

Administración de Sistemas en la Cloud

Administración básica de sistema operativo en Linux

Índice

Esquema	3
Ideas clave	4
5.1. Introducción y objetivos	4
5.2. Sistema de archivos	4
5.3. Usuarios y grupos	12
5.4. Procesos y servicios	18
5.5. Herramientas de sistema	24
5.6. Referencias bibliográficas	37
A fondo	38
Test	39

ADMINISTRACIÓN BÁSICA DE SISTEMA OPERATIVO EN LINUX

SISTEMA DE ARCHIVOS

- ▶ El primer nivel de directorios es relativamente estándar y el contenido de cada uno es similar en cualquier distribución Linux.
- ▶ Cualquier cosa es un archivo y está representado en el sistema de archivos: directorios, dispositivos, *sockets*, etc.
- ▶ La asignación de permisos se aplica por usuario, grupo y otros.

USUARIOS Y GRUPOS

- ▶ Los usuarios pertenecen a un grupo principal y cero o varios grupos secundarios.
- ▶ Cada usuario puede tener una carpeta *home*.
- ▶ Cada usuario es asignado a una *shell* para inicio de sesión, aunque puede ser una *shell* especial que no admite *login*.

PROCESOS Y SERVICIOS

- ▶ Cada proceso tiene asignado un PID (controlador proporcional, integral y derivativo).
- ▶ Los servicios son procesos que no terminan inmediatamente y ofrecen una funcionalidad al sistema operativo o al usuario.
- ▶ *Kill* sirve para enviar señales a los procesos; algunas señales son de obligado cumplimiento.
- ▶ *Systemd* es un gestor de arranque de procesos que permite administrar servicios.

Esquema

5.1. Introducción y objetivos

Este tema presentará algunos de los conceptos básicos de la administración de sistemas Linux: sistemas de archivos, usuarios, grupos, etc. Es probable que cualquier instalación de *software* requiera modificaciones y tareas sobre estos objetos. La soltura con la línea de comandos será de ayuda para poder entender los ejemplos de este tema.

Los **objetivos** que se pretenden conseguir son:

- ▶ Conocer los fundamentos de los conceptos básicos de Linux que aparecen de manera recurrente en muchas tareas administrativas.
- ▶ Aprender a manipular los objetos del sistema operativo para automatizar tareas sobre estos.

A continuación, en el vídeo *Administración de usuarios y procesos en Linux*, se verá una demostración de creación, mantenimiento y borrado de usuarios, asignación de permisos.



Accede al vídeo

5.2. Sistema de archivos

Se suele decir que cualquier cosa en Linux es un archivo. Los **archivos** normales no son más que un tipo más de archivo y los **directorios** son archivos que contienen los

nombres de otros archivos. El comando `pwd` ofrece información sobre el directorio actual en la consola. El comando `cd` permite cambiar de directorio.

```
~$ pwd
/home/ubuntu
~$ cd /var/log
/var/log$
```

Árbol de directorios

El directorio raíz se identifica por la barra oblicua (`/`). Este directorio es la base del árbol de directorios. Al contrario que Windows, todo el sistema operativo de **Linux depende de un único árbol**. No hay unidades de disco C: o D:, como en Windows. En Linux, todas las particiones, unidades y almacenamiento se alojan en algún punto bajo la raíz `/`. El almacenamiento y las unidades externas se «montan» en un directorio. Este directorio aparece como una carpeta más del árbol. Por ejemplo, la ruta `/media` suele alojar las unidades del CD (disco duro) y las unidades de USB externas, una vez montadas.

El directorio raíz es el inicio de cualquier ruta, o *path*, absoluta. Es posible cambiar de directorio usando **rutas absolutas** desde cualquier directorio. Por ejemplo, `cd /var/log` funcionará aunque la consola se encuentre en `/home/ubuntu`. Las **rutas relativas**, sin embargo, no parten de la raíz, sino del directorio en el que se encuentra la consola en ese momento. Si el directorio actual es `/var`, tanto `cd /var/log` como `cd log` cambiarán al mismo directorio. El símbolo «`..`» (dos puntos) denota el directorio padre al actual, mientras que «`.`» (un punto) denota al directorio actual. Ambos pueden usarse en rutas relativas.

```
~ $ cd /var/log
/var/log $ cd log
-bash: cd: log: No such file or directory
/var/log $ cd ..
/var $ cd log
/var/log $ cd ./cups
```

/var/log/cups \$

La mayoría de las distribuciones de Linux siguen un **mismo esquema** de árbol de directorios. Aunque hay diferencias, rutas como /bin, /boot, /etc, /home o /tmp son prácticamente una constante en cualquier Linux. La Tabla 1 lista los directorios habituales de la raíz y su contenido.

Directorios habituales de Linux	
/bin	Comandos y binarios de usuario
/boot	Ficheros del gestor de arranque
/dev	Archivos de dispositivos
/etc	Ficheros de configuración
/home	Carpetas <i>home</i> de usuarios
/lib	Librerías compartidas y módulos del <i>kernel</i>
/media	Punto de montaje habitual de medios externos
/mnt	Otro punto de montaje habitual de medios externos
/opt	<i>Software</i> adicional instalado en el sistema
/proc	Detalles del núcleo y de los procesos, almacenados en modo texto
/root	Carpeta <i>home</i> del usuario <i>root</i>
/run	Datos de aplicaciones en ejecución
/sbin	Binarios de sistema
/srv	Datos de servicios provistos por el equipo
/sys	Sistema de ficheros virtual con información del <i>kernel</i>
/tmp	Ficheros temporales
/usr	Utilidades de usuario
/var	Datos transitorios o variables: <i>logs</i> , colas de correo, trabajos de impresión, etc.

Tabla 1. Directorios habituales de Linux. Fuente: elaboración propia.

Para examinar el contenido de los directorios, se puede usar el ya conocido comando `ls`. Ejecutando `ls -la` en la *home* de un usuario, se obtiene la siguiente información (Figura 1).

```
1. ubuntu@ubuntu-VirtualBox: ~  
~ $ ls -la  
total 112  
drwxr-xr-x 16 ubuntu ubuntu 4096 May  2 21:22 .  
drwxr-xr-x  3 root   root   4096 Apr 22 12:51 ..  
-rw----- 1 ubuntu ubuntu 9619 Apr 30 13:22 .bash_history  
-rw-r--r-- 1 ubuntu ubuntu  220 Apr 22 12:51 .bash_logout  
-rw-r--r-- 1 ubuntu ubuntu 3771 Apr 22 12:51 .bashrc  
drwx----- 13 ubuntu ubuntu 4096 May  2 20:06 .cache  
drwx----- 12 ubuntu ubuntu 4096 Apr 28 18:05 .config  
drwx----- 3 root   root   4096 Apr 28 18:08 .dbus  
drwx----- 3 ubuntu ubuntu 4096 Apr 23 11:51 .gnupg  
-rw----- 1 root   root    50 Apr 30 13:05 .lessht  
drwx----- 3 ubuntu ubuntu 4096 Apr 22 12:55 .local  
-rw-r--r-- 1 ubuntu ubuntu  807 Apr 22 12:51 .profile  
-rw----- 1 ubuntu ubuntu  119 Apr 22 19:19 .python_history  
drwx----- 2 ubuntu ubuntu 4096 Apr 22 13:13 .ssh  
-rw-r--r-- 1 ubuntu ubuntu    0 Apr 22 12:56 .sudo_as_admin_successful  
-rw----- 1 ubuntu ubuntu 9328 Apr 30 13:15 .viminfo  
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Desktop  
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Documents  
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Downloads  
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Music  
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Pictures  
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Public  
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Templates  
drwxr-xr-x  2 ubuntu ubuntu 4096 Apr 22 12:55 Videos  
lrwxrwxrwx 1 ubuntu ubuntu  11 May  2 21:22 a_file -> /tmp/a_file  
-rw-rw-r-- 1 ubuntu ubuntu 2151 Apr 30 13:15 install.sh  
~ $
```

Figura 1. Comando `ls` en la *home* del usuario. Fuente: elaboración propia.

Tipos de archivos

Los ficheros que comienzan con un punto son archivos ocultos. El comando `ls` no lo muestra por defecto, a menos que se invoque con el modificador `-a`. Son archivos normales a todos los efectos. Es habitual que las configuraciones específicas de ciertas aplicaciones se guarden como archivos ocultos en la *home*. Por ejemplo, `.viminfo` son detalles de `vim`, mientras que `.python_history` guarda el histórico de la *shell* de Python.

