucede esto, hay que sustituir el disco dañado para que se regeneren los datos en el nuevo disco duro, gracias al sistema de paridad.



Fuente: Wikipedia

3.2.3.1. Discos spare/reserva

Existen variantes de RAID 5, y algunas controladoras hardware también implementan, la posibilidad de tener discos denominados "***hot spare***", o **de reserva**.

Estos discos están conectados (a la controladora, o la placa base) pero sin estar dentro del grupo RAID. Entrarán a formar parte del grupo en el momento en el sistema detecte que uno de los discos pertenecientes al grupo está fallando, y por tanto el sistema RAID esté degradado.

De esta manera, al entrar en el grupo, el RAID comenzará a arreglarse de manera automática sin tener que esperar a que el administrador de sistemas se entere de que ha habido algún error.

3.3. Sistemas de ficheros

Los sistemas de ficheros controlan cómo se almacenan y recuperan los datos. Sin un sistema de archivos, los datos colocados en un medio de almacenamiento serían un gran cuerpo de datos sin manera de saber dónde termina un dato y comienza el siguiente.

Información

Un sistema de ficheros nos proporciona una "*vista lógica*" de cómo se almacenan los datos.

Las funciones principales de los sistemas de ficheros son:

- Asignar espacio a los archivos.
- Administrar el espacio libre (reorganizándolo en caso necesario).

- Provee una API a los programas para crear, borrar, modificar y cerrar los ficheros.
- Permite gestionar el acceso a los ficheros (permisos de ficheros).
- Optimizar el rendimiento de acceso.

3.3.1. Ficheros

Un fichero (o archivo) es una secuencia de bytes almacenado en un dispositivo que **es identificado por un nombre y normalmente una extensión**. Los ficheros pueden ser almacenados en directorios, y estos a su vez en otros directorios.

En inglés sí existe distinción entre ***file*** y ***archive*** siendo la diferencia:

- **File**: Es un conjunto de información que se almacena de manera conjunta. Cómo se almacena esa información depende de cómo se ha diseñado el programa que lee y/o escribe esa información.
- **Archive**: Un archivo es un conjunto de ficheros almacenados junto con ***metadata*** (datos que aportan información sobre datos). El ejemplo más claro puede ser un archivo comprimido, archivos que permiten distribuir programas, ...

Los ficheros cuentan con un nombre para que los podamos identificar. Dependiendo del sistema de ficheros puede ser:

- **Case sensitive**: O sensible a mayúsculas/minúsculas. Esto quiere decir que “hola.txt” y “HOla.txt” son dos ficheros distintos.
- **Case insensitive**: O insensible a mayúsculas/minúsculas. No puede haber dos ficheros con las mismas letras en el mismo orden, aunque sean mayúsculas y minúsculas. “hola.txt” y “HOla.txt” no pueden convivir en el mismo directorio.

Los ficheros suelen contar con una **extensión**, que va después del nombre seguido de un punto, y normalmente suele ser de 3 letras para la mayoría de casos. Estas extensiones sirven para conocer el tipo de fichero a simple vista, pero no determina el contenido del fichero.

¡Atención!

La extensión no determina el tipo de fichero que es. Si renombramos un fichero con una extensión, el contenido del fichero sigue siendo el mismo

Un pequeño listado de extensiones habituales que podemos identificar:

- **Documentos de texto y ofimáticos**: .txt, .doc, .docx, .xls, .xlsx, .ppt, .pptx, .odt, .odp, .ods, .pdf ...
- **Ficheros multimedia, imagen, audio, vídeo**: .png, .jpg, .jpeg, .tiff, .ps, .bmp, .svg, .gif, .mp3, .mp4, .avi, .mpg, .mpeg, .mkv, ...
- **Archivos comprimidos**: .zip, .bz2, .gz, .gzip, .7z, .rar, .r00, ...
- **Archivos de programación**: .c, .java, .class, .py, .php, .rb, .pl, .sh, ...

En los sistemas Windows las extensiones están ocultas en el explorador de archivos, por lo que para un usuario sin conocimientos, no existen.

3.3.2. Sistemas de ficheros más utilizados

Existen muchos sistemas de ficheros, y cada Sistema Operativo suele utilizar uno por defecto, que es el que está optimizado para sus funciones. Eso no quita que pueda hacer uso de otros sistemas de ficheros en otros dispositivos.

Información

Los Sistemas Operativos utilizan un sistema de ficheros predeterminado, pero suelen poder acceder a dispositivos que hagan uso de otros

A continuación se expone una tabla con los sistemas de ficheros predeterminados de distintos Sistemas Operativos y los sistemas de fichero que pueden leer por defecto:

Sistema Operativo	Sistema de Ficheros	También puede leer
DOS, Windows 95		
Windows 95 OSR2, Windows 98	FAT16, FAT32	
Windows NT, 2000, XP,... Windows 10, Windows 11	NTFS (varias versiones)	FAT16, FAT32
Windows Server 2012, Windows 11	NTFS, ReFS	FAT16, FAT32, NTFS
GNU/Linux	Ext4, ReiserFS	La gran mayoría
MacOS	HFS+, APFS	FAT16, FAT32

Existen programas y drivers para permitir que los sistemas operativos puedan leer otros sistemas de ficheros que no pueden en origen. **A veces pueden tener limitaciones (en la escritura, permisos, ...)**.

En la Wikipedia existe un listado con distintas características de un gran listado de sistemas de ficheros. A continuación parte de esa información:

	FAT 32	NTFS	ReFS	EXT-4	APFS

Continúa en la siguiente página

Nombre de fichero max.	8.3 (255)	255	255	255	255
Volumen max.	16TB	16TB	1YB	1 EB	?
Tamaño max. fichero	4GB	16TB	16EB	16TB	8 EB
Permisos	No	Si	Si	Si	Si
Compresión	No	Si	No	No	Si
Cifrado	No	Si	No/Si	Si	Si

Debido a que los sistemas de ficheros van adquiriendo características nuevas, es posible que algunas no estuviesen en las primeras versiones y fueran añadidas a *posteriori*.

3.4. Jerarquía de directorios

A la hora de almacenar la información en un sistema de ficheros, se hace siguiendo una jerarquía de directorios. Esta jerarquía de directorios es creada por el sistema operativo durante la instalación y en ella se almacenan los ficheros necesarios para el sistema.

Posteriormente la jerarquía puede ser expandida para guardar información de usuarios, o para que programas y servicios puedan guardar y utilizar información. **La estructura inicial debe ser conservada y no se debe modificar si no estamos seguros de lo que hacemos**.

¡Cuidado!

Los directorios y ficheros creados durante la instalación del Sistema Operativo forman parte de una estructura "inmutable". Si se modifican/mueven es posible que el sistema deje de funcionar. ¡CUIDADO!

3.4.1. Sistemas Windows

Microsoft comenzó a utilizar el sistema denominado "letra de unidad" con el lanzamiento de MS-DOS, aunque no fueron los primeros en utilizarlos.

Este sistema utiliza una letra del alfabeto para identificar los volúmenes o unidades lógicas de sistemas de almacenamiento.

Tanto en MS-DOS como en Windows, se denomina de la siguiente manera:

- **A:** Es la unidad de disquete.
- **B:** Reservada para la segunda unidad de disquete.

- **C:** Partición o disco duro principal. Donde se instala el sistema operativo y los programas.
- **D:** Reservado para el CD-ROM/DVD-ROM.
- **E:** hasta **Z:** para otro discos duros, particiones, CD-ROMs/DVD-ROM, sistemas extraíbles de almacenamiento...

También se puede utilizar este sistema para carpetas compartidas por red, aunque no es necesario.

3.4.2. GNU/Linux

En los sistemas operativos GNU/Linux (aunque también sucede en las variantes UNIX como FreeBSD y MacOS) la jerarquía comienza en un único punto representado con "**/**" denominado **directorio raiz** o "barra".

El ***Filesystem Hierarchy Standard*** (FHS) es la referencia para conocer dónde se deben almacenar los ficheros dentro de la jerarquía, siendo algunos de los directorios más conocidos. Más adelante hablaremos de ello.

4. Software

El software es el elemento lógico que forma parte del ordenador, todo aquello que es "intangible". Es el **conjunto de programas y datos** que permiten manejar el hardware, controlando y coordinando su funcionamiento para que realice las tareas deseadas.

Dentro del software podemos diferenciar distintos tipos (aunque existen otras categorizaciones):

- **Software de sistema**: Son aquellos programas que permiten la administración del hardware (de los recursos físicos). Crean una capa de abstracción entre el hardware y los programas que utiliza el usuario. Dentro de este apartado podemos incluir: sistemas operativos, *drivers* (controladores), herramientas de diagnóstico, ...
- **Software de desarrollo**: Son aquellos programas que los desarrolladores de software utilizan para crear, depurar y mantener programas.
- **Software de aplicación**: En esta categoría entran los programas que tienen una función específica y que normalmente son utilizados por los usuarios finales del sistema.

A partir de aquí profundizaremos en cada uno de estos aspectos en los capítulos siguientes.

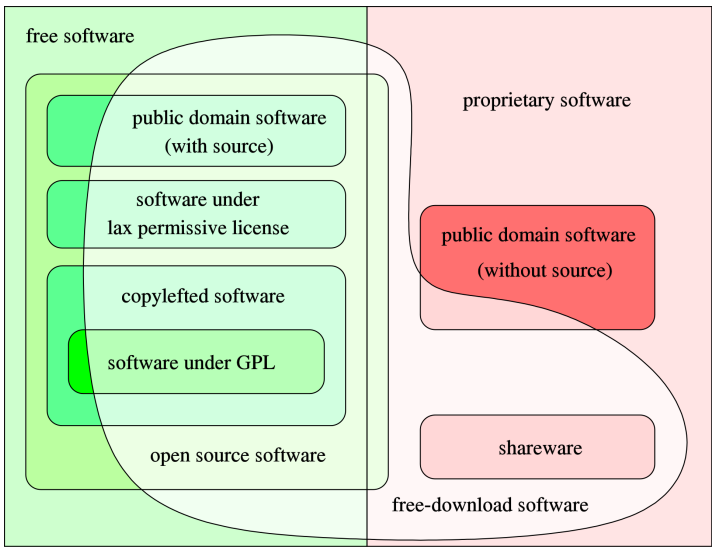
4.1. Licencias de Software

A la hora de utilizar cualquier tipo de software lo habitual es que tengamos que aceptar una licencia, que es un contrato entre el licenciante (autor o titular de los derechos de explotación y/o distribución del software) y el licenciatario (el usuario o consumidor final).

Las licencias incluyen una serie de términos y condiciones, que son un conjunto de permisos que el autor otorga al usuario.

Algunos ejemplos de condiciones en una licencia pueden ser:

- Definir el tipo de uso.
- Limitar/permitir la distribución del software.
- Plazo de cesión de los derechos.
- Limitar/permitir la modificación del software.



Esquema básico de licencias. Fuente: Wikipedia

4.1.1. Software privativo

En contraposición al Software Libre se encuentra el software privativo, el cuál **no permite** el acceso al código fuente, ya que sólo se encuentra a disposición de los desarrolladores, no permitiendo su libre modificación, adaptación o distribución.

Dentro de este tipo de software podemos encontrar algunos tipos de licenciamiento conocidos:

- **Freeware**: Normalmente se utiliza para el software que es gratuito pero que no permite la modificación del mismo.
- **Shareware**: El usuario puede evaluar de forma gratuita el producto, durante un tiempo limitado o con una funcionalidad limitada.

Entre el software más utilizado que es software privativo nos podemos encontrar a: Windows, Photoshop, juegos comerciales, ...

4.1.2. Software de dominio público

Es aquel que el autor decide publicarlo bajo el denominado dominio público, lo que hace que cualquiera pueda acceder al código fuente, modificarlo **pero también publicarlo bajo una licencia no libre**.

Este tipo de licencia suele ser aplicada a libros y música, por ejemplo, tras 70 años después de la muerte del autor.

5. Sistemas Operativos

El Sistema Operativo (**SO**) es el *software* que controla el *hardware* del ordenador, creando recursos lógicos y servicios que otro software y los usuarios pueden utilizar.

De manera muy simplificada, el núcleo del Sistema Operativo crea una abstracción del hardware real, limitando el acceso al mismo a los programas para gestionar su buen funcionamiento.

5.1. Funciones principales

Entre las funciones más importantes de las que se encarga el Sistema Operativo, destacaremos:

- **Gestionar los procesos en ejecución**: El sistema operativo se encarga de cargar el proceso que queramos ejecutar y de asignarle los recursos necesarios para su correcto funcionamiento. También se encarga de las posibles peticiones de los procesos, de pausarlos, de asignarles el procesador cuando le corresponda...
- **Gestionar la memoria RAM**: La memoria RAM es finita en el ordenador, y cuando ejecutamos un proceso este se carga en RAM para poder ser ejecutado. Si un proceso necesitase más memoria, será el sistema operativo el que se encargue de asignarle más en caso de que haya libre. Cuando los procesos mueren, esa memoria debe ser liberada, para que otros procesos puedan utilizarla.
- **Administrar la CPU** gracias a un algoritmo de programación: El sistema operativo coordina el uso de la CPU entre las diferentes tareas y procesos que se ejecutan en el sistema. Utiliza algoritmos de programación para determinar el orden y la prioridad de ejecución de los procesos, asegurando un uso equitativo de los recursos de la CPU.
- **Gestionar las entradas y salidas** de datos a través de los periféricos: Además de direccionar las entradas y salidas de datos, el sistema operativo proporciona controladores (***drivers***) para interactuar con los periféricos de entrada y salida, como teclados, mouse, impresoras, discos duros externos, entre otros. Estos controladores permiten que los dispositivos se comuniquen correctamente con el sistema operativo y las aplicaciones.
- **Asegurar el buen funcionamiento del sistema**: El sistema operativo gestiona información esencial para el funcionamiento del sistema, como la tabla de procesos, la tabla de archivos abiertos y otros datos relevantes. Además, realiza la gestión del rendimiento para asegurar un funcionamiento óptimo del sistema.
- **Seguridad**: El sistema operativo proporciona un mecanismo de autenticación y autorización para garantizar que los usuarios accedan solo a los recursos y funciones para los cuales tienen permisos. Esto incluye la gestión de cuentas de usuario, contraseñas y asignación de privilegios.
- **Administrar los archivos**: El sistema operativo maneja las operaciones relacionadas con la gestión de archivos, como la creación, modificación, eliminación y acceso a los archivos en el sistema de almacenamiento.

5.2. Breve historia de los sistemas operativos

Desde la creación de los primeros ordenadores, y al igual que ha sucedido con estos, los sistemas operativos han sufrido una evolución. A continuación un breve repaso, pudiendo ver toda la información en Wikipedia.

- **Sin sistema operativo**: Los primeros ordenadores no contaban con un sistema operativo propio. Los ordenadores electromecánicos sólo podían ejecutar el programa para el que fueron construidos.

Con la aparición de los primeros ordenadores digitales, se programaban en lenguaje máquina y ejecutaban el programa. En caso de querer ejecutar otro programa, había que cargarle el programa a mano.

- **Sistemas Batch**: A principios de los años 1950 aparecen los "monitor residente" que se encargan de cargar los programas leyéndolos de tarjetas perforadas o de cintas para su posterior ejecución.
- **Sistemas "multiprogramados" y tiempo compartido**: En la década de 1960, aparecen los sistemas operativos que son capaces de ejecutar procesos de un programa mientras otro está a la espera de una operación de E/S.

  También aparecen los sistemas multiusuario, y el "*scheduling*" del sistema, que lo que hace es compartir tiempo de ejecución del procesador entre distintos programas, dándoles un tiempo a cada uno.

  A principios de 1970 se crea Unix, un sistema operativo muy importante en la época y del cuál han salido variaciones que siguen hoy en día usándose.
- **Sistemas en red**: Con la aparición de las redes de ordenadores, empiezan a aparecer sistemas operativos con acceso a la red y ofreciendo servicios que pueden ser utilizados por usuarios o por otros ordenadores.
- **Sistemas operativos de escritorio**: En la década de 1980, con la aparición y proliferación de ordenadores personales (PC), empiezan a aparecer sistemas operativos más enfocados a usuarios que no son expertos en informática.

  Aparecen los primeros interfaces de usuario y escritorios (primero con el Macintosh de Apple en 1984) que junto al ratón facilita la usabilidad de los programas.

III

GNU/Linux

1. Introducción a GNU/Linux

1.1. Un poco de historia

Para conocer cómo nació el movimiento GNU y el kernel Linux debemos conocer un poco de historia de la informática y cómo evolucionó en los primeros años.

1.1.1. El nacimiento de Unix

1964-1969 Los laboratorios **Bell** empiezan un proyecto con el **MIT** (Instituto Tecnológico de Massachusetts) y **General Electric** para desarrollar un sistema de **tiempo compartido** ("time-sharing computing"): se llamaría **Multics** (Multiplexed Information and Computing Service).

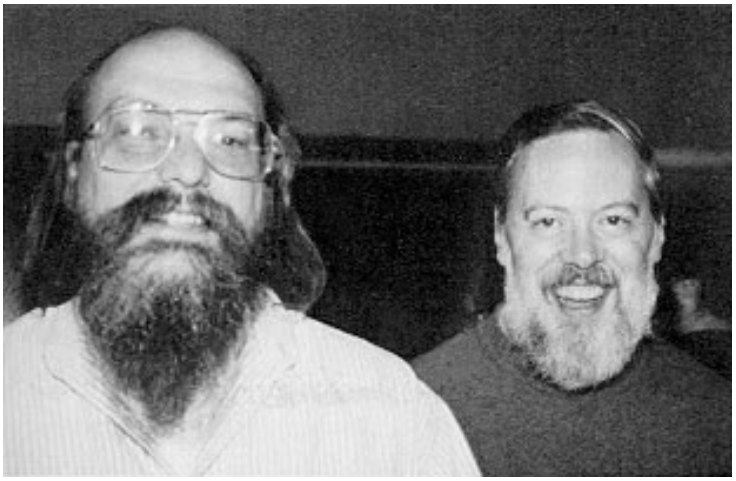
Hasta este momento, los sistemas utilizados eran de un único proceso, la CPU no era compartida por múltiples procesos sino que se ejecutaba por lotes (se les mandaba los procesos a ejecutar y se ejecutaban en orden).

Multics obtuvo licencia libre en el 2007. En Diciembre del 2016 salió la última versión 12.6f.

1969 Uno de los desarrolladores de Multics, Ken Thompson, decidió escribir su propio sistema operativo. Ken Thompson es conocido también por crear el lenguaje de programación **B**, el sistema de codificación de caracteres UTF-8 y el lenguaje de programación Go, entre otras cosas.

A Ken Thompson se le une Dennis Ritchie y otros, y empiezan a programar un sistema de ficheros jerárquico, el concepto de procesos de computación, ficheros de dispositivos, un intérprete de comandos, … El resultado de lo programado era más pequeño y simple que Multics, lo que se convertiría en Unix. En Agosto ya tendrían el sistema operativo, se auto-gestiona, tenía un assembler, un editor y una shell de comandos.

Dennis Ritchie es conocido también por crear junto con Ken el lenguaje de programación **C** (aparece por primera vez en 1972).



Ken y Dennis. Origen: Wikipedia

1970 En ese momento el nuevo sistema operativo se llamaba **Unics** (*Uniplexed Information and Computing*

Service, un juego de palabras en contraposición a Multics). No tenían todavía dinero de la organización en el desarrollo (era desarrollado por los programadores) y tampoco era multitarea todavía.

A finales de año el sistema ya era conocido como **UNIX**, y se había portado a la máquina PDP-11.

Las primeras versiones de Unix incluían el código fuente para que las universidades lo pudiesen modificar y así poder extenderlo a sus necesidades.

1971 El sistema se empieza a hacer complejo y como querían que más usuarios lo usasen, crean el sistema de manuales que es utilizado hoy en día (mediante el comando **"man"**).

Dennis Ritchie y Ken Thompson trabajando en un PDP-11. Origen: Wikipedia.

1973 La versión 4 del sistema es reescrita completamente en C. Hasta este momento el sistema había estado escrito en ensamblador, por lo que no era portable entre distintos tipos de máquinas, aunque la primera versión portada a otra plataforma fue en 1978. Se cree que había "más de 20" instalaciones del sistema.

1974 La versión 5 se licencia para ser utilizada en **instituciones educativas**.

1975 La versión 6 se licencia para poder ser utilizadas por empresas por $20.000 de la época.

1977 La universidad de Berkeley lanza su primera versión de Unix bajo la Berkeley Software Distribution (BSD).

1979 Con la salida de Unix v7, se comienza a portar a los distintos "microordenadores" de la época y a los distintos microprocesadores (Motorola 68000, Intel 8086, …).

1980 Microsoft anuncia su primer Unix para microcomputadoras de 16 bits (Xenix).

1.1.2. El nacimiento de GNU (GNU's Not Unix)

1971 Richard Stallman comienza su carrera en el MIT en el laboratorio de inteligencia artificial.

Es conocido no sólo por el movimiento GNU, si no también por crear GCC y Emacs entre otra gran cantidad de software.

En esa época el software se distribuía de manera abierta para poder ser modificado. Lo habitual era realizar modificaciones para mejorar el software y distribuirlo entre compañeros y universidades.

1982 Richard Stallman quiere modificar el firmware de unas impresoras y el fabricante le pide que firme un acuerdo de no divulgación si le enseñan el código. Esto hace que Stallman se enfurezca y es cuando decide que la situación actual debe cambiar y volver al sistema de intercambio de software anterior.

Richard Stallman: Wikimedia

1983 Se anuncia el nacimiento del proyecto **GNU**, cuya finalidad es la de construir un sistema operativo completamente libre, compatible con Unix. La idea es dar a los usuarios la libertad y el control de sus ordenadores.

1985 Se lanza el manifiesto GNU, y ya cuenta con un editor de texto (Emacs), compilador de C, una shell, varias utilidades … El núcleo inicial todavía no es funcional.

1986 Richard Stallman escribe y publica la definición de lo que es Free Software (Software Libre) a través de la Free Software Foundation.

Más adelante veremos a qué se refiere sobre libertad en el software.

1.1.3. El nacimiento de Minix

1987 Andrew S. Tanenbaum crea Minix como propósito educativo y para enseñar cómo funciona un sistema operativo.

1991 Sale la versión 1.5 de Minix y es portada a distintas arquitecturas (IBM, Motorola 68000, Amiga, Apple Macintosh, …).

1992 Debate con Linus Torvalds sobre la arquitectura del kernel Linux (núcleo monolítico) en lugar de usar un micronúcleo.

1.1.4. El nacimiento de Linux

1991 Un estudiante en la universidad de Helsinki, Linus Torvalds, comienza un proyecto personal escrito para su nuevo ordenador, un PC con procesador 80386.

El desarrollo comienza bajo **Minix**, usando el compilador **GCC** del movimiento GNU (GCC = GNU Compiler Collection).

El proyecto termina convirtiéndose en un kernel de un sistema operativo y escribió al grupo de noticias de Minix diciendo:

"Hola a todos los que estáis ahí fuera usando minix.

Estoy haciendo un sistema operativo (libre), (solamente por aficion, no será grande ni profesional como el GNU) para clones 386(486) AT.

…

PD. Sí – está libre de cualquier código de minix, y tiene un sistema de ficheros multi-hilo. NO es portable (usa el cambio de tareas del 386 etc), y probablemente nunca soporte otra cosa que no sean los discos duros AT, porque es todo lo que tengo :-(. "

Linus torvalds. Origen: Wikipedia

1992 Originalmente la licencia de Linux era propia e impedía el uso comercial de Linux. En la versión 0.99 esto cambia y se cambia a la licencia GNU Public License (**GPL**).

1993 El proyecto cuenta con más de 100 desarrolladores. El kernel se adapta al entorno del proyecto GNU. Nace la distribución **Debian** (una de las más importantes a día de hoy)

Debian

1994 Se libera la versión 1.0. El proyecto XFree86 se une y Linux consigue interfaz gráfico. Nacen las primeras distribuciones comerciales **Red Hat** y **Suse**.

1998 Empresas como **IBM**, **Compaq** y **Oracle** anuncian que apoyan a Linux. Nace el interfaz gráfico **KDE**.

1999 Nace el interfaz gráfico **GNOME** como reemplazo a KDE, ya que KDE hacía uso de una librería propietaria en aquel momento (QT).

2001 Steve Ballmer (CEO de Microsoft) dice: **"Linux es un cáncer"**.

2002 Se libera OpenOffice (originalmente suite ofimática de Sun Microsystems). Nace Mozilla (hoy día: Firefox).

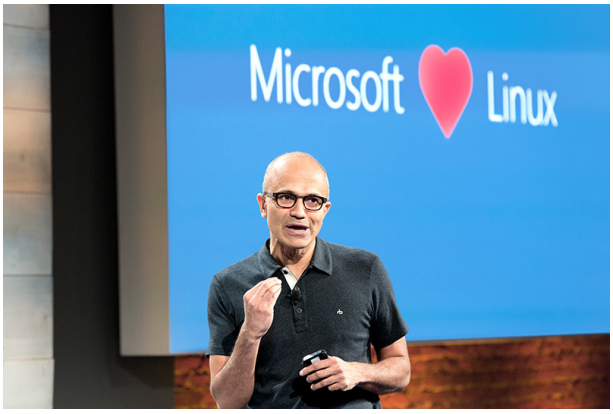
2003 IBM lanza un anuncio para la Linux Foundation: https://www.youtube.com/watch?v=x7ozaFbqg00

2004 Nace **Ubuntu** (basándose en Debian) y Steve Ballmer (CEO de Microsoft) dice que Linux infringe muchas de sus patentes.

2008 Nace **Android**, sistema operativo con kernel Linux. Actualmente es el sistema operativo de móviles que más terminales tiene.

2009 Red Hat iguala a Sun Microsystem en capitalización bursátil (un gran logro simbólico).

2014 Satya Nadella (CEO de Microsoft) muestra en una presentación la siguiente transparencia:

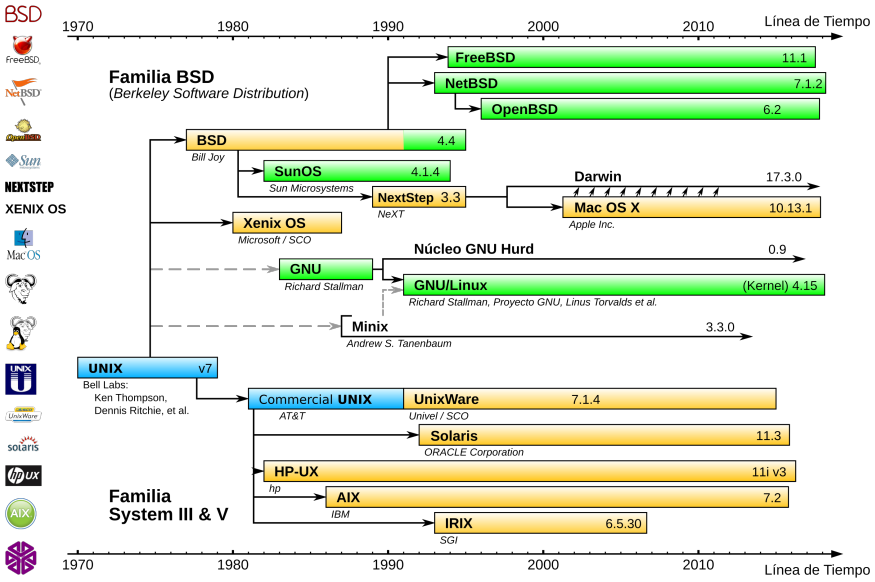


Origen: Wikipedia

2016 Microsoft anuncia WSL (*Windows Subsystem for Linux*) y se puede instalar en Windows 10 y Windows Server 2019. Permite correr ejecutables de Linux nativamente.

1.1.5. Cronograma de sistemas Unix

En el siguiente cronograma se puede ver la línea temporal de los sistemas Unix:



Origen: Wikipedia

1.2. Resumen

Linux es conocido como un sistema operativo libre pero el nombre de Linux se centra única y exclusivamente en el **kernel** (o **núcleo**) del sistema operativo.

El sistema operativo completo debería llamarse **GNU/Linux**, ya que el kernel es una "pequeña" parte (aunque muy importante) dentro de todo el sistema operativo. El resto de herramientas utilizadas en el sistema operativo pertenecen al proyecto GNU.

2. Licencias Libres

2.1. Free Software / Software Libre

En 1986 Richard Stallman saca a la luz la definición de lo que es Free Software (Software Libre) a través de la Free Software Foundation:

Información

La palabra "free" no se refiere a gratis, se refiere a libertad.

(*The word free in our name does not refer to price; it refers to freedom.*)

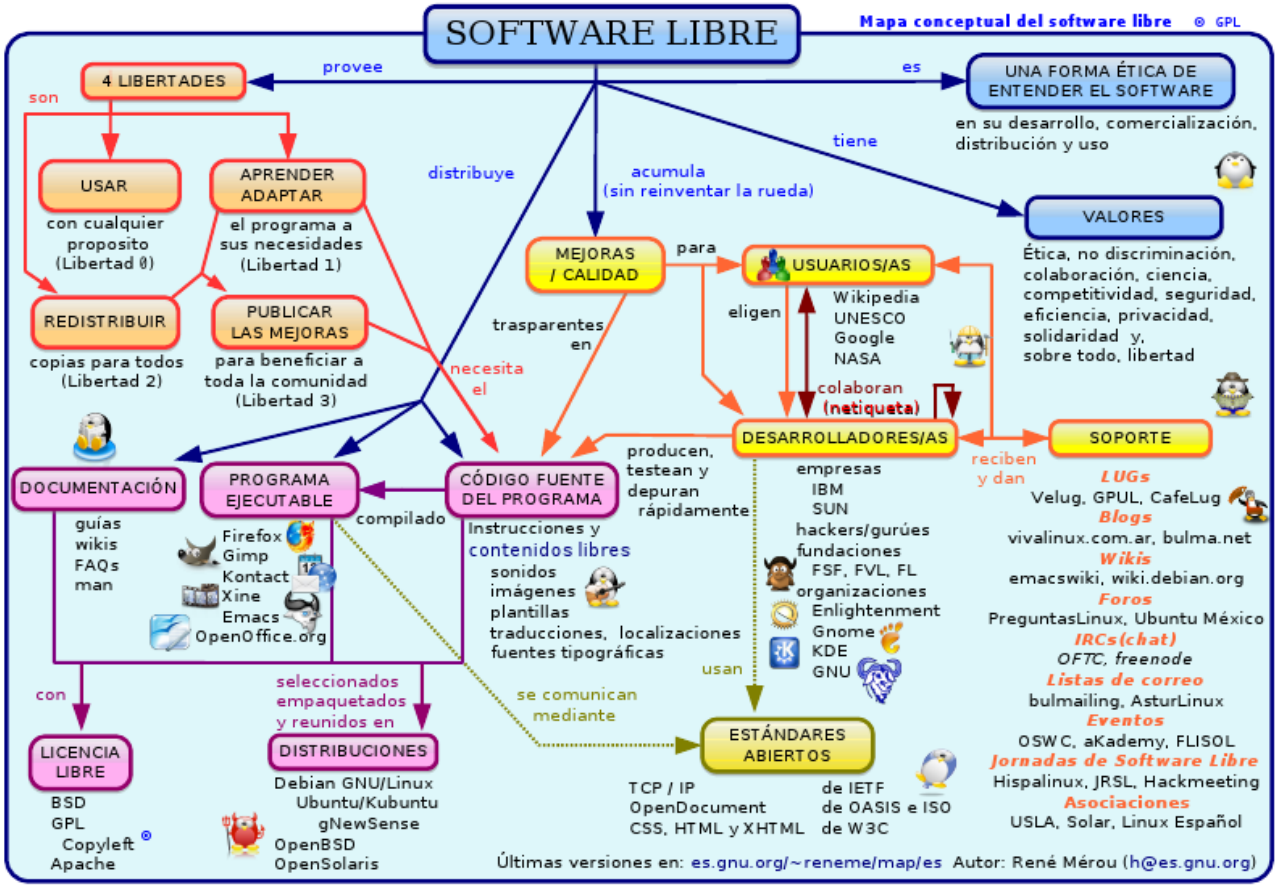
Las libertad en el software se refiere a:

1. La libertad de ejecutar el programa, **para cualquier propósito**.
2. La libertad de **estudiar cómo trabaja el programa, y cambiarlo para que haga lo que usted quiera**. El acceso al código fuente es una condición necesaria para ello.
3. La libertad de **redistribuir copias** para que pueda ayudar al prójimo.
4. La libertad de **mejorar el programa y publicar sus mejoras, y versiones modificadas en general**, para que se beneficie toda la comunidad. El acceso al código fuente es una condición necesaria.

