**Sobre cómo determinar y configurar nuestro Hardware**

Existen dos agrupaciones para cuando nos referimos a Hardware. Veamos.

**Componentes no integrados:** son los objetos fisicos que requieren ser conectados, fisicamente, a la placa base (no vienen integrados con la consola en cuestión); son componentes que compramos, de manera separada, y tienen que ser conectados a la *placa base.*

**Componentes integrados:** son los objetos fisicos que ya vienen conectados, fisicamente, a la placa base (vienen integrados con la consola en cuestión).

Para aclarar: La memoria ocupa espacio en ambas agrupaciones porque, como ya debe imaginar, toda pc viene con *memorias* integradas (componente integrado); pero, muchas veces, usted desea ampliar la capacidad de dichas *memorias* y recurre a comprar otra; entonces, en ese sentido, la memoria pasaría a ser un componente no integrado (este mismo fenomeno se puede presentar en diferentes casos con diferentes componentes, es normal).



Anotación: un ejemplo de placa base es el que vemos en la imagen; en la imagen, la placa base (o tarjeta madre), es la que está ubicada en la parte más inferior de todos los componentes (el más grande).

***placa base: es una tarjeta de circuito impreso a la que se conectan los componentes que constituyen la computadora.***

**Capas de los dispositivos**

Existen dos grupos de capas de los depositivos, mas esto depende de si nuestro depositivo es físico o virtual. Visualmente, pongamos que cada uno de los tipos de capas se ve así:



En el caso de los dispositivos fisicos, tenemos 3 capas. La parte más baja (**1ra capa**, en el fondo), que se ve de color gris, está representado con un servidor (sería el Hardware); es decir, ahí van todos nuestros componentes físicos: memoria, placa base, discos duros, etc.

**En la 2da capa (color azul):** Es una especie de software muy pequeño y básico (BIOS/UEFI) con el que se puede ocupar, en menor o mayor medida, de la *administración* *de la primera capa* (lo que es el Hardware); es decir, por medio de la 2da capa, se puede administrar el Hardware (aunque sea realmente una administración muy básica).

**& en la 3ra capa (color naranja):** Ya tiene que ver con la colocación y gestión de un sistema operativo, el que queramos instalar: Windows, Linux, etc...

Ahora, ***en cuanto a los dispositivos virtuales*** (o dedicados a la virtualización), hay una capa extra. **Una 4ta capa (cubos):** en lugar de llevar a cabo la gestión de un sistema operativo, lo que se instala es un **Hipervisor.**

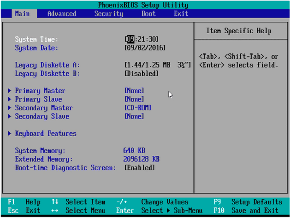
El **Hipervisor** es una pieza de software, a modo de sistema operativo de virtualización; donde, sobre él, es que se crean y se ejecutan las **Maquinas Virtuales***(emuladores de maquinas reales donde se corre el sistema operativo, y software, que querramos -cualquiera: esta es lo que nos permite un Hipervisor).*

**Entonces, la gran diferencia entre la arquitectura tradicional y la arquitectura virtual es que, en ésta última, hay una capa de software que se llama Hipervisora.**

**BIOS VS UEFI, ¿Qué son?**

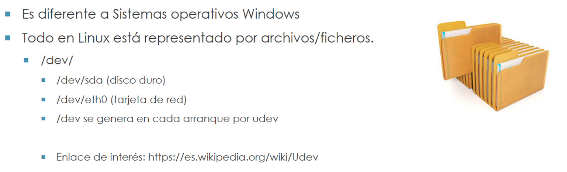
El primer nivel de determinación y configuración de nuestro Hardware se hace a nivel de la BIOS (Sistema básico de entrada y salida). La BIOS es el primer programa que se ejecuta cuando prendemos el pc, por ejemplo. La UEFI (Interfaz de sistema unificada extensible) es una versión nueva y modificada de la BIOS. En todo caso, con ambos, y aquí está la importancia de ellos, podemos administrar nuestros dispositivos fisicos o *Hardware* a nuestro antojo: por medio de ellos podemos habilitar y desabilitar componentes como los USB, hacer configuraciones personalizadas del teclado y/o de ratón, e incluso determinar el orden del arranque.