La primera columna de detalles muestra información sobre el tipo de archivo y los permisos. El primer carácter de la columna indica el tipo de archivo. La mayoría de los archivos de la Figura 1 tiene el *flag* de tipo vacío, con un guion (-). Esto significa que son archivos normales. La `d` significa que son directorios, mientras que la `l`

significa que el archivo es un enlace a otro archivo. La *c* y la *b* denotan dispositivos de bloque y de carácter, respectivamente, como los mostrados en la Figura 2. Otros tipos de archivos son los *sockets* (*s*, también en la Figura 2) y las tuberías (*p*, de *pipe*).

```

1. ubuntu@ubuntu-VirtualBox: ~
~ $ ll /dev/loop?? /dev/vcs? /var/run/snapd*
brw-rw---- 1 root disk 7, 10 May 2 20:05 /dev/loop10
brw-rw---- 1 root disk 7, 11 May 2 20:05 /dev/loop11
brw-rw---- 1 root disk 7, 12 May 2 20:05 /dev/loop12
brw-rw---- 1 root disk 7, 13 May 2 20:05 /dev/loop13
brw-rw---- 1 root disk 7, 14 May 2 20:05 /dev/loop14
brw-rw---- 1 root disk 7, 15 May 2 20:05 /dev/loop15
brw-rw---- 1 root disk 7, 16 May 2 20:05 /dev/loop16
brw-rw---- 1 root disk 7, 17 May 2 20:05 /dev/loop17
crw-rw---- 1 root tty 7, 1 May 2 20:05 /dev/vcs1
crw-rw---- 1 root tty 7, 2 May 2 20:05 /dev/vcs2
crw-rw---- 1 root tty 7, 3 May 2 20:05 /dev/vcs3
crw-rw---- 1 root tty 7, 4 May 2 20:05 /dev/vcs4
crw-rw---- 1 root tty 7, 5 May 2 20:05 /dev/vcs5
crw-rw---- 1 root tty 7, 6 May 2 20:05 /dev/vcs6
crw-rw---- 1 root tty 7, 128 May 2 20:05 /dev/vcsa
srw-rw-rw- 1 root root 0 May 2 20:05 /var/run/snapd-snap.socket=
srw-rw-rw- 1 root root 0 May 2 20:05 /var/run/snapd.socket=
~ $

```

Figura 2. Dispositivos como archivos en */dev* y *sockets* en */var/run*. Fuente: elaboración propia.

Sistema de permisos

Los permisos del archivo se indican con los últimos nueve caracteres de la primera columna. En Linux, los permisos se usan para determinar de qué acceso disponen los usuarios y grupos sobre un archivo. El control de permisos sobre archivos y aplicaciones es crítico para la seguridad y el correcto funcionamiento de un *host*. Un permiso erróneo puede suponer un agujero de seguridad (por ejemplo, si todas las subcarpetas de */home* permiten lectura al resto de usuarios en un servidor de FTP) o impedir que un servicio funcione correctamente (por ejemplo, si un servidor Nginx se ejecuta con el usuario *www-data*, pero la carpeta de archivos estáticos solo permite acceso al usuario *root*). Los tres permisos principales son:

- ▶ Lectura, indicado con el flag *r*.
- ▶ Escritura/edición, indicado con el flag *w*.

- Ejecución, indicado con el flag `x`.



Figura 3. Columna de tipo de archivo y permisos y su representación binaria y octal. Fuente: elaboración propia.

Como su propio nombre indica, el *flag* de lectura permite leer o ver un archivo normal o leer los nombres, pero no los detalles, del contenido de un directorio. El *flag* de escritura permite hacer cambios sobre un archivo o, si se trata de un directorio, permite crear, borrar y renombrar los ficheros del directorio. El *flag* de ejecución permite, efectivamente, la ejecución de un archivo. Los binarios de `/bin` deben tener el *flag* activo para que se puedan ejecutar. Un *script*, aunque sea un fichero de texto, puede tener el *flag* activo para poder ejecutarlo. Si el *flag* de ejecución se activa en un directorio, es posible «entrar» dentro del directorio usando el comando `cd`.

Los *flags* están agrupados en **tres clases**: usuario, grupo y otros. Los permisos de usuario aplican al usuario propietario del grupo. Los permisos de grupo describen los permisos que reciben los usuarios miembros del grupo propietario del archivo. El concepto de grupo propietario puede parecer chocante, pero está muy extendido en entornos Linux. La última clase describe los permisos de cualquier otro usuario.

El formato de los *flags* no es arbitrario, ya que se corresponde con la **representación binaria**, de forma que cada *flag* es un *bit*. Si el permiso está activo, el *bit* estará a uno. Una presentación muy habitual de los permisos es agrupar los tres *bits* de cada clase y mostrar la representación octal de esos *bits*. Así, según muestra la Figura 3, si los

tres *flags* están activos, la representación octal de 111 es 7, mientras que, si solo los *flags* de lectura y ejecución están activos, la representación octal de 101 es 5.

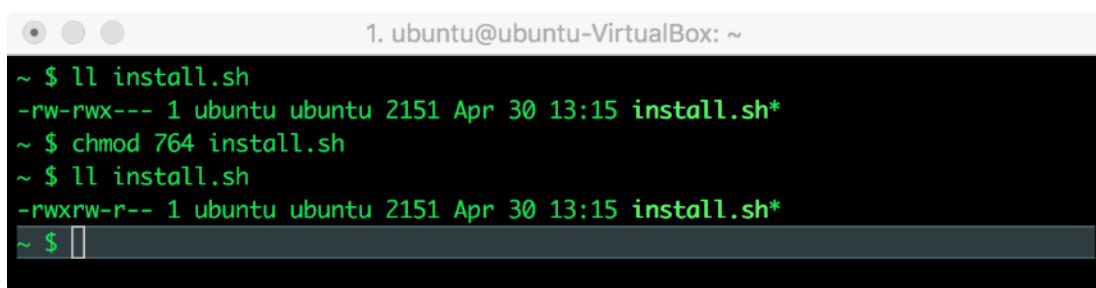
Este formato es muy útil para modificar la configuración de permisos de un archivo. El comando que permite cambiar los permisos es `chmod` y acepta dos sintaxis:

- ▶ Clase | activar/desactivar | permiso.
- ▶ Representación binaria.

Por ejemplo, dado un archivo con permisos `rw-rwx---`, es posible activar la ejecución para el usuario propietario, desactivar la ejecución para el grupo y activar la lectura para el resto con los dos comandos siguientes:

```
chmod u+x,g-x,o+r file
chmod 764 file
```

En el primer comando, cada permiso individual se activa o desactiva para cada clase. En el segundo comando, se usa la representación binaria de los *flags* para definir todos los permisos de una sola vez. La Figura 4 muestra los permisos antes y después de la ejecución de `chmod`.



```
1. ubuntu@ubuntu-VirtualBox: ~
~ $ ll install.sh
-rw-rwx--- 1 ubuntu ubuntu 2151 Apr 30 13:15 install.sh*
~ $ chmod 764 install.sh
~ $ ll install.sh
-rwxrw-r-- 1 ubuntu ubuntu 2151 Apr 30 13:15 install.sh*
~ $
```

Figura 4. Cambio de permisos en un archivo. Fuente: elaboración propia.

La Tabla 2 muestra algunos ejemplos de permisos junto a su representación en octal.

Permisos habituales	
600	rw-----
644	rw-r--r--
664	rw-rw-r--
666	rw-rw-rw-
755	rxwxr-xr-x
777	Rwxrwxrwx

Tabla 2. Ejemplos de permisos habituales en formato octal. Fuente: elaboración propia.

Enlaces

El fichero de la Figura 1 con el *flag* 1 es un enlace. Hay dos tipos de enlaces: enlaces fuertes y enlaces débiles o simbólicos. Los **enlaces fuertes**, o *hard links*, son referencias reales al archivo. Si se borran todos los enlaces fuertes a un archivo, el archivo se borra también. Un enlace fuerte solo se puede crear en la misma partición que el archivo al que referencia.

Los **enlaces simbólicos** se pueden borrar sin afectar al fichero al que referencian. Es habitual que los archivos binarios tengan enlaces simbólicos, para poder acceder a ellos con varios nombres o para poder cambiar la versión de un binario sin cambiar el nombre con el que se accede. Por ejemplo, la Figura 5 muestra los ejecutables de Python en la carpeta `/usr/bin`. La versión instalada de Python es la 3.6, tal como se indica en el binario `python3.6`. Sin embargo, es posible ejecutarlo tanto con `python` como con `python3`. Una actualización a Python 3.8 cambiaría estos enlaces simbólicos, para que apunten al nuevo binario, pero manteniendo los nombres `python` y `python3`.

```
1. ubuntu@ubuntu-VirtualBox: ~  
~ $ ll /usr/bin/python*  
lrwxrwxrwx 1 root root      9 May  2 22:43 /usr/bin/python -> python3.6*  
lrwxrwxrwx 1 root root      9 Apr 22 12:49 /usr/bin/python3 -> python3.6*  
-rwxr-xr-x 1 root root 4576440 Apr  1 2018 /usr/bin/python3.6*  
-rwxr-xr-x 1 root root 4576440 Apr  1 2018 /usr/bin/python3.6m*  
lrwxrwxrwx 1 root root     10 Apr 22 12:49 /usr/bin/python3m -> python3.6m*  
~ $
```

Figura 5. Enlaces simbólicos. Fuente: elaboración propia.

Los enlaces se crean con el comando `ln`. Los enlaces serán fuertes por defecto y simbólicos con el *flag* `-s`. La manera más sencilla de crear un enlace simbólico en la ruta actual es simplemente indicando la ruta del archivo enlazado (ver Figura 6).

```
1. ubuntu@ubuntu-VirtualBox: ~  
~ $ ln -s /usr/bin/python  
~ $ ll python*  
lrwxrwxrwx 1 ubuntu ubuntu 15 May  2 23:09 python -> /usr/bin/python*  
~ $ ./python --version  
Python 3.6.5  
~ $
```

Figura 6. Creación de enlace simbólico. Fuente: elaboración propia.