El movimiento del Free Software es un movimiento que tiene que ver más con la filosofía y la ética que con la tecnología en sí misma.

2.1.1. Copyleft y GNU Public License (GPL)

Es una práctica legal que consiste en el ejercicio del derecho de autor (copyright en inglés) con el objetivo de propiciar el libre uso y distribución de una obra, exigiendo que los concesionarios preserven las mismas libertades al distribuir sus copias y derivados (Wikipedia).



Mapa conceptual del Software Libre: Wikipedia

Con esto nació la licencia GNU GPL, la cual permite al usuario final la libertad de usar, estudiar, compartir y modificar el software recibido. Tiene que quedar claro que un programa comercial puede ser Software Libre.

2.1.2. Diferencias con el Open Source

Los programas Open Source son aquellos que podemos ver el código fuente pero esto no quiere decir que podamos modificarlo o adaptarlo a nuestras necesidades.

El Open Source es menos restrictivo que el Software Libre y se puede decir que todo Software Libre es Open Source, pero no todo Open Source tiene por qué ser libre.

2.1.3. Licencias libres más conocidas

Un listado de las licencias libres más utilizadas (en la Wikipedia existe una tabla comparativa):

- GNU GPL
- BSD
- MIT
- Licencia Apache
- Licencia PHP
- Creative Commons (no todas las versiones). Más utilizadas en contenido multimedia.

3. Sistema de ficheros en GNU/Linux

El sistema de ficheros en GNU/Linux, al igual que en Unix, es jerárquico, comenzando en la raíz denominada "/". Partiendo de esta raíz, el resto del sistema de ficheros nace en forma de ramificaciones generando lo que se denominan "rutas de ficheros", que es el camino completo para llegar al mismo.

3.1. Filesystem Hierarchy Standard #{fhs}

Debido a que en GNU/Linux todo se representa como ficheros (discos, dispositivos, programas, …) es necesario que exista un orden a la hora de ser almacenados. Con esa intención nace en 1993 el estándar de la jerarquía de ficheros de Linux, enfocado a reestructurar los archivos. Posteriormente se unieron otros derivados de UNIX (la comunidad de desarrollo de BSD) por lo que terminó adoptando el nombre FHS.

Aún siendo un estándar, no todas las distribuciones lo siguen al pie de la letra, y otros Unix, como MacOS, tienen sus propias rutas especiales.

3.2. Directorios importantes

A continuación se exponen los directorios más importantes del sistema junto con la descripción del contenido que deben de tener:

- **/boot/**: archivos de arranque del kernel, normalmente junto con la configuración utilizada para compilarlos.
- **/dev/**: contiene archivos especiales de bloque que representan los dispositivos del hardware que está corriendo el sistema operativo
- **/etc/**: contiene los archivos de configuración del servidor y de los servicios que corren en él. Está subdividido en directorios por servicios o configuraciones.
- **/home/**: los directorios de trabajo de los usuarios normales del sistema
- **/lib/**: librerías que hacen funcionar a los programas
- **/root/**: es la home del usuario root
- **/var/**: archivos variables del sistema
    - **/var/lib/**: aquí se suelen guardar los ficheros de los programas que "crecen": bases de datos, ficheros caché…
    - **/var/log/**: los logs del sistema

Junto a todos estos directorios, se ha separado los lugares en los que van los binarios, o ejecutables de los programas. Lo habitual es que se encuentren en estas rutas:

- **/bin/**: aplicaciones esenciales del sistema
- **/sbin/**: aplicaciones que en principio sólo debería ejecutar el usuario root o programas de administración del sistema

- **/usr/bin/**: ejecutables de usuario
- **/usr/sbin/**: ejecutables de superusuario

Aunque las rutas de los ejecutables denotan quién debería ejecutar el programa, en la vida real no tiene por qué ser una limitación.

3.3. Dispositivos de almacenamiento y discos duros

En sistemas operativos Windows es habitual que cada partición cuente con una letra para acceder a ella, al igual que ocurre cuando introducimos un dispositivo de almacenamiento externo (un pendrive).

Tal como se ha comentado, en sistemas Unix el sistema de ficheros es una jerarquía, y por tanto todo dispositivo de almacenamiento nuevo deberá estar montado bajo la raíz "/". Hoy día, en distribuciones con escritorio, al introducir un pendrive éste es auto-montado (es accesible) desde la ruta **/media/**, donde aparecerán tantos directorios como discos hayamos conectado.

3.3.1. Almacenamiento permanente

Si queremos que un disco duro nuevo sea permanente en nuestro sistema, podremos montarlo en cualquier lugar de la estructura jerárquica. Debido a este sistema, el usuario final no se tendrá que preocupar en almacenar los ficheros en una ruta distinta, si no que será el administrador el que haya hecho que esa ruta ahora pertenezca a un disco duro nuevo.

Imaginemos que el sistema operativo se ha instalado en un disco duro pequeño de 32Gb de espacio y se está llenando, y el directorio que más ocupa es el directorio de los usuarios. Podremos añadir al servidor un nuevo disco duro montado en /home y por tanto a partir de ahora los datos guardados en /home estarán en un nuevo disco duro más grande.

Ejemplo de discos en un sistema con "lsblk"

```
root@vega:~# lsblk
NAME                      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda                         8:0    0   1,8T  0 disk
└─sda1                      8:1    0   1,8T  0 part /home/backup

sdb                         8:16   0   3,6T  0 disk
└─sdb1                      8:17   0   3,6T  0 part /home/disco4tb
sdc                         8:32   0 447,1G  0 disk
├─sdc1                      8:33   0   529M  0 part
├─sdc2                      8:34   0   100M  0 part
├─sdc3                      8:35   0    16M  0 part
└─sdc4                      8:36   0 446,5G  0 part
nvme0n1                   259:0    0 931,5G  0 disk
```

```
├─nvme0n1p1                 259:1    0   512M  0 part
└─nvme0n1p2                 259:2    0   800G  0 part /home
nvme1n1                     259:3    0 931,5G  0 disk
├─nvme1n1p1                 259:4    0   512M  0 part /boot/efi
├─nvme1n1p2                 259:5    0    90G  0 part /
├─nvme1n1p3                 259:6    0   300G  0 part
│ ├─VMs-ubuntu--20.04--so1 254:0    0    10G  0 lvm
│ ├─VMs-manjaro            254:2    0    20G  0 lvm
│ └─VMs-win10              254:3    0    35G  0 lvm
└─nvme1n1p4                 259:7    0 156,2G  0 part
```

4. Gestión de usuarios locales en GNU/Linux

En las distribuciones GNU/Linux lo habitual suele ser que existan al menos dos usuarios tras una instalación:

- **root**: usuario administrador o súper usuario.
- **usuario no-privilegiado**: durante la instalación de la distribución nos suele preguntar para crear un usuario del sistema, que no tendrá privilegios.

El usuario root, como se ha dicho previamente, es el administrador del sistema, tiene permisos para realizar cualquier tarea dentro de nuestro sistema: instalar paquetes, desinstalarlos, modificar cualquier fichero, realizar formateos... Por lo tanto, el **realizar tareas como usuario root puede ser peligroso si cometemos algún fallo**.

Las buenas prácticas nos dicen que las tareas cotidianas del sistema deberíamos realizarlas como usuario normal y **sólo convertirnos en root cuando sea estrictamente necesario**.

4.1. Creación de usuarios locales

Tras instalar el sistema, veremos que se nos han creado varios usuarios en el sistema, aparte del usuario **root** y el usuario **no-privilegiado**. Para poder ver los usuarios que existen en nuestro sistema podemos verlo en el fichero `/etc/passwd` o podríamos obtener un listado ejecutando el siguiente comando:

Listar usuarios del sistema

```
root@vega:~# cut -d: -f1 /etc/passwd
```

Para crear un usuario:

Crear usuarios del sistema

```
root@vega:~# adduser ruben

Añadiendo el usuario `ruben' ...
```

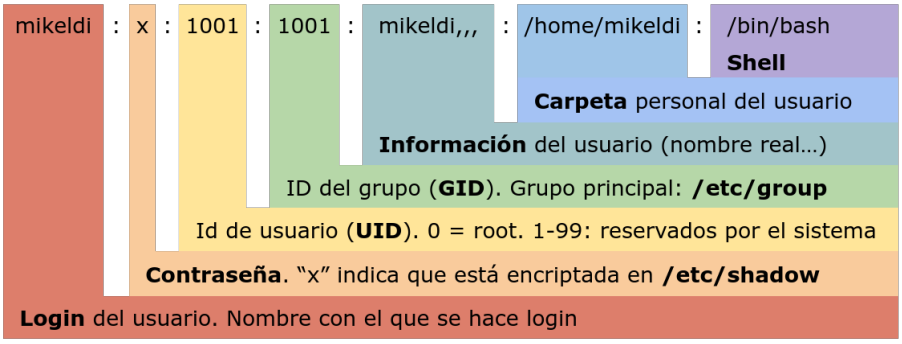
```
Añadiendo el nuevo grupo `ruben' (1001) ...
Añadiendo el nuevo usuario `ruben' (1001) con grupo `ruben' ...
Creando el directorio personal `/home/ruben' ...
Copiando los ficheros desde `/etc/skel' ...
Nueva contraseña:
Vuelva a escribir la nueva contraseña:
passwd: contraseña actualizada correctamente
Cambiando la información de usuario para ruben
Introduzca el nuevo valor, o pulse INTRO para usar el valor predeterminado
    Nombre completo []:
    Número de habitación []:
    Teléfono del trabajo []:
    Teléfono de casa []:
    Otro []:
¿Es correcta la información? [S/n]
```

Y la línea que nos creará en el fichero `/etc/passwd` es:

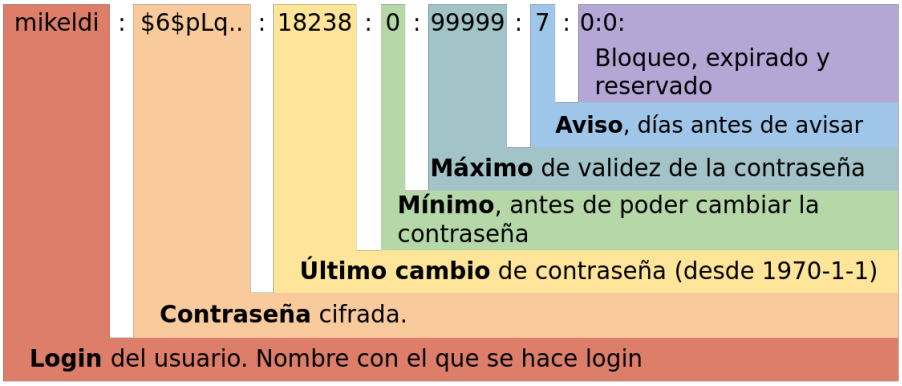
Línea de un usuario en /etc/passwd

```
ruben:x:1001:1001:ruben,,,:/home/ruben:/bin/bash
```

El fichero `/etc/passwd` nos muestra los datos de los usuarios, siendo un fichero que tiene distintos datos separados por ":", siendo cada apartado:



En las primeras versiones GNU/Linux la contraseña de los usuarios aparecía en el propio fichero /etc/passwd, lo que suponía un problema en la seguridad, ya que no estaban cifradas. Actualmente, las contraseñas de los usuarios se almacenan cifradas en el fichero `/etc/shadow`. El fichero es similar al passwd, estando separados los apartados por ":"



En el apartado de la contraseña podemos saber cierta información acerca de la misma ya que tiene el siguiente formato: **"idsalt$hashed"**

- **id**: el algoritmo utilizado para cifrar la contraseña
  - 1 – MD5
  - $2a$ – Blowfish
  - $2y$ – Eksblowfish
  - 5 – SHA-256
  - 6 – SHA-512

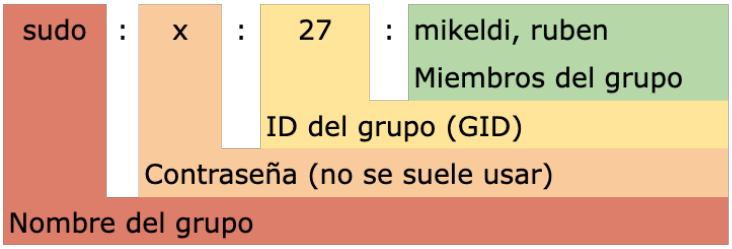
Aparte, también podemos encontrarnos con:

- **Contraseña vacía**: Si no hay contraseña, al pedirnos la contraseña a la hora de hacer login será suficiente con pulsar "intro".
- **!**, *****: la cuenta está bloqueada para la contraseña. El usuario no podrá loguearse utilizando la contraseña. Resulta útil si queremos bloquear el acceso con contraseña pero no con otros métodos (clave pública SSH).
- ***LK***: cuenta bloqueda. El usuario no podrá loguearse.
- ***NP***, **!!**: Nunca se ha puesto una contraseña

4.2. Gestión de grupos

En algunas distribuciones GNU/Linux, al crear un usuario directamente nos crea un grupo para el nuevo usuario. En otras, el usuario pertenece al grupo "users".

Para saber los grupos a los que pertenece un usuario podemos ejecutar el comando `groups`. Los grupos del sistema aparecen en el fichero `/etc/group`, y al igual que los ficheros vistos previamente, están separados por "**:**".



4.3. Permisos de ficheros

En GNU/Linux los ficheros cuentan con 3 tipos de permisos:

- lectura (**r**ead): el usuario puede leer el fichero
- escritura (**w**rite): el usuario puede escribir en el fichero
- ejecución (e**x**ecute): el usuario puede el fichero o puede ver el contenido de un directorio

Todos ello para los distintos usuarios que pueden existir en el sistema:

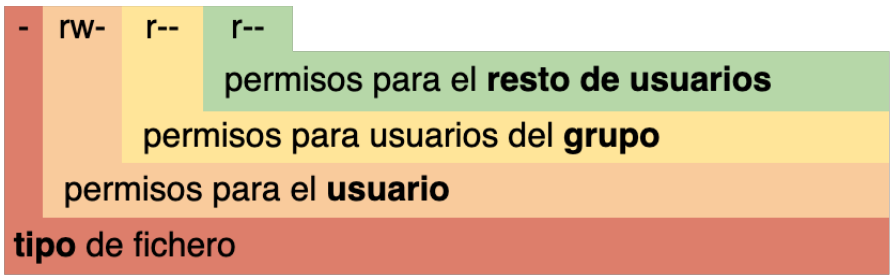
- **dueño del fichero**: la persona que ha creado el fichero
- **grupo**: los usuarios pertenecientes al grupo al que pertenece el fichero tendrán ciertos privilegios
- **el resto de usuarios**: los permisos que tendrán el resto de usuarios que no son ni el dueño ni pertenecen al grupo

Todo ello se puede visualizar en el sistema de ficheros si listamos los permisos del fichero:

Ver los permisos de un fichero

```
ruben@vega:~$ ls -lh fichero.txt
-rw-r--r-- 1 ruben ruben 0 dic  8 19:17 fichero.txt
```

Los permisos se pueden ver en los primeros 10 caracteres:



Existen los distintos tipos de ficheros:

- **-**: fichero normal
- **d**: directorio
- **b**: dispositivo de bloque (ejemplo: /dev/sda*)
- **c**: dispositivo de carácter (las consolas. ejemplo: /dev/tty*)

- **s**: socket local
- **p**: tubería (pipe)
- **l**: enlace simbólico (link)

4.3.1. Permisos especiales

Existen otros permisos especiales:

- **SUID**: permiso especial que permite que el fichero sea ejecutado con los permisos del dueño del fichero (aunque lo ejecute otro usuario). Se visualiza con una "S" en el permiso de ejecución del dueño `-rwsrw-r--`.
- **SGID**: permiso especial que permite que el fichero sea ejecutado como el grupo. Aparece una "S" en el permiso de ejecución del grupo: `-rwxr-s---`.
- **STICKY**: si el bit sticky está activado en un **directorio** sólo el usuario root, el dueño del directorio o el dueño del fichero puede borrar ficheros dentro de dicho directorio. Aparece una "t" en el permiso de ejecución del resto de usuarios: `drwxrwxrwt`. Muchas distribuciones usan este permiso en el directorio `/tmp`

4.3.2. Cambiando permisos y dueños a los ficheros y a los directorios

Para cambiar los permisos a los ficheros y a los directorios se hace con el comando **chmod**.

Para cambiar permisos de dueño a los ficheros y a los directorios se hace con el comando **chown**.

4.4. La importancia de "sudo"

En muchas distribuciones GNU/Linux el usuario no-privilegiado que se crea tiene permiso de "sudo" para poder ejecutar comandos como si se tratara del **root** (u otro usuario) para poder realizar tareas de administración. Es habitual que en estas distribuciones **el usuario root no suela tener contraseña**.

Cuando un usuario necesite realizar una tarea como administrador, deberá usar "sudo" antes del comando:

Editar un fichero con permisos de root

```
ruben@vega:~$ sudo nano /etc/passwd
```

Tras realizar este comando, el sistema nos pedirá la contraseña del usuario con el que lo estemos ejecutando y comprobará que el usuario tiene permisos de "sudo" para poder ejecutar el comando (en este caso: nano).

El comando "**sudo**" viene de "**su**per user **do**" (que en inglés sería: "super usuario haz"), y aunque su uso habitual es el de permitir realizar cualquier comando de administración, la configuración permite mucho más, pudiendo permitir a ciertos usuarios sólo realizar ciertas tareas. Por ejemplo:

- Usuario **ruben**: tendría permisos para poder realizar cualquier comando del sistema.
- Usuario **dba**: sólo tendría permisos para poder realizar el reinicio del sistema de base de datos.

- Usuario **adminweb**: sólo tendría permisos para poder realizar el reinicio del servidor web.
- …

De esta manera, la gestión de nuestro servidor estaría basada en múltiples usuarios y cada usuario sólo sería capaz de realizar pequeñas tareas, por lo que la seguridad del servidor sería mayor y limitaría lo que los usuarios puedan realizar.

4.4.1. Configurando "sudoers"

Los permisos de sudo se realizan en el fichero `/etc/sudoers`, y para su edición se hace uso del comando **visudo**, el cual abre el fichero y se asegura que a la hora de guardar la sintaxis es correcta.

Si realizamos cualquier modificación sobre el fichero, éste será tenido en cuenta la próxima vez que se realice la ejecución del comando "sudo", por lo tanto, no hay que realizar ningún reinicio de servicio.

El fichero `/etc/sudoers` tiene permisos de sólo lectura para el usuario root y el grupo root:

Permisos del fichero /etc/sudoers

```
root@vega:# ls -lh /etc/sudoers
-r--r----- 1 root root 669 jun  5  2017 /etc/sudoers
```

Un fichero sudoers suele tener el siguiente aspecto:

Contenido del fichero /etc/sudoers

```
Defaults    env_reset
Defaults    mail_badpass
Defaults    secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

La línea que más importa en este fichero es la que indica "**%sudo ALL=(ALL:ALL) ALL**" y es explicada en su comentario anterior. Lo que quiere decir es que cualquier usuario que pertenezca al grupo "sudo" podrá realizar cualquier comando del sistema como superusuario. La sintaxis de la línea es:

- **%sudo**: cualquier usuario que pertenezca al grupo "sudo"
- **ALL**= : desde cualquier host o IP

- **(ALL:ALL)**: el usuario que ejecuta puede ejecutar el comando como cualquier usuario y cualquier grupo
- **ALL**: puede ejecutar cualquier comando

Un ejemplo limitando el uso de sudo a un único comando a un usuario:

Añadimos configuración al fichero /etc/sudoers

```
ruben    ALL=(ALL:ALL) NOPASSWD:/bin/systemctl suspend
```

Con esta línea lo que estamos permitiendo es que el usuario "ruben" puede ejecutar el comando "/bin/systemctl suspend" (suspender el equipo) y sin necesidad de meter contraseña al hacer sudo, gracias a la opción "NOPASSWD").

4.5. Diferencias entre "sudo", "su" y "su -"

Como ya se ha comentado en el apartado anterior, "sudo" permite la ejecución de comandos como cualquier usuario, siendo lo habitual ejecutarlo como root. Ahora bien, en entornos donde el usuario root tiene contraseña, nos puede interesar convertirnos en él para realizar tareas sin tener que estar ejecutando "sudo" a cada comando. Al ser root, tendremos que tener especial cuidado.

4.5.1. Variables de entorno

En cualquier sistema operativo existen las denominadas "variables de entorno". Son variables que cada usuario tiene y sirven para indicar ciertos parámetros que se están utilizando (la SHELL que se está usando), o parámetros que se van a usar a la hora de ejecutar comandos o realizar tareas, ya que se consultan a ellas. En GNU/Linux las variables de entorno se pueden consultar ejecutando:

Vemos las variables de entorno del usuario ruben

```
ruben@vega:~$ printenv
LANG=es_ES.utf8
LOGNAME=ruben
XDG_VTNR=2
COLORTERM=truecolor
PWD=/home/ruben
DESKTOP_SESSION=gnome
USERNAME=ruben
SHELL=/usr/bin/zsh
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
...
```

Una variable de entorno puede consultarse haciendo:

Consultamos el contenido de la variable PATH

```
ruben@vega:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

Como se puede ver, es con un "**$**" y el nombre de la variable en mayúsculas. Existen muchas variables de entorno, y podríamos crear las nuestras propias si así lo necesitáramos.

4.5.2. La importancia de "su -"

Con el comando "**su**" nos podemos convertir en cualquier otro usuario del sistema siempre y cuando **conozcamos su contraseña**. Hay que notar la diferencia respecto a "**sudo**" que cuando lo ejecutamos nos pide **nuestra contraseña**.

Al ejecutar "su" nos convertimos en el usuario root (o ejecutando "su usuario2", nos convertimos en el usuario2), **pero no hacemos uso de sus variables de entorno**, si no que seguimos con las variables de entorno del usuario que éramos previamente. Para convertirnos en el usuario y que obtengamos sus variables de entorno es necesario ejecutar "**su -**", o lo que es lo mismo: "**su -l**", que el manual nos dice: "*Start the shell as a login shell with an environment similar to a real login*". Por ejemplo:

Consultamos el contenido de la variable PATH en distintas situaciones

```
ruben@vega:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games

ruben@vega:~$ su
Contraseña:

root@vega:/home/ruben# echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games

root@vega:/home/ruben# exit

ruben@vega:~$ su -
Contraseña:

root@vega:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

El usuario "ruben" tiene unos valores en la variable de entorno PATH (es la variable que se encarga de tener las rutas de los ejecutables de los programas). Al convertirse en root haciendo uso de "su", y mirar la variable PATH, podemos observar que es igual que el usuario prueba.

Ahora bien, si a la hora de convertirse en root hace uso de “su -”, se puede ver cómo la variable PATH obtiene otros valores, siendo lo más significativo que aparecen las rutas “/usr/local/sbin” y “/usr/sbin” que son las rutas donde se almacenan los ejecutables que (en principio) sólo deberían ejecutarse como administrador del sistema.

5. Comandos de administración básica en GNU/Linux

En este documento vamos a recopilar comandos que nos pueden ser útiles a la hora de usar un sistema GNU/Linux y realizar su administración.

5.1. Comandos sobre el sistema de ficheros

A continuación unos comandos básicos para utilizar sobre el sistema de ficheros.

Listar el contenido del directorio donde nos encontramos

```
ruben@vega:~$ ls
Descargas   Escritorio  Música      Público
Documentos  Imágenes    Plantillas  Vídeos
```

Listar el directorio actual, versión “long”

```
ruben@vega:~$ ls -l
total 36K
drwxr-xr-x 2 ruben ruben 4,0K nov 12  2022 Descargas
drwxr-xr-x 2 ruben ruben 4,0K dic  6 09:50 directorio1
drwxr-xr-x 2 ruben ruben 4,0K nov 12  2022 Documentos
drwxr-xr-x 2 ruben ruben 4,0K nov 12  2022 Escritorio
drwxr-xr-x 2 ruben ruben 4,0K nov 12  2022 Imágenes
drwxr-xr-x 2 ruben ruben 4,0K nov 12  2022 Música
drwxr-xr-x 2 ruben ruben 4,0K nov 12  2022 Plantillas
drwxr-xr-x 2 ruben ruben 4,0K nov 12  2022 Público
drwxr-xr-x 2 ruben ruben 4,0K nov 12  2022 Vídeos
```

Crear un nuevo directorio

```
ruben@vega:~$ mkdir directorio1
ruben@vega:~$ ls
Descargas    Documentos  Imágenes  Plantillas  Vídeos
directorio1  Escritorio  Música    Público
```

Borrar un directorio que está vacío

```
ruben@vega:~$ rmdir directorio1
```

Editar un fichero

```
ruben@vega:~$ nano fichero.txt
```

Obtener el contenido de un fichero de texto

```
ruben@vega:~$ cat fichero.txt
hola, qué tal
```

Paginar el contenido de un fichero de texto

```
ruben@vega:~$ more fichero.txt
hola, qué tal
```

Borrar un fichero

```
ruben@vega:~$ rm fichero.txt
```

Borrar un directorio y todo el contenido que tiene dentro. ¿Diferencias?

```
ruben@vega:~$ rm -r directorio2
ruben@vega:~$ rm -fr directorio2
```

Buscar por un contenido dentro de un fichero

```
ruben@vega:~$ grep hola fichero.txt
```

5.2. Comandos de red

Para ver los interfaces de red y las direcciones IP que tienen

Obtener los interfaces y las IPs

```
ruben@vega:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo

2: enp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
link/ether 1a:8a:1c:ff:25:15 brd ff:ff:ff:ff:ff:ff
inet 192.168.1.99/24 brd 192.168.1.255 scope global enp4s0
```

Para ver la ruta por defecto (el gateway o puerta de enlace).

Obtener la puerta de enlace

```
ruben@vega:~$ ip route show default
default via 192.168.1.1 dev enp4s0 onlink
```

Ver los puertos TCP y servicios que están a la escucha en nuestro servidor

Listar los puertos TCP a la escucha

```
root@vega:~# ss -pntal
```

5.3. Comandos sobre procesos

Listar todos los procesos

Listar todos los procesos

```
root@vega:~# ps aux
```

Listar todos los procesos en forma de árbol (para saber de quién son hijos)

Listar todos los procesos en forma de árbol

```
root@vega:~# pstree -p
```

Matar un proceso (donde PID es el identificador del proceso).

Matar un proceso

```
root@vega:~# kill -9 PID
```

5.4. Estado de la carga y memoria del servidor

Para ver los procesos y su estado por consumo de CPU, RAM…

Ver el estado del servidor

```
root@vega:~# top
```

Para ver los procesos y su estado por consumo de CPU, RAM… es necesario instalar este paquete

Ver el estado del servidor

```
root@vega:~# htop
```

5.5. Comandos sobre servicios (systemd/systemctl)

GNU/Linux cuenta con un sistema unificado (**systemd**) para administrar el sistema y los servicios que tenemos en nuestro servidor. Dado que es una pieza fundamental en el sistema operativo, debemos de

conocer ciertos comandos para poder desempeñar tareas con él.

Listar todos los servicios/unidades

Listar todos los servicios

```
root@vega:~# systemctl
```

Comprobar si algún servicio ha fallado

Comprobar servicios que han fallado

```
root@vega:~# systemctl --failed
```

Comprobar el estado de un servicio concreto (en este caso, ssh)

Comprobar servicios que han fallado

```
root@vega:~# systemctl status ssh
```

Parar un servicio concreto

Parar un servicio concreto

```
root@vega:~# systemctl stop ssh
```

Arrancar un servicio concreto

Arrancar un servicio concreto

```
root@vega:~# systemctl start ssh
```

Ver los logs de todo el sistema

Ver los logs del sistema

```
root@vega:~# journalctl
```

Ver los logs de un servicio concreto (en este caso, ssh)

Ver los logs del sistema

```
root@vega:~# journalctl -u ssh
```

Ver los logs del kernel

Ver los logs del kernel

```
root@vega:~# journalctl -k
```

5.6. Comandos para instalar/desinstalar paquetes

Los comandos que se van a exponer aquí sirven para las distribuciones que utilizan el sistema de paquetes APT (Advanced Packaging Tool) que está presente en las distribuciones derivadas de Debian, como por ejemplo Ubuntu.

Sincroniza el índice de paquetes local con lo que está en los repositorios remotos configurados en /etc/apt/sources.list

```
root@vega:~# apt update
```

Instala las versiones más nuevas de los paquetes que ya tenemos instalados

```
root@vega:~# apt upgrade
```

Instala las versiones más nuevas de los paquetes teniendo en cuenta las dependencias de los mismos y solucionando posibles conflictos.