**Cómo accedemos a ellos?** Depende del fabricante. Por lo general está entre las teclas de tipo F (f1, f2,...). En todo caso, se ve más o menos así:



**El Hardware en Linux**

**Sobre cómo se administran los dispositivos físicos, Hardware, en Linux.**



En Linux, la administración del Hardware se da por medio de *ficheros;* dichos ficheros, apuntan a cada uno de los dispositivos o componentes que tenemos conectados a nuestra PC. Como se habla de *apuntar,* podemos catalogar también a los ficheros como *punteros.*

**Cómo vemos en Linux qué Hardware tenemos conectado?**

Pues, en el directorio ***/dev/***,tal como se aclaró en la imagen. Puede navegar hacia ***dev*** por medio del comando ***cd*** (change directory)y, adicionalmente, si desea saber todos los dispositivos fisicos conectados a su PC; puede ejecutar, estando ya dentros del directorio ***dev,*** el comando ***ls*** (list). Ahí se nos mostrarán, por medio de una lista de archivos, la representación de cada hardware que tenemos conectado. Cada archivo o *fichero* apunta a su propio hardware. Por ejemplo, hay archivos para el *disco duro*, para la *tarjeta de red,* para la *camara,* entre otros.

***Udev*** vendría siendo el proceso que se encarga de alimentar el directorio ***/dev/*** con todos los dispositivos que detecta en el *Hardware*

***Una imagen de cómo se ven los dispositivos de Hardware conectados a nuestro ordenador desde Linux***



Salgamos de un par de dudas.

Nuestro primer fichero enlistado, de azul clarito y que está situado en la primera columna, es **core**. **core** es el fichero que apunta a nuestra unidad de **CD ROM.**

Ahora, el segundo fichero que cumple con las mismas caracteristicas que el anterior, es el **fd.** fd es el fichero que apunta a la disquetera **Floppy disk (el que recibe diskettes... *o recibía, esto ya no se ve mucho*).**

Además, hay un fichero muy particular que se llama: **sda,** hace referencia al primer disco duro que tenemos conectado al ordenador. En ese orden de ideas, el segundo disco duro que tengamos conectado al ordenador, si ese es el caso, sería **sdb;** el tercer disco duro conectado sería **sdc** y el cuarto **sdd,** **etc etc etc...**

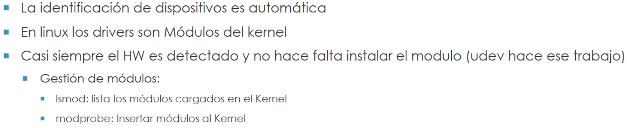
Ahora, cuando los tipos de *“sd”* vienen acompañado al final con un número; digamos, **sda1,** se refiere a una partición dentro del disco duro **sda** como tal; es decir, una partición dentro del primer disco duro que tengamos conectado a nuestro ordenador (para este caso puntual). Como es “sda1” se refiere a una única partición de momento para nuestro primer disco duro conectado. Si **sda** tuviera otra partición, no sólo una, estariamos hablando de **sda1 y** **sda2, y así...**

**Pregunta:** si el disco duro sdb tuviera 4 particiones, ¿cómo se vería enlistado? Se vería enlistado: **sdb1, sdb2, sdb3 & sdb4.**

nota:Los que ve que empiezan con “*t”* son las conocidas ***terminales.***

**Drivers para el reconocimiento de hardware**

**En Windows, a veces toca, o se hace necesario, instalar un *driver* para reconocer un hardware conectado a su ordenador. Pues bien, lo anterior cambia en Linux.**



El sistema de Linux es el **Kernel, core, el corazón del sistema operativo.** Dicho esto, entonces, para que el Kernel reconosca los distintos componentes del Hardware, necesitamos instalar los **módulos del kernel. Los *drivers* de Windows son el equivalente a los *módulos del Kernel* en Linux.** Sin embargo, aquí Linux suele tener una ventaja sobre Windows; pues, *casi nunca se asiste de módulos del kernel para poder detectar los dispositivos físicos;* es decir, lo normal es que Linux por sí solo reconosca todo tipo de Hardware. En consecuencia, en Linux, es raro que se tenga que hacer *inserción de módulos* (esto es lo ideal).

**Pongamos esto en contexto.** Por ejemplo, usted con Linux, cuando intente conectar una memoria USB a su ordenador, casi siempre logrará ser detectada sin ningún problema; lo propio con una impresora o con cualquier otro dispositivo físico (componente no integrado), y así...

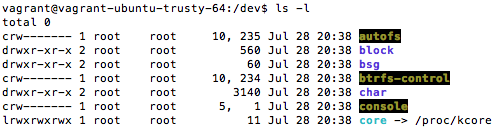
Ahora, en el caso que Linux no detecte el componente de Hardware, **tocaría trabajar con una serie de comandos** que son: **lsmod** (que nos hará un listado de los módulos que están cargados en el Kernel) y **modprobe** (que nos va a permitir insertar módulos nuevos al Kernel - también los elimina). Entonces, de esta manera, con estos dos comandos, cada vez que no se reconosca un componente, podemos hacer que Linux sea capaz de administrarlo.

