**Sistemas operativos**

Un sistema operativo (en inglés más conocido a través de las siglas OS, operating system) es un conjunto de programas que permite manejar la memoria, disco, medios de almacenamiento de información y los diferentes periféricos o recursos de nuestra computadora. Según lo que estudiamos hasta ahora, si recordamos los términos que vimos en la introducción de este módulo, podríamos decir que un sistema operativo es el software que maneja el hardware. Es decir que administra los recursos ofrecidos por el hardware y actúa como un intermediario entre la computadora y su usuario.

**¿Qué es el sistema operativo?**

“El sistema operativo es el soporte lógico que controla el funcionamiento del equipo físico.”

“Desde el punto de vista usuario, es un conjunto de programas y funciones que ocultan los detalles del hardware, ofreciendo al usuario una vía sencilla y flexible de acceso al mismo.”

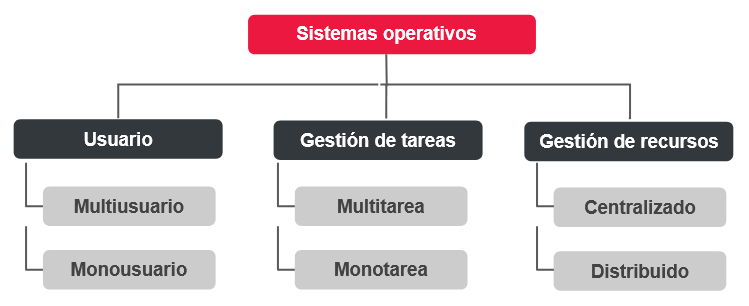
**Funcionalidad del sistema operativo**

**Recursos** administrados por el sistema operativo

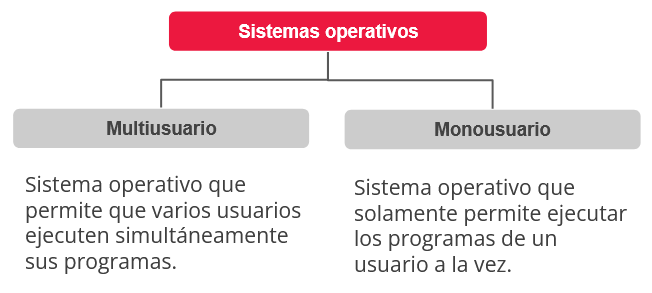
* Gestionar la **memoria** de acceso aleatorio y ejecutar las aplicaciones, designando los recursos necesarios.
* Administrar la **CPU**, gracias al algoritmo de programación.
* Direccionar las **entradas y salidas** de datos (a través de drives), por medio de los periféricos de entrada y salida.
* Administrar **la información** para el buen funcionamiento de la PC.
* Dirigir **las autorizaciones** de uso para el usuario.
* Administrar **los archivos**.

**Tipos de sistema operativo**

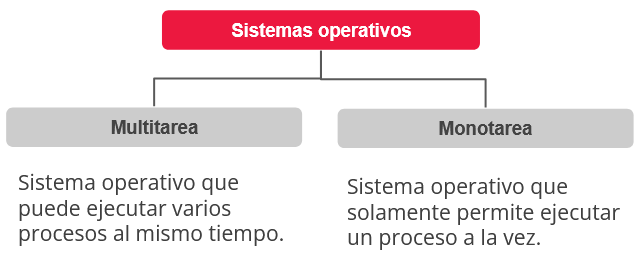
“Los tipos de sistemas operativos varían según el hardware y la función de cada dispositivo.”



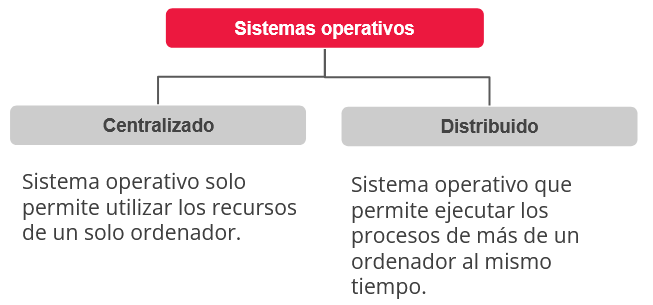
**Según el usuario** pueden ser**:**



**Según la gestión** de tareas puede ser:

****

**Según la gestión** de recursos pueden ser:

****

****

**Monotarea**  
Los sistemas operativos monotarea se caracterizan por poder realizar solo una tarea a la vez sin que se pueda interrumpir. Estos son los sistemas operativos más primitivos, como DOS. Por ejemplo, si queremos imprimir algún archivo en estos sistemas operativos, no vamos a poder realizar ninguna otra tarea hasta que la computadora imprima y pueda recibir otra instrucción.