```
root@vega:~# apt full-upgrade
```

5.7. Comandos para apagar/reiniciar

Para realizar el apagado o reinicio de un equipo se pueden utilizar varios comandos distintos.

Reiniciar el equipo

```
root@vega:~# reboot
```

Apagar el equipo en un minuto

```
root@vega:~# shutdown
```

Apagar el equipo ahora

```
root@vega:~# shutdown now
```

Suspender el equipo

```
root@vega:~# systemctl suspend
```

IV

Sistemas de comunicación y redes

1. Introducción a los sistemas de comunicación

Desde el principio de los tiempos, el ser humano se ha comunicado con sus congéneres de distintas maneras: comenzó a través de la voz (se cree que hace unos 100.000 años), con algún tipo de protolenguaje, para posteriormente comenzar a utilizar sistemas de comunicaciones permanentes (la escritura).

Por todos es conocido la evolución histórica de distintos sistemas de comunicación, entre los que podemos destacar (referencia):

- **Pinturas rupestres**: Realizadas en cuevas o rocas en las que se pueden observar escenas de caza, distintos animales, grabado de manos, figuras humanas... Algunas de las pinturas encontradas cuentan con más de 50.000 años. Tenemos un ejemplo cercano en las cuevas de Santimamiñe en donde tenemos pinturas datadas entre 14.000 y 9.000 años a. C.

- **Escritura cuneiforme**: Es uno de los primeros sistemas de escritura realizados, y se utilizaban tablillas de arcilla húmeda en las que se grababa mediante un tallo vegetal. Con este sistema se han datado tablas anteriores al 3.200 a.C. y en distintos idiomas.

- **Escritura jeroglífica y el papiro**: En el antiguo Egipto se crea la escritura mediante signos que comienza por escribirse en paredes para posteriormente inventar el papiro (cuya datación más antigua es del 2.500 a.C.) y de esta manera se comienza a tener un sistema de comunicación fácilmente manejable e intercambiable.

- **Uso de palomas mensajeras**: El uso de palomas mensajeras para el envío de comunicaciones data de la época anterior a 1.500 a.C. y se ha estado utilizando hasta este siglo en algunos países durante desastres naturales.

- **Imprenta**: El primer documento impreso data de china del año 868, pero la imprenta moderna la creó en 1440, más o menos, Johannes Gutemberg. Gracias a la imprenta la creación de documentos escritos se realizaba de manera más rápida y esto permitió que la expansión del conocimiento escrito se acelerase.

- **Telégrafo**: A partir de mediados del siglo XVIII y durante el inicio del siglo XIX hubo bastantes avances en las investigaciones del electromagnetismo y de esta manera se comenzó a investigar cómo usarlo para el envío de señales. En 1837 Samuel Morse patenta el telégrafo. En **1858** se une Irlanda y Terranova mediante el **primer cable trasatlántico**.

- **Teléfono**: Como evolución al telégrafo, que sólo permitía el envío de señales, nace el teléfono de la mano de Antonio Meucci (aunque normalmente se le atribuye el invento a Alexander Graham Bell). En 1860 realizó una demostración pública transmitiendo voz a una considerable distancia.

Tal como podemos ver, ha habido distintos sistemas de comunicación utilizados durante siglos para el envío y recepción de información.

1.1. Comunicación de la información

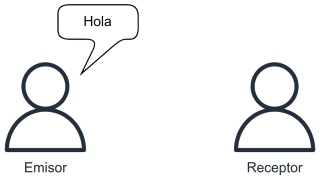
Tal como hemos visto, los sistemas de comunicación de la información no es algo nuevo, ¿pero qué necesidades tiene un sistema de comunicación?

- **Emisor**: Es el origen y la fuente de la información que se pretende comunicar.
- **Receptor**: Es el destinatario, el que va a recibir la información.
- **Mensaje**: Es la información que queremos transmitir entre el emisor y el recepetor.
- **Código**: Es el conjunto de reglas utilizadas a la hora de representar el mensaje. El emisor y receptor deben utilizar el mismo código para que la comnunicación sea correcta.
- **Canal**: Es el medio físico por el que se va a enviar el mensaje.
- **Señal**: Es el componente físico por el que se envía la información.

Para entender de mejor manera un sistema de comunicación y los componentes que lo forman, vamos a poner dos ejemplos:

Ejemplo 1: Comunicación oral

En este ejemplo vemos que hay dos personas, las cuales se han identificado cada una de ellas como “Emisor” y “Receptor”, y así de esta manera conocemos quién es el origen y quién el destino de la comunicación.

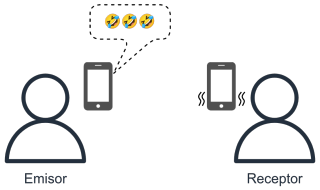


En este caso, el **mensaje** es “Hola”, haciendo uso del **código** conocido como “castellano”. La **señal** que se va a utilizar es la voz, ya que están hablando y el **canal** por el que se envía el mensaje es el aire.

Es un ejemplo sencillo que utilizamos cada día.

Ejemplo 2: Comunicación escrita por mensajería

AL igual que en el ejemplo anterior, vemos que hay dos personas, las cuales se han identificado cada una de ellas como “Emisor” y “Receptor” pero que en este caso se van a comunicar haciendo uso de un teléfono móvil, tal como hacemos en nuestro día a día a través de una aplicación de mensajería o red social.



Teniendo en cuenta esto, en este ejemplo realmente existen dos sistemas de comunicación que están mezclados y uno está por encima del otro:

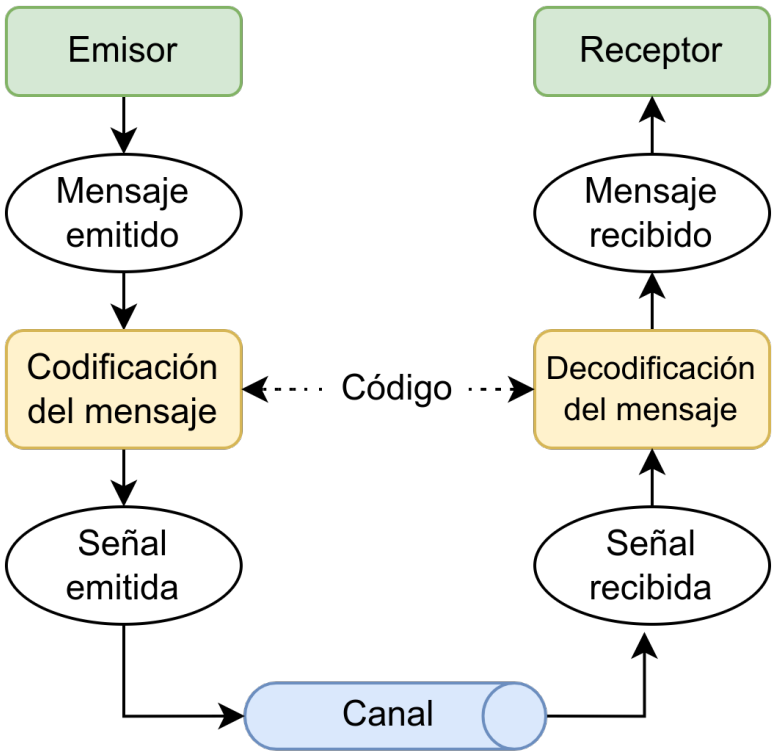
- **Entre personas**: Similar al ejemplo anterior, el emisor y el receptor se están comunicando, con el mensaje compuesto por tres emojis que representan estar riendo. El **código** es el idioma que estén utilizando, el **canal** sería el programa utilizado y la **señal** podríamos decir que es el móvil.
- **Entre dispositivos**: En este caso, el emisor y receptor es el móvil de cada usuario. El mensaje es el mismo, pero convertido a un sistema digital (como el binario). El **canal** en este caso sería el aire y la

señal es la utilizada por el móvil, por ejemplo el 5G.

Tal como se puede ver en este caso, una comunicación puede depender a su vez de otro sistema de comunicación.

1.1.1. Esquema de la comunicación

Para simplificar cómo se realiza la comunicación, podemos utilizar el siguiente esquema:



2. Redes de comunicación

En el ámbito informático una red de comunicaciones es representada como una red de ordenadores para poder compartir información, recursos u ofrecer servicio (ya sea al usuario o a otros ordenadores).

Las redes de ordenadores son un conjunto de equipos hardware que están conectados entre sí, mediante cables o de manera inalámbrica, y que a través de un software especializado envían y reciben impulsos eléctricos (u ondas electromagnética) para el transporte de datos.

2.1. Breve historia de las redes

A continuación una breve cronología mostrando los hitos más importantes dentro de las redes de comunicaciones, en lo que a ordenadores se refiere (Referencia).

~1950 En la década de los 50 se desarrollan los circuitos integrados. Esto hará que en el futuro los ordenadores cada vez se vayan haciendo más pequeños.

Las redes de ordenadores comienzan a aparecer en las bases militares americanas, en principio para sistemas de radares.

~1960 Se realiza una conexión entre dos mainframes en EEUU para el sistema de reservas aéreas comerciales.

El MIT utiliza un ordenador para enrutar y mantener conexiones telefónicas.

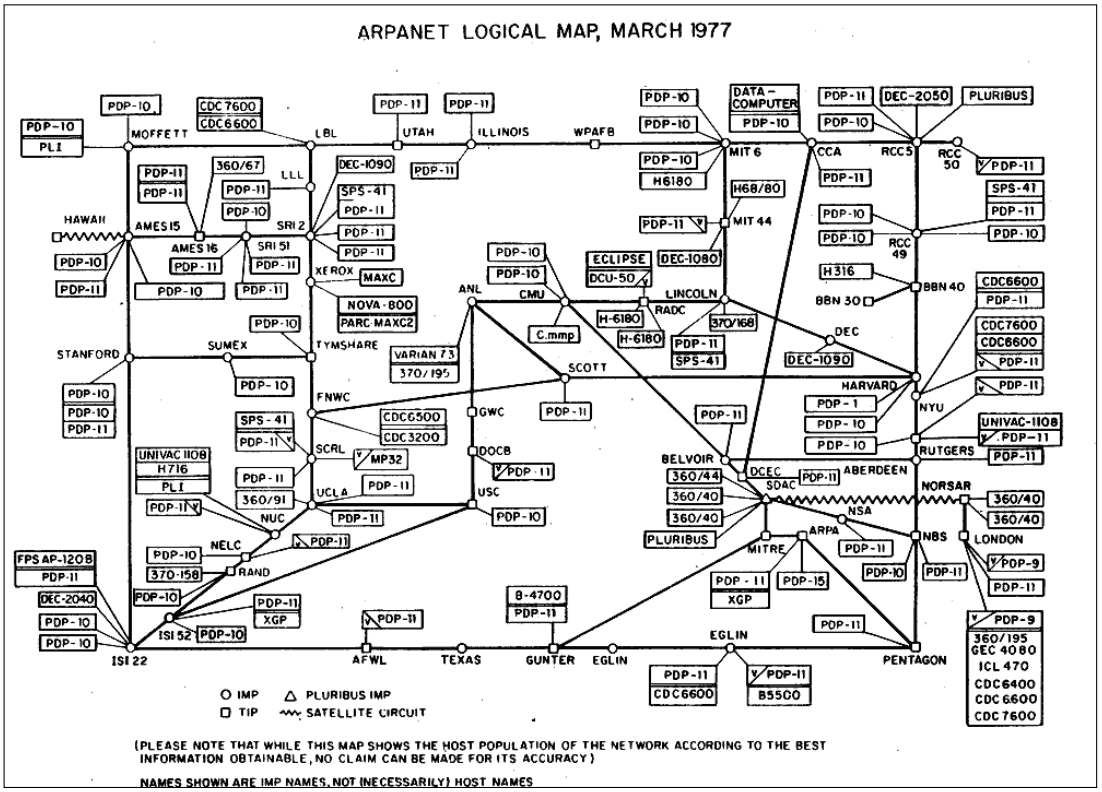
En 1966, aparece un paper (artículo científico) describiendo las WAN.

En 1969 ARPANET (red de ordenadores creadas por el Departamento de Defensa de Estados Unidos) cuenta con 4 nodos (a 50kbit/s de velocidad).

~1970 En 1972 se hace la primera demostración pública de ARPANET.

A comienzos de la década (1973) se crea Ethernet en la compañía Xerox Parc.

A finales de la década Xerox intenta hacer que Ethernet se convierta en un estándar de conexión para terminar con las competencias (token ring, …).



Mapa lógico de ARPANET, marzo de 1977. Origen: Wikipedia

~1980 Los ordenadores personales empiezan a generalizarse.

Aparece el protocolo para enviar y recibir e-mails (SMTP).

El protocolo TCP/IP se convierte en el utilizado por ARPANET (1983) y es declarado como su estándar para las comunicaciones.

Aparece el servicio DNS.

Se crea el modelo de referencia OSI.

Aparece el primer gusano por la red (Morris worm, 1988). Se estima que infectó al 10 % de los

ordenadores conectados a la red.

Se crea el protocolo BGP.

El protocolo Ethernet evoluciona y permite conexiones a 10Mbit/s.

~1990 Tim Berners-Lee desarrolla el código para WWW y crea el primer servidor web (1991).

Se puede decir que aquí es cuando nace la Internet que conocemos actualmente.

En 1995 Ethernet permite conexiones a 100Mbit/s

Se establece un control para los nombres de dominio (posteriormente lo asumirá ICANN).

Aparece Amazon, ebay, Craiglist, IMDB, hotmail, google, yahoo, ...

Aparece el protocolo IPv6 (1998).

Aparece el protocolo wifi 802.11b.

~2000 Crisis de las ".com".

Internet se generaliza.

Empiezan a permitirse más TLDs, que no corresponden sólo a países.

Ethernet permite conexiones a 1Gbit/s

~2010 Ethernet permite conexiones a 400Gbit/s (2018).

Starlink comienza a desplegar su constelación de satélites para dar cobertura en todo el planeta.

2.2. Tipos de redes

A la hora de diferenciar las redes de ordenadores podemos diferenciarlas por distintos conceptos:

- Por el medio de transmisión utilizado.
- Por la dirección de los datos.
- Por el alcance.
- Por el grado de acceso.
- Por la topología.
- ...

2.2.1. Por el medio de transmisión utilizado

Más adelante veremos distintos sistemas de transmisión, pero para ir diferenciando podemos crear dos grandes grupos teniendo en cuenta el medio utilizado:

- **Guiados**: Es decir, a través de cables que se encargarn de realizar la transmisión de la señal desde un punto de origen al punto de destino.

- **No guiados**: Se hace uso de algún sistema inalámbrico (mediante antenas) para realizar la transmisión de los datos.

2.2.2. Por la dirección de los datos

Si tenemos en cuenta la dirección de los datos en la transmisión, podemos diferenciarlos como:

- **Simplex**: la comunicación sólo se realiza en un único sentido, por lo que sólo es necesario un único canal de transmisión.
- **Half-duplex**: se permite la comunicación en ambos sentidos, pero no de manera simultánea, por lo que emisor y receptor se reparten el tiempo de emisión. Por ejemplo, el ***walkie-talkie***.
- **Duplex**: O también conocido como *full-duplex*, permite la comunicación en ambas direcciones y de manera simultánea. Por ejemplo, el **teléfono**. Para ello es necesario tener una de estas dos opciones:
  - Dos canales Half-duplex: uno para cada dirección de la comunicación.
  - Un único canal por el que se envía la comunicación, pero para ello es necesario algún sistema de multiplexación (como puede ser usar frecuencias separadas).

2.2.3. Por alcance

Teniendo en cuenta el alcance al que llegan las redes, podríamos realizar la siguiente distinción:

2.2.3.1. Red de área personal (PAN)

Del inglés *Personal Area Network*, es aquella en la que interactúan distintos dispositivos de muy corto alcance, limitado al área de una persona.

El ejemplo más habitual hoy día sería la comunicación mediante tecnología inalámbrica por Bluetooth en la comunicación entre ordenador, móvil y dispositivos como un *smartwatch*.

2.2.3.2. Red de área local (LAN)

Del inglés *Local Area Network*, es una red que puede abarcar un cierto área de tamaño como una casa, una oficina, un colegio, una universidad...

El ejemplo de una oficina sería una red en la que existen distintos ordenadores, que pueden comunicarse entre sí o compartir información con un servidor ya sea a través de una red cableada o también inalámbrica.

2.2.3.3. Red de área metropolitana (MAN)

Del inglés *Metropolitan Area Network*, y como su nombre indica, el área es mayor y suele abarcar una ciudad para ofrecer los servicios necesarios en la misma.

En este caso también puede ser de manera cableada (normalmente haciendo uso de tecnología más rápida como es la fibra óptica) y también de manera inalámbrica.

Dentro de los servicios que pertenecerían a una MAN podemos poner como ejemplos:

- Despliegue de zonas WIFI gratuito en la ciudad.

- Comunicación entre sistemas de información (paradas de autobuses, marquesinas, ...).
- Sistemas de video-vigilancia municipal.

Algunos de estos servicios que están en una MAN pueden ser públicos (como el WIFI) o de acceso restringido (sistemas de seguridad).

2.2.3.4. Red de área amplia (WAN)

Del inglés *Wide Area Network*, es una red que abarca grandes extensiones geográficas y normalmente construidas por grandes empresas o proveedores de internet (ISP, *Internet Service Provider*).

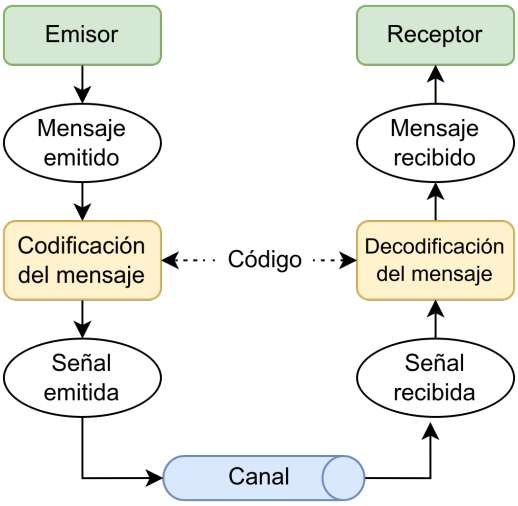
2.2.4. Por el grado de acceso

Teniendo en cuenta quién puede acceder a la red, podríamos definir dos tipos de redes:

- **Red privada**: es una red que sólo ciertas personas pueden acceder y que no normalmente no es accesible desde otras redes. El ejemplo más sencillo es la red que tenemos en casa.
- **Red pública**: es una red a la que puede acceder cualquier persona y que interconecta otras redes sin importar su situación geográfica. Internet es una red pública.

3. Arquitectura en capas

Un sistema de comunicación se pueden diferenciar en distintos niveles en los que cada uno realiza una función independiente, pero que a su vez interactúan con los niveles limítrofes.



Este ejemplo es un modelo simplificado de comunicación, y dentro de una arquitectura de red de ordenadores pueden existir más capas en las que pueden existir distintas funciones extra que no aparecen en este esquema.

3.1. Origen

Al comienzo de las redes de ordenadores cada empresa creaba su propio sistema de comunicación creando su propio hardware y software, lo que hacía imposible la interconexión entre equipamiento de distintas empresas.

Estos sistemas de comunicación constan de unas reglas que los nodos deben conocer para poder comunicarse entre sí, y a ese conjunto de reglas se les denomina **protocolo de comunicación**.

Para que eso hoy en día no suceda ya que las redes están definidas en varios estándares, como veremos más adelante.

Información

Un **estándar** es un conjunto de normas que pueden abarcar distintos niveles (tanto software como hardware) que ha sido aceptado, o creado, por la gran mayoría de las empresas del sector para poder realizar la interconexión e intercomunicación entre sí.

3.2. Ventajas de la división en capas

La división en capas nos permite:

- Dividir el proceso de comunicación en procesos más pequeños.
- Aislar las funciones de cada capa. De esta manera, en caso de realizar modificaciones en la misma, no afecta al resto de capas.
- Ocultar la implementación al resto de capas. Siguiendo con el punto anterior, una capa utilizará los servicios de su capa inferior sin saber cómo realiza sus funciones.
- Cada capa puede constar de distintos estándares, facilitando la interconexión de distintas tecnologías

Una arquitectura de red en capas se implementa por medio de distintos protocolos, formando una familia de protocolos para facilitar la comunicación de distintos sistemas y equipos en la red.

Información

Una arquitectura en capas nos permite que cada capa actúe de manera independiente y que incluya sus propios protocolos. Cada capa dispone de una serie de servicios que ofrece a su capa limítrofe superior.

Desde el comienzo de las redes de ordenadores han existido distintas familias de protocolos, y se puede considerar que hubo una guerra de protocolos durante las décadas de 1970 a 1990. Empresas, organizaciones y países se posicionaban sobre cuál sería el mejor protocolo de comunicaciones y el que saldría ganador para el uso a nivel internacional.

Por destacar algunos protocolos que ya no se usan:

- **SNA** creado en 1974 por IBM.
- **NetBEUI** de Microsoft. Que evolucionó a NetBIOS sobre TCP/IP que hoy día se usa en Windows Server.
- **IPX/SPX** de Novell.

3.3. Modelo de referencia OSI

El modelo de interconexión de sistemas abiertos, conocido como "modelo **OSI**" (***O**pen **S**ystems **I**nterconnection* en inglés) es un **modelo de referencia (teórico)** que busca estandarizar las funciones de comunicación para un sistema informático siendo agnóstico a la tecnología utilizada para realizar la implementación y a los protocolos utilizados en cada capa.

El diseño comenzó en 1977 tratando de terminar con la guerra de protocolos comentada previamente, y la Organización Internacional de Estandarización (*International Organization for Standardization*, o **ISO** en inglés) terminó por definir el estándar ISO-7498 en 1984.

3.3.1. Capas en el modelo OSI

El modelo OSI está compuesto por siete capas numeradas del 1 al 7 siendo la 1 la más baja y haciendo referencia a la parte física de la red.

Dentro de cada una de las capas a las unidades de datos se les llama de manera distinta, por lo que es conveniente referirse de manera correcta a ellas. Más adelante también aprenderemos que existen distintos componentes *hardware* que actúan en algunas de las capas.

Capa	Nombre de la unidad de datos	Función
7ª - Aplicación	Datos	APIs de alto nivel, como compartir recursos y acceso remoto a archivos.
6ª - Presentación	Datos	Traducción de datos entre un servicio de red y una aplicación, que incluye la codificación de caracteres, la compresión de datos y el cifrado y descifrado de datos.
5ª - Sesión	Datos	Manejo de sesiones de comunicación, por ejemplo el continuo intercambio de información en forma de múltiples transmisiones hacia ambos lados entre dos nodos.
4ª - Transporte	Segmento, Datagrama	Transmisión de segmentos de datos confiable entre puntos de red, incluyendo la segmentación, el acknowledgement y la multiplexación. Aquí actúan los protocolos **TCP** y **UDP**, junto con los **puertos**.

Continúa en la siguiente página

3ª - Red	Paquete	Estructura y manejo de una red multinodo. Incluye el direccionamiento, el ruteo y el control de tráfico. Aquí actúan los **routers**.
2ª - Enlace de datos	Trama	Transmisión de datos confiable entre dos nodos conectados mediante una capa física. Aquí actúan los **switches**.
1ª - Física	Bit, Baudios	Transmisión y recepción de flujos de bits sin procesar por un medio físico.

3.4. Pila de protocolos TCP/IP

Durante el surgimiento de las redes de comunicaciones entre ordenadores cada fabricante creaba su propio estándar, lo que hacía que la comunicación entre ellos no fuese posible. En 1974 Vint Cerf and Bob Kahn describen un protocolo para compartir recursos usando envío de paquetes a través de una red de comunicación. Es el comienzo del protocolo **TCP**.

En 1984 fue el protocolo elegido por parte del Departamento de Defensa de Estados Unidos, y poco tiempo después se convirtió en el estándar de facto de la comunicación de red que posteriormente dio lugar a Internet.

Tal como se puede ver, la pila de protocolos TCP/IP es funcional, al contrario que ocurre con el modelo OSI que sólo es teórico.

La pila TCP/IP cuenta con cuatro capas:

Capa	Protocolos conocidos	Función
4ª - Aplicación	HTTP, FTP, POP, SMTP, …	Es la capa más cercana al usuario, utilizada por las aplicaciones a la hora de enviar datos.
3ª - Transporte	TCP, UDP	Estructura y manejo de una red multinodo. Incluye el direccionamiento, el ruteo y el control de tráfico.
2ª - Internet	IPv4, IPv6	Transmisión de datos confiable entre dos nodos conectados mediante una capa física.

Continúa en la siguiente página

1ª - Acceso al medio	Ethernet, IEEE 802, 802.11 (wifi)	Es una mezcla de las capas 1 y 2 del modelo OSI.

3.4.1. Protocolo TCP

El objetivo del protocolo TCP es crear conexiones dentro de una red de datos compuesta por redes de ordenadores para intercambiar datos. La ventaja es que el protocolo busca garantizar que los datos son entregados a su destino sin errores y en el mismo orden en el que se transmitieron.

Por otro lado, también proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de **puerto**.

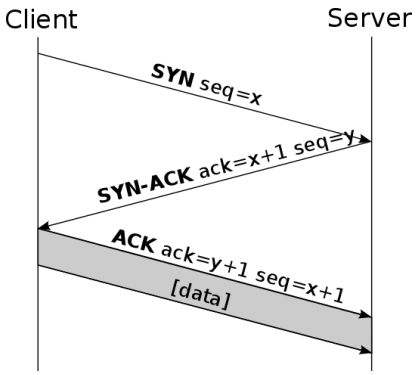
Hoy día TCP es el protocolo más utilizado dentro de las aplicaciones que hacen uso de comunicación en red. Por ejemplo, los protocolos HTTP, SMTP, SSH, FTP... hacen uso de TCP como protocolo de transporte (**capa 4 del modelo OSI**).

3.4.1.1. Establecer comunicación

A la hora de establecer una comunicación usando el protocolo TCP lo habitual es que en el lado del **servidor (el receptor de la comunicación)** exista un servicio que esté "*escuchando*" en un puerto previamente levantado (por ejemplo, un servidor web que escucha en el **puerto 80 y 443**).

El cliente comenzará la comunicación enviando un paquete **SYN** a la IP del servidor y al puerto con el que se quiere comunicar, a lo que si todo ha ido bien, el servidor responderá con la respuesta **SYN/ACK**.

Finalmente, el cliente debería responderle al servidor con un ACK, completando así la **negociación en tres pasos** (SYN, SYN/ACK y ACK) y la fase de establecimiento de conexión.

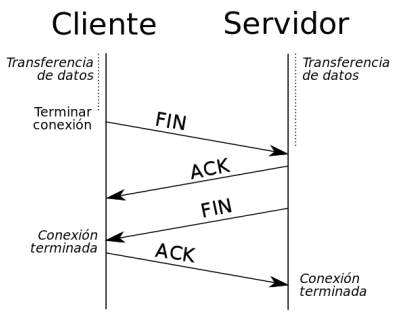


De esta manera, comenzará la transferencia de datos, a lo que se le añade una serie de mecanismos que determinan la fiabilidad y robustez del protocolo. Entre ellos están incluidos:

- Uso del número de secuencia para ordenar los segmentos TCP recibidos y detectar paquetes duplicados.
- *Checksums* para detectar errores.
- Indicación por parte del receptor que ha recibido los paquetes para detectar pérdidas.
- Temporizadores para detectar retrasos o necesidad de reenvío de información.

3.4.1.2. Cierre de la comunicación

La fase de finalización de la conexión utiliza una negociación en cuatro pasos (four-way handshake), terminando la conexión desde cada lado independientemente.



Cuando uno de los dos extremos de la conexión desea parar su "mitad" de conexión transmite un segmento con el flag FIN en 1, que el otro interlocutor asentirá con un ACK. Por tanto, una desconexión típica requiere un par de segmentos FIN y ACK desde cada lado de la conexión.

Una conexión puede estar "medio abierta" en el caso de que uno de los lados la finalice pero el otro no. El lado que ha dado por finalizada la conexión no puede enviar más datos pero la otra parte si podrá.

3.4.1.3. Puertos TCP

TCP usa el concepto de número de puerto para identificar a las aplicaciones emisoras y receptoras. Cada lado de la conexión TCP tiene asociado un número de puerto (de 16 bits sin signo, con lo que existen 65536 puertos posibles) asignado por la aplicación emisora o receptora.

Los **puertos bien conocidos** son asignados por la Internet Assigned Numbers Authority (IANA), van del 0 al 1023 y son usados normalmente por el sistema o por procesos con privilegios. Algunos ejemplos son: FTP (21), SSH (22), Telnet (23), SMTP (25) y HTTP (80).

Los **puertos registrados** son normalmente empleados por las aplicaciones de usuario de forma temporal cuando conectan con los servidores, pero también pueden representar servicios que hayan sido registrados por un tercero (rango de puertos registrados: 1024 al 49151).

Los **puertos dinámicos/privados** también pueden ser usados por las aplicaciones de usuario, pero este caso es menos común. Los puertos dinámicos/privados no tienen significado fuera de la conexión TCP en la que fueron usados (rango de puertos dinámicos/privados: 49152 al 65535).

4. Direccionamiento IPv4

La **dirección IP** es el identificador (que debe ser único) de un dispositivo dentro de una red. Los dispositivos pertenecientes a la misma red podrán comunicarse entre sí mediante dicha dirección IP. Todas las características que conforman esta IP están explicados en el **protocolo IP** (Internet Protocol).

Debido a que en el origen de la creación del protocolo IPv4 no se pensaba que llegase a haber tantos dispositivos conectados a internet, **el límite que se puso para el número de posibles direcciones está llegando a su fin**. Es por ello que ya existe un nuevo protocolo, **IPv6** desde hace tiempo, que sustituirá a IPv4 en un futuro, pero para ello debe de realizarse una transición que no termina de finalizar, aunque ya es posible hacer uso de ello.

El direccionamiento IP proporciona un mecanismo para la asignación de identificadores a cada dispositivo conectado a una red. Antes de dar información más técnica, exponemos los principales conceptos:

- Una dirección IP es un conjunto de bits (que formarán 4 números decimales), que sirve para identificar de forma única a un dispositivo dentro de la red.
- La asignación de dicha IP se puede realizar de dos maneras:
  - **Estática**: El administrador del dispositivo deberá configurar la dirección de manera manual teniendo en cuenta los parámetros necesarios que se ajusten a la red a la que se quiere conectar.
  - **Dinámica**: Una vez el dispositivo se conecte de manera física a la red (por cable, o en caso de ser de manera inalámbrica, realizando la conexión al SSID e introduciendo la contraseña) habrá un servicio (**DHCP**) que le otorgará una IP.

El protocolo IPv4 permite la existencia de dos tipos de direcciones:

- **Direcciones públicas**: Son las utilizadas en Internet. Cualquier dispositivo que se conecte de manera directa a Internet debe tener un direccionamiento público. Existe la organización ICANN que se encarga de repartir estos direccionamientos entre los proveedores de internet.
- **Direcciones privadas**: Son direcciones asignadas a dispositivos que se encuentran dentro de una red que no tiene visibilidad desde Internet. Los dispositivos que tienen este tipo de direccionamiento privado no pueden acceder a internet a través de su IP, por lo que debe de haber un dispositivo que le oculte su IP privada y se la "cambie" por una IP del rango público (el **router** realizando **NAT**, lo veremos más adelante).

4.1. Formato de una dirección IPv4

Como ya se ha comentado, una IP es un conjunto de bits, concretamente 32, que normalmente son representados en 4 grupos de 8 bits pasados a decimal, que es lo que normalmente estamos acostumbrados a ver:

- Dirección IP en formato de 32 bits:

```
11000000101010000000000101100100
```

- IP en formato de bits, separados en grupos de 8 bits:

```
11000000 10101000 00000001 01100100
```

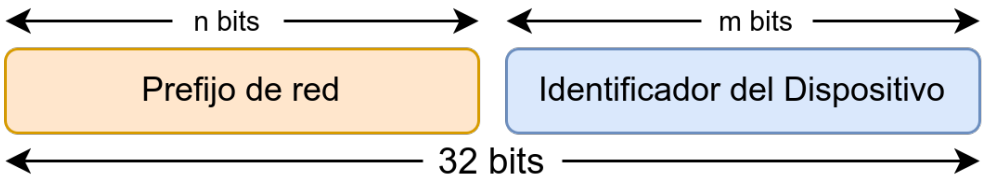
- Dirección IP en formato decimal:

```
192 . 168 . 1 . 100
```

Al tener 32 bits, se realizan 4 grupos de 8 bits, por lo que obtendremos 2^8 posibles combinaciones en cada grupo. Por lo que nos lleva a poder representar cualquier número desde 0 (`00000000`) hasta 255 (`11111111`) en cada bloque.

4.2. Máscara de red

Una dirección IP no sólo contiene la dirección única de un dispositivo, sino que también contiene la red a la que pertenece dicho dispositivo. Esto nos lleva a ver que una IP pertenece a una jerarquía, ya que pertenece a una red "superior". A simple vista no podemos saber a qué red pertenece una IP, pero el formato es el siguiente:



Dependiendo del número de bits que sea "n", el número "m" variará a la par hasta que ambos sumen los 32 bits que debe de tener una dirección IP. Aquí es donde entra en juego la **máscara de red**.

La máscara de red es un número binario de 32 bits que nos permitirá conocer qué número es "n" y "m" en una dirección IP, para así obtener el prefijo de la red y conocer cuántos dispositivos puede llegar a existir en la red.

Al igual que la IP, **una máscara de red son 32 bits** que contendrán a la izquierda "n" unos y a la derecha "m" ceros.

¡Cuidado!

En la máscara de red a la izquierda irán los "1" y a la derecha los "0". ¡NUNCA SE MEZCLAN!

Por ejemplo:

- **IP decimal**: `192.168.1.100`
- **IP binario**: `11000000101010000000000101100100`
- **Máscara de red**: `11111111111111110000000000000000`

Si contamos el número de unos que tenemos en el lado izquierdo de la máscara de red veremos que tenemos 16, por lo que los primeros 16 bits de la IP serán los que nos digan a qué red pertenece esa IP. En este caso, y para favorecer el visionado, realizamos una separación de los bits:

- **IP binario**: `11000000 10101000 00000001 01100100`
- **Máscara de red**: `11111111 11111111 00000000 00000000`

Con esto obtenemos:

- **Prefijo de la red**: `192.168`
- **Identificador del dispositivo:** `1.100`

La máscara de red, al ser 32 bits, también se puede escribir en formato decimal, aplicando al igual que antes, la creación de 4 grupos de 8 bits. En el ejemplo anterior **la máscara sería:** `255.255.0.0` .

Existe un tercer método para escribir la máscara de red: "**/n**", donde "**n**" es un número indicando el número de **unos** (y por tanto, los bits que identifican el prefijo de la red) que tiene la máscara de red. En nuestro caso, la máscara también se puede escribir como **/16**, ya que es el número de unos que tiene nuestra máscara.

4.3. Nombre de la red, broadcast y dispositivos

De una máscara y una IP podemos obtener más información. Para conocer el **número de posibles IPs que puede llegar a haber en esa red** tendremos que realizar el cálculo de 2^m, donde "m" es el número de ceros que tiene la máscara.

En nuestro ejemplo es $2^{16} = 65536$ posibles IPs. **De estas IPs existen dos IPs especiales**:

- **Nombre de la red**: El nombre de la red sirve para identificar y diferenciar las distintas redes que pueden llegar a existir.

  Es muy importante nombrarlas de manera correcta ya que nos dará la información necesaria para conocer el prefijo de la red y el número posible de dispositivos que puede haber.

  Para crear el nombre de la red lo haremos usando el prefijo de la red y el resto de bits ponerlos a 0. En nuestro ejemplo:

  `11000000 10101000 00000000 00000000 = 192.168.0.0`

  Al nombre de la red siempre se le debe añadir la máscara, por lo que quedaría: `192.168.0.0 /16`

  ¡Cuidado!

  Si no se pone la máscara de red, podría ser una IP suelta de cualquier red.

- **Broadcast**: Sirve para poder mandar un mensaje a todos los dispositivos de la red. Para formar esta dirección usamos el prefijo de la red y el resto de bits ponerlos a 1, por lo que en nuestro ejemplo quedaría:

  `11000000 10101000 11111111 11111111 = 192.168.255.255`

Teniendo esto en cuenta, al total de posibles IPs de una red tendremos que restarle 2 para darnos el número total de dispositivos que podremos tener en una red:

Información

Para saber el número de posibles IPs de una red:

$2^m - 2$

siendo "m" el número de `0` de la máscara

En nuestro ejemplo vemos que “m” es 16 (porque en la máscara tenemos 16 ceros), y sabemos que tenemos 2 IPs especiales (**el nombre de la red** y **el broadcast**) que tendremos que restar para tener el número total de dispositivos (tablets, ordenadores, móviles conectados) que podrá llegar a tener esa red: $2^{16} - 2 = 65534$ posibles dispositivos. El rango de estos dispositivos será desde la IP **`192.168.1.1`** hasta **`192.168.255.254`**.

4.4. Bloques de IPs reservadas

Dentro de todas las posibles IPs y redes que podemos llegar a crear, existen unos bloque que están reservados por unas razones u otras:

- **Redes públicas**: son todas las IPs que no entran en los siguientes bloques, y por tanto, son las utilizadas públicamente en Internet.
- **Redes privadas**: Son redes que sólo pueden existir en el ámbito privado y no se podrá configurar de cara internet.
- **Reservado**: Existen varios bloques que están reservados por diversas razones, de los cuales veremos los ejemplos más característicos.

Es importante conocer los bloques reservados para no cometer errores a la hora de crear redes. El listado completo se puede encontrar en la Wikipedia.

4.4.1. Bloque reservado: 127.0.0.0 /8

La primera dirección de este bloque, 127.0.0.1, es el utilizado como bucle local (o **loobpack**). Este conjunto de IPs hacen referencia al propio equipo en el que nos encontramos y se suelen utilizar para hacer pruebas locales.

4.4.2. Redes privadas

Una red privada es una red de computadoras que usa el espacio de direcciones IP especificadas más adelante. A los equipos o terminales se les puede asignar direcciones de este espacio cuando deban comunicarse con otros terminales dentro de la red interna/privada.

Los nombre de las redes privadas son:

- **`10.0.0.0 /8`**
- **`172.16.0.0 /12`**
- **`192.168.0.0 /16`**

Estas redes privadas no son siempre utilizadas con su máxima máscara posible, ya que en la mayoría de los casos se estarían desperdiciando IPs.

El utilizar en nuestra red privada un rango que no sea uno de estos tres signfica que estamos yendo en contra de las reglas establecidas por la ICANN.

4.4.3. Reservado para despliegues Carrier Grade NAT

En abril de 2012, el IANA asignó el rango **100.64.0.0/10** para uso en escenarios de Carrier Grade NAT en el RFC 6598.

Este bloque de direcciones no debe ser usado en redes privadas o en la Internet pública: ha sido pensado para operaciones de uso interno en redes de teleoperadores. El tamaño del bloque de direcciones (aproximadamente 4 millones de direcciones, 2^{22}) ha sido seleccionado para ser suficientemente grande para acomodar todos los dispositivos de acceso de un solo operador en un punto de presencia en una red de área metropolitana como la de Tokio (Fuente: Wikipedia).

4.4.4. Bloque reservado para el futuro

Existe el bloque completo **`240.0.0.0 /8`** cuyas IPs están reservadas para pruebas y para el futuro. Es conocido como la "clase E".

4.4.5. Otros bloques

Como ya se ha comentado, en la Wikipedia se pueden encontrar todos los bloques reservados que no son públicos y la razón por las que han sido reservados.

4.5. Clases de IP

Durante el inicio de la expansión de internet y la creación de redes se crearon clases que nos indican el número de subredes que deberían existir, la máscara y más información que se puede encontrar en la Wikipedia. **Hoy en día se considera obsoleta**.

¡Cuidado!

Este sistema fue el utilizado desde 1981 hasta 1993. ¡Hoy en día está obsoleto!.

- **Clase A**: se asigna el primer octeto para identificar la red, reservando los tres últimos octetos (24 bits) para que sean asignados a los hosts, de modo que la cantidad máxima de hosts es $2^{24} - 2$. El bit más significativo (el de más a la izquierda) empieza por `0`. Sería desde `0.0.0.0` hasta `127.255.255.255`.
- **Clase B**: se asignan los dos primeros octetos para identificar la red, reservando los dos octetos finales (16 bits) para que sean asignados a los hosts. Los primeros bits más significativos son 10. Sería desde `128.0.0.0` hasta `191.255.255.255`.
- **Clase C**: se asignan los tres primeros octetos para identificar la red, reservando el octeto final (8 bits) para que sea asignado a los hosts, de modo que la cantidad máxima de hosts por cada red es $2^8 - 2$. Los primeros bits más significativos son 110. Sería desde `192.0.0.0` hasta `223.255.255.255`.
- **Clase D**: serían las IPs que empiezan por 1110. Desde `224.0.0.0` hasta `239.255.255.255`.

- **Clase E**: serían las IPs que empiezan por 1111. Desde `240.0.0.0` hasta `255.255.255.255`.

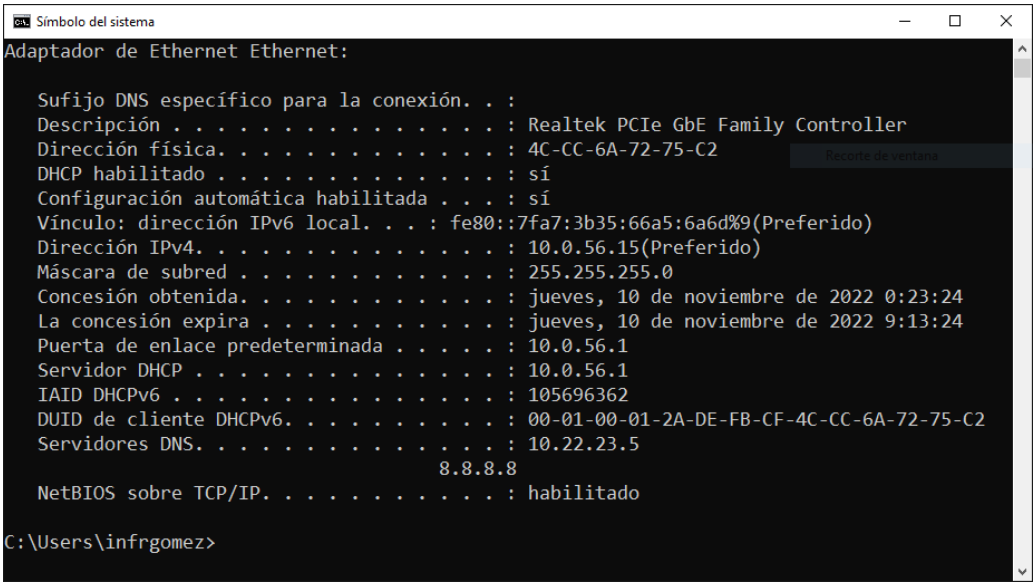
4.6. Configurar IPv4

Una vez sabemos cómo funciona IPv4, es el momento de aplicarlo en el sistema operativo en el que nos encontremos.

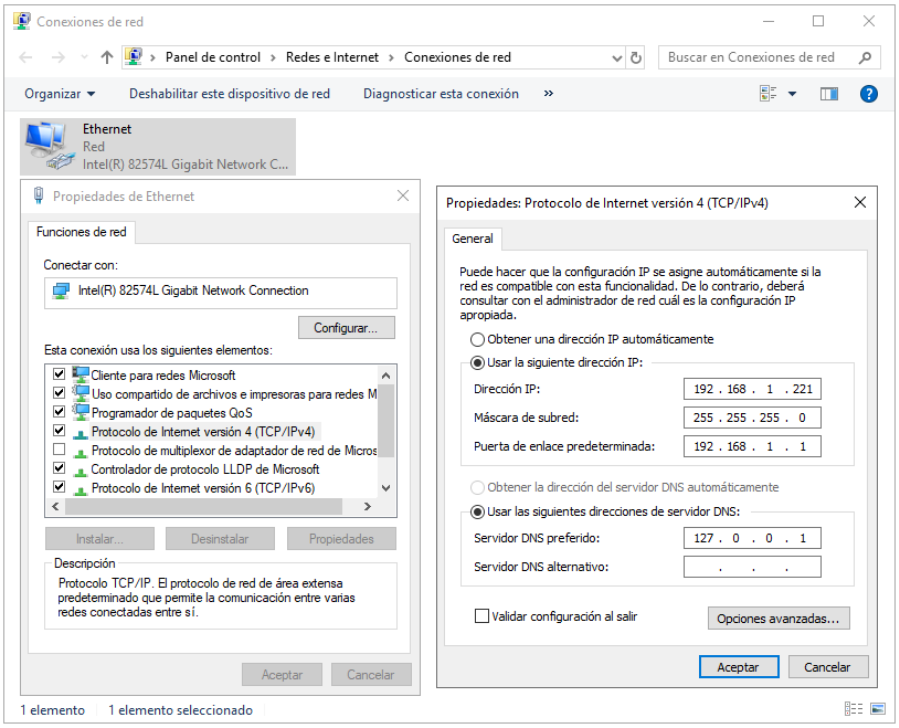
4.6.1. IPv4 en Windows

Por defecto, la configuración de red en Windows solicita una dirección IP de manera automática cuando el cable está conectado en el interfaz de red.

Para conocer la configuración actual de la red podemos utilizar el comando `>_ ipconfig` desde el terminal de windows. Si le pasamos el parámetro "/all" (`>_ ipconfig /all`), el resultado del comando nos dará más información, como es la configuración del DNS, la dirección física (MAC), ...



Si queremos realizar la modificación de la red, podremos hacer uso del icono de red que está al lado de la hora. Para ver toda la configuración, podemos ir al "Panel de Control → Redes e Internet → **Configuración de Red e Internet**". Elegimos "Protocolo de Internet versión 4", le damos a "Propiedades" y podremos realizar la configuración siguiente:



En la imagen se puede ver la configuración realizada:

- **Dirección IP**: La dirección IP que queremos que tenga el equipo.
- **Máscara de subred**: La máscara de red a la que pertenece la IP que hemos configurado.
- **Puerta de enlace predeterminada**: Para que el equipo tenga conexión con otras redes, debemos indicar qué IP tiene el ***gatway***.

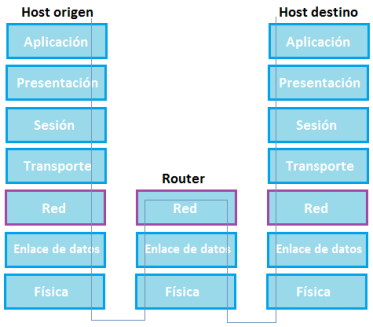
Aparte, podemos realizar la configuración del servidor DNS primario y secundario.

5. Componentes básicos en una red

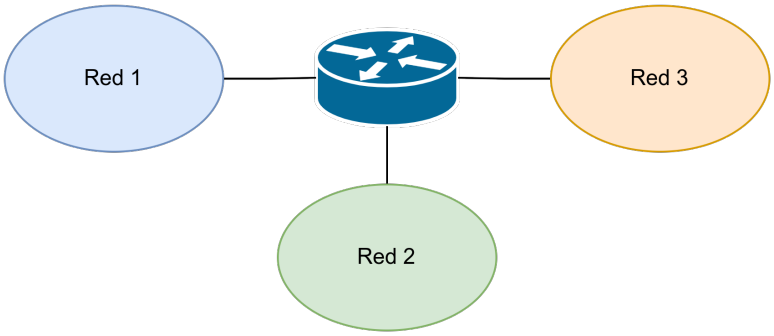
5.1. Router

Un router (o encaminador) es el encargado de enrutar (o encaminar) los paquetes que recibe de una red a otra red buscando la ruta más adecuada para ello. Pertenecen a la **capa 3 del modelo OSI**.

Un router puede "unir" redes, por lo que para ello es necesario que tenga interfaces configuradas (IPs) en las redes que quiere comunicar. Tendrá tantas interfaces configuradas como redes a las que esté unido.



Router en el modelo OSI (Fuente: Wikipedia)



Router conectado a 3 redes

El ejemplo más sencillo de router lo tenemos en casa, que es proporcionado por nuestro ISP. Este router une la red privada donde conectamos nuestros equipos (PCs, tablets, móviles) y la red del proveedor que a su vez nos dará acceso a Internet.

5.1.1. Puerta de enlace predeterminada

La puerta de enlace predeterminada (en inglés ***default gateway***) es el dispositivo por defecto por el que irá la comunicación de un equipo cuando trate de comunicarse con una red que no es la suya.

Sin una puerta de enlace, nuestro PC sólo podría comunicarse con otros equipos de la misma red, ya que el switch se encarga de ello, pero no podríamos realizar ninguna comunicación con ningún equipo que estuviese fuera de la red.

¡Atención!

Los routers también pueden tener puertas de enlace predeterminadas.

Los gateway también tienen otras funciones a la hora de intercomunicar redes, como por ejemplo:

- Traducir la información que se envía utilizando el protocolo de la red de origen al protocolo utilizado en la red de destino.
- Realizar enmascaramiento de la IP de la red origen cambiándola por la IP del dispositivo en la red de destino (también conocido como NAT).

Un equipo sólo podrá contar con una puerta de enlace predeterminada configurada, pero mediante rutas estáticas podremos elegir cómo encaminar tráfico a otras redes destino.

Información

Un equipo sólo podrá contar con una puerta de enlace predeterminada configurada.

Para saber cuál es la puerta de enlace de un equipo informático, dependeremos del sistema operativo en el que nos encontremos. En un entorno GNU/Linux actual podremos obtenerlo por consola de la siguiente manera:

Obtener puerta de enlace en GNU/Linux

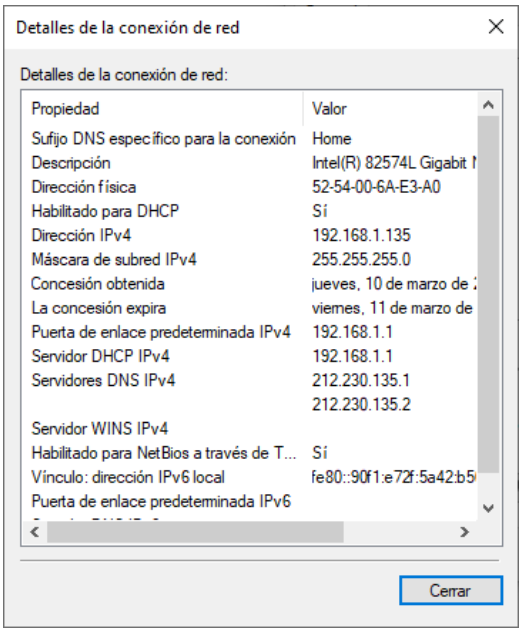
```
ruben@ubuntu:~$ ip route
default via 192.168.1.1 dev enp1s0 proto dhcp metric 100
```

En distribuciones antiguas se hacía

Obtener puerta de enlace en GNU/Linux

```
ruben@ubuntu:~$ route -n

Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    100    0        0 enp1s0
```

En un entorno Windows, podremos verlo a través del interfaz gráfico yendo a "**Configuración → Estado de red → Cambiar opciones del adaptador**", donde veremos los adaptadores que tiene nuestro equipo. Seleccionaremos uno, y haciendo click derecho le daremos a "**Estado → Detalles**".



Para no dar tantos pasos, si ejecutamos en la terminal **CMD** el siguiente comando también lo veremos:

Obtener puerta de enlace en Windows

```
C:\Users\ruben> ipconfig

Configuración IP de Windows
Adaptador de Ethernet Instancia de Ethernet 0 2:

Sufijo DNS específico para la conexión. . : Home
```

```
Vínculo: dirección IPv6 local. . . : fe80::90f1:e72f:5a42:b50d%7
Dirección IPv4. . . . . . . . . . . . . : 192.168.1.135
Máscara de subred . . . . . . . . . . . : 255.255.255.0
Puerta de enlace predeterminada . . . . . : 192.168.1.1
```

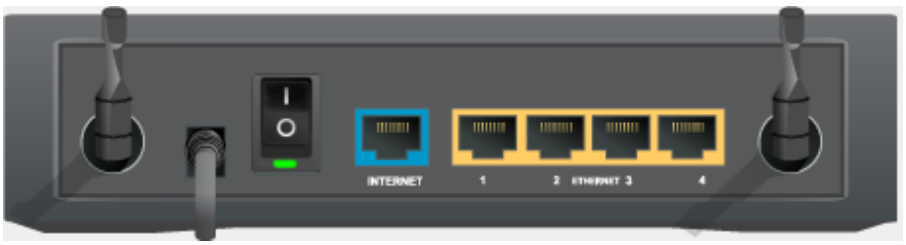
Pueden existir equipos sin puerta de enlace, pero no suele ser lo habitual. Esto sucede cuando queremos que equipos puedan ver a otros equipos de la red, pero no puedan comunicarse con el exterior. Suele ser más habitual realizar bloqueos a nivel de firewall.

5.1.2. Router casero

Los routers caseros son dispositivos que nos permiten conectarnos a internet. El problema es que normalmente suelen ser muy básicos y su funcionalidad es limitada.

Eso no quita que realmente cumpla la función para la que han sido creados, que es la de permitir la interacción de una red como la de un hogar hacia internet.

Los routers caseros suelen tener diferenciada en la parte de atrás el interfaz donde se debe conectar el cable que va hacia Internet (en este caso de color azul) y las bocas que van a formar la red LAN interna de casa (color amarillo), que realmente son un pequeño switch de 4 bocas.



Home-Router en Packet Tracer

Dependiendo del tipo de conexión que tangamos en casa, el interfaz que nos conecta a internet será distinto, pudiendo ser hoy día los más habituales:

- **Cable coaxial**: En conexiones con Euskaltel en el que no nos llega la fibra hasta casa.
- **Conector de fibra**: Cuando la fibra nos llega hasta nuestra casa.

Los routers que tenemos en casa también tiene otras funcionalidades básicas que vienen pre-configuradas como veremos a continuación, entre las que podemos destacar:

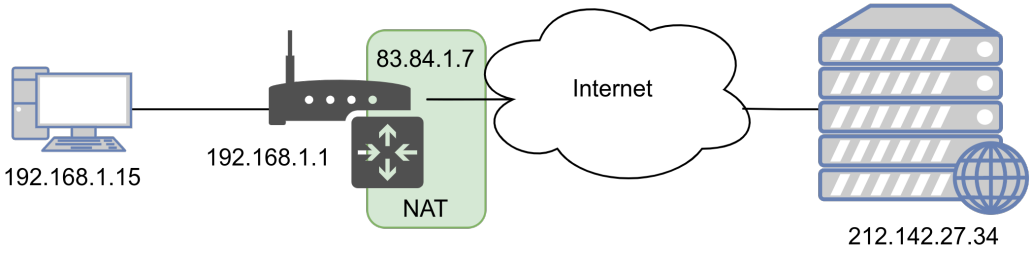
- **Direccionamiento LAN**: Normalmente viene configurado con la red 192.168.1.0/24 o 192.168.0.0/24
- **DHCP Server**: Servicio que otorga IPs en nuestra red LAN, teniendo en cuenta el direccionamiento que tengamos.
- **Configuración Wifi**: Dependiendo del modelo tendremos distintas redes inalámbricas (en distintos rangos). Podremos configurar el nombre de la red, el canal y el tipo de seguridad de acceso a la misma.

- **Restricción hacia internet**: Algunos routers permiten limitar el acceso a Internet en ciertos horarios (ideal para restringir el acceso a menores de edad).
- **Redirección de puertos**: Para redirigir conexiones desde Internet a un equipo concreto dentro de la red.
- **Filtrado MAC**: Normalmente para el apartado Wifi, y así limitar quién se puede o no se puede conectar.

5.2. NAT

La traducción de direcciones de red, también llamado enmascaramiento de IP o NAT (del inglés ***Network Address Translation***), es un mecanismo utilizado por routers que conectan dos (o más) redes para que el paquete que llega a un equipo destino no parezca que llega desde la red de origen.

Vamos a tomar como ejemplo el siguiente esquema que es un ejemplo que podemos entender teniendo en cuenta la conexión de nuestra casa:



En este ejemplo tenemos conectado un equipo con IP 192.168.1.15 al router que tenemos en casa (cuya IP en la LAN es 192.168.1.1). Este router cuenta con una IP pública proporcionada por el ISP para el acceso a internet (83.84.1.7). Cuando nuestro equipo quiere comunicarse con algún equipo que no está en su red, al salir a internet, el router cambia la cabecera del paquete sustituyendo 192.168.1.15 por 83.84.1.7, y de esta manera al servidor remoto (212.142.27.34) el paquete le llega desde una IP pública.

Este es el ejemplo más sencillo y que hacemos uso cada día en casa, pero esto no significa que NAT sólo se realice entre redes públicas y privadas. Lo podemos utilizar entre dos redes públicas o dos redes privadas también.

Las traducciones de dirección se almacenan en una tabla, para recordar qué dirección y puerto le corresponde a cada dispositivo cliente y así saber donde deben regresar los paquetes de respuesta.

Existen distintos tipos de NAT:

- **NAT de sobrecarga**: Varios equipos de la red de origen se traducen por una única dirección de la red de salida. Es el método más habitual (el que realizan nuestros routers en casa).
- **NAT estática**: También conocida como “NAT 1:1”, ya que una dirección IP privada se traduce siempre por una única dirección IP pública, y siempre será la misma. Este método es el habitual cuando queremos tener un servidor en la red interna y queremos que su comunicación con el exterior siempre sea con la misma IP pública

- **NAT dinámica**: Similar al caso anterior, pero en este caso el router contará con una tabla de IPs públicas y se asignará una que esté libre de esta tabla a un equipo interno cuando necesite comunicarse de manera pública. Cuando deje de necesitarlo, la IP se marcará como "libre" y se podrá asignar a otro equipo interno posteriormente.

5.3. DHCP

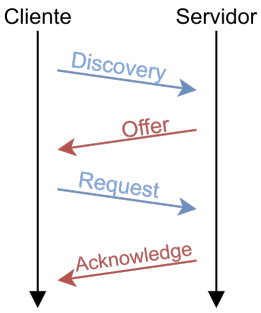
El protocolo de configuración dinámica de host (*Dynamic Host Configuration Protocol*, también conocido por sus siglas de **DHCP**) es un protocolo de red que nos permite configurar la IP de un dispositivo dentro de una red.

Normalmente el protocolo DHCP enviará la configuración de los siguientes parámetros para que el equipo los use:

- **IP**: IP asignada al equipo.
- **Máscara de red**: para identificar el tamaño de la red.
- **Default gateway**: para que el equipo se pueda comunicar con otra red.
- **DNS**: el servidor DNS que utilizará el equipo para la resolución de nombres.

Existen muchas más configuraciones que se pueden enviar a un equipo que realiza una petición de configuración por DHCP, pero las arriba expuestas son las más habituales.

El protocolo funciona en forma "Cliente/Servidor", siendo el equipo el que realiza la búsqueda de un servidor DHCP para el inicio de la configuración. DHCP hace uso de los puertos 67/UDP para el servidor y 68/UDP para los clientes.



Referencia

Lo más habitual es que el servidor DHCP esté configurado en los routers (tal como sucede en los que nos proporcionan los ISP), pero no tiene por qué ser así, pudiéndose instalar en un equipo con Windows Server, GNU/Linux, …

5.3.1. Configuración

Dependiendo del equipo en el que estemos realizando la configuración del servidor DHCP, el interfaz a configurar podrá ser distinto.

5.4. Swith

Un *switch* (o conmutador) es el dispositivo digital que interconecta equipos que están en la misma red, y actúa en la **capa 2 del modeo OSI**. También permiten conectar distintos segmentos de la misma red, uniendo distintos switches entre sí.

Switch profesional Cisco Business CBS350-48T-4X

Los switches poseen la capacidad de aprender y almacenar las direcciones de red de la capa 2 (**direcciones MAC**) de los dispositivos alcanzables a través de cada uno de sus puertos. Por ejemplo, un equipo conectado directamente a un puerto de un switch provoca que el propio switch almacene su dirección MAC. Esto permite que la información dirigida a un dispositivo vaya desde el puerto origen al puerto de destino.

Los switches profesionales tienen un interfaz de configuración (tanto por consola como interfaz web) para poder realizar distintas tareas de configuración, tanto a nivel general como a nivel de puerto individual.

V

PfSense

1. Introducción

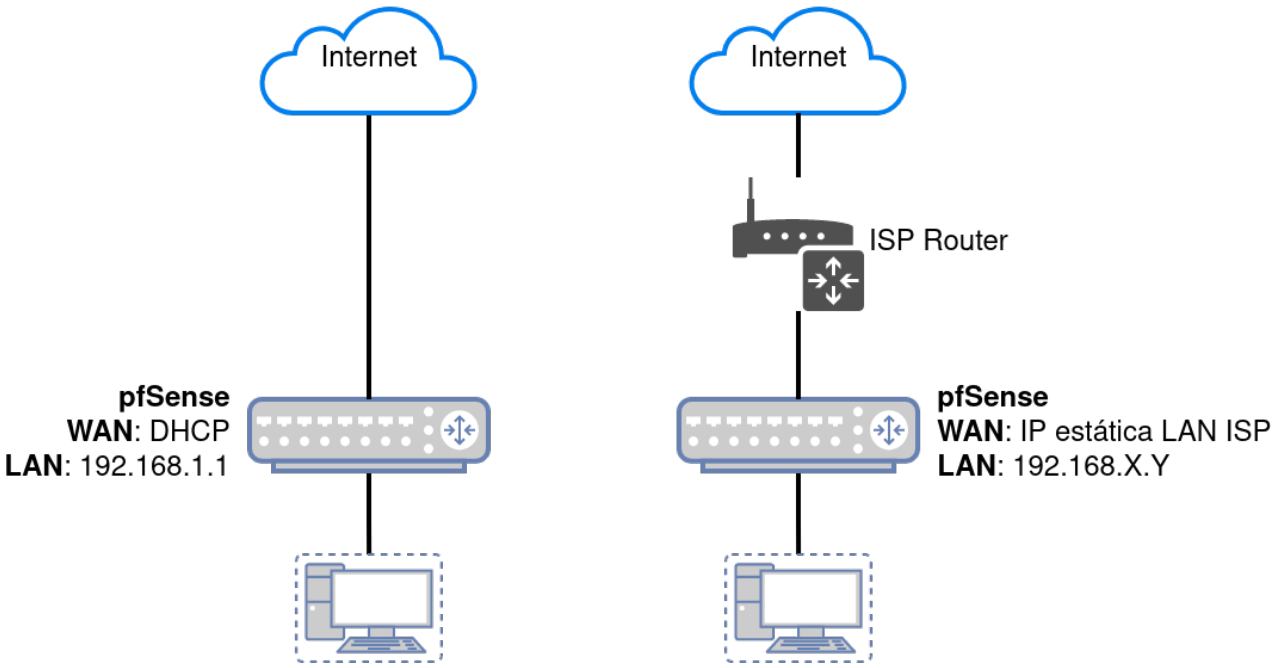
En este documento se va a explicar cómo realizar la instalación y configuración de un servidor que hará las funciones de cortafuegos basado en la distribución pfSense.

Para completar la simulación, se usará un equipo que va a pertenecer a la LAN de la infraestructura, y lo usaremos para poder ver cómo se bloquea el tráfico y para realizar pruebas en la configuración del pfSense recién instalado.

1.1. Antes de empezar

La idea de este documento es crear una pequeña infraestructura de red haciendo uso de un firewall (o cortafuegos) basado en pfSense. Con ello vamos a ver cómo funciona el sistema de creación de reglas de filtrado de tráfico para un equipo que estará dentro de la red LAN detrás de dicho firewall.

El esquema de infraestructura real quedaría de la siguiente manera, dependiendo de cómo realicemos la instalación y las posibilidades que tengamos con nuestro proveedor de internet:



- **1ª opción**: el pfSense actúa como conexión directa a internet. Para ello estará conectado a un router neutro, cable-modem, ONT o lo habremos configurado como nuestro ISP nos indique. Por lo tanto pfSense tendrá IP pública a internet y actuará como firewall directo.
- **2ª opción**: haciendo doble NAT, detrás del router de nuestro proveedor de Internet. Para que todo funcione de manera correcta, en el router del proveedor deberemos hacer una redirección de puertos para que todas las conexiones vayan a la IP del servidor pfSense.

1.2. Requisitos

Esta simulación se puede realizar de dos maneras:

- haciendo uso de hardware dedicado.

- usando máquinas virtuales.

La manera más sencilla es realizarlo haciendo uso de máquinas virtuales con un virtualizador como es Virtualbox, por lo que se explicará esta modalidad.

2. PfSense

PfSense es una distribución de FreeBSD (no confundir con GNU/Linux, ya que FreeBSD es Unix) que está adaptada para que actúe como un sistema de firewall, enrutador, control de tráfico, servidor DNS, DHCP, VPN, proxy y muchos servicios más.

Tal como veremos a continuación, la instalación es sencilla y la configuración de los servicios se realiza a través de un interfaz web desde el que se puede controlar las reglas de filtrado que se pueden crear, las configuraciones que se van a realizar, …

Hoy en día, la empresa que está detrás de pfSense, Netgate, vende unos sistemas appliance (hardware específico para realizar las funciones de firewall, con la distribución preinstalada), pero lo habitual suele ser que la instalación se realice sobre un sistema hardware de servidor o virtualizado.

A nivel de características técnicas lo único que se pide es un procesador basado en la arquitectura x86_64 (Intel o AMD) de 600MHz, 512 MB de RAM y 4GB de disco duro. Lógicamente, este es el hardware mínimo recomendado, y dependiendo de la cantidad de tráfico que tenga nuestra infraestructura deberemos tener un hardware adecuado para el mismo. En la documentación oficial hacen referencia al hardware que podamos necesitar dependiendo del tráfico que vayamos a tener.

Dado que se va a optar por realizar la instalación en una máquina virtual, se va a necesitar un sistema de virtualización (Virtualbox) y el CD de instalación. La instalación será idéntica si se realiza en hardware físico, o en otro sistema de virtualización.

2.1. Detalles de la máquina virtual

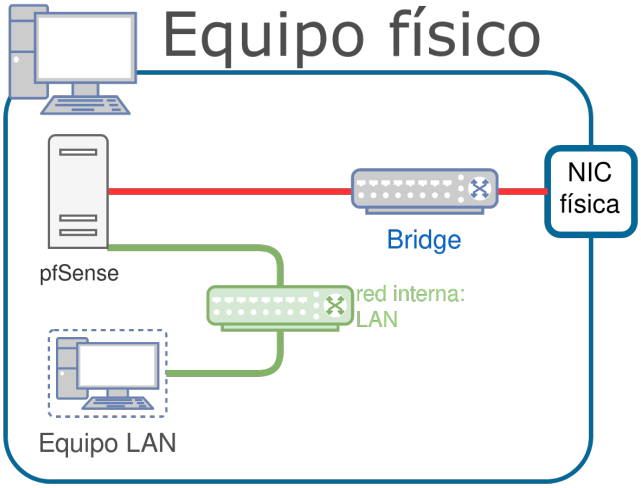
No se va a detallar cómo crear una máquina virtual, pero si las características técnicas que debe tener cuando se crea en Virtualbox.

Dado que pfSense está basado en un sistema Unix FreeBSD, la máquina tiene que crearse indicando el tipo “BSD” y la versión “FreeBSD” de 64 bits, tal como aparece en la imagen.

Por otro lado, a la máquina virtual se le van a añadir dos interfaces de red:

- El primer adaptador de red será de tipo “Adaptador puente”, ya que en el sistema haremos que sea la interfaz que actuará como “WAN”.
- El segundo adaptador de red se creará de tipo “Red interna” y le pondremos el nombre de “LAN”, que hará esas funciones en nuestra infraestructura.

Dadas las explicaciones previas, una vez creada la máquina virtual nuestra infraestructura virtual quedaría de la siguiente manera:



Visto este dibujo, la máquina virtual que actuará como PC dentro de la LAN le tendremos que modificar el adaptador virtual para que sea de tipo "Red interna" y escribiremos "**LAN**", por lo que ambas máquinas estarán conectadas mediante un "switch virtual".

El resto de parámetros de la máquina virtual, como se trata de una para pruebas, será:

- 8 GB de disco duro
- 1GB de memoria RAM

2.1.1. Máquina virtual en la LAN

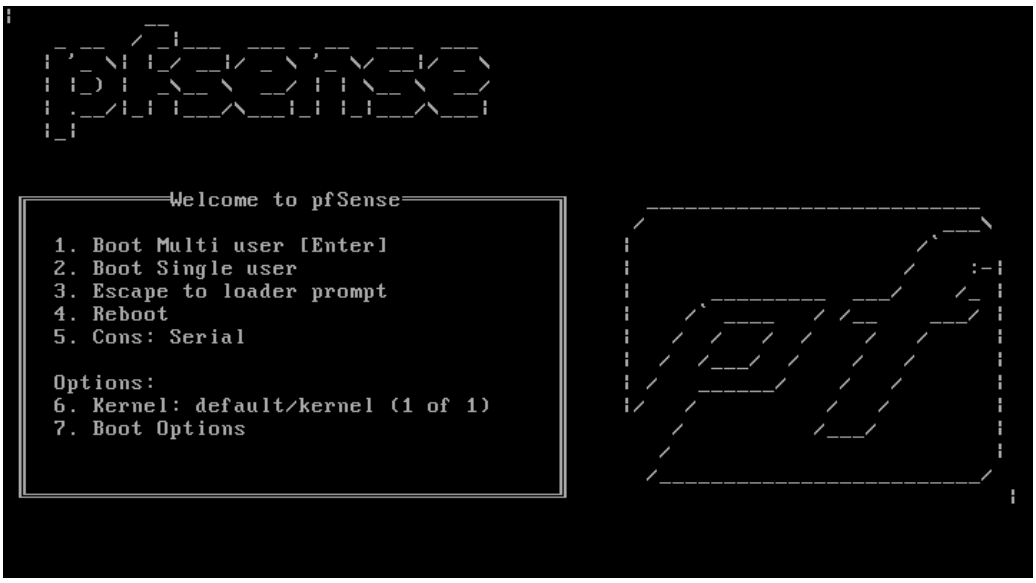
Tal como se ha visto en el dibujo anterior, vamos a usar una máquina virtual dentro de la *LAN virtual* de pfSense.

Para ello necesitamos configurar el interfaz de la máquina virtual en modo "Red interna" y seleccionaremos la "**LAN**" creada previamente.

Queda por parte del lector el crear esta máquina virtual, pero se recomienda realizar la instalación de una distribución GNU/Linux en ella.

2.2. Instalación

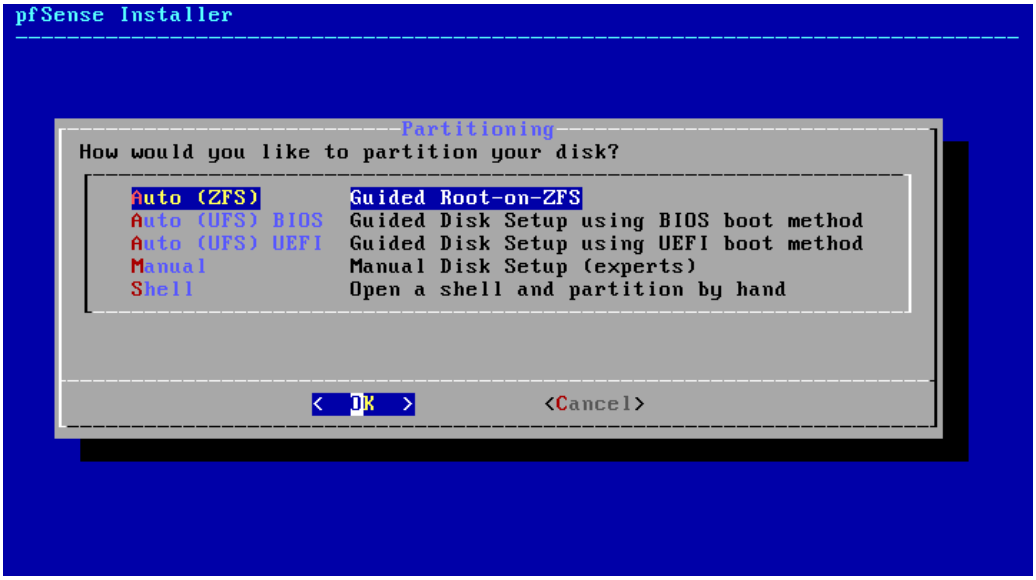
Tras poner el CD de instalación en la máquina virtual y arrancar veremos un pequeño menú como muestra la siguiente captura de pantalla:



El menú contará con un sistema de cuenta atrás y si no se selecciona nada entrará en la primera opción por defecto. Se podrá ver cómo el sistema arranca y detecta el hardware y al finalizar nos mostrará un menú con las opciones:

- **Install**: Instalar pfSense
- **Rescue Shell**: Lanza una terminal para poder recuperar una instalación que esté dando problemas
- **Recover config.xml**: recupera el fichero de configuración config.xml de una instalación previa.

Tras seleccionar la opción de **instalar**, nos aparecerá un menú para seleccionar la distribución del teclado y a continuación el tipo de partición que queremos utilizar:



- **Auto (ZFS)**: Sistema de particionado con el sistema de ficheros ZFS. Es el sistema por defecto, aunque ZFS puede consumir más RAM.
- **Auto (UFS) BIOS**: si nuestro sistema usa BIOS.
- **Auto (UFS) UEFI**: si nuestro sistema usa UEFI.

- **Manual**: Nos permite particionar de manera manual, para expertos.
- **Shell**: Abre una consola y podremos realizar el particionado a mano.

Al seleccionar la opción “Auto (ZFS)”, el sistema nos permitirá crear un particionado en modo:

- **stripe**: sin redundancia, usando el disco duro instalado.
- **mirror**: haciendo uso del sistema RAID1
- **raid10**: hacer uso de un sistema RAID1+0
- **raidz1**: redundancia con un disco de paridad
- **raidz2**: redundancia con dos disco de paridad
- **raidz3**: redundancia con tres disco de paridad

Tras terminar, nos preguntará si queremos abrir una terminal en el sistema recién instalado. Le diremos que no y que reinicie el sistema. Nos tendremos que asegurar de quitar la ISO de la máquina virtual para que arranque desde el disco duro en lugar del CD.

3. Configuración básica

En este apartado se va a detallar cómo realizar una configuración básica de pfSense para que actúe como firewall dentro de la red simulada que hemos creado.

3.1. Primer arranque

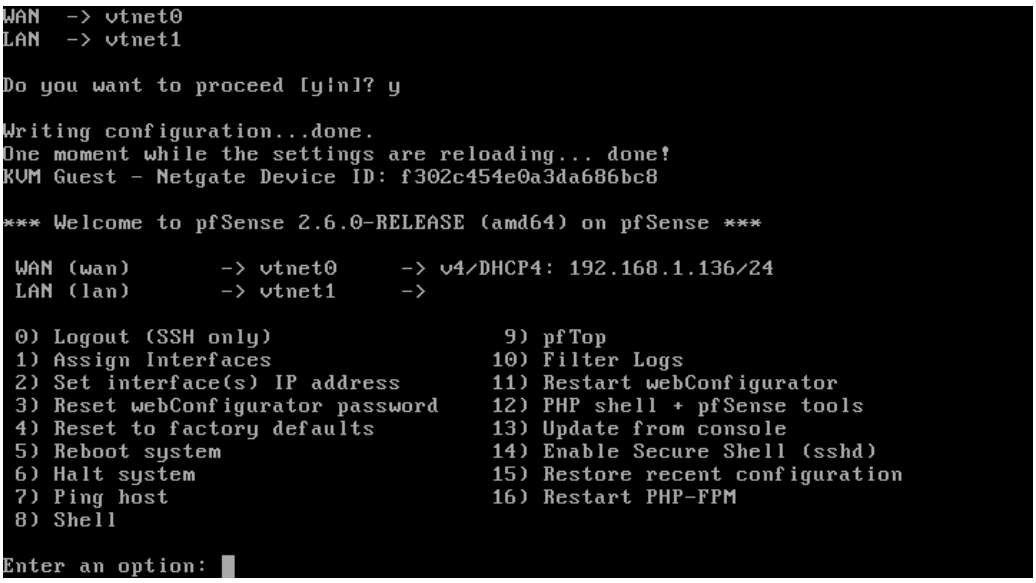
Tras realizar la instalación y reiniciar, en el primer arranque desde el disco duro aparecerá un pequeño asistente que comprobará el hardware, detectará los interfaces de red que tenemos y nos realizará una serie de preguntas:

- Si es necesario configurar VLANs.
- De los interfaces existentes, cuál es el WAN.
- De los interfaces existentes, cuál es el LAN.

Dependiendo del direccionamiento de red en el que nos encontremos, es posible que haya que configurar la LAN virtual de nuestra infraestructura de red virtualizada.

3.1.1. A tener en cuenta en redes 192.168.1.0 /24

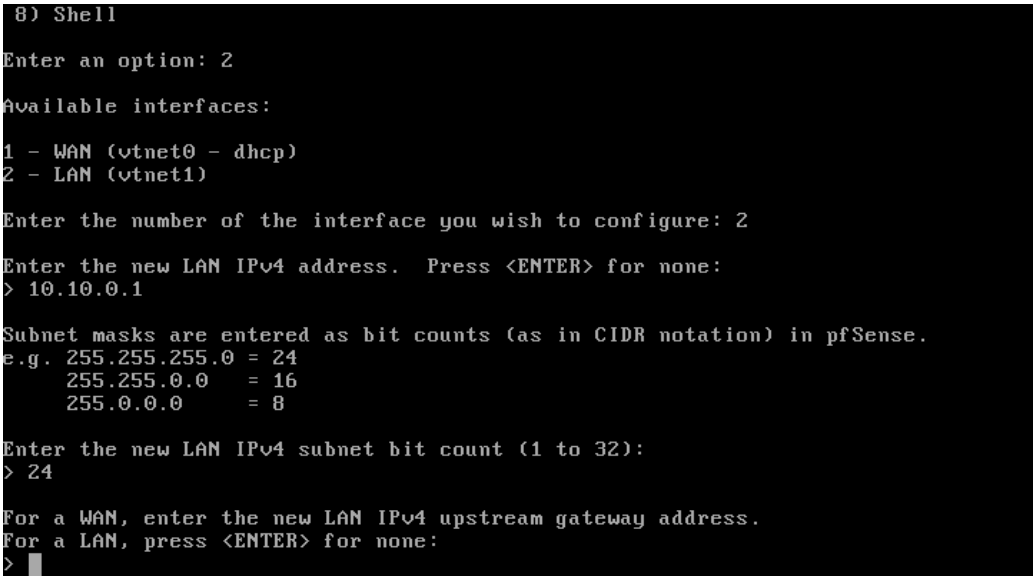
Dado que pfSense por defecto hace uso de una red LAN 192.168.1.0/24, en caso de que nuestra LAN física contenga ese direccionamiento, el interfaz LAN de pfSense no será configurado. Tendremos un menú como el siguiente:



Tal como se puede ver en la imagen, el asistente ha cogido IP por DHCP para el interfaz WAN, pero el interfaz LAN no se ha configurado ya que la WAN ya tiene el direccionamiento 192.168.1.0/24.

3.1.2. Configurar LAN virtual

En las situaciones mencionadas en el paso anterior, o en casos de que queramos modificar la red LAN, podremos cambiarla desde el menú seleccionando la opción 2.



Tal como se puede ver en la imagen anterior, al elegir la opción 2 el asistente nos pregunta por el interfaz que queremos configurar. Durante el proceso nos realiza las siguientes preguntas:

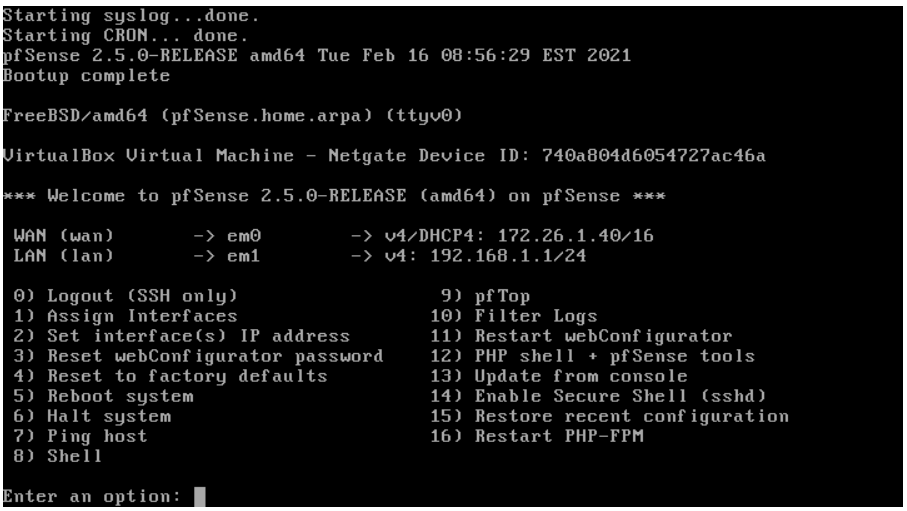
- **Interfaces disponibles**: En nuestro caso, la LAN.
- **Dirección IPv4**: La dirección IPv4 para el interfaz seleccionado.
- **Máscara de red**: Máscara del direccionamiento para la IP anterior.
- **Dirección IPv6**: En caso de querer configurar el interfaz en IPv6
- **Activar servidor DHCP**: En el caso configurar la LAN, es interesante configurar el servidor DHCP.

- **IP inicial del rango**: Dentro de la LAN, la primera IP que se asignará por DHCP.
- **IP final del rango**: Dentro de la LAN, la última IP que se asignará por DHCP.

Al terminar el asistente, nos aparecerá de nuevo el menú que pasaremos a explicar a continuación.

3.2. Menú de configuración desde consola

Tal como se ha comentado, el menú cuenta con distintas opciones de administración.

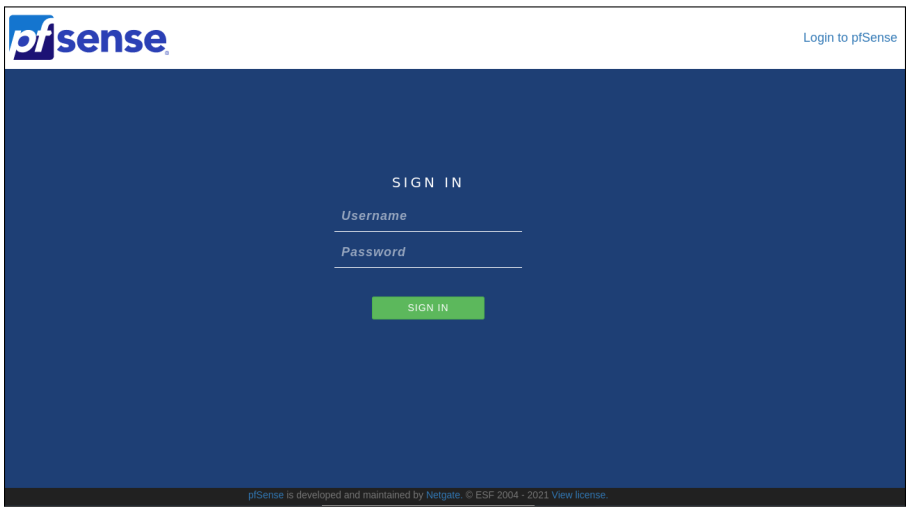


Como se puede ver en la imagen previa, aparecen 16 posibles opciones a elegir, entre las que destacaremos:

1. **Assign interfaces**: Para poder reconfigurar a qué red pertenecen los interfaces que tiene nuestro firewall (si es WAN, LAN, …).
2. **Set interfaces IP address**: Para poder modificar la IP de los interfaces que tiene nuestro firewall.
3. **Reset webConfigurator password**: Cambiar la contraseña de acceso.
4. **Reset to factory defaults**: Restaurar el servidor a los valores de “fábrica”, es decir, resetea todas las configuraciones propias realizadas.
5. **Reboot**: Reinicia el servidor.
6. **Halt**: Apaga el servidor.
7. **Ping** host: Realiza un ping al equipo indicado.
8. **Shell**: Nos abre una consola en el sistema para poder realizar modificaciones mediante comandos.
9. **pfTop**: Nos muestra un listado de las conexiones establecidas en tiempo real.
10. **Enable Secure Shell (sshd)**: Habilita el servidor SSH para poder realizar conexiones. Por defecto, sólo podremos conectarnos desde la LAN.
11. **Restart PHP-FPM**: Reinicia el servicio del interfaz web.

3.3. Interfaz web de configuración

El acceso a la interfaz web de configuración **por defecto sólo está disponible desde la red LAN**, por lo que accederemos desde la máquina virtual dentro de la LAN abriendo un navegador y apuntando a la IP por defecto de la LAN "https://192.168.1.1" (o la que hayamos puesto si hemos configurado la LAN). Tendremos que aceptar el certificado de seguridad (ya que es auto-firmado) y nos aparecerá la web de login.



Los credenciales por defecto son:

- **username**: admin
- **password**: pfsense

El usuario "admin" de la web tendrá la misma contraseña que el usuario root por consola.

¡Cuidado!

Dada la importancia de los cortafuegos, la contraseña que usemos debe ser segura y sólo los administradores del servidor la deben conocer.

Cuando nos logueamos por primera vez nos aparecerá la primera página del asistente de configuración de pfsense, que nos guiará a través de nueve pasos donde podremos hacer una configuración básica de pfSense. Los pasos serán los siguientes:

0. **Bienvenida al asistente**: página de inicio.
1. **Soporte de Netgate**: detrás de pfSense hay una empresa, Netgate, que ofrece sus servicios de soporte de pago para pfSense.
2. **Información general**: configuración básica del servidor:
   - **hostname**: nombre del servidor.
   - **domain**: nombre del dominio al que pertenece el servidor.
   - **DNS Server**: servidores DNS externos al que mandar las peticiones.
3. **Servidor de tiempo**: configurar la zona horaria y el servidor NTP.

4. **Configurar interfaz WAN**: Dependiendo de cómo sea la infraestructura real, podremos configurar el interfaz WAN con IP pública estática, por DHCP, por PPPoE, … Por defecto está en DHCP. Si estamos tras doble NAT, habría que desactivar la opción "RFC1918 Networks".
5. **Configurar interfaz LAN**: La IP de pfSense en el direccionamiento LAN. Por defecto es 192.168.1.1 con una máscara "/24".
6. **Cambio de contraseña del admin**: Para poner una contraseña más segura.
7. **Recargar la configuración**: Si se ha realizado algún cambio, recargará la configuración.

4. Reglas de filtrado

PfSense es un servidor que actúa como **firewall** (entre otras de sus posibles funciones) y por tanto permite la creación de reglas de filtrado de tráfico para todos los interfaces que tiene configurados.

Información

Las reglas se ejecutan **cuando el tráfico entra al interfaz** y son evaluadas en base a "primer acierto".

Las acciones principales de las reglas pueden ser:

- **Pass**: Permite el tráfico al destino.
- **Reject**: Rechaza el paquete y avisa al emisor.
- **Block**: Rechaza el paquete de manera silenciosa.

Si el tráfico no coincide con alguna regla que sea explícitamente **pass** el tráfico será denegado.

¡Atención!

Por defecto pfSense deniega todo el tráfico entrante a sus interfaces.

Aunque las opciones "*block*" y "*reject*" rechazan el paquete, la diferencia puede suponer una gran diferencia, ya que "reject" responde con **TCP RST** (o "*port unreacheable*") y eso puede permitir la posibilidad de recibir un ataque de denegación de servicio (***DoS***).

¡Cuidado!

¡Nunca se debería usar "*reject*" en el interfaz WAN!

En redes privadas es útil hacer uso de "*reject*", porque avisa a los programas que intentan realizar conexiones que la conexión está bloqueada, y por tanto la respuesta es más rápida ya que no se esperan timeouts.

4.1. Ciclo de vida de una conexión

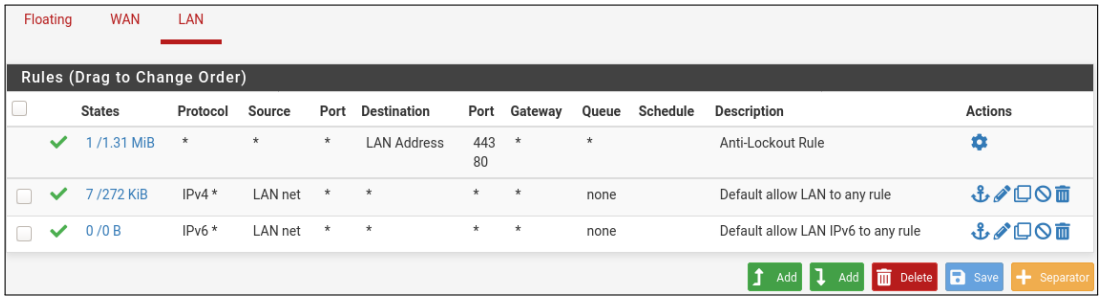
A continuación se explica cómo actúa pfSense al recibir un paquete de una nueva conexión:

- El paquete, como parte de una nueva conexión, llega a un interfaz.
- Se comprueban las reglas de filtrado **en orden descendiente** contra el paquete.
- Cuando existe coincidencia, se ejecuta la acción de la regla de filtrado.
- Se para las comprobaciones de las reglas.
- Si no ha existido una coincidencia tras comprobar todas las reglas, **el paquete es bloqueado por defecto**.

4.2. Reglas creadas por defecto

Tras la instalación de PfSense podemos ir a "Firewall → Rules" y ahí aparecen los interfaces que están configurados actualmente y en los que se pueden crear reglas de filtrado. Los interfaces que existirán en nuestra infraestructura serán:

- **Floating**: No es una interfaz al uso. Es para crear reglas de filtrado especiales que se pueden saltar el orden y/o aplicar a todos los interfaces. **Es mejor no utilizarlo salvo que sepamos qué estamos haciendo y hayamos leído detenidamente** la documentación sobre ello.
- **WAN**: Bloquea todos los paquetes.
- **LAN**: Existen varias reglas creadas por defecto que permiten tráfico:



- ✔Permite el acceso a la IP de la LAN del pfsense al puerto 80 y 443 para poder administrarlo vía web. Esta regla está especialmente creada para que no se pueda eliminar desde este apartado, ya que el borrarla podría suponer no poder configurar pfSense vía web.
- ✔Permite cualquier tráfico de tipo IPv4
- ✔Permite cualquier tráfico de tipo IPv6

Las últimas dos reglas explicadas previamente permiten que el tráfico pueda salir de la LAN hacia internet y así se permita navegar por internet. Si se crease una nueva red LAN (para una red wifi, DMZ, …) estas reglas no estarían creadas, por lo que desde esa nueva LAN no se podría acceder a ninguna otra red, por lo que habría que crear reglas que permitieran el tráfico.

4.3. Crear regla de denegación

Tal como se ha comentado, por defecto desde la LAN se permite acceder a cualquier red, por lo que podemos hacer ping a cualquier equipo de Internet. Como ejemplo se va a crear una regla que impedirá el acceso a un servidor de Internet. Esta regla servirá de ejemplo para poder realizar cualquier otra regla que sea necesaria.

La regla se va a crear en el interfaz LAN, ya que queremos limitar el tráfico cuyo origen es esa red. Para crear la regla existen dos botones de creación:

- **Añadir la regla al principio de la lista**: como su propio nombre indica, creará la regla al principio de la lista en la que aparecen todas las reglas que ya están creadas.
- **Añadir la regla al final de la lista**: en este caso creará la lista al final de todas las reglas.

No importa dónde se cree la nueva regla, ya que se podrá modificar después su posición. Al crear la regla, tendremos que tener en cuenta los siguientes apartados:

- **Action**: Qué queremos hacer con la regla: **pass**, **block** o **reject**.
- **Disabled**: Para deshabilitar la regla. Suele ser buena idea deshabilitar temporalmente las reglas que no se necesiten en lugar de borrarlas, por si nos hemos confundido y hay que recuperarlas.
- **Interface**: El interfaz sobre la que se va a crear la regla.
- **Familia de dirección**: Si queremos aplicar la regla sobre IPv4, IPv6 o ambas.
- **Protocolo**: Protocolo de la conexión: TCP, UDP, ICMP, Any…
- **Source**: Origen de la conexión. Si es “Any” será desde cualquier equipo de la red elegida. Podremos elegir un único equipo u otras opciones.
- **Destination**: El destino de la conexión. Si es “Any” será a cualquier equipo. Podremos elegir un equipo u otras opciones.
- **Extra options**: Opciones extra y avanzadas para la regla (limitar número de conexiones, modificar el gateway de salida, …).
- **Description**: Suele ser recomendable añadir una descripción a las reglas para identificar el servicio o el servidor sobre el que se aplica.

Para el ejemplo se va a bloquear todo el tráfico desde la LAN, al servidor 1.1.1.1 (servidor DNS de la empresa Cloudflare). La regla quedaría:

Una vez creada la regla aparecerá un botón para aplicar los cambios, por lo que hasta que no sea pulsado ese botón, las nuevas reglas que se hayan creado no tendrán efecto y por tanto no entrarán en funcionamiento.

4.4. Orden de las reglas

Tal como se ha identificado en el ciclo de vida de las conexiones, el orden de las reglas es muy importante, ya que en el momento en el que el tráfico entra sobre el interfaz se comprobará si coincide con las reglas en orden descendente.

Por lo tanto, si existe una regla muy general que permite el tráfico y después una muy específica de bloqueo, es bastante probable que la regla específica de bloqueo no llegue a entrar en funcionamiento, ya que el tráfico coincidirá con la regla general que permite dicho tráfico.

Teniendo en cuenta la regla creada en el apartado anterior, vamos a analizar cuál es el comportamiento dependiendo del orden en el que se sitúa:

- Al **final** del todo:

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✓	1 /1.52 MiB	*	*	*	LAN Address	443 80	*	*		Anti-Lockout Rule	
✓	5 /397 KiB	IPv4 *	LAN net	*	*	*	*	none		Default allow LAN to any rule	
✓	0 /0 B	IPv6 *	LAN net	*	*	*	*	none		Default allow LAN IPv6 to any rule	
✗	0 /0 B	IPv4 *	*	*	1.1.1.1	*	*	none		bloqueo DNS cloudflare	

  Teniendo en cuenta las reglas creadas sobre el interfaz LAN en este orden, la regla de bloqueo al servidor 1.1.1.1 no entrará nunca en funcionamiento. El tráfico cuyo origen sea la LAN coincidirá siempre con la regla que le permite ir a cualquier parte, por lo que al coincidir con esa regla no se analizará ninguna más.

- Al **comienzo** de las reglas:

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✓	0 /1.58 MiB	*	*	*	LAN Address	443 80	*	*		Anti-Lockout Rule	
✗	0 /0 B	IPv4 *	*	*	1.1.1.1	*	*	none		bloqueo DNS cloudflare	
✓	7 /399 KiB	IPv4 *	LAN net	*	*	*	*	none		Default allow LAN to any rule	
✓	0 /0 B	IPv6 *	LAN net	*	*	*	*	none		Default allow LAN IPv6 to any rule	

  En este caso la regla más específica de denegación se ha puesto al principio, por lo que si el tráfico coincide con esta hará lo que indica la regla, bloquear el tráfico al servidor 1.1.1.1. Si no coincide, se seguirán analizando el resto de reglas, y en este caso se permitirá el resto de tráfico.

Es de vital importancia tener en cuenta el orden de las reglas, analizarlo con cuidado y realizar las pruebas oportunas para confirmar que lo que se pretende hacer es lo que termina sucediendo.

¡Cuidado!

¡El orden de las reglas de filtrado es muy importante en cualquier Firewall!

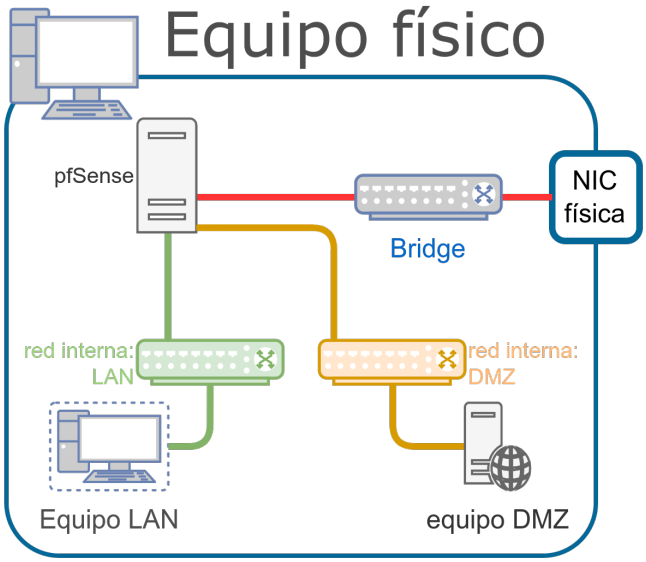
5. Crear una nueva red

Teniendo en cuenta lo visto hasta ahora, nuestra infraestructura cuenta con una única red LAN detrás del firewall y, aparte, el acceso a WAN. En este apartado se va a explicar cómo crear una nueva red para una sección DMZ (zona desmilitarizada) en la que se instalarán servidores. Normalmente lo habitual suele ser que el acceso desde y hasta la DMZ cumpla con las siguientes restricciones:

- **LAN → DMZ: permitido/limitado**, para el acceso a servicios como carpetas compartidas, páginas web, … Se permitirá sólo a los servicios que se ofrecen.
- **DMZ → LAN**: bloqueado. El acceso desde la DMZ a la LAN suele estar bloqueado ya que los servidores no deberían poder acceder a los equipos de los usuarios.
- **DMZ → Internet**: limitado. Dependerá de los servicios que estén instalados en los servidores. Para el acceso total a internet (para actualizaciones de software, por ejemplo) se suele permitir el acceso a través de un proxy que realizará el bloqueo y sólo permitirá las webs correspondientes.
- **Internet → DMZ**: limitado. Dependerá de los servicios que estén instalados en los servidores. Sólo se permitirá el acceso a los puertos de los servicios necesarios.

5.1. Modificando la infraestructura creada

Dado que se va a crear una red nueva, deberemos realizar cambios en la máquina virtual de nuestro servidor pfSense. La nueva infraestructura será la siguiente:



Tal como se puede apreciar al compararlo con la infraestructura anterior, se ha creado una nueva red interna, por lo que al pfSense se le tendrá que activar una nueva interfaz de tipo “Red interna” y le daremos el nombre de DMZ.

Virtualbox no nos permite realizar esta modificación “en caliente”, por lo que habrá que apagar la máquina virtual de pfSense. Algunos sistemas profesionales de virtualización sí que permiten hacer algunas modificaciones hardware en las máquinas virtuales sin tener que apagarlas, y dependiendo del sistema

operativo, se dará cuenta de dichos cambios y por lo tanto no será necesario realizar un apagado.

Tenemos que crear otra máquina virtual que actuará a modo de servidor en la red DMZ, por lo que en su configuración la configuraremos también como “Red interna” en DMZ.

5.2. Modificación de la configuración en pfSense

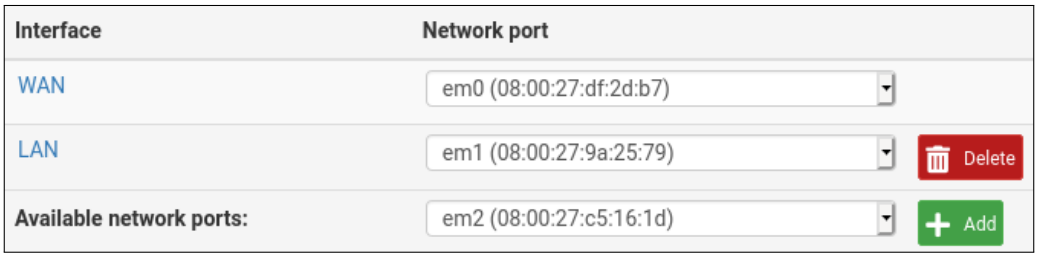
Tal como se ha visto, tras la instalación de pfSense aparece un pequeño asistente de configuración que nos lleva a través de unos pasos que nos permitirá realizar la configuración de una infraestructura basada en una red con parte WAN y LAN.

En caso de tener más interfaces de red durante la instalación, o a posteriori como sucede ahora, no se configurarán y por tanto queda en nuestra mano realizar dicha configuración.

5.2.1. Añadir y configurar interfaz

Dado que pfSense cuenta con un nuevo interfaz, por defecto aparece deshabilitado y sin configurar. Debemos activarlo, darle el direccionamiento que veamos adecuado y lo habitual también suele ser darle un nombre representativo de la red que va a servir.

Todas estas modificaciones se realizan desde el interfaz web, a través de “***Interfaces → Assignments***”:



Tal como se puede ver, aparece la nueva interfaz junto a un botón “Add” que indica que lo podemos añadir a la configuración. Una vez pulsado el botón se le asigna el nombre “OPT1” como nombre por defecto, pero la idea es cambiarlo. Para ello se hace click en el interfaz, y se entra en su configuración.

Entre las modificaciones que vamos a realizar están:

- **Enable**: hay que habilitar el interfaz, ya que aunque esté configurado, si no está habilitado no entrará en funcionamiento.
- **Description**: El nombre que le daremos al interfaz, en este caso “DMZ”.
- **IPv4 Configuration Type**: Tipo de IPv4 que se le va a asignar. Crearemos un direccionamiento estático, por ejemplo: 172.16.0.1 /25 . La configuración de la red se hace en la misma página, un poco más abajo.

Una vez realizados los cambios, habrá que aplicar los cambios.

5.2.2. Configurar DHCP Server

Aunque no suele ser habitual configurar el DHCP Server en la red DMZ, se va a explicar cómo realizar la configuración ya que en otro tipo de redes (para el WIFI, departamento de marketing, …) es útil.

Para realizar la configuración se hace en "***Services → DHCP Server***", donde aparecerán las interfaces sobre las que podemos gestionar dicho servicio, actualmente en LAN y DMZ.

La configuración que se tendrá en cuenta a la hora de modificar el DHCP en el interfaz será:

- **Enable**: Habilitar servicio
- **Range**: Si queremos limitar el DHCP para que sólo de IPs a una parte de la red
- **Other Options**: Los servidores DHCP no sólo se encargan de dar IPs, por lo que hay muchas opciones extra que se pueden modificar

5.2.2.1. Mapear IPs de DHCP a estática

Existe la posibilidad de dar a los equipos siempre la misma IP cuando hacen una petición DHCP. Los servidores DHCP suelen crear una pequeña base de datos con las IPs que han otorgado y durante cuánto tiempo las tienen reservadas para esos equipos.

Es posible realizar mapeos de IPs a equipos que sean permanentes, por lo que los equipos que pidan DHCP, si tienen dicho mapeo creado, siempre se les asignará la misma IP y esa IP estará reservada sólo para ese equipo.

Suele ser habitual el realizar estos mapeos para tener controlados los equipos y conocer sus IPs sin tener que ir a ellos a configurarlos de manera estática, ya que todo se queda centralizado en el servidor DHCP.

Para realizar estos mapeos hay que ir a "Status → DHCP Leases" donde podremos ver todas las IPs que se han otorgado:

Leases

IP address	MAC address	Client Id	Hostname	Description	Start	End	Online	Lease Type	Actions
172.16.0.100	08:00:27:be:bb:6f		debian		2021/03/22 11:49:21	2021/03/22 13:49:21	online	active	

5.2.3. Modificación de reglas de filtrado

Tras realizar los pasos previos, si se enciende la máquina virtual que actuará de servidor en la DMZ, se podrá comprobar que cogerá IP por DHCP en el direccionamiento que se haya configurado en el paso anterior. El problema es que si se intenta realizar algún intento de comunicación con el exterior de la red veremos que no será posible ya que actualmente no existe ninguna regla que lo permita en pfSense. Lo podemos comprobar yendo a "***Firewall → Rules***" y mirando el apartado "DMZ"

¡Cuidado!

¡pfSense deniega cualquier tipo de tráfico por defecto en los interfaces!

Teniendo en cuenta las reglas que normalmente suele tener una red DMZ, que se han comentado al principio de este capítulo, habría que realizar las configuraciones oportunas para permitir el tráfico tal como sea necesario.

Teniendo en cuenta las reglas que se necesitan crear, y dado que pfSense aplica las reglas “al entrar al interfaz”, habrá que entender dónde crear cada regla y tener cuidado con el orden con el que se aplican. Como pruebas, se ha aplicado las siguientes reglas:

Rules (Drag to Change Order)

	States	Protocol	Source	Port	Destination	Port	Gateway
✗	0 /420 B	IPv4 *	DMZ net	*	LAN net	*	*
✓	1 /336 B	IPv4 *	172.16.0.100	*	8.8.8.8	*	*

¿Es realmente necesaria la primera regla?

VI

Docker

1. Introducción

Hoy en día es muy habitual hacer uso de los sistemas de contenedores, siendo el más conocido Docker en el mundo del desarrollo de *software*. Este sistema trae consigo una serie de ventajas que veremos más adelante, que nos permite asegurar, entre otras cosas, que las versiones utilizadas en el entorno de producción son las mismas que durante las etapas de desarrollo.

En este documento se va a explicar cómo realizar la instalación y configuración de un sistema basado en contenedores Docker para poder arrancar servicios, y ciertas configuraciones que son necesarias conocer.

2. Sistemas de contenedores

Los sistemas de contenedores son un método de virtualización (conocido como "virtualización a nivel de sistema operativo"), en el que se permite ejecutar sobre una capa virtualizadora del núcleo del sistema operativo distintas instancias de "espacio de usuario".

Este "espacio de usuario" (donde se ejecutarán aplicaciones, servicios...) se les denomina **contenedores**, y aunque pueden ser como un servidor real, están bajo un mecanismo de aislamiento proporcionado por el *kernel* del sistema operativo, y sobre el que se pueden aplicar límites de espacio, recursos de memoria, de acceso a disco...

Información

Un contenedor es un espacio de ejecución de servicios al que se les puede aplicar límites de recursos (como la memoria, el acceso a disco...)

Desde el punto de vista del usuario, que un servicio se ejecute en una máquina virtual o en un contenedor es indistinguible. En cambio, desde el punto de vista de un administrador de sistemas o de un desarrollador, el uso de contenedores trae consigo una serie de ventajas que veremos en apartados posteriores.

2.1. Un poco de historia

Aunque está muy en boga el despliegue de aplicaciones haciendo uso de contenedores, no es un concepto nuevo, ya que lleva existiendo desde la década de los 80 en sistemas UNIX con el concepto de chroot.

Chroot, también conocido como "jaulas chroot", permitían ejecutar comandos dentro de un directorio sin que, en principio, se pudiese salir de dicha ruta. Tenía muy pocas restricciones de seguridad, pero era un primer paso al sistema de contenedores.

LXC nace en 2008 utilizando distintas funcionalidades del kernel Linux para proveer un entorno virtual donde poder ejecutar distintos procesos y tener su propio espacio de red. Con LXC nacen distintas herramientas para controlar estos contenedores, así como para crear plantillas y una **API que permite interaccionar con LXC** desde distintos lenguajes de programación.

Ha habido otras tecnologías en Linux, como OpenVz, pero nos centraremos en Docker, ya que es lo más conocido actualmente.

2.2. Qué es un contenedor y cómo se crea

Para entender qué es un contenedor dentro de la infraestructura Docker y cómo se crea, tenemos que diferenciar distintos conceptos:

- **Imagen Docker**
- **Contenedor Docker**

A continuación se van a detallar en profundidad.

2.2.1. Imágenes Docker

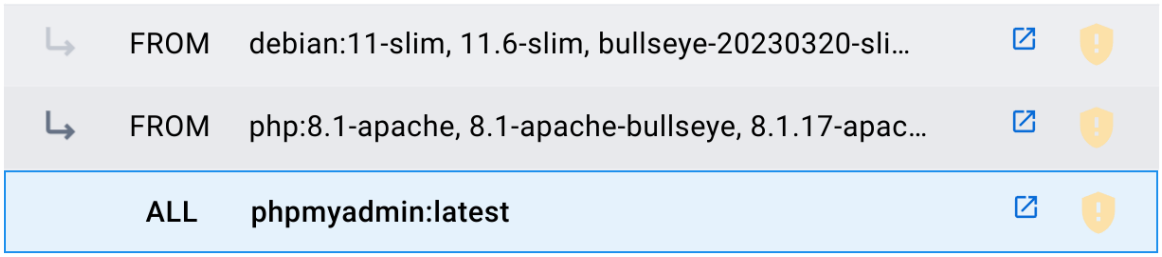
Para crear un contenedor necesitamos hacer uso de una **"imagen", que es un archivo inmutable (no modificable) que contiene el código de la aplicación que queremos ejecutar y todas sus dependencias necesarias**, para que pueda ser ejecutada de manera rápida y confiable independientemente del entorno en el que se encuentre.

Las imágenes, debido a su origen **sólo-lectura**, se pueden considerar como **"plantillas"**, que son la representación de una aplicación y el entorno necesario para ser ejecutada en un momento específico en el tiempo. **Esta consistencia es una de las grandes características de Docker**.

Información

Una imagen contiene el servicio que nos interesa ejecutar junto con sus dependencias, y son independientes del servidor donde se ejecuta.

Una imagen puede ser creada utilizando otras imágenes como base. Por ejemplo, la imagen de PHPMyAdmin empaqueta la aplicación PHPMyAdmin sobre la imagen **PHP** (versión 8.1-apache), que a su vez hace uso de la imagen **Debian** (versión 11-slim).



Jerarquía de imágenes usadas por PHPMyAdmin. Fuente.

A las imágenes creadas se les suele añadir etiquetas (**tags**) para diferenciar versiones o características internas. Cada creador determina las etiquetas que le interesa crear. Por ejemplo:

- **latest**: Se le denomina a la última imagen creada.
- php:**8.1-apache**: Indica que en esta imagen PHP la versión es la 8.1 y además cuenta con Apache.

Podemos utilizar imágenes públicas descargadas a través de un ***registry*** público, que no es otra cosa que un repositorio de imágenes subidas por la comunidad. El *registry* principal más utilizado es Docker Hub.

Información

Las imágenes Docker pueden ser públicas o privadas y se almacenan en un repositorio llamado *registry*, siendo el más conocido Docker Hub

Se pueden crear nuestras propias imágenes privadas, que pueden ser almacenadas en nuestros equipos o a través de un **registry privado** que podemos crear (también existen servicios de pago).

2.2.2. Contenedores Docker

Un contenedor Docker es un **entorno de tiempo de ejecución virtualizado donde los usuarios pueden aislar aplicaciones**. Estos contenedores son unidades compactas y portátiles a las que se les puede aplicar un sistema de limitación de recursos.

Información

Un contenedor se crea a través de una imagen, es la versión ejecutable de la misma que se crea en un entorno virtualizado

Un contenedor se crea a través de una imagen y es la versión ejecutable de la misma. Lo que se hace es crear una capa de escritura sobre la imagen inmutable, donde se podrán escribir datos. Se pueden crear un número ilimitados de contenedores haciendo uso de la misma imagen base.

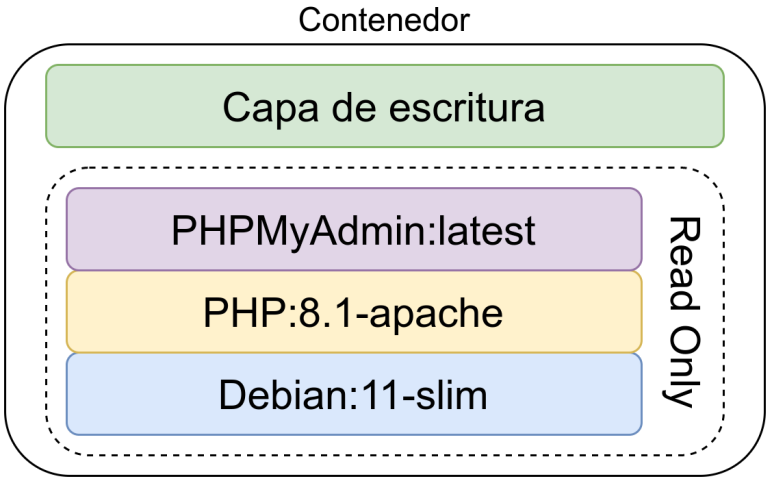


Imagen “interna” de un contenedor

La capa de escritura no es persistente y se pierde al eliminar el contenedor, es decir, **los datos de un contenedor se eliminan al borrar el contendor**. Para evitar este comportamiento se puede hacer uso de un volumen persistente de datos, de esta manera esos datos no se pierden.

¡Cuidado!

Los datos creados dentro de un contenedor se borran al eliminar el contenedor

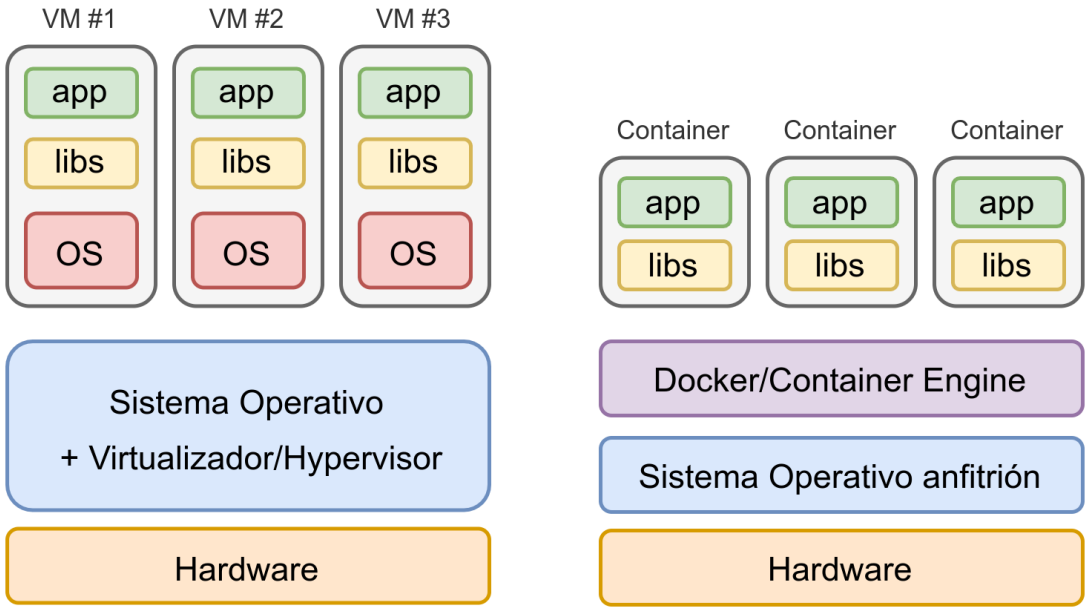
2.3. Contenedores vs. Máquinas virtuales

El uso de máquinas virtuales está muy extendido gracias a que cada vez es más sencillo crearlas. Esto no quiere decir que siempre sea la mejor opción, por lo que se va a realizar una comparativa teniendo en cuenta distintos aspectos a la hora de realizar un desarrollo con máquinas virtuales y con sistemas de contenedores.

2.3.1. Infraestructura

La creación de máquinas virtuales nos permite crear entornos aislados en los que poder instalar el Sistema Operativo que más nos interese y con ello poder instalar el software y los servicios que necesitemos.

Las máquinas virtuales se virtualizan a nivel de hardware, donde debe existir un Sistema Operativo con Hypervisor que permita dicha virtualización. Por otro lado, los contenedores se virtualizan en la capa de aplicación, haciendo que este sistema sea mucho más ligero, permitiendo utilizar esos recursos en los servicios que necesitamos hacer funcionar dentro de los contenedores.



Infraestructura Máquinas Virtuales vs Docker

En la imagen se puede apreciar una comparativa diferenciando cómo quedaría una infraestructura de 3 aplicaciones levantadas en distintas máquinas virtuales o en distintos contenedores.

Tal como se puede ver en la imagen, **al tener cada servicio en una máquina virtual separada**, se va a tener que virtualizar todo el Sistema Operativo en el que se encuentre, con el consiguiente **coste de recursos (memoria RAM y disco duro) y con el coste en tiempo de tener que realizar la configuración y securización del mismo**.

Información

Usando contenedores la infraestructura se simplifica notáblemente

Por otro lado, en un sistema de contenedores, cada contenedor es un servicio aislado, en el que sólo tendremos que preocuparnos (en principio) de configurar sus parámetros.

2.3.2. Ventajas durante el desarrollo

A la hora de desarrollar una aplicación es habitual hacer pruebas utilizando distintas versiones de librerías, *frameworks* o versiones de un mismo lenguaje de programación. De esta manera, podremos ver si nuestra aplicación es compatible.

Cuando se hace uso de una máquina virtual dependemos de las versiones que tiene nuestra distribución y es posible que no podamos instalar nuevas versiones u otras versiones en paralelo.

Por ejemplo, la última versión de PHP actualmente es la 8.4.11 y de Apache la 2.4.65:

- En **Debian 12** sólo se puede instalar PHP 8.2.29 y Apache 2.4.62.
- En **Ubuntu 24.04** la versión de PHP es la 8.3.6 y la de Apache la 2.4.58.

Con Docker, podremos levantar contenedores con distintas versiones del servicio que nos interese en paralelo para comprobar si nuestra aplicación/servicio es compatible.

Información

Con Docker es posible levantar servicios con distintas versiones en paralelo

Por otro lado, si un desarrollador quiere utilizar un sistema operativo distinto, no se tendrá que preocupar de si su distribución tiene las mismas versiones. O en el caso de usar Windows/Mac, no tener que estar realizando instalaciones de las versiones concretas.

2.3.3. Ventajas durante la puesta en producción

Ligado al apartado anterior, durante la puesta en producción es obligatorio hacer uso de las mismas versiones utilizadas durante el desarrollo para asegurar la compatibilidad.

¡Cuidado!

Para asegurar la compatibilidad en producción, siempre se debe usar la misma versión de los servicios que en desarrollo

Si tenemos un servidor que no está actualizado, o en el mismo servidor tenemos distintas aplicaciones que requieren utilizar distintas versiones de software, en un entorno de máquinas virtuales se hace muy complejo, ya que lo habitual será tener que instalar nuevas máquinas virtuales.

¡Atención!

No siempre es posible tener distintas versiones del mismo software en un mismo servidor

En un entorno con contenedores, al igual que se ha comentado antes, esto no es problema.

2.3.4. Rapidez en el despliegue

Ligado a todo lo anterior, realizar el despliegue de un entorno de desarrollo/producción es más rápido utilizando contenedores, sin importar el sistema operativo en el que nos encontremos.

Información

El despliegue con contenedores es más rápido.

Más adelante se verá cómo realizar el despliegue de distintos servicios haciendo uso de un único comando.

3. Docker

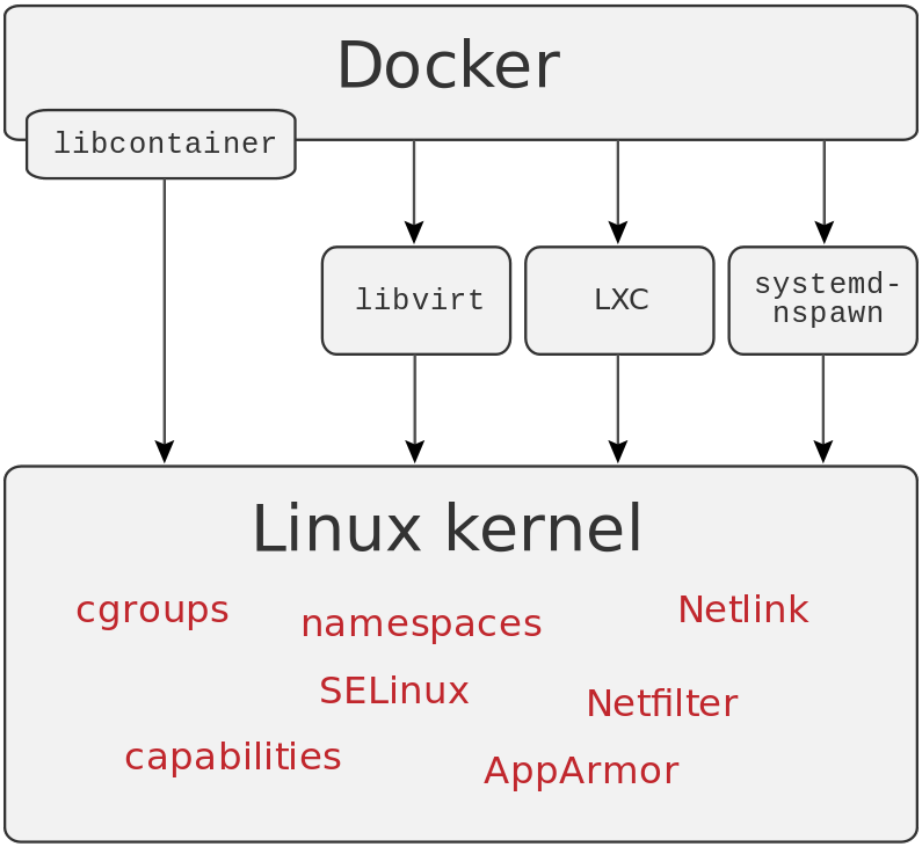
Docker es un proyecto de Software Libre nacido en 2013 que permite realizar el despliegue de aplicaciones y servicios a través de contenedores de manera rápida y sencilla, tal como veremos más adelante.

Estos contenedores proporcionan una capa de abstracción y permiten aislar las aplicaciones del resto del sistema operativo a través del uso de ciertas características del kernel Linux.

Dentro del contenedor, se puede destacar el aislamiento a nivel:

- Árbol de procesos
- Sistemas de ficheros montados
- ID de usuario
- Aislamiento de recursos (CPU, memoria, bloques de E/S...)
- Red aislada

Al igual que sucede con otro tipo de *software*, para que Docker haga uso de todas estas características, está construido haciendo uso de otras aplicaciones y servicios.



Tecnologías usadas por Docker. Fuente: Wikipedia

En el 2015 la empresa Docker creó la ***Open Container Initiative***, proyecto actualemente bajo la Linux Foundation, con la intención de diseñar un estándar abierto para la virtualización a nivel de sistema operativo.

3.1. Instalación

Dependiendo del sistema operativo en el que nos encontremos, Docker tiene la opción de instalarse de distintas maneras. En sistemas operativos GNU/Linux cada distribución tiene un paquete para poder realizar la instalación del mismo.

Instalación de Docker en Ubuntu

