Qué es un usuario Root en Linux?

Es el usuario que tiene acceso administrativo a todo el sistema y puede realizar lo que desee. Este usuario tiene poder y control absoluto sobre el equipo y puede hacer y deshacer a su antojo, sin ningún tipo de limitación más allá de sus conocimientos. Este tipo de usuario es, si lo comparamos con Windows, el usuario administrador, aunque en Linux tenemos mucho más poder de decisión que en Windows.

No es recomendable iniciar sesión con permisos root, si no tenemos muy claro que es lo que estamos haciendo, si no queremos **poner en riesgo la estabilidad** del sistema si realizamos algún cambio.

¿Por qué Ubuntu no me deja establecer la contraseña durante la instalación?

Por precaución y por un tema de protección el sistema Ubuntu evita ceder el control o el inicio de sesión de forma directa, como usuario root o super usuario.

En la distribución de Ubuntu, el sistema está configurado de forma predeterminada para no permitir el inicio de sesión directo como usuario root. En su lugar, Ubuntu utiliza el comando "sudo" para permitir a los usuarios autorizados ejecutar comandos con privilegios de superusuario temporalmente.

Tenemos algo llamado el mecanismo sudo para manejar eso. En su lugar, se añaden usuarios a la cuenta `admin` cuenta. Todos esos usuarios pueden entonces ejecutar comandos o programas como root ejecutando `sudo command` para los comandos de terminal o `gksu command` para que las aplicaciones GUI se ejecuten como root, como `gksu gcalctool` aunque ese ejemplo obviamente no tiene sentido)

Cuando te pidan una contraseña al instalar cosas, etc., es tu propia contraseña la que debes usar. De esta manera, es posible permitir a otros hacer tareas administrativas sin tener que compartir contraseñas y claves. También es configurable para permitir que alguien ejecute un comando específico como root, pero no otros, pero normalmente no tocarás eso.

El primer usuario creado es por defecto la cuenta de administrador.

¿Cuáles son los procesos típicos de Linux?¿Cómo identificarlos?

Es importante saber lo que es un proceso para luego saber cuáles son esos procesos típicos.

Un proceso en Linux es una serie de instrucciones que vienen de un programa que está en ejecución, existen diferentes elementos que incorpora un proceso como la prioridad de ejecución del proceso que le indica a Linux cuanto CPU utilizar y el tiempo máximo de ejecución del proceso.

Cuando Linux se ejecuta, el kernel de Linux tiene la primera prioridad de ejecución, conocida como PID 1 (Process ID). En versiones anteriores de Linux, este proceso era conocido como **init** que está basado en la forma en la que sistemas antiguos de Unix arrancaban el sistema.

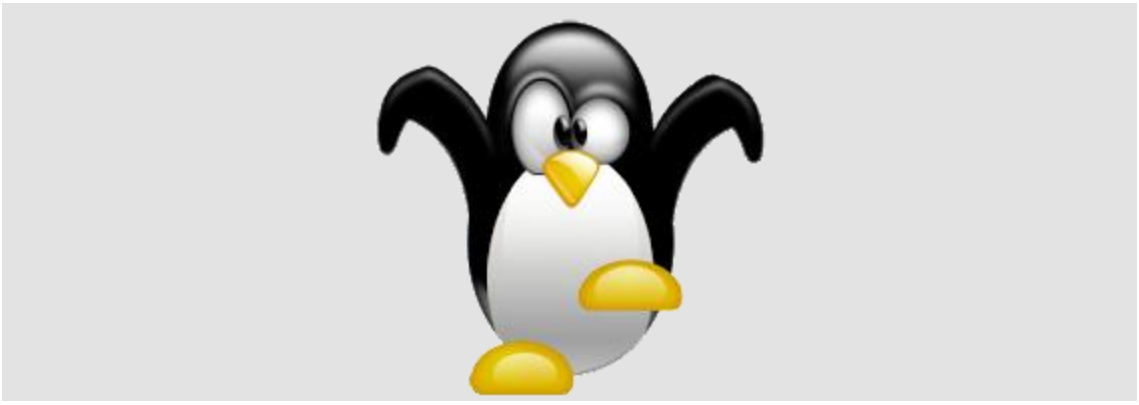
Las versiones modernas de Linux utilizan **systemd** que intenta coordinar la manera en que los procesos son manejados. Como comentamos PID1 es el proceso padre, todos los demás procesos ejecutados a partir de este son procesos hijos.