5.3. Usuarios y grupos

Linux es un sistema operativo **multiusuario** (Matotek et al., 2017). Esto significa que permite el inicio de sesión de múltiples usuarios simultáneamente, ya sea por la línea de comandos o en una sesión de escritorio. También hay usuarios específicos para ciertos componentes del sistema operativo. Por ejemplo, el servidor web Nginx suele ejecutarse bajo el usuario `nginx` o el usuario `www-data`, según cómo se haya instalado el paquete.

Linux también tiene el concepto de **grupo**. Los usuarios pueden pertenecer a uno o más grupos y suelen agregarse a grupos para dar permisos de acceso a ciertos recursos. Por ejemplo, una carpeta compartida ofrecida en un servidor FTP puede dar

acceso a un grupo concreto. Los miembros de ese grupo podrán leer los ficheros, facilitando la labor administrativa.

Por regla general, al crear un usuario, se crea una carpeta *home*, típicamente debajo de la ruta `/home`. Esta carpeta ofrece un lugar en el que los usuarios pueden guardar sus ficheros, además de ser la ubicación por defecto que muchas aplicaciones usan para guardar las configuraciones específicas de cada usuario. Por ejemplo, las claves para las conexiones SSH se guardan en la carpeta `~/.ssh` y el histórico de Bash se guarda en `~/.bash_profile`. Efectivamente, el símbolo «~» hace referencia a la carpeta *home* de un usuario, con independencia de la ruta absoluta de la misma. Un usuario Ubuntu que tenga su carpeta en `/home/ubuntu` podría crear un archivo tanto con `touch ~/new_file` como con `touch /home/ubuntu/new_file`.

Sudo

El comando `sudo` permite la ejecución de comandos administrativos a usuarios distintos de `root`. Las ventajas de Sudo son:

- ▶ Incrementa la seguridad.
- ▶ Permite más granularidad en el control de comandos administrativos.
- ▶ Incorpora auditoría de ejecución.
- ▶ Algunas distribuciones deshabilitan el usuario `root`, por lo que Sudo es la única opción.

Los comandos de creación y modificación de usuarios son ejemplos de comandos administrativos que solo el usuario `root` puede ejecutar. Las distribuciones que deshabilitan el usuario `root` habilitan el comando Sudo para todos los comandos para un usuario normal: Ubuntu lo habilita para el usuario que se crea durante la instalación y, en Amazon Linux, es el usuario `ec2-user` el que tiene permisos.

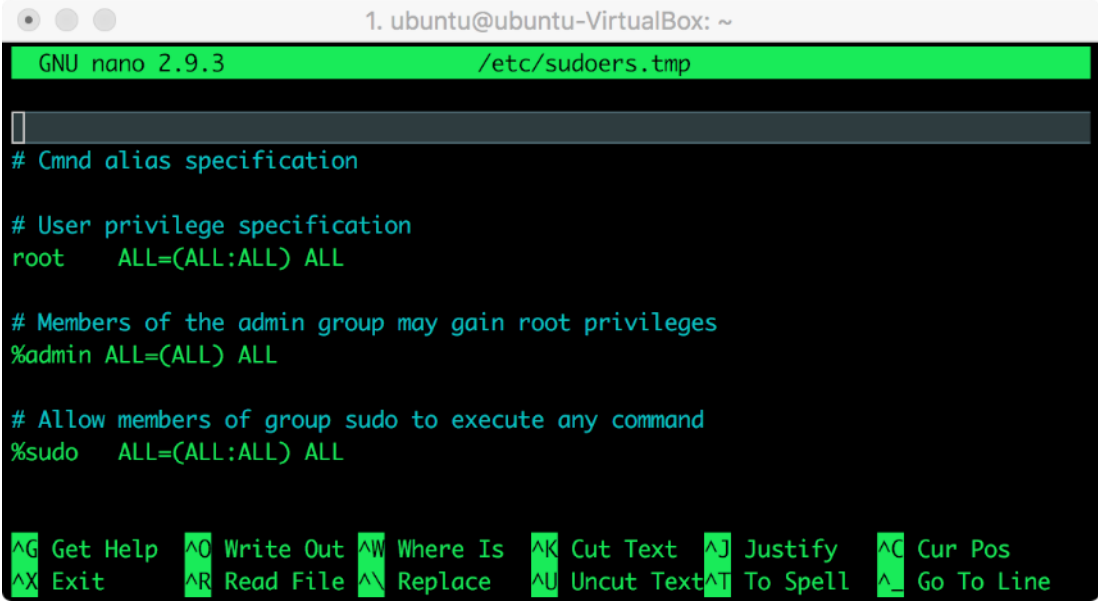
Para modificar permisos de Sudo, hay que editar el fichero `/etc/sudoers` con el comando especial `visudo`. Este comando también necesita privilegios, así que habría

que ejecutarlo con `sudo visudo`. Se abrirá el editor del sistema y se comprobará la sintaxis del fichero. Un fichero `/etc/sudoers` erróneo impedirá que se puedan ejecutar comandos administrativos y, por tanto, ni siquiera se podrá ejecutar `sudo visudo` para arreglar el fichero.

Según el contenido de `/etc/sudoers` de la Figura 7, hay dos formas de dar permisos de Sudo a un usuario:

- ▶ Añadiendo el usuario individualmente con una línea nueva, por ejemplo, `ubuntu ALL=(ALL:ALL) ALL`.
- ▶ Añadir el usuario como miembro de un grupo con permisos, como los grupos `admin` o `sudo`.

La sintaxis del fichero se puede consultar con `man sudoers`.



```
1. ubuntu@ubuntu-VirtualBox: ~
GNU nano 2.9.3 /etc/sudoers.tmp

# Cmnd alias specification

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

# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

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

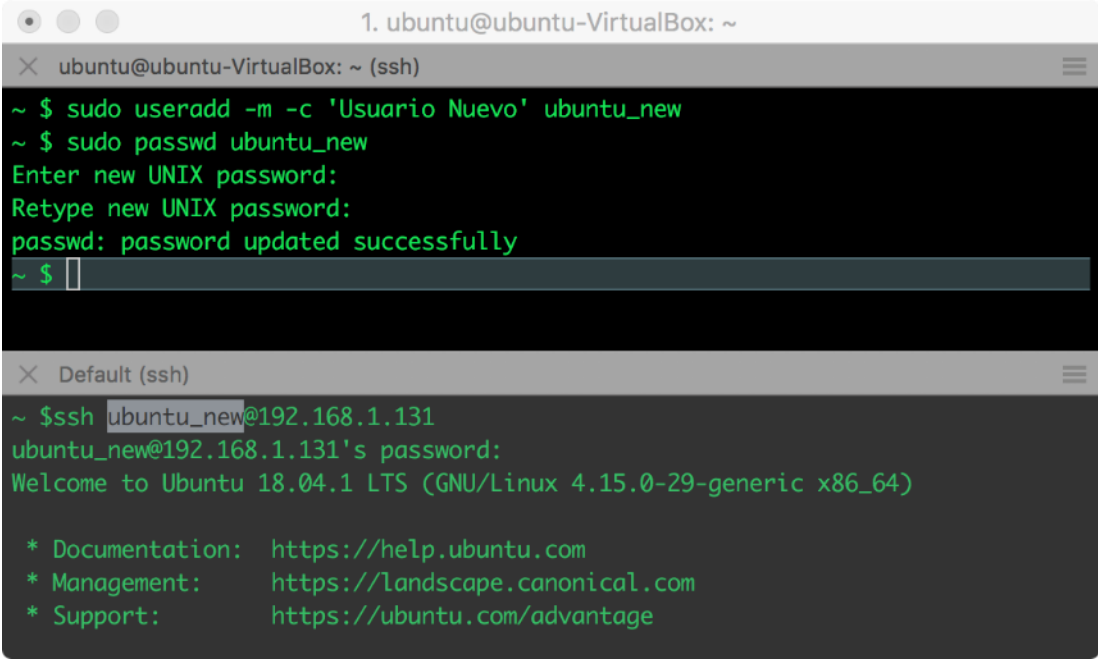
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^N Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

Figura 7. Editor nano tras ejecutar `sudo visudo`. Fuente: elaboración propia.

Administración de usuarios

El comando `useradd` permite crear usuarios nuevos. Durante la creación, se especifican el nombre, la descripción (con el *flag* `-c`), la *shell* por defecto del usuario

(con `-s`) y si se desea crear la *home* del usuario (con `-m`). El usuario estará desactivado y sin contraseña, que habrá que añadir con el comando `passwd`. La Figura 8 muestra la creación de un usuario nuevo, la fijación de la contraseña y el inicio de sesión con este nuevo usuario.