```
ruben@vega:~$ sudo apt install docker.io
```

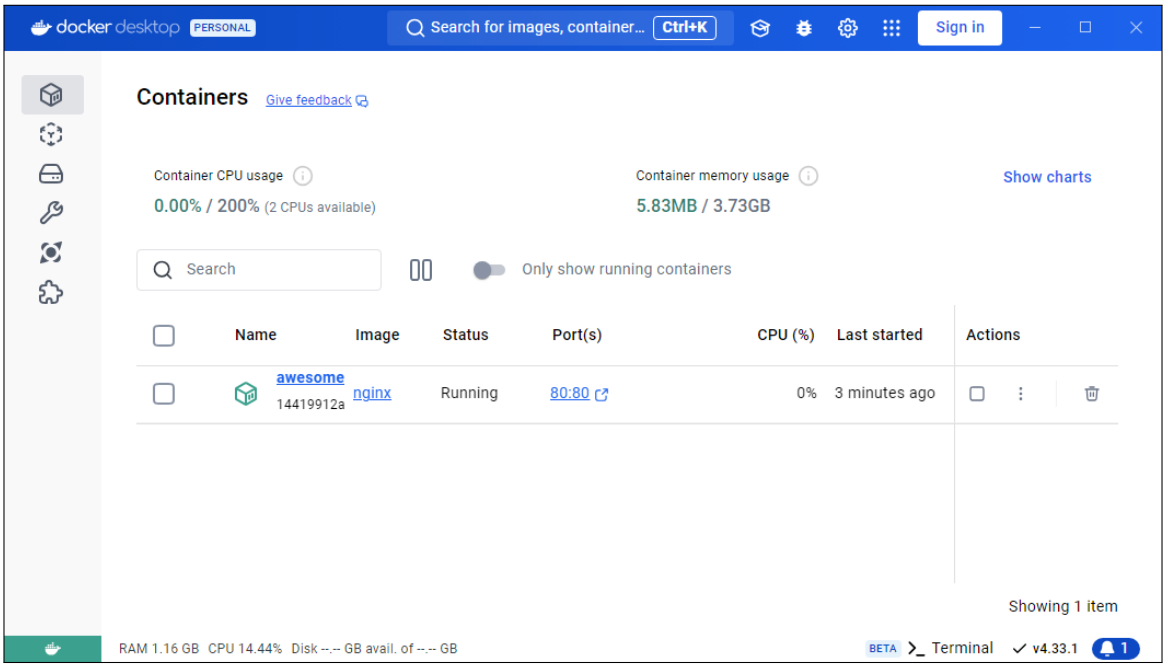
¡Cuidado!

El nombre del paquete en Ubuntu y Debian es “docker.io”.

3.1.1. Instalación en Windows y/o Mac

En sistemas Windows y MacOS existe la opción de instalar Docker Desktop, una versión que utiliza una máquina virtual para simplificar la instalación en estos sistemas. De todas maneras, también se instala el

CLI para poder usar los comandos que veremos a continuación.



Docker Desktop

En caso de Windows, se requiere tener las extensiones de virtualización habilitadas en la BIOS/UEFI y una de estas dos opciones, que habrá que configurar antes de instalar Docker Desktop:

- Usar WSL2.
- Usar Hyper-V y el sistema de contenedores de Windows.

3.2. Configuración

Tras realizar la instalación veremos cómo el servicio Docker ha levantado un interfaz nuevo en nuestra máquina, cuya IP es **172.17.0.1/16**, siendo el direccionamiento por defecto.

Nueva IP en el equipo

```
ruben@vega:~$ ip a
...
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    link/ether 02:42:9c:1f:e2:90 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
      valid_lft forever preferred_lft forever
```

Esta IP hará de **puente** (similar a lo que sucede con las máquinas virtuales) cuando levantemos contenedores nuevos. Los contenedores estarán dentro de ese direccionamiento 172.17.0.0/16, por lo tanto, aislados de la red principal del equipo.

Información

Los contenedores que levantemos estarán en la red 172.17.0.0/16

3.3. Usar Docker con usuario sin privilegios

Para poder hacer uso de Docker con un usuario sin permisos de root/administrador, se debe añadir a los usuarios no privilegiados dentro de un grupo. Dependiendo de dónde usemos Docker, tendremos que realizarlo de una manera u otra.

3.3.1. Linux

En este caso, el grupo que debe tener el usuario es "**docker**", que se lo podemos añadir al usuario de distintas maneras:

- Editar el fichero `/etc/group`, y añadir el usuario al grupo
- Ejecutar estos comandos que ponemos a continuación:

Añadir el grupo docker al usuario correspondiente

```
ruben@vega:~$ sudo addgroup ruben docker
[sudo] password for ruben:
Adding user `ruben' to group `docker' ...
Adding user ruben to group docker
Done.
ruben@vega:~$ newgrp docker
```

A partir de ahora ya se puede hacer uso de Docker con el usuario al que hayamos añadido al grupo.

3.3.2. Windows

Para que un usuario en Windows pueda usar Docker Desktop tiene que pertenecer al grupo "**docker-users**". Para añadirlo, desde un PowerShell **con permisos de administrador**, ejecutaremos:

Añadir al usuario "usuario" al grupo docker-users

```
PS C:\Users\ruben> net localgroup "docker-users" "usuario" /add
```

3.4. Primeros pasos

El comando `docker` tiene muchas opciones, por lo que es recomendable ejecutarlo sin parámetros. De esta manera se pueden ver todas las opciones y una ayuda simplificada para cada una de ellas.

Algunas de las opciones del comando docker

```
ruben@vega:~$ docker
Usage:  docker [OPTIONS] COMMAND

Management Commands:
builder     Manage builds
completion  Generate the autocompletion script for the specified shell
config      Manage Docker configs
container   Manage containers
context     Manage contexts
image       Manage images
manifest    Manage Docker image manifests and manifest lists
network     Manage networks
node        Manage Swarm nodes
plugin      Manage plugins
secret      Manage Docker secrets
service     Manage services
stack       Manage Docker stacks
swarm       Manage Swarm
system      Manage Docker
trust       Manage trust on Docker images
volume      Manage volumes

Commands:
...
```

Para cada una de estas opciones, se le puede añadir el parámetro `--help` para mostrar la ayuda. Hay un segundo apartado que se ha cortado, en el que se incluyen más comandos.

Para asegurar que el servicio Docker está funcionando, podemos hacer uso de `docker info`, que nos mostrará mucha información acerca del servicio. Pero si lo que queremos es comprobar si tenemos algún contenedor corriendo, es más sencillo hacer `docker ps` (que es la versión simplificada de `docker container ls`):

Comprobar estado de Docker y contenedores levantados

```
ruben@vega:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

ruben@vega:~$ docker container ls
```

```
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

En este caso, como no hay ningún contenedor levantado, sólo muestra las cabeceras de las columnas del listado.