**Multitarea**  
Los sistemas operativos multitarea nos permiten realizar varias tareas al mismo tiempo, estos son muchos más comunes hoy en día.

**Monousuario**  
Los sistemas operativos monousuarios, como su nombre lo indica, soportan solo a un usuario a la vez, no importa cuántos procesadores tenga la computadora o cuántas tareas realice el usuario, solo podrá dar servicio a uno. Ejemplos podrían ser las primeras versiones de Windows para computadoras domésticas, como Windows 1.0.

**Multiusuario**Los sistemas operativos multiusuario pueden dar servicio a varios usuarios al mismo tiempo, ya sea por medio de terminales conectadas a la computadora o por sesiones remotas en una red de comunicaciones. Ejemplos de estos sistemas operativos pueden ser Unix, Linux o Solaris.

**Monolítica**La estructura monolítica está constituida por un solo programa compuesto de una serie de rutinas entrelazadas entre sí, de tal forma que pueden comunicarse entre ellas. Estos sistemas operativos suelen estar hechos a medida, por lo que son muy rápidos, pero no tienen flexibilidad para soportar diferentes tipos de aplicaciones.

**Máquina virtual**Los sistemas operativos tipo máquina virtual separan dos conceptos que suelen estar unidos en el resto de sistemas: la multiprogramación y la máquina extendida. El objetivo de los sistemas operativos de máquina virtual es el de integrar distintos sistemas operativos dando la sensación de ser varias máquinas diferentes.

**Jerárquica**A medida que fueron creciendo las necesidades de los usuarios y se perfeccionaron los sistemas, se hizo necesaria una mayor organización del software del sistema operativo, donde una parte del sistema contenía subpartes y estaba organizado en forma de niveles. A este sistema operativo se le conoce como estructura jerárquica, ya que se subdivide en capas o anillos, perfectamente definidas y con una clara interfaz con respecto al resto de los recursos.

**Cliente-Servidor**El sistema operativo cliente-servidor sirve para toda clase de aplicaciones, por lo tanto, es de propósito general y cumple con las mismas actividades que los sistemas operativos convencionales. La idea es mantener la visión que tiene un usuario de una computadora personal, pero la red le permite compartir el espacio del disco o la impresora con el fin de economizar recursos.

# El centro de computos

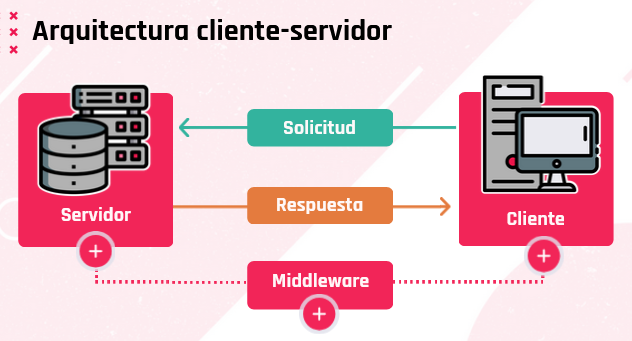
# Arquitectura cliente-servidor

La arquitectura cliente-servidor persigue el objetivo de procesar la información de un modo distribuido. De esta forma, pueden estar dispersos en distintos lugares y acceder a recursos compartidos.

Además de la transparencia y la independencia del hardware y del software, una implementación cliente-servidor debe tener las siguientes características:

* Utilizar protocolos asimétricos, donde el servidor se limita a escuchar en espera de que un cliente inicie una solicitud.
* El acceso es transparente, multiplataforma y multiarquitectura.
* Se facilitará la escalabilidad, de manera que sea fácil añadir nuevos clientes a la infraestructura —escalabilidad horizontal— o aumentar la potencia del servidor o servidores, aumentando su número o su capacidad de cálculo —escalabilidad vertical—.

Ahora veamos las características de los tres componentes de la arquitectura cliente-servidor.



