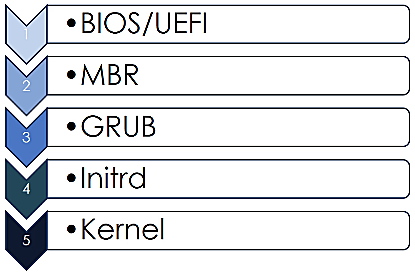
**Sobre el proceso de Arranque de Linux**

Cómo se produce, qué secuencia tiene, etc,...

**A continuación la secuencia de Arranque de un Sistema Operativo Linux:**



**1. Se inicia la BIOS o UEFI (depende de la que tenga seleccionada para trabajar).**

**2. Cargar el MBR (Master Boot Record).**

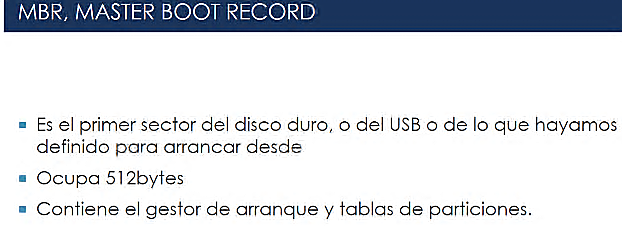
**3. Cargar el GRUB, un gestor de arranque.**

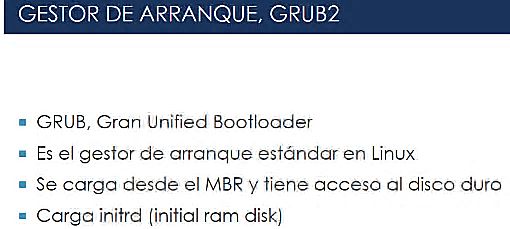
**4. Arranca el Initrd (Initial RAM Disk).**

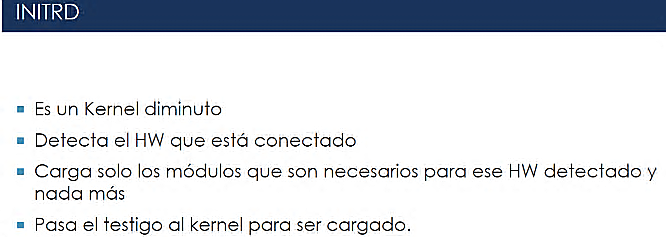
**5. Arranca el Kernel con sus módulos, iniciados por Init; y así, en definitiva, se carga nuestro Linux.**

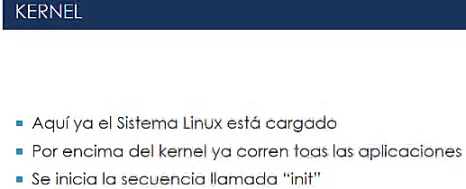
**A modo de resumen:**

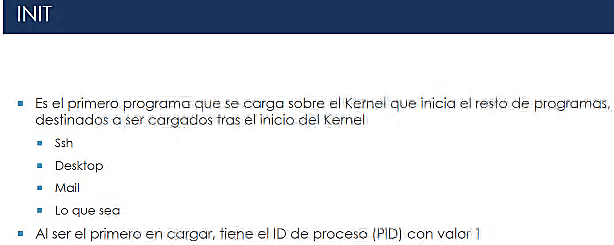






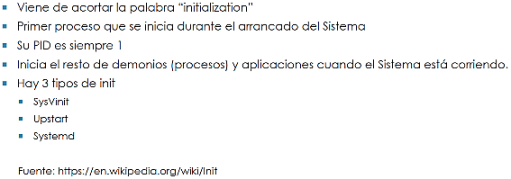






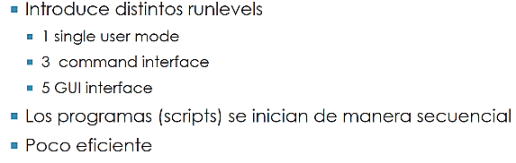
**INIT a profundidad**

**Qué es INIT?**



***Init*** es el primer programa ejecutado; por lo cual, su PID siempre será 1.

**Qué es SYSVINIT?**



Nos dará la facultad de elegir e inicializar nuestros programas en diferentes niveles de ejecución ***(runlevels).***