Dependiendo de la forma en que corren estos programas en LINUX se los puede clasificar en tres grandes categorías:

- Procesos Normales.
- Procesos Daemon.
- Procesos Zombie

Los procesos en GNU/Linux son organizados de forma jerárquica, cada proceso es lanzado por un proceso padre y es denominado proceso hijo. De esta forma, todos los procesos en GNU/Linux son hijos de init ya que este es el primer proceso que se ejecuta al iniciar el ordenador y init es padre de todos los procesos. Si se mata al proceso padre, también desaparecerán los procesos hijos.

Procesos Normales



Los procesos de tipo normal generalmente son lanzados en una terminal (tty) Y corren a nombre de un usuario, es decir, son los programas que utiliza el usuario generalmente y se encuentran conectados a una terminal. El programa aparecerá el pantalla e interactuará con el usuario.

Proceso Daemon

Los procesos de tipo Daemon corren a nombre de un usuario y no tienen salida directa por una terminal, es decir corren en 2° plano. Generalmente los conocemos como servicios. La gran mayoría de ellos en vez de usar la terminal para escuchar un requerimiento lo hacen a través de un puerto.

Proceso Zombie

En sistemas operativos Unix un proceso zombie es un proceso que ha completado su ejecución, pero aún tiene una entrada en la tabla de procesos. Esto se debe a que dicho proceso (proceso hijo) no recibió una señal por parte del proceso de nivel superior (proceso padre) que lo creó informándole que su vida útil ha terminado. Se pueden deber a errores de programación, a situaciones no contempladas por el programador y generalmente provocan lentitud y/o inestabilidad en el Sistema.

Los principales estados en los que pueden encontrarse los procesos en Linux/Unix son los siguientes:

running (R) : Procesos que están en ejecución.
sleeping (S) : Procesos que están esperando su turno para ejecutarse.
stopped (D) : Procesos que esperan a que se finalice alguna operación de Entrada/Salida.
zombie (Z) : Procesos que han terminado pero que siguen apareciendo en la tabla de procesos

¿Cómo identificarlos?

Existen varias herramientas para ver los procesos en ejecución, la más importante es el comando ps

`ps` (*process status*)

Lista los procesos con su PID, datos de usuario, tiempo, identificador del proceso y linea de comandos usada

```
$ ps
  PID TTY          TIME CMD
 6368 pts/0    00:00:00 bash
 7441 pts/0    00:00:00 ps
```

sin opciones, ps sólo muestra los procesos lanzados desde el terminal actual y con el mismo EUID que el usuario que lo lanzó

Opciones de ps

ps tiene un gran número de opciones, que se pueden especificar de 3 maneras:

1. opciones UNIX: pueden agruparse y se preceden por un guión: `ps -ef`
2. opciones BSD: pueden agruparse y van sin guión: `ps uxa`
3. opciones largas GNU: precedidas de dos guiones: `ps --user tomas`

Algunas opciones:

- `-e` o `ax`: muestra todos los procesos
- `-u` (o `U` o `--user`) *usuario*: muestra los procesos de un usuario
- `u`: salida en formato usuario
- `j`: salida en formato *job* (muestra PID, PPID, etc.)
- `-f` o `l`: salida en formato largo
- `f`: muestra un árbol con la jerarquía de procesos
- `k` (o `--sort`) *campo*: ordena la salida por algún campo (p.e. `ps uxak rss`)
- `-o` (o `o` o `--format`) *formato*: permite definir el formato de salida `ps -o ruser,pid,comm=Comando`

Ejemplo:

```
$ ps axu
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
COMMAND
root         1  0.0  0.0   1516   536 ?        S    09:43   0:00 init
[2]
root         2  0.0  0.0      0     0 ?        S    09:43   0:00
[migration/0]
root         3  0.0  0.0      0     0 ?        SN   09:43   0:00
[ksoftirqd/0]
root         4  0.0  0.0      0     0 ?        S    09:43   0:00
[migration/1]
.............
tomas     6475  0.1  4.9 140180 50920 ?        Sl   09:51   0:18
/usr/lib/mozilla-thunderbird/mozilla-thunderbird-bin
tomas     6528  0.2  3.6 116396 37948 ?        Sl   10:01   0:25
/usr/lib/mozilla-firefox/firefox-bin -a firefox
```