**Servidor**  
Genéricamente un servidor es un ordenador, pero con prestaciones elevadas. Sin embargo, desde este enfoque, un servidor es un proceso que ofrece recursos y servicios a los clientes que lo solicitan (back end).

Según el tipo de servidor implantado, tendremos un tipo de arquitectura cliente-servidor diferente. A su vez, debido a que los programas y datos se encuentran centralizados, se facilita la integridad y el mantenimiento.

**Cliente**  
De manera genérica, nos referimos a una computadora, normalmente con prestaciones ajustadas, sin embargo, en entornos cliente-servidor, se utiliza el término front end, ya que es un proceso que solicita los servicios del servidor a través de una petición del usuario.

Un proceso cliente se encarga de interactuar con el usuario, por lo que estará construido con alguna herramienta que permita implementar interfaces gráficas (GUI).

**Middleware**  
Es la parte del software del sistema que se encarga del transporte de los mensajes entre el cliente y el servidor y facilita la interconexión de sistemas heterogéneos sin utilizar tecnologías propietarias. Por lo cual, se ejecuta en ambos lados de la estructura.

El middleware permite independizar a los clientes y a los servidores. Además, ofrece más control sobre el negocio, debido a que permite obtener información desde diferentes orígenes —uniendo tecnologías y arquitecturas distintas— y ofrecer de manera conjunta.

Otra característica es que los sistemas están débilmente acoplados ya que interactúan mediante el envío de mensajes.



**Soporte de red**   
Es indispensable que tengan un soporte completo para poder brindar conectividad.

**Amplia compatibilidad con el hardware**   
Un punto fundamental para el aprovechamiento pleno de las características del servidor es que el S.O. sea capaz de exprimir al máximo las características técnicas del hardware en donde se ejecuta, es por ello que se debe priorizar el uso de S.O. actualizados y con un soporte importante de controladores. Por ejemplo, que nuestros controladores permitan acceder a características avanzadas de gestión de discos rígidos, con el propósito de realizar arreglos redundantes para tener mayor velocidad y tolerancia a fallos.

**Seguridad**Teniendo en cuenta el rol que cumple el servidor, es de vital importancia que el S.O. instalado sea seguro; eso implica no solo que este actualizado con todos los parches/actualizaciones, sino que además debe tener aplicadas políticas estrictas de acceso para prevenir accesos no autorizados o ataques.

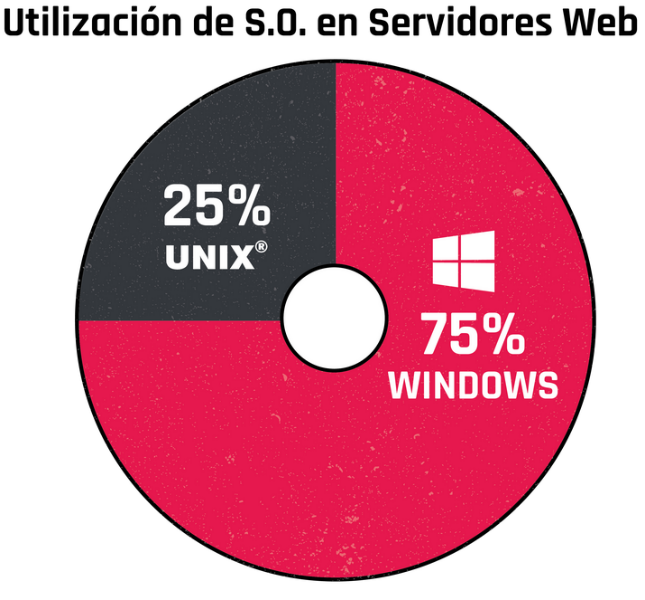
Adicionalmente esa seguridad debe reforzarse con la instalación de Firewalls (por software o hardware) y antivirus. En este ítem también debemos incluir el respaldo de la información, ya sea por medio de herramientas que el propio S.O. nos ofrezca o instalando software externo, con el propósito de tener la menor pérdida de datos posible en caso de fallos fatales.

**Tolerancia o fallos**   
Debemos priorizar S.O. que en su arquitectura permita una tolerancia a fallos, ya sea mediante la generación de granja de servidores, que interconectados, operen como una gran unidad de proceso, dando la posibilidad que ante la caída de uno de los integrantes de la granja, otro puede tomar su rol y responsabilidad.