***Nivel 1:*** Single user mode.

***Nivel 2:*** Command interface (Interfaz solo de comandos).

***Nivel 3:*** GUI interface.

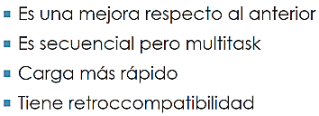
***Hay más niveles, mas éstos son los más usados.***

Adicionalmente, hay que decir que los programas *o scripts* (archivos ejecutables) tienen una enumeración a la hora de ser cargados; es decir, *se inician de manera secuencial* y se ejecutan en orden según de menor a mayor enumeración.En otras palabras, cuánto más baja es la enumaración del programa en cuestión, mayor es la prioridad a la hora de ser inicializado; y, hasta que no finalice la ejecución de un programa, no comienza el programa inmediatamente ulterior.

Lo lamentable es que **SysVinit** es muy poco eficiente por su propia naturaleza y puede dilatar mucho el arranque. Por ejemplo, pongamos el siguiente caso: Si suponemos que tenemos un problema en el programa número 14 y queremos solucionarlo; para solucionarlo, tendríamos que esperar entonces a que todos los programas *o procesos* que van desde el número 1 hasta el número 13 se ejecuten y se cierren, ¿no es eficiente, cierto?

Pues bien, para solucionar ese tipo de problema de eficiencia, la familia Ubuntu ha optado por incorporar otro tipo de init: ***Upstart.***

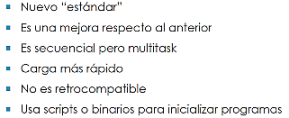
**Qué es Upstart?**



La gran diferencia de ***Upstart*** con relación a ***SysVinit*** es que puede inicializar más de una aplicación a la vez; esto si, entre ellas, no hay interferencias (multitask); en todo caso, su carga es más rápida.

Por último, tenemos el último tipo de init: ***Systemd***

**Qué es Systemd?**



La diferencia de ***Systemd*** con relación a los otros dos tipos de INIT es que ***Systemd*** también puede ejecutar o inicializar su programas a base de binarios (y no sólo con ***Scripts***). Aparte de que acá hay una mayor gestión a la hora de inicializar los procesos.

**Run Levels (Niveles de ejecución)**

**Distintos niveles de ejecución SYSVINIT (7 en concreto).**

**0. Halt:** *Cuando lo ejecutamos le notificamos a nuestro Hardware que detenga todas las funciones de la CPU; por lo que, va a mantener a nuestro servidor o PC en una especie de “estado de coma”. Él seguirá encendido, pero sólamente podrá ejecutar tareas de mantenimiento muy básicas.*

**1. Single user mode:** *Siendo un chico “malo”; con él, por ejemplo, podríamos resetear la contraseña de root; ahora, siendo un chico“bueno” (todo depende de cómo lo vea y bajo qué circunstancias), podríamos resetear la contraseña de root en caso que nosotros mismos seamos el administrador de sistemas y realmente se nos haya olvidado, con genuinidad, nuestra contraseña.*

**2. Multiusuario sin red:** *Esto quiere decir que vamos a poder cargar nuestro sistema operativo sin necesidad de una red. Y de paso, con Multiusuario, aún te puedes logear con muchos usuarios más (por eso se llama multiusuario)*

**3. Multiusuario con red:** *Es exactamente lo mismo que Multiusuario con red. Esto quiere decir que vamos a poder cargar nuestro sistema operativo; pero, ahora, con una red.*

**4. No tiene uso**

**5. Multiuser con GUI:** *Graphic User Interface (GUI), carga lo que sería nuestro entorno de escritorio: escritorios, ventanas, iconos, etc.*

**6. Reiniciar:** *El PC o servidor se va a reiniciar.*

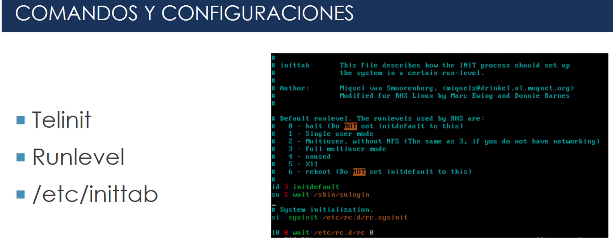
**Cada familia de Linux tiene unos niveles de ejecución diferentes.**