```
1. ubuntu@ubuntu-VirtualBox: ~
X ubuntu@ubuntu-VirtualBox: ~ (ssh)
~ $ sudo useradd -m -c 'Usuario Nuevo' ubuntu_new
~ $ sudo passwd ubuntu_new
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
~ $

X Default (ssh)
~ $ ssh ubuntu_new@192.168.1.131
ubuntu_new@192.168.1.131's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-29-generic x86_64)

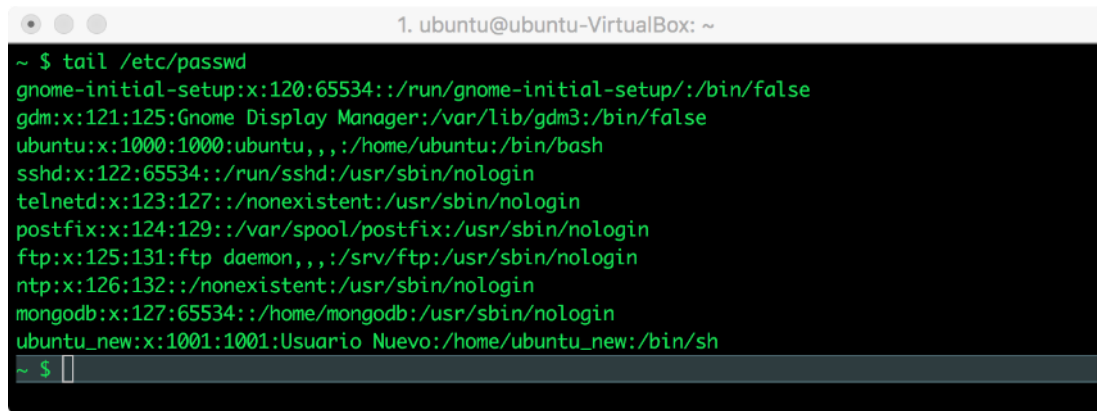
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage
```

Figura 8. Creación de usuario con `useradd`. Fuente: elaboración propia.

La información del nuevo usuario se almacena en dos ficheros, `/etc/passwd` y `/etc/shadow` (Frampton, s. f.). El primero contiene detalles del usuario: nombre de usuario, ID, ID de grupo, ruta de la *home* y ruta de la *shell* por defecto, entre otros.

Los ID son identificadores numéricos. Las utilidades de administración permiten que los usuarios trabajen con nombres de usuario y de grupo, aunque internamente el sistema trabaje con esos identificadores. La ruta de la *home* puede ser inválida (por ejemplo, `/nonexistent`) si el usuario no dispone de ella. De la misma manera, la ruta de la *shell* puede apuntar a `/usr/sbin/nologin`: esta es una *shell* especial que impide el inicio de sesión, pero registra el intento de inicio de sesión en los *logs*. Un usuario puede no recibir una *shell* en algunos casos. Un servidor FTP, por ejemplo, puede ofrecer acceso a sus usuarios exclusivamente por FTP, pero no por *shell* (se podría

deshabilitar el acceso por SSH completamente, pero esto cierra la puerta al acceso de administradores).

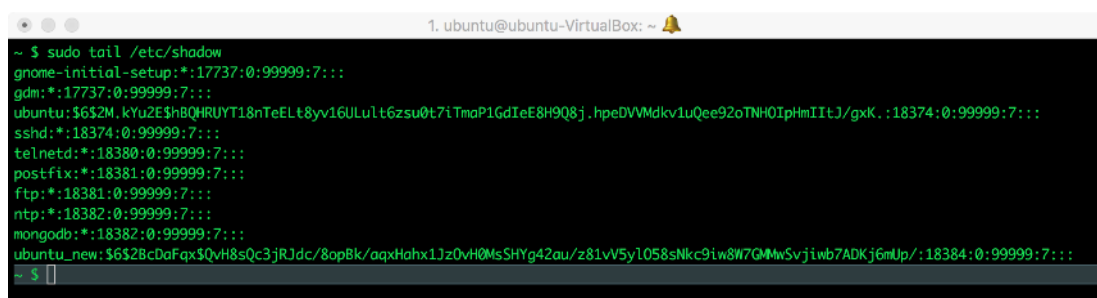
A terminal window titled '1. ubuntu@ubuntu-VirtualBox: ~' showing the output of the command 'tail /etc/passwd'. The output lists system and regular users with their hashed passwords (indicated by 'x') and shell locations. The users listed are gnome-initial-setup, gdm, ubuntu, sshd, telnetd, postfix, ftp, ntp, mongodb, and ubuntu_new.

```
~ $ tail /etc/passwd
gnome-initial-setup:x:120:65534:./run/gnome-initial-setup:/bin/false
gdm:x:121:125:Gnome Display Manager:/var/lib/gdm3:/bin/false
ubuntu:x:1000:1000:ubuntu,,,:/home/ubuntu:/bin/bash
sshd:x:122:65534:./run/sshd:/usr/sbin/nologin
telnetd:x:123:127:./nonexistent:/usr/sbin/nologin
postfix:x:124:129:./var/spool/postfix:/usr/sbin/nologin
ftp:x:125:131:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
ntp:x:126:132:./nonexistent:/usr/sbin/nologin
mongodb:x:127:65534:./home/mongodb:/usr/sbin/nologin
ubuntu_new:x:1001:1001:Usuario Nuevo:/home/ubuntu_new:/bin/sh
~ $
```

Figura 9. Fichero /etc/passwd. Fuente: elaboración propia.

El fichero /etc/passwd también contiene la contraseña, aunque, en el ejemplo de la Figura 9, todos los usuarios contienen una x en el campo reservado para la misma. Originalmente, las contraseñas se alojaban en ese campo como un *hash* unidireccional, pero este mecanismo era susceptible a ataques de fuerza bruta, especialmente si el fichero era robado, lo que no era imposible, porque el fichero debía ser legible por todos los usuarios.

Para intentar reducir el riesgo, las contraseñas se movieron a un segundo fichero, /etc/shadow, con un *hash* más complejo. Este fichero, además, solo es legible por el usuario root. La Figura 10 muestra el fichero /etc/shadow del mismo sistema que el fichero /etc/passwd de la Figura 9. Solo los usuarios ubuntu y ubuntu_new tienen contraseña.

A terminal window titled '1. ubuntu@ubuntu-VirtualBox: ~' showing the output of the command 'sudo tail /etc/shadow'. The output lists system users with empty password fields (indicated by ':') and regular users with their complex hashed passwords. The users listed are gnome-initial-setup, gdm, ubuntu, sshd, telnetd, postfix, ftp, ntp, mongodb, and ubuntu_new.

```
~ $ sudo tail /etc/shadow
gnome-initial-setup*:17737:0:99999:7:::
gdm*:17737:0:99999:7:::
ubuntu:$6$2M.kYu2E$hh8QHRUYT18nTeELt8yv16ULuLt6zsu0t7iTmaP1GdIeE8H9Q8j.hpeDVVMdkv1uQee92oTNH0IpHmIItJ/gxK.:18374:0:99999:7:::
sshd*:18374:0:99999:7:::
telnetd*:18380:0:99999:7:::
postfix*:18381:0:99999:7:::
ftp*:18381:0:99999:7:::
ntp*:18382:0:99999:7:::
mongodb*:18382:0:99999:7:::
ubuntu_new:$6$2BcDaFqx$QvH8sQc3jRJdc/8opBk/aqxHahx1Jz0vH0MsSHYg42au/z81vV5yL058sNkc9iw8W7QMMvSvjwb7ADKj6mUp/:18384:0:99999:7:::
~ $
```

Figura 10. Fichero /etc/shadow. Fuente: elaboración propia.

Administración de grupos

En Linux, cada usuario debe pertenecer al menos a un grupo. La mayoría de las distribuciones crean un nuevo grupo para cada nuevo usuario. Solo este usuario pertenecerá a este grupo, que será el grupo primario del usuario. Los demás grupos de los que sea miembro serán grupos suplementarios. El comando `id` muestra tanto el ID del usuario como los grupos a los que pertenece. En el siguiente ejemplo, el usuario `ubuntu` pertenece a `ubuntu` como grupo principal y a los grupos `cdrom` y `sudo`, entre otros, como suplementarios.

```
$ id ubuntu
uid=1000(ubuntu) gid=1000(ubuntu)
groups=1000(ubuntu),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),116(lpadmin),126(sambashare)
```

Los grupos suplementarios pueden definirse durante la creación del usuario:

```
$ sudo useradd -m -G sudo,cdrom ubuntu_sudo
$ id ubuntu_sudo
uid=1002(ubuntu_sudo) gid=1002(ubuntu_sudo)
groups=1002(ubuntu_sudo),24(cdrom),27(sudo)
```

También es posible modificar la membresía de grupos una vez creado el usuario con `usermod`.

```
$ sudo usermod -a -G plugdev ubuntu_sudo
$ id ubuntu_sudo
uid=1002(ubuntu_sudo) gid=1002(ubuntu_sudo)
groups=1002(ubuntu_sudo),24(cdrom),27(sudo),46(plugdev)
```

El comando `groupadd` permite crear grupos, mientras que `groupdel` los borra.