Conocidas estas características casi innegociables a la hora de elegir un S.O. para servidores, es común preguntarnos por qué se utilizan sistemas operativos y versiones específicas para el entorno de servidores y no usamos los mismos S.O. que tenemos instalados habitualmente en nuestras computadoras o estaciones de trabajo; eso se basa en algunas diferencias, a destacar:

* **Manejo diferente del hardware:** Debido al diferente propósito, los S.O. de estaciones de trabajo no pueden aprovechar todo el hardware disponible, como, por ejemplo, el manejo de memoria RAM —teniendo el caso de Windows 10 64-bit que puede manejar 6TB de RAM, mientras que Windows Server 2019 alcanza los 24TB—.
* **Características soportadas:** Hay funcionalidades que nativamente un S.O. de estación de trabajo no es capaz de brindar, ya que en su versión de kernel están limitadas o deshabilitadas —casos tales como la virtualización en algunas versiones de Windows 10—.
* **Soporte:** Algo muy importante a tener en cuenta es, cuando nuestro negocio o aplicación depende de un S.O., es el soporte por parte del fabricante/desarrollador. En el caso de los S.O. de estación de trabajo, el soporte/cobertura que tenemos es para un uso específico, si sobre esa base quisiéramos desplegar una arquitectura, por ejemplo, de servidor web, si bien es probable que nos funcione, vamos a carecer de soporte técnico ya que el fabricante nos indicará que para ese propósito esta la version “Server” del producto.

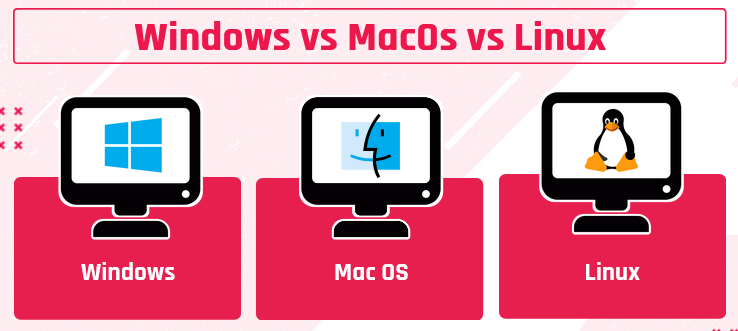
Ya entrando en el terreno de las opciones disponibles para un S.O. de servidor, nos vamos a encontrar con dos opciones predominantes en el mercado:



En el grupo de los UNIX (o UNIX-like) tenemos a:

* GNU Linux
* FreeBSD
* macOsServer

En el grupo de Windows, integrado por toda la familia de Windows Server, tenemos las versiones que van desde la versión 2003 a 2019.

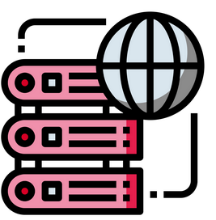
En el transcurso de la cursada nos vamos a centrar en GNU Linux y Windows, ya que son las opciones más utilizadas; iremos conociendo algunas características que ofrecen, como así también sus diferencias. 

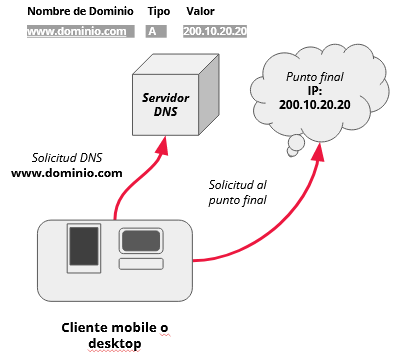
# Servicios que ofrecen los sistemas operativos

**Servicios más utilizados**    
“Un servicio es una puerta de entrada que el S.O. nos ofrece para que podamos ingresar y consumirlo. Los mismos pueden ser útiles para que ejecutemos programas, manipulemos archivos o asignemos recursos.”