**Es, por ejemplo, diferente los niveles de ejecución de la familia Debian/Ubuntu si los comparamos con la familia Centos/Suse.**





**Ahora, necesitamos saber sobre cómo interactuar con los niveles de ejecución (Run Levels) en Consola**



Al ejecutar **Telinit,** por ejemplo, vamos a poder cambiar entre los distintos niveles de ejecución; sin embargo, para poder correr este comando, necesitamos permisos de *root.*

Al ejecutar el 2do comando, **Runlevel,** se nos dice en qué nivel de ejecución estamos. Al igual, el mismo comando, nos refleja cuál era el anterior nivel de ejecución; es decir, cuál era el nivel de ejecución en el que estabamos antes de haber hecho el cambio.

Adicionalmente, algo para agregar: Muchas veces el resultado que arroja este comando, al principio, refleja una N y luego un número (#); pues bien, la ***N*** significa ***None*** (Ninguno) y el significado de ***None*** nos está diciendo que no habíamos corrido ningún nivel de ejecución antes de hacer el cambio; es decir, estaríamos corriendo realmente nuestro primer cambio de nivel de ejecución, y por eso el resultado arroja *N* en ocasiones. Y si por ejemplo aparece en los números un: ***5*** y luego ***3,*** quiere decir entonces que, anteriormente estabamos en el nivel de ejecución ***5) Multiuser con GUI*** y que ahora estamos en el nivel de ejecución ***3) Multiusuario con red*** (para saber de dónde vinimos y hacía dónde vamos).

**Práctica: Run Levels (Niveles de ejecución)**

Antes de poner en práctica los comandos del ***run levels;*** nos interesaría saber, primero que todo, a qué familia o versión de Linux pertenece nuestro PC; para en consecuencia saber, con cierta precisión, cuántos niveles de ejecución *(run levels)*tenemos. El comando que nos ayuda a saber exactamente a qué familia de Linux pertenecemos es el siguiente: ***cat /etc/os-release.***



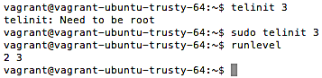
*Al parecer, somos de la familia* ***Debian/Ubuntu;*** *si estamos en lo cierto, entonces, tenemos 4 niveles de ejecución: 0, 1, 2 & 6.*

Ahora nos interesa saber en qué ***run level*** estamos al ejecutar el comando ***runlevel,*** veamos:



***N*** significa que “venimos” del *run level* ***none;*** es decir, como lo explicamos antes, realmente no venimos de ningún *run level* porque no hemos cambiado; o sea, estamos sobre nuestro primer *run level* posicionado; el cual, es ***2.***

Entonces, para ***cambiar*** de *run level,* tenemos que usar el comando ***telinit*** y, justo al lado, pasarle el número correspondiente al *run level* que queremos visitar ahora. Probemos lo anterior al cambiarnos al *run level* 3, tal que así:



En efecto, **cambiamos al run level 3 viniendo,** anteriormente, **del run level 2. Ahora, lo ideal es que al haber cambiado al *run level 3* (Multiusuario con red)*;* cambie, también, de interfaz gráfica: pasar a ser de *usuario* a *interfaz de línea de comando.***

Ahora, probemos con ***telinit 6.***



Si bien recuerda, al cambiar al ***run level 6*** se nos reinicia todo (puede observarlo); en este caso, como estabamos trabajando dentro de una maquina virtual, ella se nos cerró. Y, de hecho, si intenta saber en que *run level* está dentro de la maquina virtual,... no encontrará información, pues, ya no estamos dentro de la maquina como tal: fue cerrada completamente; por lo cual, se perdió todo el registro de los comandos que ahí habiamos puesto en marcha y, de hecho, se restauró de todo el sistema dentro de la misma maquina virtual en sí (como si no hubieramos ejecutado ningún comando ahí, puede probar ejecutando de nuevo el comando *runlevel* y verá que estará en el *run level* que por defecto tiene su PC -o maquina virtual en mi caso).