En este ejemplo:

- `%CPU y %MEM`: porcentajes de uso de CPU y memoria
- `VSZ`: memoria virtual del proceso, en KBytes
- `RSS`: tamaño de la memoria residente (*resident set size*) en KBytes
- `STAT`: estado del proceso; puede ser:

Código	Significado
D	*Uninterruptible sleep* (usualmente IO)
R	Ejecutándose(*running*) o en cola de ejecución
S	*Interruptible sleep* (p.e. esperando un evento)
T	Detenido
Z	Proceso *zombie*

- Cuando se usa formato BSD puede aparecer otro código acompañando al principal:

Código	Significado
<	alta prioridad
N	baja prioridad
L	páginas bloqueadas (*locked*) en memoria
S	líder de sesión
L	multi-threaded
+	proceso en foreground

pstree

Muestra el árbol de procesos (similar a ps f)

```
    init-+-acpid
         |-atd
         |-bonobo-activati
         |-clock-applet
         |-cron
         |-cupsd
         |-dbus-daemon-1
         |-dcopserver
         |-dirmngr
         |-2*[esd]
         |-events/0-+-aio/0
         |          |-ata/0
         |          |-ata/1
         |          |-kblockd/0
         |          |-khelper
         |          `-pdflush
         |-events/1-+-aio/1
         |          |-kacpid
         |          |-kblockd/1
         |          `-pdflush
         |-exim4
         |-famd
         |-firefox-bin---wvMime---ggv
         ...
```

top

`ps` da una versión estática de los procesos

`top` nos da una lista actualizada a intervalos

```
top - 17:34:08 up  7:50,  6 users,  load average: 0.12, 0.31, 0.27
Tasks: 111 total,   1 running, 110 sleeping,   0 stopped,   0 zombie
Cpu(s):  6.2% us,  2.0% sy,  0.0% ni, 91.0% id,  0.0% wa,  0.8% hi,
0.0% si
Mem:   1026564k total,   656504k used,   370060k free,    65748k
buffers
Swap:  2048248k total,        0k used,  2048248k free,   336608k
cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 6130 root      15   0 63692  48m 9704 S  8.7  4.9   8:03.34 XFree86
 6341 tomas     15   0 14692 8852 6968 S  4.3  0.9   1:55.13 metacity
 6349 tomas     16   0 32792  14m 9232 S  1.3  1.5   0:41.60 gnome-
terminal
 6019 tomas     15   0  7084 3184 1896 D  0.3  0.3   0:23.22 famd
 6401 tomas     15   0 16756 8280 6856 S  0.3  0.8   0:02.49
geyes_applet2
 6427 tomas     15   0 18288  10m 8112 S  0.3  1.0   0:09.04 wnck-
applet
 7115 tomas     15   0 26312  13m  11m S  0.3  1.4   0:00.61
kio_uiserver
 7390 tomas     15   0 45016  30m  18m S  0.3  3.0   0:38.69 kile
    1 root      16   0  1516  536  472 S  0.0  0.1   0:00.61 init
    2 root      RT   0     0    0    0 S  0.0  0.0   0:00.00
migration/0
    ......................
```

- en la cabecera nos muestra un resumen del estado del sistema
  - hora actual, tiempo que el sistema lleva encendido, el número de usuarios conectados y la carga media del sistema para los últimos 1, 5, y 15 minutos
  - número total de tareas y resumen por estado
  - estado de ocupación de la CPU y la memoria
- por defecto, los procesos se muestran ordenados por porcentaje de uso de CPU (los más costosos arriba)
- pulsando `h` mientras se ejecuta `top`, obtenemos una lista de comandos interactivos
- para salir, `q`