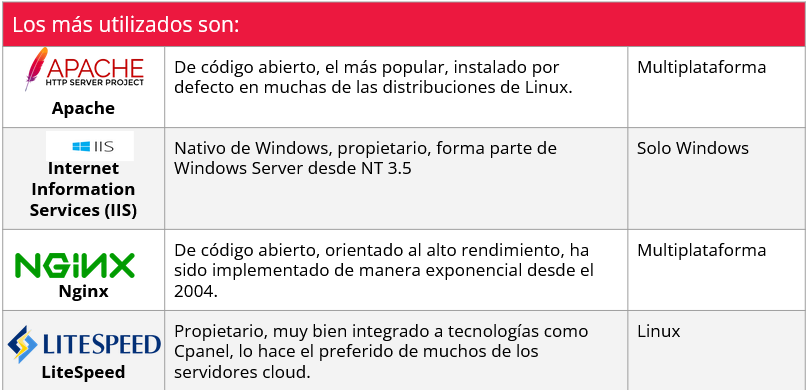
Los sistemas operativos nos pueden ofrecer una amplia variedad de servicios, en algunos casos de manera nativa o instalando un software específico para que funcionen de manera simultánea, tales como:

# 

**Servicios de publicación web**  
Es el software que se encarga de despachar el contenido de un sitio web al usuario. Este proceso de despacho que, a simple vista parece muy simple, es en realidad más complejo de lo que parece, pues toda la “magia” de un webserver ocurre fuera de quien está navegando por un sitio web.

El proceso de despacho comienza en nuestro navegador web. Al escribir la dirección de un sitio web y presionar Enter, comienza la siguiente secuencia: el sistema hace una búsqueda DNS para encontrar en cuál servidor está alojado el sitio en cuestión.

Cuando el server es encontrado, el navegador le pide el contenido del sitio web y, acto seguido, el webserver procesa este pedido y envía dicho contenido al navegador, lo cual da como resultado la visualización del sitio en nuestra pantalla.

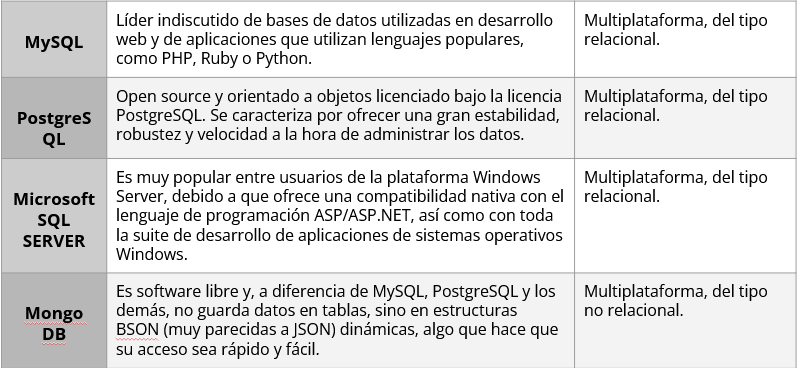
****

# Servicio de base de datos

Un servidor de base de datos, también conocido como database server o RDBMS (Relational DataBase Management Systems) en caso de bases de datos relacionales, es un tipo de software de servidor que permiten la organización de la información mediante el uso de tablas, índices y registros.   
En la actualidad se hace cada vez más extendido el uso de bases de datos no relacionales, conocidas popularmente como No-SQL.

Si vamos a la función fundamental de un servidor de base de datos, encontraremos que es el servicio que provee de información a otras aplicaciones web o equipos/hosts, tal como se especifica en el modelo cliente servidor.

En las diferentes bases de datos mediante un usuario, contraseña y nombre de host, se le permite realizar diferentes tareas, dependiendo del nivel de privilegios que posea, tales como selección, actualización y borrado de datos.