Ante todo, toca activar de nuevo nuestro entorno de trabajo *(maquina virtual).*

Ahora, si realmente nos queremos asegurar de cuántos *run levels* tenemos en nuestro dispositivo, toca ejecutar el comando: ***ls /etc/rc,*** lo que hará es enlistarnos todos los ficheros del directorio ***“rc”;*** dichos ficheros serían, entonces, la totalidad de nuestros ***run levels.*** Veamos:



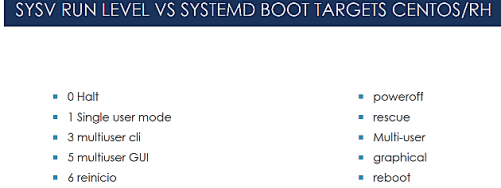
En nuestro caso, realmente los ficheros *run levels* están dentro del directorio */etc;* es decir, toca enlistar solamente a */etc.* Por lo demas, contamos con todos los niveles del *run level* (del 0 al 6, todos).

Aparte, algo para aclarar: Si se da cuenta esos ficheros de *rc* son directorios; en los cuales se puede navegar, uno a uno, también; y, de paso, enlistar todo el contenido que tienen cada uno de ellos por separado.

**Boot Targets**

**Sobre qué son los *Boot Targets* y la relación que tienen con los *Run Levels***

*Boot Targets* es como el primo hermano de los *run levels* y, de hecho, hay equivalencia entre ambos; por ejemplo, observemos lo siguiente:



Mientras que en SYSVINIT (SYSV) teníamos números como niveles de ejecución, *run levels,...* Aquí tenemos *“palabrejas”* que podrían reemplazar a dichos número de los *run levels;* con lo cual, se nos hará la vida más *fácil* a la hora de elegir entre recordar los números de los *run levels* o, más bien, mejor, las palabrejas de los *boot targets*.

***Pongamos así:***

*0 Halt* es para los *run levels* como lo es *poweroff* para los *Boot Targets*

*1 Single user mode* es para los *run levels* como lo es *rescue* para los *Boot Targets*

*3 multiuser cli* es para los *run levels* como lo es *Multi-user* para los *Boot Targets*

*5 multiuser GUI* es para los *run levels* como lo es *graphical* para los *Boot Targets*

*6 reinicio* es para los *run levels* como lo es *reboot* para los *Boot Targets*

**Luego, si ahora en adelante vamos a usar los *Boot Targets, entonces ¿cómo cambiamos de Boot Target?* Sabemos cambiar de *run level*, pero, no sabemos hacer lo propio con los Boot Targets. Veamos cómo se procede.**

Acá nos vamos a asistir de la herramienta ***Systemctl*** para poder lograr el cambio entre Boot Targets (y de paso ejecutarlo), se tendría que correr el siguiente comando en consola: ***Systemctl isolate +*** nombre ***Boot Target*** destino; entonces, de la anterior forma, es como se ejecutaría cada uno de los posibles ***Boot targets*** que pasemos.

Es decir, pongamos el siguiente caso, por ejemplo: Si queremos que se reinicie el ordenador,... tendríamos que correr, en el caso de los ***Boot Targets,*** la siguiente sentencia: ***“systemctl isolate reboot”***

Ahora, para saber cuál es el ***Boot Target*** que tenemos definido por defecto en nuestro sistema,... tendríamos que *teclear* la siguiente sentencia: ***“systemctl get-default”.*** De la anterior manera es que se nos va a mostrar, por pantalla, cuál es el ***Boot target*** que tenemos configurado en nuestro sistema.

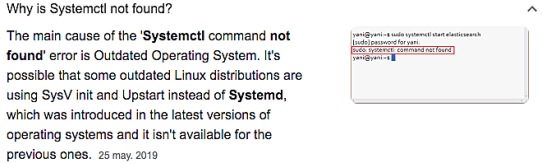
Ahora, en caso que queramos configurar a voluntad propia otro ***Boot target***, que sea el establecido por defecto cada vez que prendamos la PC (o en mi caso el servidor),... tendríamos que correr la siguiente sentencia en consola: ***Systemctl set-default +*** nombre ***Boot Target.*** Un ejemplo: **“*systemctl set-default graphical*”.**

**Path: /etc/system/system** (lugar donde se alojan los *boot targets*).

**Práctica: Boot Targets**

Lo primero que hay que hacer, ya estando en consola, es saber qué ***Target*** es el que tenemos por defecto cada vez que se inicializa el sistema; es decir, habría que ejecutar: ***systemctl get-default.*** Veamos: ...

Parece que no funciona, mas, estas son las razones:



***Ahora, si le funciona, podría considerar hacer lo siguiente:***