- Algunos campos de top
  - VIRT: Tamaño total del proceso (código, datos y librerías compartidas cargadas), VIRT=SWAP+RES
  - SWAP: Memoria que ha sido *swapped out* o que aún no ha sido cargada
  - RES: Memoria residente (RAM ocupada por el proceso)
  - CODE y DATA: Memoria ocupada por el código y datos (datos y pila, pero no librerías compartidas) del proceso
  - SHR: Memoria compartida (memoria que puede ser compartida con otros procesos)
  - P: Última CPU usada (SMP)
  - nFLT: Número de fallos de página para el proceso

strace

Muestra las llamadas al sistema realizadas por un proceso en ejecución

Ejemplo de un `strace` sobre un `top` en ejecución

```
$ strace top
gettimeofday({1195811866, 763977}, {4294967236, 0}) = 0
open("/proc/meminfo", O_RDONLY)         = 3
fstat64(3, {st_mode=S_IFREG|0444, st_size=0, ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0) = 0xb7f55000
read(3, "MemTotal:      2066348 kB\nMemFre"..., 1024) = 728
close(3)                                = 0
munmap(0xb7f55000, 4096)                = 0
open("/proc", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY) = 3
fstat64(3, {st_mode=S_IFDIR|0555, st_size=0, ...}) = 0
fcntl64(3, F_SETFD, FD_CLOEXEC)         = 0

getdents(3, /* 52 entries */, 1024)     = 1020
getdents(3, /* 64 entries */, 1024)     = 1024
stat64("/proc/1", {st_mode=S_IFDIR|0555, st_size=0, ...}) = 0
open("/proc/1/stat", O_RDONLY)          = 4
read(4, "1 (init) S 0 1 1 0 -1 4194560 44"..., 1023) = 185
```

```
close(4)
......
```

Ejecución en segundo plano

Por defecto, los comandos corren en primer plano (foreground): El shell espera a que termine el comando antes de aceptar uno nuevo

- para ejecutar un comando en segundo plano (background) hacerlo con &

```
$ sleep 10
$ sleep 10 &
```

- para terminar un proceso en foreground Ctrl-C
- para pausar un comando en foreground usar Ctrl-Z
  - `bg` pasa el proceso a background
  - `fg` lo devuelve a foreground
- Ejemplo:

```
$ sleep 20
Ctrl-Z
[3]+ Stopped         sleep 20
$ bg
[3]+ sleep 20 &
$ fg
sleep 20
```

- El comando `jobs` permite ver la lista de comandos (*jobs*) en background lanzados desde el shell, así como su estado (`fg` y `bg` pueden referirse a uno de los jobs)

```
$ gedit nada.txt &; sleep 100 &
$ jobs
[2]- Running         gedit nada.txt &
[3]+ Running         sleep 100 &
$ fg 3
sleep 100
Ctrl-Z
[3]+ Stopped         sleep 100
$ jobs
[2]- Running         gedit nada.txt &
[3]+ Stopped         sleep 100
$ bg 3
[3]+ sleep 100 &
$ jobs
[2]- Running         gedit nada.txt &
[3]+ Running         sleep 100 &
```

Investigar y establecer una contraseña para el usuario root.

¿Cómo agregar password (clave) a root en Ubuntu/Linux?

Cuando instalamos ubuntu por primera vez en nuestra computadora, establece una clave para root que es desconocida para todos. Root es el usuario que por default es el administrador del sistema. Muy raramente necesitamos hacer login como root, esto para instalar programas, modificar archivos del systema etc. El primer usuario que es creado a la hora de instalar nuestro sistema es el que tiene acceso a root.

Cuando queremos agregar una clave o password a root; vamos a necesitar que nosotros hagamos login con una cuenta que tenga acceso sudo (acceso a comandos de root).

Necesitamos abrir una terminal; para establecer una clave o password a root, y escribir el siguiente comandó:

```
sudo passwd root
```

Tendrás que escribir tu password actual de tu cuenta con acceso sudo. Después te pedirá crear la clave UNIX, la cual sera la clave de root. a la hora de escribir los passwords o claves no verás ningún carácter escribiéndose, por seguridad. De [Enter] para finalizar cada clave.

para hacer login con la nueva clave de root, usa el siguiente comandó.

```
su
```

para salirte del login como super usuario, escribe:

```
exit
```

Ten mucho cuidado cuando haces loging como root, pues puedes eliminar archivos importantes del sistema que podrían dejarlo inutilizable etc. una de las razones de ponerle una clave a root es por la razón de que tu computadora tenga que ser usada por más usuarios que deban tener permisos se super usuario root para modificar el sistema, o razones administrativas que eviten usar repetidamente el comando sudo.