```
$ sudo groupadd new_group
$ sudo usermod -a -G new_group ubuntu_sudo
$ id ubuntu_sudo
```

```
uid=1002(ubuntu_sudo) gid=1002(ubuntu_sudo)
groups=1002(ubuntu_sudo),24(cdrom),27(sudo),46(plugdev),1003(new_group)
$ sudo groupdel new_group
```

5.4. Procesos y servicios

Al igual que otros sistemas operativos, en Linux existe el concepto de servicio, también llamado *daemon*. Los **servicios** son procesos que no terminan inmediatamente tras la ejecución, como el comando `pwd`, sino que se ejecutan continuamente, ofreciendo ciertas funcionalidades al sistema operativo o a los usuarios. Algunos ejemplos de servicios son, por ejemplo, el *daemon* de SSH, cuyo proceso se llama `sshd`, o el servidor web Apache, cuyo proceso es `httpd`. El comando `ps aux` muestra todos los procesos en ejecución, tanto servicios como otros procesos.

```
1. ubuntu@ubuntu-VirtualBox: ~
~ $ ps aux | grep -v '.*\['
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.4 160084  9212 ?        Ss   May02   0:02 /sbin/init splash
root       228  0.0  1.5 127908 31056 ?        Ss   May02   0:00 /lib/systemd/systemd-journald
root       238  0.0  0.2  46744  4744 ?        Ss   May02   0:00 /lib/systemd/systemd-udevd
systemd+  313  0.0  0.3  70744  6344 ?        Ss   May02   0:00 /lib/systemd/systemd-resolved
root       488  0.0  0.4  427256  8848 ?        Ssl  May02   0:00 /usr/sbin/ModemManager
syslog    492  0.0  0.2  263036  4208 ?        Ssl  May02   0:00 /usr/sbin/rsyslogd -n
root      502  0.0  0.1  38428  3340 ?        Ss   May02   0:00 /usr/sbin/cron -f
message+  505  0.0  0.2  51492  5632 ?        Ss   May02   0:00 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
root       526  0.0  0.9 580244 18872 ?        Ssl  May02   0:00 /usr/sbin/NetworkManager --no-daemon
root       527  0.0  0.2  44752  5116 ?        Ss   May02   0:00 /sbin/wpa_supplicant -u -s -O /run/wpa_supplicant
root       530  0.0  0.6 518004 13216 ?        Ssl  May02   0:00 /usr/lib/udisks2/udisksd
root       531  0.0  0.0   4552   752 ?        Ss   May02   0:00 /usr/sbin/acpid
root       535  0.0  0.8 177636 17308 ?        Ssl  May02   0:00 /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
root       541  0.0  0.2  70688  6036 ?        Ss   May02   0:00 /lib/systemd/systemd-logind
root       543  0.0  0.4 311004   9092 ?        Ssl  May02   0:00 /usr/lib/accountsservice/accounts-daemon
root       561  0.0  1.2 658696 25736 ?        Ssl  May02   0:03 /usr/lib/snapd/snapd
avahi     564  0.0  0.0   47076   336 ?        S   May02   0:00 avahi-daemon: chroot helper
root       596  0.0  0.5 311308 11304 ?        Ssl  May02   0:00 /usr/lib/policykit-1/polkitd --no-debug
root       650  0.0  0.1  28676  2812 ?        Ss   May02   0:00 /usr/sbin/vsftpd /etc/vsftpd.conf
root       653  0.0  0.2  72296  5680 ?        Ss   May02   0:00 /usr/sbin/sshd -D
root       667  0.0  0.4 308180  8664 ?        Ssl  May02   0:00 /usr/sbin/gdm3
whoopsie  729  0.0  0.6 466612 12808 ?        Ssl  May02   0:00 /usr/bin/whoopsie -f
root       730  0.0  0.1  33992  3232 ?        Ss   May02   0:00 /usr/sbin/inetd
kernelo+  751  0.0  0.0   56932   420 ?        Ss   May02   0:00 /usr/sbin/kerneloops --test
kernelo+  755  0.0  0.0   56932   416 ?        Ss   May02   0:00 /usr/sbin/kerneloops
ubuntu    786  0.0  0.4  77024  8188 ?        Ss   May02   0:00 /lib/systemd/systemd --user
ubuntu    795  0.0  0.1 114132  2704 ?        S   May02   0:00 (sd-pam)
ubuntu    959  0.0  0.3 288380  6688 ?        SLL  May02   0:00 /usr/bin/gnome-keyring-daemon --daemonize --login
ubuntu    965  0.0  0.3 212124  6144 tty1     Ssl+ May02   0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_SESSION_MODE=ubuntu gnome-session --session=ubuntu
ubuntu    974  0.0  3.4 409948 70408 tty1     Sl+  May02   0:02 /usr/lib/xorg/Xorg vt1 --displayfd 3 -auth /run/user/1000/gdm/Xauthority -b
```

Figura 11. Comando `ps aux`. Fuente: elaboración propia.

La lista en la Figura 11 muestra los procesos ordenados por PID, de *process ID*. El PID se usa para controlar e identificar procesos. El proceso con PID 1, `/sbin/init`, es el primer proceso que arranca y el que se encarga de arrancar otros procesos. El

proceso init era originalmente parte del sistema System-V. La mayoría de las distribuciones usan **Systemd** como gestor de arranque de procesos; en esos casos, el proceso con el PID 1 será /sbin/systemd. Ubuntu ha conservado el nombre del proceso init, aunque utiliza Systemd (Ubuntu usó [Upstart](#) como gestor hasta la versión 16.04).

El comando top es una herramienta de monitorización interactiva que muestra los procesos en funcionamiento en el equipo. Al contrario que ps, top se actualiza cada pocos segundos. Por defecto, top ordena los procesos por el uso instantáneo del CPU, por lo que los primeros procesos de la lista son los que más cargan el equipo.

```

top - 02:10:21 up 6:04, 3 users, load average: 0.16, 0.03, 0.01
Tasks: 182 total, 3 running, 148 sleeping, 0 stopped, 0 zombie
%Cpu(s): 61.1 us, 9.8 sy, 0.0 ni, 16.8 id, 0.4 wa, 0.0 hi, 11.9 si, 0.0 st
KiB Mem : 2041316 total, 367668 free, 709472 used, 964176 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used, 1150312 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR S %CPU  MEM   TIME+ COMMAND
 4769 _apt      20   0 69064 15412 6352 R 57.4  0.8   0:01.74 store
 4708 _apt      20   0 88904 8848 7908 R 21.5  0.4   0:00.67 http
 4703 root      20   0 62072 8392 6588 S  0.7  0.4   0:00.02 apt-get
  32 root      20   0      0      0   0 I  0.3  0.0   0:05.29 kworker/0:1
2378 ubuntu  20   0 110076 3580 2556 S  0.3  0.2   0:01.13 sshd
 4604 ubuntu  20   0 48884 4024 3336 R  0.3  0.2   0:00.18 top
 4692 ubuntu  20   0 110076 3628 2608 S  0.3  0.2   0:00.02 sshd
   1 root      20   0 160084 9212 6588 S  0.0  0.5   0:03.09 systemd
   2 root      20   0      0      0   0 S  0.0  0.0   0:00.00 kthreadd
   4 root      0 -20      0      0   0 I  0.0  0.0   0:00.00 kworker/0:0H
   6 root      0 -20      0      0   0 I  0.0  0.0   0:00.00 mm_percpu_wq
   7 root      20   0      0      0   0 S  0.0  0.0   0:00.29 ksoftirqd/0
   8 root      20   0      0      0   0 I  0.0  0.0   0:00.80 rcu_sched
   9 root      20   0      0      0   0 I  0.0  0.0   0:00.00 rcu_bh
  10 root      rt    0      0      0   0 S  0.0  0.0   0:00.00 migration/0
  11 root      rt    0      0      0   0 S  0.0  0.0   0:00.07 watchdog/0
  12 root      20   0      0      0   0 S  0.0  0.0   0:00.00 cpuhp/0
  13 root      20   0      0      0   0 S  0.0  0.0   0:00.00 kdevtmpfs
  14 root      0 -20      0      0   0 I  0.0  0.0   0:00.00 netns
  15 root      20   0      0      0   0 S  0.0  0.0   0:00.00 rcu_tasks_kthre
  16 root      20   0      0      0   0 S  0.0  0.0   0:00.00 kauditd
  17 root      20   0      0      0   0 S  0.0  0.0   0:00.01 khungtaskd
  18 root      20   0      0      0   0 S  0.0  0.0   0:00.00 oom_reaper
  19 root      0 -20      0      0   0 I  0.0  0.0   0:00.00 writeback
  20 root      20   0      0      0   0 S  0.0  0.0   0:00.00 kcompactd0
  21 root      25   5      0      0   0 S  0.0  0.0   0:00.00 ksmd
  22 root      39  19      0      0   0 S  0.0  0.0   0:00.00 khugepaged

```

Figura 12. Comando top durante una ejecución de apt-get update. Fuente: elaboración propia.

Los procesos pueden terminar por sí mismos si terminan su cometido o si sufren un error que provoque que se cierren con un código de salida diferente de 0. En algunos casos, no obstante, es necesario detener los procesos antes de tiempo. La familia de comandos de kill facilitan la tarea de enviar «señales» a los procesos (Van Vugt, 2015). Algunas señales pueden ser ignoradas y, en todo caso, el *software* tiene que estar preparado para atenderlas. Las principales señales se muestran en la Tabla 3.

Señales		
SIGHUP	1	Fuerza un proceso a releer la configuración sin llegar a parar el proceso.
SIGKILL	9	Termina el proceso de manera forzada. El proceso no tiene opción de ignorar la señal y tampoco tiene tiempo para terminar las tareas en curso.
SIGTERM	15	Solicita la terminación del proceso, aunque el proceso puede optar por ignorar esta señal.
SIGUSR1 y SIGUSR2	10 y 12	Envía una señal específica. El proceso tiene que estar diseñado para entender estas señales; y diferentes procesos pueden reaccionar de manera diferente.

Tabla 3. Señales habituales. Fuente: elaboración propia.

El comando `kill` envía una señal a un proceso a partir del PID. Es necesario conocer el PID, que es fácil de obtener a partir de `ps`. Por ejemplo, en la Figura 13, se muestra el PID de un proceso `top` al que se envía la señal `SIGKILL`.