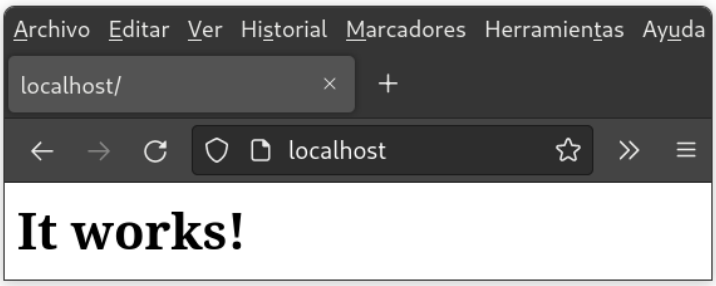
3.5. Levantando nuestro primer contenedor

Es momento de crear nuestro primer contenedor. Para ello, dado que se está usando la consola, hay que hacer uso del comando `>_ docker` con una serie de parámetros. En este caso se ha optado por levantar el servicio **Apache HTTPD**:

>_ Levantando el primer contenedor

```
ruben@vega:~$ docker run -p 80:80 httpd
AH00558: httpd: Could not reliably determine the server's ...
AH00558: httpd: Could not reliably determine the server's ...
[Fri Mar 24 18:25:14.194246 2023] [mpm_event:notice] ...
[Fri Mar 24 18:25:14.194347 2023] [core:notice] [pid  ...
172.17.0.1 - - [24/Mar/2023:18:25:41 +0000] "GET / HTTP/1.1" 304 -
```

Vemos los logs del servicio Apache al arrancar y si vamos al navegador a la dirección http://localhost muestra lo siguiente:



Y para entender lo que hace el comando, los parámetros son:

- **docker**: Cliente de consola para hacer uso de Docker.
- **run**: Ejecuta un comando en un nuevo contenedor (y si no existe lo crea).
- **-p 80:80**: Publica en el puerto 80 del servidor el puerto 80 utilizado en el contenedor. Se puede pensar que es como hacer un **port-forward** en un firewall.
- **httpd**: Es la **imagen** del contenedor que se va a arrancar. En este caso, la imagen del servidor Apache HTTPD.

Y si vemos qué muestra el estado de docker, vemos cómo aparece el contenedor levantado.

Comprobar estado de Docker y contenedores levantados

```
ruben@vega:~$ docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS                NAMES
a1c3362b0d6c   httpd     "httpd-..."   3 seconds ago    Up 2 seconds    0.0.0.0:80->80/tcp   great
```

En la columna PORTS se puede apreciar cómo aparece que se ha levantado el puerto **`0.0.0.0:80`** (escucha en el puerto 80 para cualquier IP del sistema operativo) que es una redirección al puerto **`80/TCP`** interno del contenedor.

3.6. Contenedores en *background* y más opciones

Tal como se puede ver en el ejemplo anterior, el contenedor se queda en primer plano, viendo los logs del Apache. Esto para ver qué es lo que está sucendiendo durante el desarrollo puede ser útil, pero lo ideal es que el contenedor arranque en modo ***background***, y cuando necesitemos vayamos a ver los logs.

A continuación se va a arrancar un nuevo contenedor de Apache con nuevos parámetros:

Crear un contenedor Web en el puerto 8080

```
ruben@vega:~$ docker run --name mi-apache -d -p 8080:80 httpd
```

Los nuevos parámetros son:

- **`--name mi-apache`**: De esta manera se le da un nombre al contenedor, para poder identificarlo de manera rápida entre todos los contenedores creados.
- **`-d`**: Este parámetro es para hacer el ***detach*** del comando, y de esta manera mandar a ***background*** la ejecución del contenedor.
- **`-p 8080:80`**: Publica en el puerto 8080 del servidor el puerto 80 utilizado en el contenedor. Se puede pensar que es como hacer un **port-forward** en un firewall.

3.7. Parar, arrancar y borrar contenedores

Hasta ahora hemos aprendido a crear contenedores, pero en ciertos momentos nos puede interesar parar un contenedor que no estemos utilizando, o una vez haya cumplido su función, borrarlo.

3.7.1. Parar contenedores

Para parar un contenedor, debemos conocer el nombre del mismo o su ID (que es único). Estos datos los podemos conocer a través del comando `docker ps`.

Con esto, podemos ejecutar:

Parar un contenedor

```
ruben@vega:~$ docker stop mi-apache
```

3.7.2. Arrancar un contenedor parado

Una vez parado un contenedor, o al reiniciar el servidor, si queremos arrancar un contenedor parado, debemos conocer también su ID o nombre.

Para visualizar todos los contenedors (tanto los arrancados como los parados), lo podemos hacer a través del comando `>_ docker ps -a`.

Gracias a ese listado, podemos volver a arrancar un contenedor que esté parado con `>_ docker start mi-apache`, siendo "mi-apache" el contenedor que queremos arrancar.

3.7.3. Borrar un contenedor

Si queremos borrar un contenedor, éste debe estar parado, ya que Docker no nos va a dejar borrar un contenedor que está en ejecución.

Es interesante borrar contenedores que hayamos creado de pruebas o contenedores que ya no se vayan a utilizar más, para de esta manera liberar recursos.

Para borrarlo, similar a los casos anteriores, se hará con `>_ docker rm mi-apache`.

4. Variables de entorno

Algunos contenedores tienen la opción de recibir variables de entorno al ser creados. Estas variables pueden afectar al comportamiento del contenedor, o para ser inicializado de alguna manera distinta a las opciones por defecto.

El creador de la imagen Docker puede crear las variables de entorno que necesite para después utilizarlas en su aplicación. A modo de ejemplo, se va a utilizar la imagen de la aplicación PHPMyAdmin.

A continuación se van a crear 2 contenedores de PHPMyAdmin, diferenciados por el puerto, el nombre, y la variable de entorno **PMA_ARBITRARY**:

- El primer contenedor va a estar en el puerto 8081, se le va a dar el nombre "myadmin-1" y no va a tener la variable de entorno incializada.
- El segundo contenedor usará el puerto 8082, llamado "myadmin-2" y la variable **PMA_ARBITRARY** inicializada a "1", tal como aparece en la documentación de la imagen de PHPMyAdmin.

Para ello, se han ejecutado los siguientes comandos:

>_ Creación de dos contenedores PHPMyAdmin

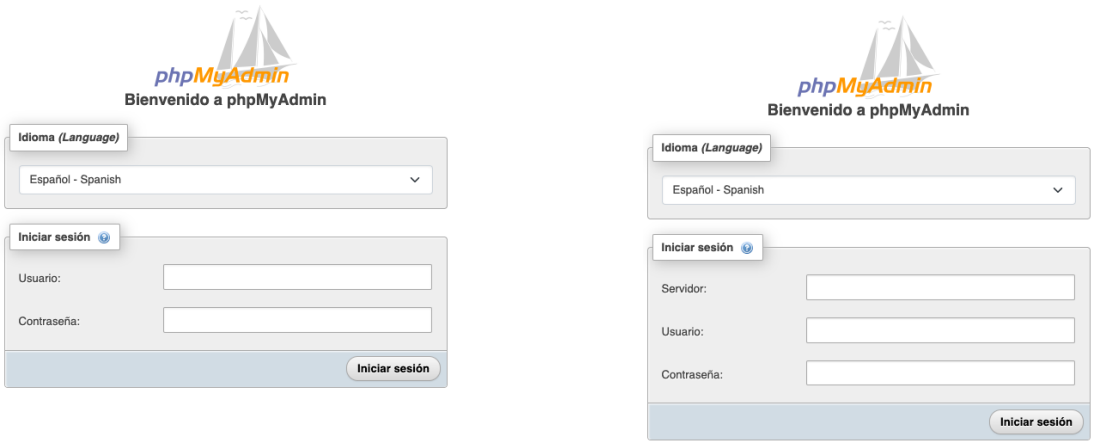
```
ruben@vega:~$ docker run --name myadmin-1 -d -p 8081:80 phpmyadmin

ruben@vega:~$ docker run --name myadmin-2 -e PMA_ARBITRARY=1 -d -p 8082:80 phpmyadmin
```

Tal como se puede ver, al segundo contenedor se le ha pasado un nuevo parámetro **-e**, que significa que lo que viene a continuación es una variable de entorno (en inglés **environment**). En este caso, la variable de

entorno es **PMA_ARBITRARY** que se ha inicializado a **1**.

Si ahora en nuestro navegador web apuntamos al puerto 8081 y al puerto 8082 de la IP de nuestro servidor, veremos cómo existe una ligera diferencia en el formulario que nos muestra la web.

En el formulario del puerto 8081 (donde no hemos inicializado la variable) sólo podemos indicar el usuario y la contraseña. Por el contrario, en el formulario del puerto 8082, al inicializar la variable **PMA_ARBITRARY**, y tal como nos dice la documentación de la imagen, nos permite indicar la IP del servidor MySQL al que nos queremos conectar.



A la izquierda formulario del puerto 8081, sin variable inicializada. A la derecha, puerto 8082 con variable inicializada.

Dado que una variable puede afectar al comportamiento (o la creación) del servicio que levantemos a través de un contenedor, es importante leer la documentación e identificar las variables que tiene por si nos son de utilidad.

Información

Es recomendable leer la documentación de las imágenes Docker para identificar las posibles variables de entorno que existen y ver si nos son útiles.

5. Volumen persistente de datos

Hasta ahora hemos levantado un contenedor a través de una imagen que levanta el servicio Apache, mostrando su página por defecto. Podríamos escribir en el contenedor la página HTML que nos interesase, pero hay que entender que **los datos de un contenedor desaparecen cuando el contenedor se elimina**.

Para que los cambios realizados dentro de un contenedor se mantengan, tenemos que hacer uso de los denominados **volúmenes de datos**. Esto no es más que **hacer un montaje de una ruta del disco duro del sistema operativo dentro de una ruta del contenedor**.

Estos volúmenes que le asignamos al contenedor pueden ser de dos tipos:

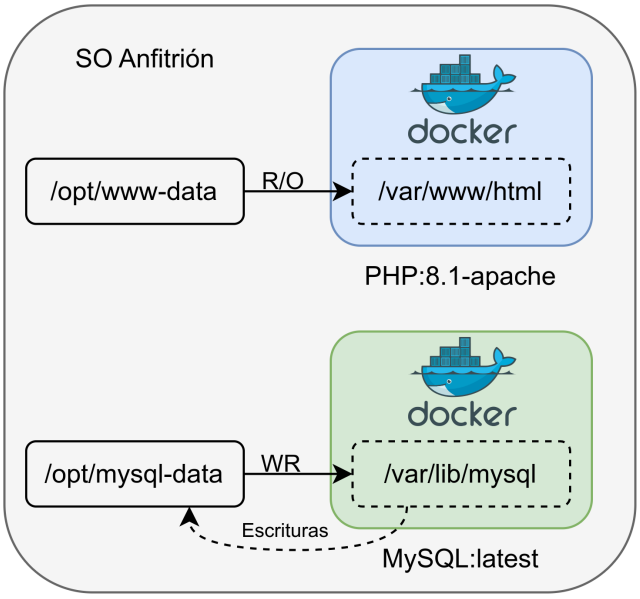
- **Sólo lectura**: Nos puede interesar asignar un volumen de sólo lectura cuando le pasamos ficheros de configuración o la propia web que queremos visualizar.

- **Lectura-Escritura**: En este caso se podrá escribir en el volumen. Por ejemplo, el directorio donde una base de datos guarda la información o una web donde deja imágenes subidas por usuarios.

De esta manera, tendremos que asignar el número de volúmenes necesarios a cada contenedor dependiendo de la imagen utilizada, el servicio que se levanta y lo que queremos hacer con los datos que le asignemos o generemos en el contenedor.

En la siguiente imagen se puede ver una infraestructura con dos contenedores y dos volúmenes:

- **Contenedor Web**: Se le asigna un volumen en modo **sólo lectura** cuya ruta original está en `/opt/www-data`, que dentro del contenedor está en `/var/www/html`.
- **Contenedor MySQL**: Dado que los datos de la base de datos deben ser guardados, en este caso se le asigna un volumen que permite escritura. Por lo tanto, lo que se crea dentro del contenedor en `/var/lib/mysql` realmente se estará guardando en el sistema operativo anfitrión en `/opt/mysql-data`.



Ejemplo de dos volúmenes asignados a distintos contenedores

5.1. Añadir volumen de escritura al crear un contenedor

Al añadir un volumen cuando creamos un contenedor hace que por defecto sea en modo lectura-escritura. Cualquier escritura realizada dentro del contenedor en la ruta especificada va a resultar en que el fichero se creará en la ruta indicada del sistema operativo anfitrión.

Añadir volumen de escritura al crear un contenedor

```
ruben@vega:~$ ls /opt/mysql-data
ruben@vega:~$ docker run -d -p 3306:3306 --name mi-db \
    -v /opt/mysql-data:/var/lib/mysql \
    -e MYSQL_ROOT_PASSWORD=my-secret-pw \
    mysql:latest
```

```
ruben@vega:~$ ls /opt/mysql-data
auto.cnf      client-key.pem      '#innodb_temp'      server-cert.pem   ...
```

Para este ejemplo se ha creado un contenedor usando la imagen de MySQL, al que se le ha asignado un volumen (**por defecto se asigna permitiendo la escritura**) y un parámetro necesario para realizar la posterior conexión con contraseña.

- **-v /opt/mysql-data:/var/lib/mysql**: A través del parámetro **-v** se le indica al contenedor que se le va a pasar un volumen. Posteriormente se le indica la ruta del sistema operativo anfitrión `/opt/mysql-data` que se montará dentro del contenedor en `/var/lib/mysql`.
- **-e MYSQL_ROOT_PASSWORD=my-secret-pw**: El parámetro "**-e**" sirve para pasarle al contenedor **variables de entorno**. En este caso, y tal como dice la web de la imagen MySQL, esta es la manera de asignar la contraseña del usuario **root** durante la inicialización de la base de datos.

Tras crear el contenedor, y asegurarnos que está levantado haciendo uso del comando `docker ps`, podemos realizar la conexión desde el sistema operativo anfitrión o desde cualquier otro lugar usando la contraseña indicada previamente.

Comprobar estado de MySQL

```
ruben@vega:~$ mysql -h127.0.0.1 -uroot -P3306 -p
    Enter password:
    ...
    MySQL [(none)]> show databases;
    +--------------------+
    | Database           |
    +--------------------+
    | information_schema |
    | mysql              |
    | performance_schema |
    | sys                |
    +--------------------+
    4 rows in set (0,007 sec)
```

5.2. Añadir volumen en modo sólo-lectura

A continuación se van a explicar los pasos para levantar un contenedor que contiene una web simple creada en PHP, que está alojada en la ruta `/opt/www-data` del sistema operativo anfitrión.

Añadir volumen sólo lectura al crear un contenedor

```
ruben@vega:~$ ls /opt/www-data
index.php
ruben@vega:~$ docker run -d -p 80:80 --name mi-web \
```

```
    -v /opt/www-data:/var/www/html:ro \
    php:8.2.4-apache
```

El parámetro nuevo asignado en la creación de este contenedor es:

- **`-v /opt/www-data:/var/www/html:ro`**: Para indicarle que le vamos a asignar un volumen siendo la ruta real en el sistema de ficheros del sistema operativo anfitrión `/opt/www-data` y la ruta destino **dentro del contenedor** y que va a ser en modo **read-only** `/var/www/html`.

Más adelante, cuando veamos cómo entrar dentro de un contenedor Docker, se podría usar el comando para ir a la ruta dentro del contenedor y comprobar que efectivamente está en modo sólo-lectura.

5.3. Entrar dentro de un contenedor Docker

Normalmente no suele ser necesario entrar dentro de un contenedor, ya que, tal como se ha dicho antes, cualquier modificación realizada dentro de él se perderá (salvo que sea dentro de un volumen persistente).

Aún así, para realizar pruebas o comprobaciones del correcto funcionamiento de una imagen puede ser interesante entrar dentro de un contenedor. Para ello, el comando a ejecutar es el siguiente:

Acceder a un contenedor

```
ruben@vega:~$ docker exec -it mi-db /bin/bash
```

Los parámetros utilizados son:

- **`exec`**: Indicamos que queremos ejecutar un comando dentro de un contenedor que está corriendo.
- **`-it`**: Son dos parámetros unidos, que sirven para mantener la entrada abierta (modo interactivo) y crear una TTY (consola)
- **`mi-db`**: Es el nombre del contenedor al que se quiere entrar. También se puede indicar el **ID** del contenedor.
- **`/bin/bash`**: el comando que queremos ejecutar. En este caso, una shell **bash**. En algunos casos esta shell no está instalada y debemos usar **`/bin/sh`**

Hay que tener en cuenta que dentro de un contenedor está el mínimo software posible para que la aplicación/servicio funcione, por lo que habrá muchos comandos que no existan.

6. Otros comandos útiles

Para obtener toda la información de un contenedor, incluido su estado, volúmenes utilizados, puertos, …

Obtener toda la información de un contenedor

```
ruben@vega:~$ docker inspect mi-db
```

Listar las imágenes descargadas en local. Al tener las imágenes en local, no hará falta volver a descargarlas, por lo que crear un nuevo contenedor que haga uso de una de ellas será mucho más rápido.

Listado de imágenes en local

```
ruben@vega:~$ docker image ls
REPOSITORY       TAG             IMAGE ID       CREATED         SIZE
php              8.2.4-apache    de23bf333100   3 days ago      460MB
httpd            latest          192d41583429   3 days ago      145MB
mysql            latest          483a8bc460a9   3 days ago      530MB
```

Borrar una de las imágenes que no se esté utilizando en ningún contenedor.

Borrar una imagen concreta

```
ruben@vega:~$ docker image rm httpd
```

Cuando un contenedor está en modo ***detached*** no aparecen los logs, por lo que para poder visualizarlos tenemos un comando especial para ello.

Ver los logs de un contenedor

```
ruben@vega:~$ docker logs mi-web -f
172.17.0.1 - - [26/Mar/2023:18:05:29 +0000] "GET / HTTP/1.1" 200 248
172.17.0.1 - - [26/Mar/2023:18:05:29 +0000] "GET / HTTP/1.1" 200 248
172.17.0.1 - - [26/Mar/2023:18:05:29 +0000] "GET / HTTP/1.1" 200 248
```

Listar los volúmenes existentes en el sistema. El listado muestra los que se están utilizando en contenedores (activos o parados) o los que se han utilizado en contenedores que ya no existen.

Listar volúmenes

```
ruben@vega:~$ docker volume ls
DRIVER    VOLUME NAME
local     0d6c400a6407f5cdea81a2f0158222fdd87d7f3b3e2b5969ca466d743fc71f5c
local     1d2f52018e17af0689e070a55337154c1dd68517c54435ecc24d597f7509d43c
local     6b72797227ef4708ca23ee1dfcb4b651b42eeacefd4166b898407ad4aadda10c
```

Si queremos realizar una limpieza de todos los recursos (contenedores, imágenes, volúmenes) que no se estén utilizando, se puede utilizar el siguiente comando.

Borrar recursos que no estén activos

```
ruben@vega:~$ docker system prune -a
WARNING! This will remove:
- all stopped containers
```

```
- all networks not used by at least one container
- all images without at least one container associated to them
- all build cache

Are you sure you want to continue? [y/N] y
```

¡Cuidado!

El comando anterior hace que se borren contenedores parados

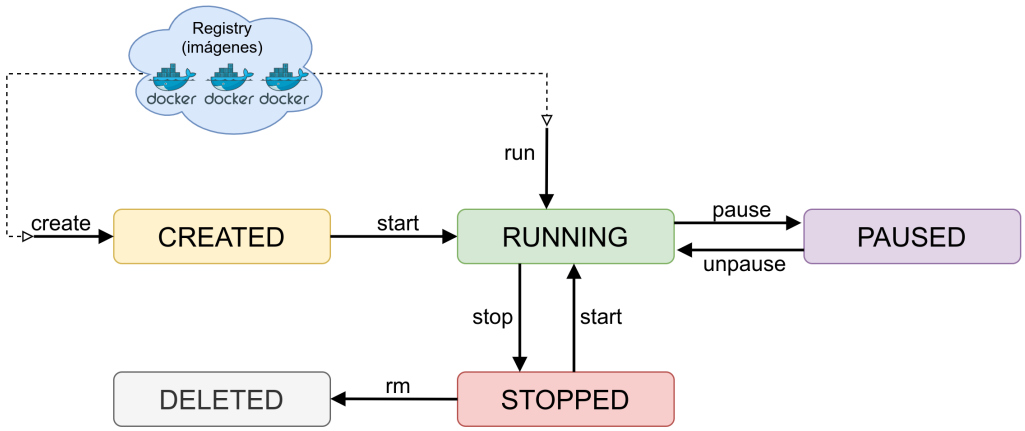
Para conocer las estadísticas de uso de cada contenedor.

Ver las estadísticas de los contenedores

```
ruben@vega:~$ docker stats
CONTAINER ID   NAME    CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O   PIDS
2fcf97530766   mi-web  0.00%   54.62MiB / 15.47GiB   0.34%   9.44MB / 172kB    0B / 0B     6
413d6e9f590f   mi-db   0.55%   357.9MiB / 15.47GiB   2.26%   319kB / 280B      0B / 0B     38
```

7. Ciclo de vida de un contenedor Docker

Un contenedor tiene un ciclo de vida que puede pasar por distintos estados. Para pasar entre estados se debe realizar a través de distintos comandos de Docker.



Estados de un contenedor

En la imagen se representa los estados más básicos junto con los comandos para pasar entre ellos.

VII

Anexos

1. Virtualbox y adaptadores de red

1.1. Introducción

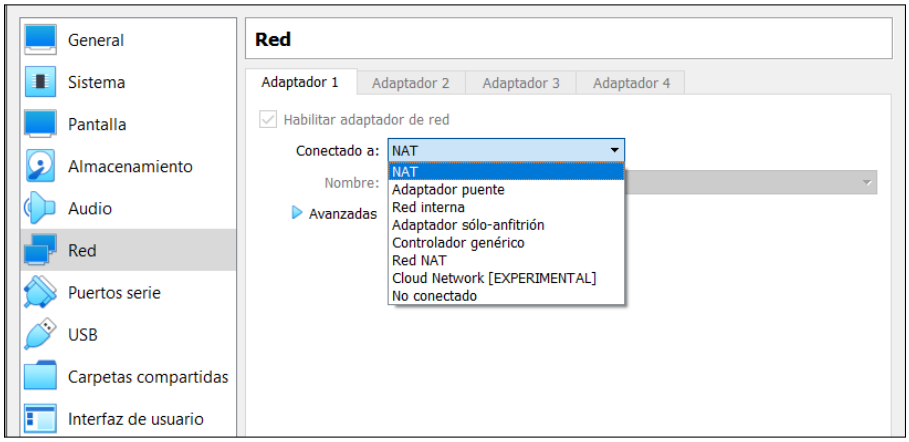
Virtualbox es una herramienta de virtualización para crear máquinas virtuales de manera sencilla. Es multiplataforma por lo que se puede utilizar en Windows, MacOS y Linux y aparte, es Software Libre.

Este documento no va a entrar en detalle en cómo se crean las máquinas virtuales, sino que va a explicar los distintos modos y adaptadores de red que puede tener una máquina virtual en este sistema de virtualización.

1.2. Adaptadores de red

Virtualbox permite que cada máquina virtual cuente con hasta cuatro adaptadores de red, lo que comúnmente se llaman interfaces o NIC (network interface controller).

Al crear las máquinas virtuales sólo tienen un único adaptador activo y suele estar configurado en modo NAT, pero tal como se ve a continuación, en el desplegable se puede ver que existen otras opciones:



En la documentación oficial aparece la explicación de los distintos modos, y es buena práctica leer y entender la documentación del software que utilizamos. También hay que entender que cada tipo de adaptador contará con una serie de ventajas y una serie de limitaciones que aparecen reflejadas en la documentación. Estos modos son comunes a otros sistemas de virtualización (VmWare, Proxmox, …), pero el nombre o el modo de uso puede variar así como las posibles limitaciones que puedan existir.

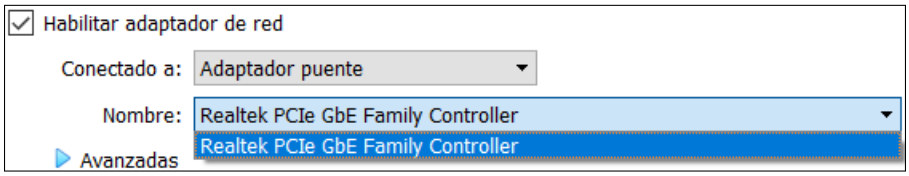
A continuación se va a dar una pequeña introducción a cada tipo de adaptador.

1.2.1. Adaptador puente

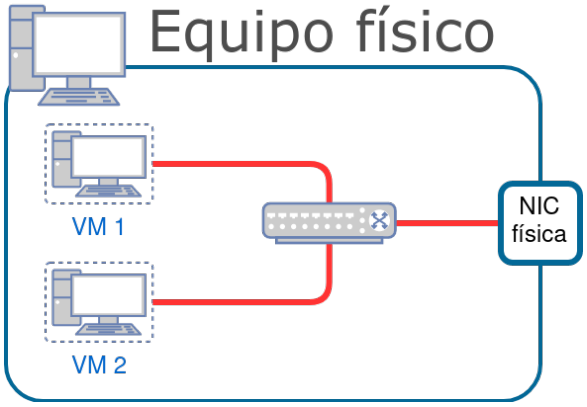
Es el tipo de adaptador que se usará si queremos que las máquinas virtuales aparezcan en la red física como si fueran un equipo más. Para poder entenderlo de mejor manera, podríamos pensar que este tipo de adaptador lo que hace es crear un “switch virtual” entre las máquinas virtuales y el interfaz físico, por lo que es como si fueran un equipo más en la red física.

Si el equipo físico anfitrión cuenta con más de un NIC (por ejemplo, en un portátil el NIC por cable y el NIC

wifi) tendremos que elegir en la máquina virtual sobre qué NIC queremos hacer el puente. En la siguiente imagen en el desplegable sólo se puede seleccionar un interfaz porque el equipo sólo cuenta con un NIC físico.



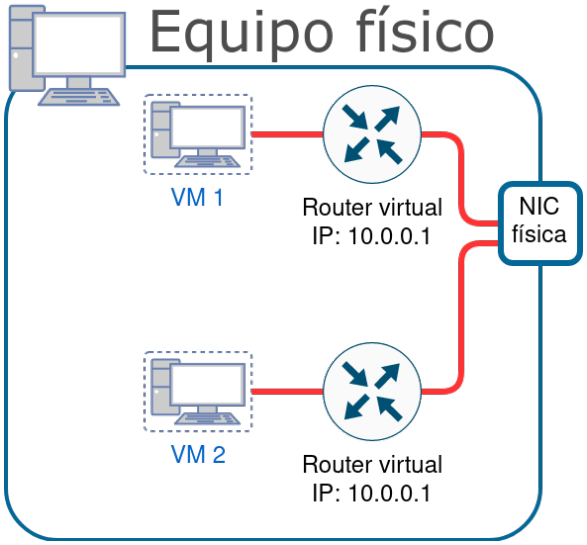
Es el método utilizado cuando virtualizamos servidores, ya que podrán dar sus servicios a toda la red.



Red como Adaptador Puente

1.2.2. NAT

Cada máquina virtual contará con su propio “router virtual” que hará NAT, y por eso todas las máquinas virtuales que usen este modo suelen tener la misma IP, pero no pertenecen a la misma red. Por defecto no se puede realizar conexión desde la red física al equipo virtualizado.



Red en modo NAT

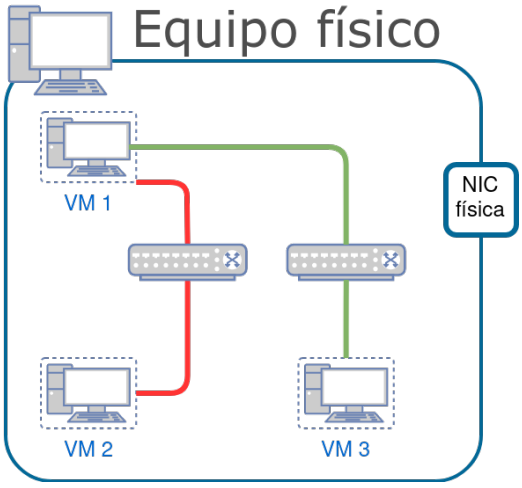
1.2.3. Red interna

Este tipo de adaptador lo que hace es “crear” un “switch virtual” que unirá las distintas máquinas virtuales que estén conectadas al nombre de esa red interna.

En el siguiente ejemplo la VM1 tiene 2 NICs, cada una con una red interna distinta. La VM2 tiene un NIC conectado a una de las redes internas creadas previamente y VM3 está conectada a la otra red interna.

Virtualbox no se encarga de dar IPs en estas redes, por lo que deberemos configurar cada interfaz de la máquina virtual con el direccionamiento que nos interese.

Este método se utiliza si queremos comunicar máquinas virtuales entre sí y que estén aisladas, ya que no podrán conectarse con el exterior, ni siquiera con el propio equipo físico.

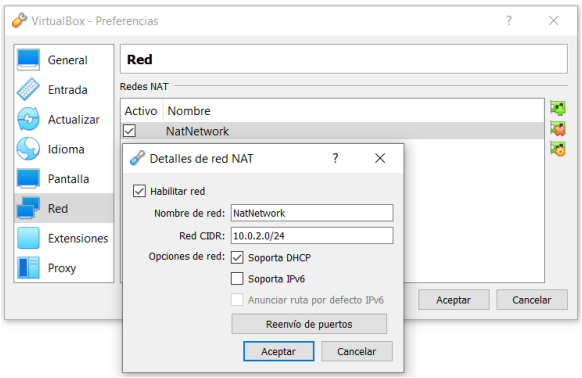


Red Interna

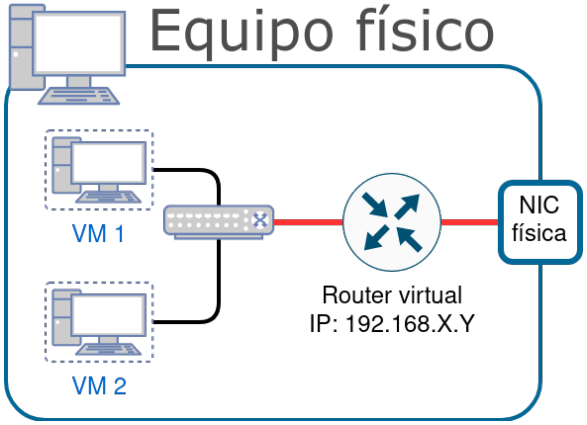
1.2.4. Red NAT

Podría definirse como una mezcla de NAT y red interna. Las máquinas virtuales podrán pertenecer a una única red, se podrán comunicar entre ellas, estarán detrás de un NAT de la red física y se podrán comunicar con el exterior.

Para poder usar ese modo hay que crear la “red NAT” en Virtualbox yendo a “*Archivo → Preferencias → Red*” y ahí se creará las redes NAT que queramos con el direccionamiento interno que nos interese.



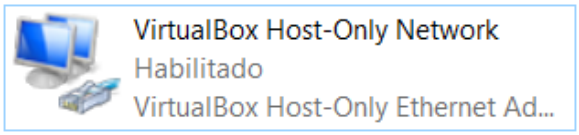
A la hora de crear la máquina virtual y elegir la opción “Red NAT” se podrá elegir entre las redes creadas en el paso anterior.