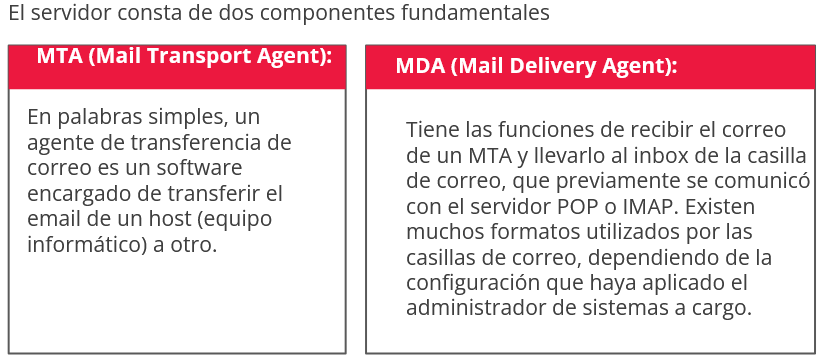


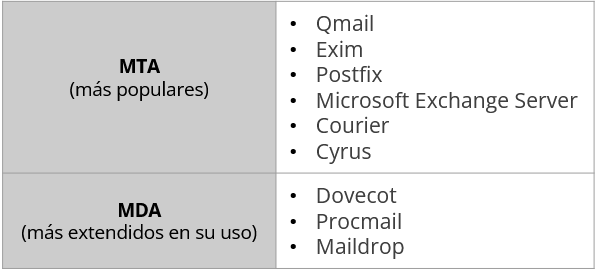
# Servicio de correo electrónico

Un servidor de correo es el encargado de enviar y recibir mensajes de correo electrónico entre hosts, usuarios o servidores.

Entre sus funciones se incluyen el procesado de los mensajes, filtrado, almacenamiento, envío, recepción y reenvío de correos.

Es una de las aplicaciones más populares en usar el protocolo TCP/IP, y que permite en cuestión de segundos comunicarnos con cualquier persona en otra parte del mundo.

****

****

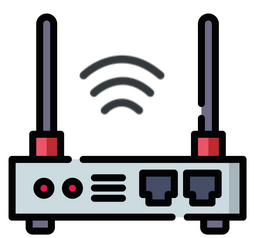
**Servicio de archivos**

Un file server (o servidor de archivos) es un servidor central de una red de ordenadores, que pone a disposición del cliente conectado un conjunto de archivos o parte del mismo. De este modo, proporciona a los usuarios un **lugar de almacenamiento centralizado** para los archivos en sus propios soportes de datos, **disponible para todos los clientes autorizados**.

El administrador del servidor establece unas directrices estrictas para determinar qué usuarios tienen derechos de acceso y a qué datos.

Cada sistema operativo tiene su propio servicio de compartición de archivos, es por ello que la elección muchas veces depende de las plataformas de clientes. Los servicios de archivos más utilizados son:

* CIFS/Samba (utilizado en Linux)
* NTFS Share (nativo de los sistemas Microsoft)
* NFS (desarrollado por Sun y originario de los sistemas UNIX)

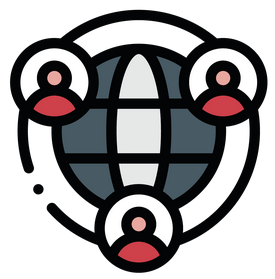
**Servicio de red**Es muy extendido el uso de los propios servicios del sistema operativo con el propósito de satisfacer necesidades de ruteo, firewall o proxy.

Este tipo de implementaciones se aplica más tomando como base sistemas Linux, implementando sobre ellos una serie de paquetes/softwares que dan como resultado routers, firewalls o proxies.

Las soluciones más utilizadas son:

* **PFSense**: Stack basado en BSD, que cumple con múltiples funciones tales como Router, VPN, Firewall.
* **OPNSense:** Derivado del primero y de Monowall, muy presente en el mercado de los routers corporativos.
* **DD-WRT:** Basado en Linux,  extendido su uso en routers hogareños, destaca por los pocos requerimientos para funcionar.

**Servicio de dominio**

Un controlador de dominio (DC) es un servidor que responde a las solicitudes de autenticación y verifica a los usuarios en las redes informáticas. Los dominios son una forma jerárquica de organizar usuarios y equipos que trabajan juntos en la misma red y, el controlador de dominio mantiene todos esos datos organizados y protegidos.

La responsabilidad principal del DC es autenticar y validar el acceso de los usuarios a la red. Cuando los usuarios inician sesión en su dominio, el DC verifica su nombre de usuario, contraseña y otras credenciales para permitir o denegar el acceso.

Los más utilizados son:

* En el ecosistema Windows, Active Directory, presente desde Windows 2000. Utiliza como base los  protocolos LDAP, Kerberos y DNS.
* En el ecosistema Linux, existe NIS y NIS+, originalmente llamado YellowPages. También es posible implementar LDAP en Linux y convivir con ActiveDirectory.

## **¡SI QUERES SABER MÁS!**

## **Powershell**

Vimos que la GUI es la manera más extendida de administrar servidores Windows. Hablamos de sus ventajas en lo que a curva de aprendizaje se refiere y sus pocos cambios con el paso del tiempo, lo que hace que una vez familiarizados con estas consolas podemos dominar en varios aspectos cualquiera de las versiones de Windows Server.