```

2. ubuntu@ubuntu-VirtualBox: ~
ubuntu@ubuntu-VirtualBox:~$ ps aux | grep top
ubuntu 1719 0.0 2.4 770412 48992 tty1 S1+ May02 0:00 nautilus-desktop
ubuntu 4996 0.3 0.1 48884 3976 pts/1 S+ 02:50 0:00 top
ubuntu 5000 0.0 0.0 21508 1000 pts/3 S+ 02:51 0:00 grep --color=auto top
ubuntu@ubuntu-VirtualBox:~$ kill -9 4996
ubuntu@ubuntu-VirtualBox:~$ ps aux | grep top
ubuntu 1719 0.0 2.4 770412 48992 tty1 S1+ May02 0:00 nautilus-desktop
ubuntu 5002 0.0 0.0 21508 1000 pts/3 S+ 02:51 0:00 grep --color=auto top
ubuntu@ubuntu-VirtualBox:~$

```

Figura 13. Comando `kill` sobre un proceso `top`. Fuente: elaboración propia.

Se puede hacer uso de la sustitución al vuelo de comandos de Bash para obtener el PID del proceso con `pidof` e insertarlo en `kill` en una sola línea:

```
$ sudo kill -9 `pidof top`
```

El comando `killall` es un poco más cómodo de usar para un usuario, ya que permite especificar el nombre del proceso. Como su nombre indica, `killall` terminará todos los procesos con ese nombre. También permite especificar un usuario, por lo que terminará los procesos de ese usuario. Si el usuario ha iniciado una sesión interactiva,

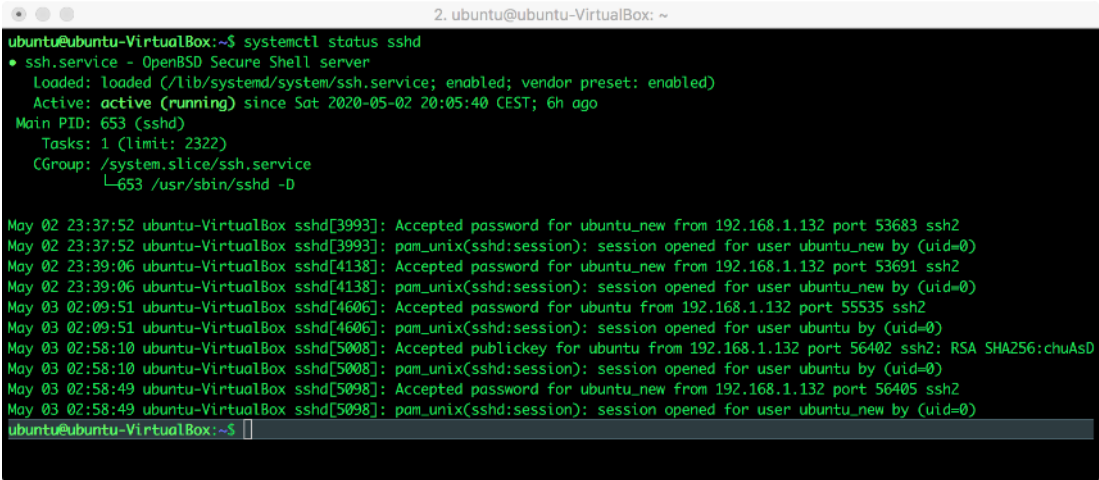
killall terminará también el proceso de consola y, por tanto, terminará la sesión del usuario.

```
$ killall -u username
$ killall -r httpd
```

Finalmente, pkill permite terminar procesos con el nombre del proceso. Es también capaz de terminar los procesos hijos de un proceso padre con el *flag* -P.

```
$ pkill -P pid
```

Los servicios se pueden administrar como procesos con el comando kill, pero es posible usar la funcionalidad de Systemd para obtener más información. La utilidad systemctl permite arrancar, parar, recargar y obtener información de los servicios arrancados con Systemd. Por ejemplo, para el servicio de SSH, systemctl muestra la información que se encuentra en la Figura 14.

A terminal window titled '2. ubuntu@ubuntu-VirtualBox: ~' showing the command 'systemctl status sshd' and its output. The output includes details about the 'ssh.service' (OpenBSD Secure Shell server), its status (loaded, enabled), and a list of recent log messages showing successful SSH logins for 'ubuntu_new' and 'ubuntu' users from IP 192.168.1.132.

```
ubuntu@ubuntu-VirtualBox:~$ systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2020-05-02 20:05:40 CEST; 6h ago
     Main PID: 653 (sshd)
       Tasks: 1 (limit: 2322)
      CGroup: /system.slice/ssh.service
              └─653 /usr/sbin/sshd -D

May 02 23:37:52 ubuntu-VirtualBox sshd[3993]: Accepted password for ubuntu_new from 192.168.1.132 port 53683 ssh2
May 02 23:37:52 ubuntu-VirtualBox sshd[3993]: pam_unix(sshd:session): session opened for user ubuntu_new by (uid=0)
May 02 23:39:06 ubuntu-VirtualBox sshd[4138]: Accepted password for ubuntu_new from 192.168.1.132 port 53691 ssh2
May 02 23:39:06 ubuntu-VirtualBox sshd[4138]: pam_unix(sshd:session): session opened for user ubuntu_new by (uid=0)
May 03 02:09:51 ubuntu-VirtualBox sshd[4606]: Accepted password for ubuntu from 192.168.1.132 port 55535 ssh2
May 03 02:09:51 ubuntu-VirtualBox sshd[4606]: pam_unix(sshd:session): session opened for user ubuntu by (uid=0)
May 03 02:58:10 ubuntu-VirtualBox sshd[5008]: Accepted publickey for ubuntu from 192.168.1.132 port 56402 ssh2: RSA SHA256:chuAsD
May 03 02:58:10 ubuntu-VirtualBox sshd[5008]: pam_unix(sshd:session): session opened for user ubuntu by (uid=0)
May 03 02:58:49 ubuntu-VirtualBox sshd[5098]: Accepted password for ubuntu_new from 192.168.1.132 port 56405 ssh2
May 03 02:58:49 ubuntu-VirtualBox sshd[5098]: pam_unix(sshd:session): session opened for user ubuntu_new by (uid=0)
ubuntu@ubuntu-VirtualBox:~$
```

Figura 14. Información del servicio SSH con systemctl. Fuente: elaboración propia.

El campo *loaded*, en la segunda línea, indica el fichero utilizado para arrancar el servicio. Este *unit file* no es el binario como tal, sino un fichero de configuración de Systemd que indica, entre otras cosas, condiciones de ejecución, la ruta al binario con los *flags* necesarios para el arranque y ficheros con variables de entorno. El fichero de Systemd de sshd se muestra a continuación.

```
$ cat /lib/systemd/system/ssh.service
[Unit]
Description=OpenBSD Secure Shell server
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=-/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/usr/sbin/sshd -t
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify
RuntimeDirectory=sshd
RuntimeDirectoryMode=0755

[Install]
WantedBy=multi-user.target
Alias=sshd.service
```

Estos ficheros son editables por un administrador y es posible definir ficheros propios para configurar aplicaciones propias como servicios.

La información provista por `systemctl` va más allá. Indica también el estado del servicio, `active` (`running`) en este caso. El estado puede ser `inactive` (`dead`) si se ha parado el servicio o `failed`, con el código de salida, si el proceso terminó de manera abrupta. Además, las últimas líneas del fichero de *log* se incluyen en la salida de `systemctl status`, lo que puede ser útil si el proceso ha detectado un error antes de detenerse.

Además de ofrecer información, `systemctl` permite arrancar y parar los servicios con `systemctl start` y `systemctl stop`. La Figura 15 muestra ejemplos de ambos comandos. Es necesario identificar el servicio con el nombre y el servicio no siempre

tiene el mismo nombre que el proceso que ejecuta. Es posible listar todos los servicios en ejecución filtrando la salida de `systemctl` con `grep running`, como en la Figura 16, o todos los servicios habilitados (que pueden no estar en ejecución si se han parado manualmente o si han fallado) con `grep enabled`.