Red NAT

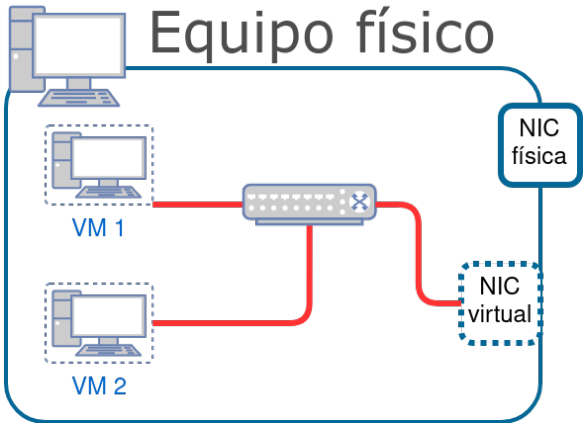
1.2.5. Adaptador sólo-anfitrión

Este tipo de adaptador es similar al de "red interna" pero con la posibilidad de comunicarse con el equipo físico anfitrión. En el equipo físico se crea un interfaz virtual y a través de él se podrá comunicar con las máquinas virtuales.



El direccionamiento que existe entre las VMs y el host se define en Virtualbox, dentro de "*Archivo* → ***Administrador de red de anfitrión***". Las máquinas virtuales podrán coger IP de ese direccionamiento por DHCP.

Mismo uso que "red interna" pero añadiendo la opción de comunicarnos con el host anfitrión.



Red en modo "Sólo Anfitrión"

1.3. Resumen de los adaptadores

A continuación se expone una tabla que resume los distintos tipos de adaptadores que existen y la conectividad posible entre las máquinas virtuales que usan esos adaptadores y el host anfitrión (fuente).

Modo \ Conectividad	VM → Host	VM ← Host	VM1 ← → VM2	VM → LAN real	VM ← LAN real
Adaptador puente	✔	✔	✔	✔	✔
NAT	✔	Redirección de puertos	✘	✔	Redirección de puertos
Red interna	✘	✘	✔	✘	✘
Sólo-anfitrión	✔	✔	✔	✘	✘

En la documentación se explica cómo realizar la redirección de puertos.

2. Gestión de copias de seguridad (backups)

2.1. Introducción

Una copia de seguridad, o ***backup***, **es una copia de los datos originales** que se realiza con el fin de disponer de un medio **para recuperarlos en caso de pérdida**. Las copias de seguridad son útiles en caso de que los datos originales se hayan perdido por un fallo humano, fallo de sistema (rotura de disco duro), modificado por un virus, evento natural (incendio del edificio), un ataque (borrado deliberado de los datos), etc…

En el proceso de copia de seguridad también **se tiene que tener en cuenta el método de restauración de los datos**, que es el momento en el que necesitaríamos realizar la restauración de los datos del *backup* para dejarlos en la ubicación original.

A la hora de pensar un sistema de copias de seguridad, tendremos que tener en cuenta los siguientes puntos:

- Importancia de los datos
- Espacio ocupado por la copia de seguridad
- Ubicación de la copia de seguridad
- Plan de recuperación ante desastre
- Punto máximo que podemos recuperar
- Tiempo estimado de recuperación de los datos

Por otro lado, también tendremos que tener en cuenta el método que vamos a utilizar para realizar la copia de seguridad, y cómo va a ser el proceso. Este proceso dependerá del servicio/sistema al que vayamos a realizar el backup.

Y por último, la estrategia a utilizar:

- Estrategia multi-backup completos
- Incrementales y/o diferenciales
- Estrategia personalizada
- La que el programa que vayamos a utilizar nos permita
- …

Antes de realizar la copia de seguridad de los datos tendremos que tener muy en cuenta distintos apartados, que deberán ser analizados en detalle y puede que modificados en el futuro.

2.2. A tener en cuenta

Durante la **planificación** de la creación de las copias de seguridad hay ciertos aspectos que tenemos que tener en cuenta para que la gestión de copias de seguridad se realice de manera satisfactoria.

Esta planificación debe realizarse conociendo los datos que maneja la empresa, la importancia de los mismos, el volumen que ocupan, la cantidad de cambios que reciben, ...

Información

Una buena planificación para la gestión de copias de seguridad de una empresa puede involucrar a varias personas de distintos ámbitos internos.

2.2.1. Conocer los datos y la importancia de los mismos

Para saber sobre qué datos tenemos que realizar las copias de seguridad, debemos conocer los datos que manejamos en la empresa, la importancia que tienen y su ubicación. **Ejemplo**: no es lo mismo la importancia de los datos gestionados en la base de datos de clientes, o el directorio "Descargas" el PC de un usuario.

También tendremos que conocer el tamaño actual de esos datos, y saber cuánto volumen total alcanzan. Cuanto mayor tamaño tengan, también mayor será el espacio ocupado por las copias de seguridad.

Otro aspecto a tener en cuenta es saber las modificaciones que sufren esos datos, y estimar el volumen de esos cambios. **Ejemplo**: no es lo mismo que una base de datos reciba 10 clientes nuevos al día o que reciba 15.000 ventas a la hora en nuestra tienda online.

Información

Debemos conocer los datos que existen en nuestra empresa, la ubicación, la importancia y el volumen que ocupan para así poder realizar un buen plan de gestión de copias de seguridad.

2.2.2. Espacio ocupado por la copia de seguridad

Estimar el espacio que nos va a ocupar una copia de seguridad puede parecer sencillo, pero no siempre es así, y es por eso que conocer los datos, tal como hemos dicho antes, es de vital importancia.

Tendremos que tener en cuenta cuánto ocupan nuestros datos originales, cuál va a ser la estrategia de backup a utilizar, estimar el incremento de espacio que puede haber en los datos originales…

Por lo tanto, siempre deberíamos estimar por alto el espacio ocupado, y siempre pudiendo realizar la expansión de ese espacio "en caliente", para que las copias de seguridad funcionen cada día.

¡Cuidado!

No realizar copias de seguridad debido a falta de espacio debería ser considerado un fallo de primer orden y que debe ser subsanado lo antes posible.

2.2.3. Punto mínimo en el tiempo que queremos recuperar

Es posible tener copias de seguridad que estén completamente actualizadas en cuanto se realizan cambios en los datos originales, pero la puesta a punto de estos sistemas puede no estar a nuestro alcance (por no

conocer cómo se realizan o por un coste elevado).

En ciertos casos, algunas empresas pueden aceptar la pérdida de ciertos datos, ya que el gestionar que las copias se realicen “en tiempo real” es más caro que el repetir el proceso de crear esos datos de nuevo.

Ejemplo: Una empresa puede no necesitar hacer backup de la base de datos de cuándo fichan los empleados en tiempo real, porque si se pierden los datos de un día es asumible (el backup se hace cada noche). En cambio, la base de datos de marketing tiene que realizarse en tiempo real.

2.2.4. Punto máximo en el tiempo que queremos recuperar

Otro aspecto importante a tener en cuenta es **pensar cuánto tiempo para atrás queremos poder llegar para poder realizar la restauración de los datos**. No será lo mismo querer realizar una restauración de los datos de hace una hora, un día, de hace una semana o de hace un año, ya que esta decisión influirá en toda estrategia de copia de seguridad.

Es por eso, que a los datos hay que darle la importancia que se merece, y es bastante probable que distintos datos tengan distintos puntos de restauración máximo.

Ejemplo: No es lo mismo los datos de facturación de una empresa, que la ley nos marca que debemos guardar dichos datos (4 años), los datos de una página web que apenas cambia (igual nos sirve una copia a la semana) o la carpeta personal de un usuario en su equipo (dado que en su equipo no debería guardar nada importante de la empresa, se podría asumir una pérdida de datos).

2.2.5. Ubicación de la copia de seguridad

Es muy importante también tener en cuenta cuál va a ser la ubicación de la copia de seguridad y las consecuencias que eso puede conllevar. Vamos a poner varios ejemplos:

- Si decidimos hacer la copia de seguridad en el mismo equipo informático, un problema de electricidad podría suponer la pérdida de los datos originales y los backups.
- Si decidimos realizar la copia de seguridad en la misma sala de ordenadores donde se sitúan los datos originales, si esa sala se ve involucrada por un factor que destruya todo (un incendio, un robo, …), perderíamos todos los datos.
- Si decidimos realizar la copia en la misma oficina, edificio, puede pasar lo mismo que el punto anterior
- Si decidimos realizar la copia en la “nube”, el tiempo para realizar la copia de seguridad y el tiempo de restauración de los datos puede incrementarse
- Si decidimos realizar la copia en otra oficina, dependemos de la conexión, los datos deberían ir cifrados, tiempo de creación de la copia de seguridad y restauración…

Como se puede ver, cada ejemplo puede tener unos pros y unos contras.

¡Cuidado!

La copia de seguridad nunca debería estar en el mismo equipo informático, y mucho menos en el mismo disco duro físico, que los datos originales.

2.2.6. Tiempo estimado de la copia

Teniendo en cuenta lo comentado previamente, el tiempo estimado de la copia de seguridad es muy importante, y es posible que varíe en el tiempo. No es lo mismo realizar la copia de seguridad de unos pocos megabytes o de varios gigabytes.

Es por eso por lo que tendremos que tener en cuenta el tiempo que tarda en realizarse nuestras copias de seguridad para así asegurar que se realizan de manera correcta y no se solapan en el tiempo.

2.2.7. Plan de recuperación ante desastre

Tan importante es tener una copia de seguridad como un plan ante un posible desastre. De nada sirve tener una copia de seguridad si restaurarla nos va a llevar semanas por no saber el estado de las mismas, o si su restauración es compleja.

Por lo tanto, **es muy importante disponer de un plan de recuperación ante desastres que esté actualizado**, que contemple si ha habido modificaciones en la gestión de copias de seguridad, que tenga en cuenta los pasos a realizar…

Este plan de recuperación **debería ser puesto en práctica cada cierto tiempo para asegurar que sigue estando vigente**, y si no, realizar las modificaciones oportunas.

¡Atención!

Es muy importante disponer de un plan de recuperación ante desastres que esté actualizado y probar que siga vigente cada cierto tiempo.

El plan de recuperación ante desastre debería contener:

- Los distintos métodos y estrategias de copias de seguridad que existen en la empresa
- Ubicaciones donde se realizar las copias de seguridad
- Importancia de los datos de los que se realizar la copia de seguridad
- Cómo realizar la restauración de cada sistemas de los que se ha realizado la copia de seguridad
- Orden para realizar las restauraciones y dada la importancia de los datos, la prioridad de las restauraciones
- ...

Información

El plan de recuperación es algo que toda compañía debe tener a la hora de crear sus sistema de copias de seguridad, ya que deben de ir de la mano.

2.2.8. Tiempo estimado de recuperación de los datos

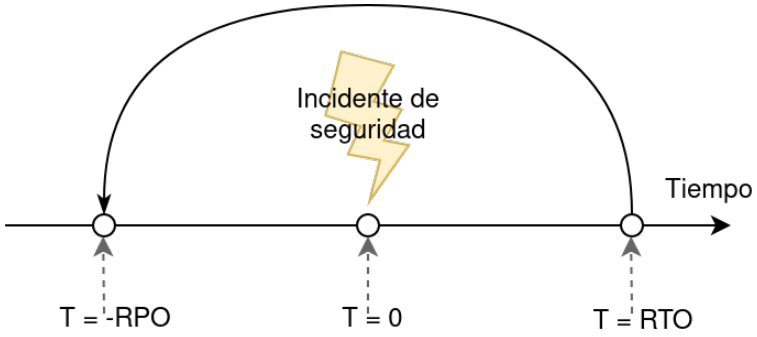
El último punto, pero no por ello menos importante, es el tiempo estimado de recuperación de los datos.

¡Cuidado!

De poco sirve tener copias de seguridad y un plan de recuperación, si luego es posible que estemos dos semanas para recuperar los datos.

Es por eso que tenemos que conocer el tiempo estimado de recuperación de los datos para poder gestionar los tiempos con los empleados, los clientes…

Este tiempo estimado, al igual que el tamaño de las copias, puede variar en el tiempo, por lo que **debemos asegurar cada cierto tiempo el realizar simulacros para asegurar qué tiempos se manejan**.



Teniendo en cuenta el gráfico anterior:

RTO: es el objetivo de tiempo de recuperación (*Recovery Time Objective*), y es el tiempo que se tarda en restablecer el servicio, o al menos los mínimos acordados. Cuanto menor sea el tiempo de recuperación, mejor.

RPO: es el objetivo de punto de recuperación (*Recovery Point Objective*), y es el último instante de tiempo previo al incidente al que los sistemas son capaces de regresar. Habitualmente suele ser marcado por la frecuencia con que se realicen las copias de seguridad. De nuevo, cuanto más cercano a T=0 mejor.

2.3. Métodos de Backup

Existen distintas estrategias a la hora de crear un backup, por lo que tendremos que analizar los datos que queremos salvaguardar y decidir la mejor estrategia.

2.3.1. Copiar los ficheros

Sería la metodología más sencilla. Simplemente haremos una copia de los datos importantes en otra ubicación/soporte. Este sistema hace que cada X tiempo, se guarden los ficheros y obtendremos una copia

completa de los datos.

El problema es que **no tendremos la posibilidad de recuperar un fichero en un estado anterior a la última copia**. Es decir, si se realiza la copia de seguridad a las 6:00 de la mañana, no podremos obtener el estado del fichero previo a ese. Si a las 11:00 el usuario corrompe el fichero, podríamos recuperar al estado de las 6:00, pero no al estado de ayer.

2.3.2. Sistema incremental a nivel de bloque

Se basa en copiar sólo los bloques físicos que han sido modificados. Tendremos que tener guardado el fichero original por un lado, y a partir de ahí sólo guardaremos las modificaciones de manera incremental.

Para recuperar el fichero, primero tendremos que elegir restaurar el fichero original completo, y posteriormente aplicar los bloques incrementales hasta llegar al punto de restauración que nos interesa.

2.3.3. Sistema incremental o diferencial binaria

Similar al anterior, pero esta vez en lugar de hacer uso de bloques (1K, 4K, 8K…), la parte incremental o diferencial sería a nivel binario.

2.3.4. Versionado de ficheros

El versionado de ficheros se puede realizar de distintas maneras:

- **Utilizando el sistema de ficheros**: Existen sistemas de ficheros que cuando uno es modificado, automáticamente crea una versión nueva del mismo.
- **Versionado manual**: cuando el usuario quiere crear una nueva versión del fichero, debe ejecutar a mano una acción para guardar la nueva versión del mismo.

2.4. Estrategias de backup

Existen muchas estrategias a la hora de realizar una copia de seguridad, quizá tantas como entornos existen, ya que se pueden utilizar estrategias generales o crear una para cada caso concreto. Vamos a explicar las más habituales.

2.4.1. Multi-backup completos

Este método sería el más sencillo, pero también el que más espacio nos ocuparía, y posiblemente el que más tiempo puede tardar, dependiendo de los datos a copiar.

En este método, podríamos hacer una copia completa de los datos que queremos guardar en tantas ubicaciones como queramos. Supongamos que queremos poder llegar a restaurar hasta 3 días de un fichero corrupto, eso supondría que tendríamos que tener 3 ubicaciones distintas donde cada día haríamos un backup completo de los datos. Por ejemplo:

- Ubicación **1**: el **lunes** haríamos el primer backup completo
- Ubicación **2**: el **martes** haríamos el segundo backup completo

- Ubicación **3**: el **miércoles** haríamos el tercer backup completo
- Ubicación **1**: el **jueves** haríamos el backup completo (machando los datos previos que había del lunes)
- …

Como se puede ver, es una estrategia cíclica, sencilla, y que en ciertos casos puede ser más que suficiente. Pero como se ha comentado previamente, cada día se realizará un backup completo, por lo que habrá que tener en cuenta el tiempo que tarda, el espacio que ocupa, ...

2.4.2. Incrementales y/o diferenciales

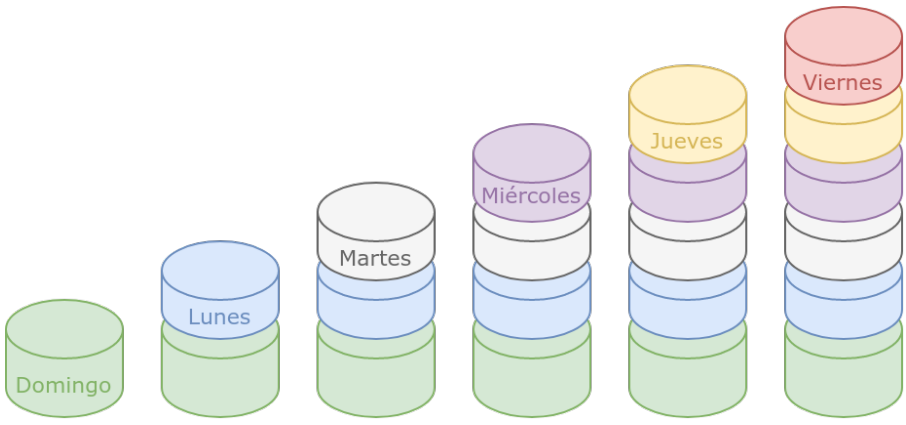
Normalmente es el método más habitual y todos los programas suelen tener estos métodos a la hora de realizar copias de seguridad.

El hacer uso de copias incrementales y/o diferenciales nos permitirá poder restaurar los datos hasta un punto concreto. Por eso será muy importante tener en cuenta cuánto tiempo queremos poder llegar a restaurar, porque no es lo mismo a la hora de estimar espacio ocupado, si queremos restaurar los datos de una semana, de hace un mes o de hace un año.

Por lo tanto, a la hora de utilizar esta metodología, como ya se ha comentado, que es la más habitual, la estimación de espacio debe de ser holgada, y pensar a futuro, en cuánto puede llegar a aumentar el espacio ocupado.

2.4.2.1. Incrementales

Supongamos que realizamos una copia completa de los datos el domingo, esta será la base para las siguientes copias incrementales de los datos:

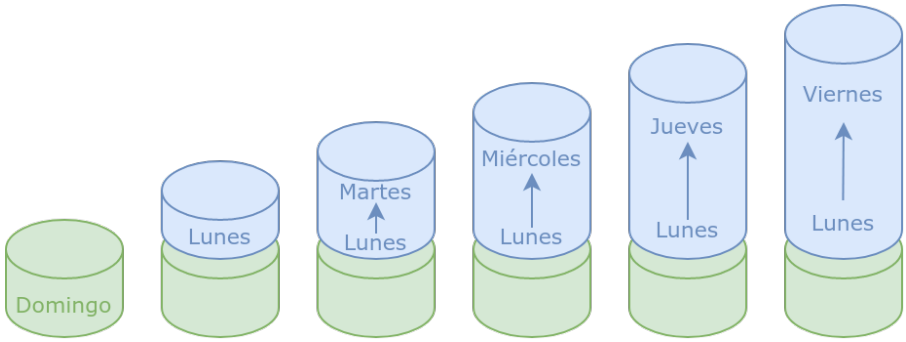


El lunes se hará una copia guardando sólo los datos que han cambiado respecto a la copia del domingo. El martes, se hará una copia de los datos que han sido modificados desde la copia del lunes… Y así sucesivamente.

Este método es rápido, fácil de realizar y sabemos claramente cuándo se han modificado los datos, por lo que si tenemos cada incremental separado en distintas carpetas, la restauración sería también sencilla.

2.4.2.2. Diferencial

Similar al incremental, pero cada día se guarda todos los datos que han sido modificados desde el último backup completo:



Esto hace que cada backup diferencial pueda ocupar más que un incremental, pero cada día obtendremos todos los cambios desde el último backup completo.

2.4.2.3. Incremental y diferencial

También suele ser habitual realizar la mezcla de ambas estrategias:

- El primer domingo de cada mes realizar la copia completa.
- Hacer uso del método incremental de lunes a sábado
- Los domingos realizar un diferencial de toda la semana.

2.4.3. Estrategia personalizada

Como se ha comentado previamente, pueden existir tantas estrategias como entornos existan, por lo que dependiendo de nuestros datos, nuestro entorno, nuestro modelo de negocio, tendremos que crearnos una estrategia propia.

Este método puede ser el más complejo, ya que quizá tengamos que realizarlo a mano, pero con ello nos aseguraremos de darle la importancia que se merece a cada parte de los datos que tengamos que realizar un backup.

2.4.4. La que el programa nos marque

Si hacemos uso de un programa de gestión de backups, al final estaremos limitados a lo que el programa nos permita. Como se ha comentado, la estrategia incremental/diferencial suele ser la más ampliamente utilizada, por lo que casi todos los programas deberían tener esta opción.

Dependiendo del programa, nos permitirá hacer mezclas de estrategias, realizar copias a sistemas remotos (la nube, por FTP, SSH…), varios cada día… Por lo que quedaría en nuestra mano analizar distintos programas y ver cuál es el que mejor se adecúa a nuestras necesidades.

2.5. Regla 3-2-1

Con todo lo dicho anteriormente, una de las estrategias más habituales, de manera generalizada y simplificada, es la conocida como la "**Regla 3-2-1**".

Esta regla se resume en:

- **3 copias completas** de los datos.
- En **2 tipos de almacenamiento distintos**
- **1 de las copias debe estar fuera**

Imaginemos una base de datos de una tienda online que está situada en un servidor en un RACK junto con otros servidores nuestros en un proveedor contratado. En esta base de datos tenemos **los datos originales**.

Haremos uso de otro de los servidores del RACK en el que realizaremos una copia de seguridad completa. Esta será la **segunda copia completa**, que está en otro servidor y que podremos acceder de manera rápida a ella.

La **tercera copia** de los datos la tendremos en nuestra oficina, ya que realizaremos una copia remota de los datos. Esta tercera copia está situada fuera del proveedor contratado, por lo que si en ese CPD hubiese un incendio, tendríamos una copia en local.

2.6. Realización de simulacros de recuperación de los datos

De nada sirve lo comentado hasta ahora, si no se realizan simulacros de desastre y de recuperación de los datos para asegurar que todo funciona de manera correcta. Desgraciadamente, **también puede haber fallos en los sistemas de backups**, por lo que aunque creamos que se están realizando correctamente, quizá el día que vayamos a necesitarlos nos damos cuenta que no es así, por lo que no podríamos realizar la recuperación de datos.

¡Cuidado!

Deberíamos crear simulacros de desastre para asegurar que todos los eslabones de la cadena de nuestro sistema de copias de seguridad es funciona de manera correcta.

Estos simulacros nos deberían ayudar a asegurar que:

- Las copias de seguridad funcionan de manera correcta
- Podemos realizar restauraciones de los datos a fecha que nos interesa
- El plan de recuperación ante desastre está actualizado y es funcional
- El tiempo estimado de recuperación es el adecuado una vez restaurados los datos, nuestro sistema es funcional y podemos seguir con la actividad comercial

¡Atención!

Si alguno de los puntos anteriores falla y no es correcto, deberemos repasar todo el plan estratégico de copias de seguridad para solventar los fallos, y así poder volver a realizar un nuevo simulacro para confirmar que esta vez es correcto.

3. Creación de copias de seguridad en GNU/Linux

Tal como hemos visto, la gestión de copias de seguridad es de vital importancia para preservar los datos de nuestra empresa. Debemos interiorizar que el realizar copias de seguridad, tanto en nuestro ámbito personal como profesional, es una parte en la gestión de nuestra información así como el comprobar que se realizan de manera correcta.

Al igual que sucede en Windows, existen múltiples sistemas para realizar copias de seguridad en GNU/Linux, pero uno de los más sencillos de utilizar ya suele venir instalado en la gran mayoría de las distribuciones hoy día: **rsync**.

3.1. Rsync como sistema para sincronizar directorios

Rsync es una aplicación que ofrece la posibilidad de realizar la sincronización de directorios (de manera local o en remoto) de manera eficiente, ya que es capaz de sincronizar sólo las modificaciones realizadas en los ficheros.

Por defecto, **rsync** copia las modificaciones o ficheros nuevos que existen en el directorio de origen y los copia al directorio de destino. Si un fichero es borrado en el directorio origen, **por defecto no se borra en el directorio de destino**.

La manera más sencilla de utilizar rsync es para crear una sincronización de un directorio local a un servidor remoto en el que almacenar una copia de los datos.

¡Cuidado!

Esta sincronización remota equivale a realizar una copia completa de todos los datos (de manera recursiva), o lo que es lo mismo, un *full-backup*.

Ejemplo para sincronizar dos directorios locales

```
ruben@vega:~$ rsync -av directorio_origen  directorio_destino
```

En el ejemplo anterior lo que hace es sincronizar todo el “directorio_origen” dentro del “directorio_destino”. En cambio:

Ejemplo para sincronizar dos directorios locales

```
ruben@vega:~$ rsync -av directorio_origen/  directorio_destino2
```

En este ejemplo, como hemos añadido una barra “/” al final del directorio origen, lo que estamos indicando es que el contenido (y sólo el contenido) se va a sincronizar dentro de “directorio_destino2”.

¡Cuidado!

Hay que tener cuidado con esa "**/**", ya que ponerla o no ponerla da resultados distintos en la sincronización.

3.1.1. Rsync para sincronizar de manera remota

Tal como sabemos, nunca deberíamos realizar copias de seguridad de manera local en el mismo disco duro, ni tenerlas en el mismo equipo, por lo que es necesario poder realizar copias de seguridad desde otros servidores o a otros servidores.

Imaginemos que queremos copiar los datos de una carpeta compartida `/home/sistemas` al servidor remoto 10.40.30.200, y dejar una copia en `/home/backups`:

Sincronizar directorio local a servidor remoto

```
ruben@vega:~$ rsync -av /home/sistemas  root@10.40.30.200:/home/backups
```

Con este comando realizaremos la sincronización al servidor remoto haciendo uso de SSH y conectándonos con el usuario root remoto. Explicación de los distintos apartados del comando:

- **rsync** es el programa que usamos para realizar la sincronización.
- **-av** son los parámetros que realizan la sincronización en moodo archivos y en modo "verbose" (nos indica qué está sincronizando).
- **/home/sistemas** es el directorio con los datos de origen que vamos a sincronizar
- **root** es el usuario con el que nos vamos a conectar al servidor remoto.
- **@10.40.30.200:** es el servidor al que nos vamos a conectar por SSH para realizar en él la copia. **¡OJO!** Es importante esa "@" y los ":" al final.
- **/home/backups** es el lugar donde dejaremos una copia de los datos a sincronizar.

Tal como se ha comentado, rsync hará uso de SSH para realizar la conexión remota, por lo que los datos se enviarán de manera segura y cifrada.

3.1.2. Obtener datos remotos

Si queremos realizar la sincronización en el orden inverso, obtener los datos estando en el servidor de backup y traernos los datos de un servidor remoto, el comando sería:

Sincronizar directorio remoto a directorio local

```
root@backups:~# rsync -av 10.40.30.5:/home/sistemas  /home/backups
```

En este caso, estando en el servidor de backups, vamos a traernos los datos del servidor 10.40.30.5. En este caso, antes de la IP del servidor no hemos puesto usuario, por lo que se realizará la conexión con el mismo usuario que somos actualmente, en este caso **root**.

3.1.3. Opciones extra

Hasta ahora hemos visto las opciones más básicas para sincronizar directorios (tanto locales como remotos), pero rsync cuenta con muchas opciones extra que es interesante conocer:

- **-z** o **--compress** sirve para comprimir los datos antes de realizar el envío o recepción de los datos. Sólo es útil si vamos a realizar la sincronización de manera remota.
- **--progress** sirve para ver el progreso de cómo va la transferencia de los ficheros
- **--delete** sirve para borrar en el destino los ficheros que no existen en el origen.
- **--exclude "*txt"** excluye los ficheros que termina con la extensión "txt" al hacer la sincronización.

VIII

Ejercicios

1. Sistemas de numeración: tabla de conversión

La siguiente tabla sirve a modo de resumen de los sistemas de numeración.

Información

Esta tabla no hay que aprenderla de memoria. Hay que entender los sistemas de numeración y de esta manera se puede crear.

Decimal	Binario	Octal	Hexadecimal
$0_{(10}$	$0_{(2}$	$0_{(8}$	$0_{(16}$
$1_{(10}$	$1_{(2}$	$1_{(8}$	$1_{(16}$
$2_{(10}$	$10_{(2}$	$2_{(8}$	$2_{(16}$
$3_{(10}$	$11_{(2}$	$3_{(8}$	$3_{(16}$
$4_{(10}$	$100_{(2}$	$4_{(8}$	$4_{(16}$
$5_{(10}$	$101_{(2}$	$5_{(8}$	$5_{(16}$
$6_{(10}$	$110_{(2}$	$6_{(8}$	$6_{(16}$
$7_{(10}$	$111_{(2}$	$7_{(8}$	$7_{(16}$
$8_{(10}$	$1000_{(2}$	$10_{(8}$	$8_{(16}$
$9_{(10}$	$1001_{(2}$	$11_{(8}$	$9_{(16}$
$10_{(10}$	$1010_{(2}$	$12_{(8}$	$A_{(16}$
$11_{(10}$	$1011_{(2}$	$13_{(8}$	$B_{(16}$
$12_{(10}$	$1100_{(2}$	$14_{(8}$	$C_{(16}$
$13_{(10}$	$1101_{(2}$	$15_{(8}$	$D_{(16}$
$14_{(10}$	$1110_{(2}$	$16_{(8}$	$E_{(16}$
$15_{(10}$	$1111_{(2}$	$17_{(8}$	$F_{(16}$

Continúa en la siguiente página

$16_{(10}$	$10000_{(2}$	$20_{(8}$	$10_{(16}$
$17_{(10}$	$10001_{(2}$	$21_{(8}$	$11_{(16}$
...	...	...	...
$29_{(10}$	$11101_{(2}$	$35_{(8}$	$1D_{(16}$
$30_{(10}$	$11110_{(2}$	$36_{(8}$	$1E_{(16}$
$31_{(10}$	$11111_{(2}$	$37_{(8}$	$1F_{(16}$
$32_{(10}$	$100000_{(2}$	$40_{(8}$	$20_{(16}$

2. Conversiones

2.1. De decimal ...

... a binario

30 =	145 =	278 =	329 =
512 =	776 =	1024 =	1376 =

... a octal

31 =	88 =	127 =	234 =
524 =	876 =	1098 =	2475 =

... a hexadecimal

29 =	340 =	530 =	940 =
1212 =	1512 =	2120 =	3201 =

2.2. De binario ...

... a decimal

111001 =	11001010 =	110101101 =	1010101101 =
10101010101110 =	10101111101 =	111100010110 =	111000100110110 =

... a octal

111001 =	11001010 =	110101101 =	1010101101 =
10101010101110 =	10101111101 =	111100010110 =	111000100110110 =

... a hexadecimal

111001 =	11001010 =	110101101 =	1010101101 =
10101010101110 =	10101111101 =	111100010110 =	111000100110110 =

2.3. De octal ...

... a decimal

54 =	77 =	134 =	267 =
345 =	376 =	412 =	564 =

... a binario

54 =	242 =	356 =	654 =
1235 =	3457 =	7652 =	21315 =

... a hexadecimal

36 =	175 =	657 =	1456 =
3245 =	7541 =	71727 =	754315 =

2.4. De hexadecimal ...

... a decimal

1F =	23B =	86F =	AA1 =
FF3 =	2F1C =	4AD7 =	5CABD =

... a binario

1D =	72A =	F5C =	157A =
9FAF =	18ABFD =	2A3D5F =	F6A7DE1 =

... a octal

3E =	7F =	1AD =	FAD =
4D1C =	7A9D =	A2B7C =	741FA3 =

2.5. Mezcladas

Ten en cuenta la base de origen y la base de destino.

$175_{(10}$ = (16	$475_{(8}$ = (2	$9A3_{(16}$ = (10
$175_{(8}$ = (2	$754_{(8}$ = (16	$101110111_{(2}$ = (8
$274_{(10}$ = (2	$4751_{(8}$ = (16	$5742_{(16}$ = (8
$1789_{(10}$ = (2	$1175_{(8}$ = (16	$3AB1_{(16}$ = (2
$101000111100_{(2}$ = (10	$101010_{(8}$ = (2	$1011101_{(16}$ = (10
$101000111100_{(2}$ = (8	$74513_{(8}$ = (2	$78954_{(16}$ = (2
$10100011011001100_{(2}$ = (8	$724123_{(8}$ = (2	$7AB1FE4_{(16}$ = (2