Las consolas de administración, al ser componentes específicos a determinadas funciones —administración de dispositivos, discos físicos, usuarios de dominio, etc.— en determinados escenarios, tales como la automatización de procesos, pueden no ser suficientes y es allí cuando nos surge la necesidad de administrar nuestro Windows con comandos, tal como lo haríamos en Linux.

Windows siempre tuvo una interfaz de línea de comandos, pero limitada en funciones y apoyada en un lenguaje casi obsoleto como Batch.

### **EL LENGUAJE**

El lenguaje de la consola incluye declaración de variables, variables especiales predefinidas, operadores matemáticos, incluyendo igualdades y desigualdades numéricas, manejo de vectores, comparación entre estos, operadores de asignación, vectores asociativos —hashtables—, valores booleanos, ciclos y ruptura de los mismos, operadores de expansión para simplificación de ejecuciones complejas —creación de vectores por medio de llamados a procedimientos, creación dinámica de vectores, etc.—; comentarios, operadores de comparación binaria, caracteres de escape, orden de ejecución, ciclos del tipo "foreach", creación de procedimientos y funciones, creación de filtros, estructuras condicionales complejas —if/then/else/elseif/switch—, operador de invocación dinámica del contenido de variables.

### **LOS COMANDOS**

Los comandos de PowerShell llamados cmdlets —por command-applets—. Están estructurados de la siguiente manera: un verbo y un nombre separados por un guión (-): verbo-nombre. Vamos a ver un ejemplo, para ello abrimos nuestra consola de

Powershell y ejecutamos:

Get-Command

El verbo —evidentemente en inglés— describe la acción a realizar sobre el nombre. En el anterior ejemplo, recuperamos —Get— los comandos —Command—. En este caso, el comando nos devolverá una lista de los comandos disponibles de Powershell. Con PowerShell encontramos numerosos verbos genéricos tales como Get, Set, Add, Remove, entre otros, que se combinan con diferentes nombres como Path, Variable, Item, Object, Computer, etcétera. Los nombres que constituyen los comandos están siempre en singular y esto es válido también para los parámetros. Por lo tanto, es posible, mezclando verbos y nombres, acordarse fácilmente de un buen número de comandos. Hay que tener en cuenta que los comandos, así como sus parámetros asociados, pueden escribirse indistintamente en mayúsculas o en minúsculas. El analizador sintáctico PowerShell no es case sensitive.

## **Bash**

Ya vimos que para poder interactuar con los S.O. teníamos a la interfaz gráfica (GUI) o la interfaz de línea de comandos (CLI), y supimos que la primera es más utilizada en Windows y la segunda, en Linux.

La CLI de Linux, además de ser muy potente, tiene de manera nativa la interpretación de lenguaje llamado BASH, o Bash Shell, el cual tiene como propósito la creación de scripts que facilitan la administración del sistema.

Las funciones Bash pueden:

* Eliminar tareas repetitivas.
* Ahorrar tiempo.
* Proporcionar una secuencia de actividades bien estructurada, modular y formateada.
* Con scripts, podemos proporcionar valores dinámicos a comandos usando argumentos de línea de comando.
* Puede simplificar comandos complejos en una sola unidad en ejecución.
* Una vez creada, se puede ejecutar cualquier cantidad de veces por cualquier persona. Construye una vez y ejecuta muchas veces.
* Los flujos lógicos se pueden construir utilizando funciones Bash.
* Las funciones Bash se pueden ejecutar al inicio del servidor o agregando un cron job programado.
* Los comandos pueden ser depurados.
* Tener comandos de shell interactivos.

Un ejemplo sencillo de una función Bash sería

#!/bin/bash

testfunction

testfunction(){

echo "My first function"

}

El cual como resultado de ejecución devolverá en pantalla

My first function

Las funciones bash pueden aceptar cualquier número de parámetros. El siguiente ejemplo acepta dos parámetros:

#!/bin/bash

testfunction(){

echo $1

echo $2

}

Los scripts bash soportan:

* Bucle *while*
* Bucle *for*
* Declaración *if*
* Elemento lógico *and*
* Elemento lógico *or*
* Declaración *Else If*
* Declaración *case*