```

2. ubuntu@ubuntu-VirtualBox: ~
ubuntu@ubuntu-VirtualBox:~$ sudo systemctl status vsftpd
● vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabled)
   Active: failed (Result: exit-code) since Sun 2020-05-03 03:14:41 CEST; 3min 2s ago
   Process: 5998 ExecStart=/usr/sbin/vsftpd /etc/vsftpd.conf (code=exited, status=2)
   Process: 5997 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=exited, status=0/SUCCESS)
   Main PID: 5998 (code=exited, status=2)

May 03 03:14:41 ubuntu-VirtualBox systemd[1]: Starting vsftpd FTP server...
May 03 03:14:41 ubuntu-VirtualBox systemd[1]: Started vsftpd FTP server.
May 03 03:14:41 ubuntu-VirtualBox systemd[1]: vsftpd.service: Main process exited, code=exited, status=2/INVALIDARGUMENT
May 03 03:14:41 ubuntu-VirtualBox systemd[1]: vsftpd.service: Failed with result 'exit-code'.
ubuntu@ubuntu-VirtualBox:~$ sudo systemctl start vsftpd
ubuntu@ubuntu-VirtualBox:~$ sudo systemctl status vsftpd
● vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2020-05-03 03:17:47 CEST; 2s ago
   Process: 6013 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=exited, status=0/SUCCESS)
   Main PID: 6014 (vsftpd)
     Tasks: 1 (limit: 2322)
    CGroup: /system.slice/vsftpd.service
            └─6014 /usr/sbin/vsftpd /etc/vsftpd.conf

May 03 03:17:47 ubuntu-VirtualBox systemd[1]: Starting vsftpd FTP server...
May 03 03:17:47 ubuntu-VirtualBox systemd[1]: Started vsftpd FTP server.
ubuntu@ubuntu-VirtualBox:~$ sudo systemctl stop vsftpd
ubuntu@ubuntu-VirtualBox:~$

```

Figura 15. Arranque y parada de servicios con `systemctl`. Fuente: elaboración propia.

```

2. ubuntu@ubuntu-VirtualBox: ~
ubuntu@ubuntu-VirtualBox:~$ systemctl | grep running
acpid.service loaded active running ACPI Events Check
caps.path loaded active running CUPS Scheduler
init.scope loaded active running System and Service Manager
session-1.scope loaded active running Session 1 of user ubuntu
session-14.scope loaded active running Session 14 of user ubuntu
accounts-daemon.service loaded active running Accounts Service
acpid.service loaded active running ACPI event daemon
avahi-daemon.service loaded active running Avahi mDNS/DNS-SD Stack
bolt.service loaded active running Thunderbolt system service
colord.service loaded active running Manage, Install and Generate Color Profiles
cups-browsed.service loaded active running Make Remote CUPS printers available locally
cups.service loaded active running CUPS Scheduler
dbus.service loaded active running D-Bus System Message Bus
fwupd.service loaded active running Firmware update daemon
gdm.service loaded active running GNOME Display Manager
inetd.service loaded active running Internet superserver
kerneloops.service loaded active running Tool to automatically collect and submit kernel crash signatures
ModemManager.service loaded active running Modem Manager
networkd-dispatcher.service loaded active running Dispatcher daemon for systemd-networkd
NetworkManager.service loaded active running Network Manager
packagekit.service loaded active running PackageKit Daemon
polkit.service loaded active running Authorization Manager
postfix.service loaded active running Postfix Mail Transport Agent (instance -)
rsyslog.service loaded active running System Logging Service
rtkit-daemon.service loaded active running RealtimeKit Scheduling Policy Service
snappy.service loaded active running Snappy daemon
ssh.service loaded active running OpenSSH Secure Shell server
systemd-journald.service loaded active running Journal Service
systemd-logind.service loaded active running Login Service
systemd-resolved.service loaded active running Network Name Resolution
systemd-udevd.service loaded active running udev Kernel Device Manager
udisks2.service loaded active running Disk Manager
unattended-upgrades.service loaded active running Unattended Upgrades Shutdown
upower.service loaded active running Daemon for power management
user@1000.service loaded active running User Manager for UID 1000
whoopsie.service loaded active running crash report submission daemon
wpa_supplicant.service loaded active running WPA supplicant
acpid.socket loaded active running ACPI Dbus Listen Socket
avahi-daemon.socket loaded active running Avahi mDNS/DNS-SD Stack Activation Socket
caps.socket loaded active running CUPS Scheduler
dbus.socket loaded active running D-Bus System Message Bus Socket
snappy.socket loaded active running Socket activation for snappy daemon
syslog.socket loaded active running Syslog Socket
systemd-journald-audit.socket loaded active running Journal Audit Socket
systemd-journald-devlog.socket loaded active running Journal Socket (/dev/log)
systemd-journald.socket loaded active running Journal Socket
systemd-udevd-control.socket loaded active running udev Control Socket
systemd-udevd-kernel.socket loaded active running udev Kernel Socket

```

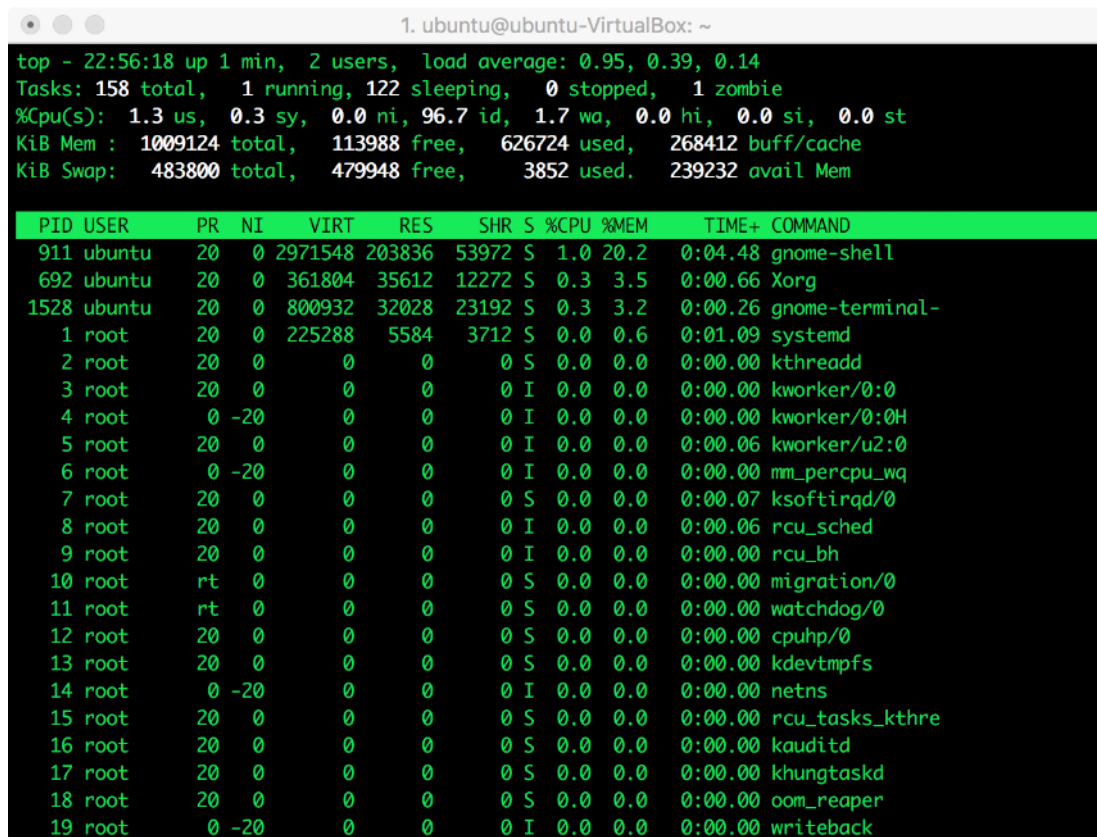
Figura 16. Servicios en ejecución con `systemctl`. Fuente: elaboración propia.

5.5. Herramientas de sistema

Una tarea habitual de los administradores es controlar el rendimiento del sistema. Algunos de los parámetros habituales son el uso del CPU, memoria, uso de *swap* y espacio libre del disco duro. Las siguientes secciones presentan algunas herramientas de Linux para monitorizar estas métricas.

Top

El comando *top* muestra los comandos con más uso del CPU, ordenados en orden descendente de uso, junto con otras métricas, como uso de memoria, *swap*, etc. La pantalla de *top* se actualiza cada cinco segundos, por defecto, y tendrá un aspecto similar a la Figura 17.