**Navegando por la terminal...**

La idea es ir poniendo en práctica el manejo de la consola; en este caso, para navegar por medio de comandos a todo nuestro Hardware y, concretamente, identificar los ficheros *punteros* (enlaces simbólicos). Veamos.



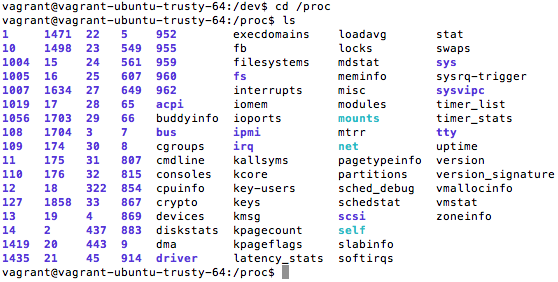
Enlistemos ahora todos los elementos en columnas para tener una mejor visualización, y con mayor contenido, de nuestro Hardware. *Por ejemplo:*



Resaltemos esto último: 

Pues bien, aquí *core* es un enlace simbólico que está apuntando (->) al archivo *kcore.* Recordemos, un *enlace simbólico* es, en resumidas cuentas, un acceso directo. Esos ficheros que apuntan (->) son justamente nuestros enlaces simbólicos. Profundicemos más.

Queremos saber si el archivo  existe en la ruta 

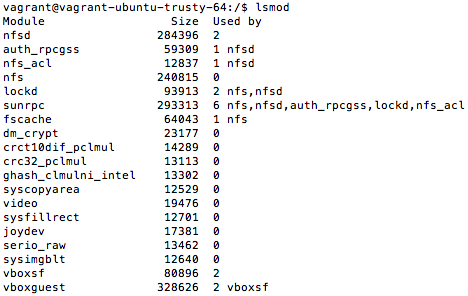


Efectivamente existe un tal *“kcore”*; entonces, cada vez que ejecutamos desde */dev,* el archivo *core,* realmente se está yendo hacia la ruta:  y ejecutando el archivo ; es eso lo que hace la función de *apuntar* desde un fichero A *(core)* a un fichero B *(kcore),...* se ejecuta fichero B. Es esta la aplicación del concepto *Acceso directo* que, en Linux, se conoce como: *Enlace Simbólico.* Fin de nuestra práctica en terminal.

**Sobre cómo configurar módulos nuevos en el Kernel (o incluso quitarlos)**

Recordemos que **lsmod,** que nos hace un listado de los módulos que están cargados en el Kernel y **modprobe**, que nos permite insertar módulos nuevos al Kernel - o también eliminarlos por medio del parametro u *opción* **“-r”**,... son los comandos que gestionan el acceso a la totalidad de los módulos en el Kernel (los selecciona, inserta y elimina). Veamos cómo se desenvuelve esto en consola.

***Enlistemos algunos módulos...***



***Ahora, a modo de práctica y siempre sobre un entorno de trabajo de prueba (maquina virtual),*** *vamos a eliminar un módulo al kernel y, posteriormente, lo volveremos a insertar.* El módulo de prueba a seleccionar será: , que es el módulo del kernel que le permite reconocer, al corazón del sistema operativo (kernel), el componente no integrado: *“Mouse”* o ratón. *psmouse* es quién reconoce el dispositivo *mouse.*

Observe que *psmouse* está entre *e1000* y *ahci:*



Veamos esta dinámica en consola: eliminar el módulo *psmouse* del kernel, tal que así:



Lo que a continuación vemos es un mensaje de “error” que tiene que ver con límites de acceso. Resulta que, la operación que pretendes ejecutar es *delicada* y sólo puede ser llevada a cabo por el administrador (en Linux conocido como Superusuario).



Esto devela que, en la terminal, ciertas acciones sólo pueden ser ejecutadas por el Superusuario (sudo); dicho lo anterior, para intentar ejecutar esta operación denegada, habría que ejecutarla como Superusuario, tal que así (eso sí, en ocasiones le pedirá una clave para acceder a la terminal como Supersuario o, al menos, ejecutar dicha acción como uno):



Se agrega un *sudo* al inicio antes de correr la línea de comando, y listo!

En efecto, ya no sale nuestro módulo *psmouse* enlistado en el Kernel (que salía entre los módulos *e1000* y *ahci*):



Ahora, suponiendo que necesitamos usar el *mouse* (tener acceso a él)nuevamente dentro de nuestra maquina virtual, podemos insertar el módulo suprimido *psmouse*, tal que así:



Y, de nuevo, aparece enlistado dentro de los módulos del Kernel (ahora sale de primero por ser el más reciente):