```
top - 22:56:18 up 1 min,  2 users,  load average: 0.95, 0.39, 0.14
Tasks: 158 total,   1 running, 122 sleeping,   0 stopped,   1 zombie
%Cpu(s):  1.3 us,  0.3 sy,  0.0 ni, 96.7 id,  1.7 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 1009124 total, 113988 free,  626724 used,  268412 buff/cache
KiB Swap: 483800 total,  479948 free,   3852 used,  239232 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
911	ubuntu	20	0	2971548	203836	53972	S	1.0	20.2	0:04.48	gnome-shell
692	ubuntu	20	0	361804	35612	12272	S	0.3	3.5	0:00.66	Xorg
1528	ubuntu	20	0	800932	32028	23192	S	0.3	3.2	0:00.26	gnome-terminal-
1	root	20	0	225288	5584	3712	S	0.0	0.6	0:01.09	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
5	root	20	0	0	0	0	I	0.0	0.0	0:00.06	kworker/u2:0
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.07	ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	0:00.06	rcu_sched
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
14	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditd
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd
18	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback

Figura 17. Comando *top* en un sistema de escritorio. Fuente: elaboración propia.

Las primeras líneas muestran un resumen de información del sistema, en este orden:

- ▶ Hora actual, *uptime* del sistema, número de usuarios que han iniciado sesión y la carga media del CPU durante el último minuto, últimos cinco minutos y últimos quince minutos.
- ▶ Número total de procesos y el número en cada estado.
- ▶ Uso del CPU y porcentaje del CPU dedicado a procesos de usuario, a procesos del núcleo del sistema y sin uso.
- ▶ Uso de memoria física: total, libre y en uso, en *bytes*.
- ▶ Uso de memoria virtual: espacio total de *swap*, libre y en uso.

A continuación, la tabla lista los procesos, ordenados por uso de CPU. Es posible cambiar los campos que se muestran, el orden y los campos de resumen. Por ejemplo, para ordenar los procesos por uso de memoria, habría que pulsar *f* para entrar en el menú de la Figura 18, seleccionar *%MEM*, pulsar *s* para ordenar por ese campo y pulsar *q* para volver a la pantalla de *top*. También se podrían añadir nuevas columnas, seleccionándolas y pulsando *d*.

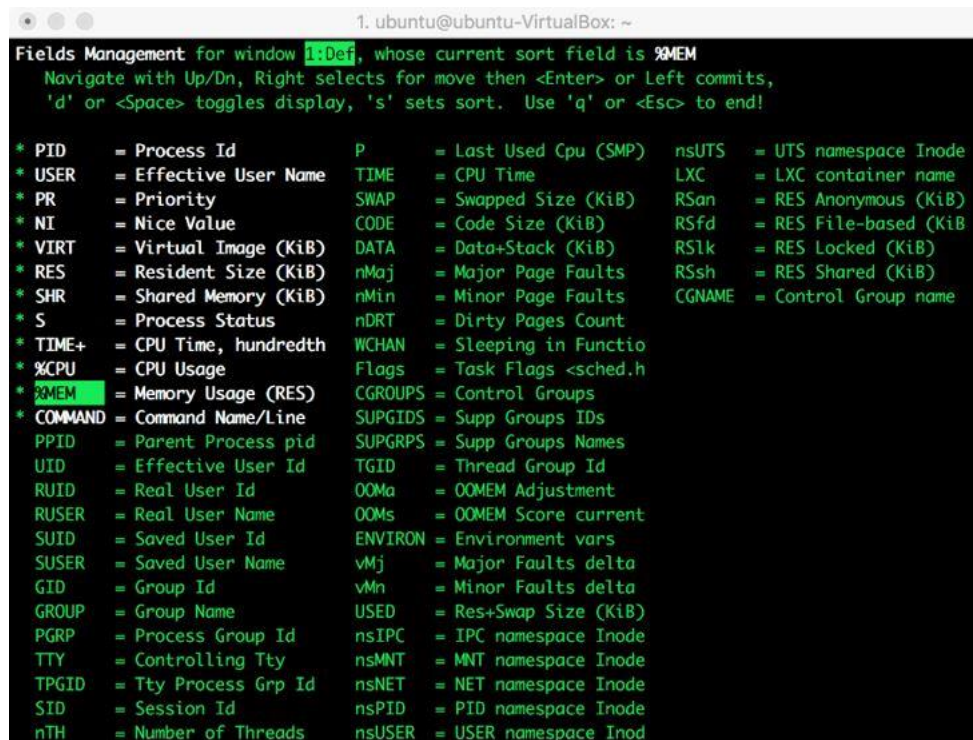


Figura 18. Configuración de columnas de *top*. Fuente: elaboración propia.

La Tabla 4 lista los campos habituales de `top` y su descripción.

Columnas de top	
PID	ID del proceso.
USER	Nombre del usuario bajo el que se ejecuta el proceso.
PR	Prioridad del proceso.
NI	Valor <i>nice</i> del proceso. Este valor representa la prioridad relativa del proceso: un proceso con un valor alto es <i>nice</i> respecto a otros procesos, porque les cede ciclos de CPU, mientras que un proceso con un valor bajo consume más CPU que el resto.
VIRT	Memoria virtual total en KB.
RES	Memoria física total en KB.
SHR	Memoria compartida.
S	Estado del proceso: dormido (S), hibernado (D), en ejecución (R), zombi (z) o parado (T).
%CPU	Porcentaje de CPU.
%MEM	Porcentaje de memoria física.
TIME+	Tiempo del CPU desde el arranque del proceso.
COMMAND	Comando de arranque.

Tabla 4. Columnas habituales de `top`. Fuente: elaboración propia.

Uptime

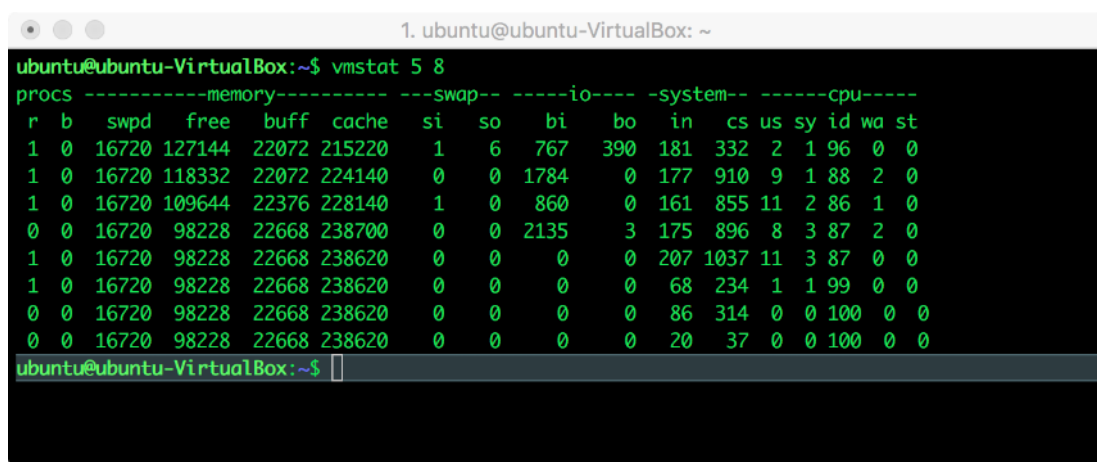
El comando `uptime` muestra por consola un resumen del estado del sistema: la hora actual, el tiempo que el sistema ha estado arrancado, el número de usuarios conectados y el uso medio del CPU en el último minuto, últimos cinco minutos y últimos quince minutos.

```
$ uptime
```

```
23:32:56 up 37 min,  2 users,  load average: 0.00, 0.00, 0.05
```

Vmstat

El comando `vmstat` ofrece un resumen de métricas de rendimiento en intervalos definidos por el usuario. En vez de actualizar los valores en la consola, como `top`, imprime los valores de las métricas en una línea periódicamente. El ejemplo de la Figura 19 imprime las métricas cada cinco segundos, hasta un máximo de ocho. Las columnas están organizadas en categorías. La descripción de cada columna se detalla en la Tabla 5.



```
1. ubuntu@ubuntu-VirtualBox: ~  
ubuntu@ubuntu-VirtualBox:~$ vmstat 5 8  
procs -----memory----- --swap-- -----io----- -system-- -----cpu-----  
r  b  swpd  free  buff  cache   si   so    bi    bo    in  cs  us  sy  id  wa  st  
1  0  16720 127144 22072 215220    1    6   767   390  181 332  2  1 96  0  0  
1  0  16720 118332 22072 224140    0    0  1784    0  177 910  9  1 88  2  0  
1  0  16720 109644 22376 228140    1    0   860    0  161 855 11  2 86  1  0  
0  0  16720  98228 22668 238700    0    0  2135    3  175 896  8  3 87  2  0  
1  0  16720  98228 22668 238620    0    0    0    0  207 1037 11  3 87  0  0  
1  0  16720  98228 22668 238620    0    0    0    0   68 234  1  1 99  0  0  
0  0  16720  98228 22668 238620    0    0    0    0   86 314  0  0 100  0  0  
0  0  16720  98228 22668 238620    0    0    0    0   20  37  0  0 100  0  0  
ubuntu@ubuntu-VirtualBox:~$
```

Figura 19. Comando `vmstat`. Fuente: elaboración propia.

Columnas de vmstat	
procs	r: procesos en ejecución b: procesos dormidos
memory	swpd: memoria virtual free: memoria física libre buff: memoria usada como <i>buffers</i> caché: memoria virtual cacheada
swap	si: memoria transferida desde disco (KB) so: memoria transferida a disco (KB)
io	bi: bloques escritos en disco (bloques/s) bo: bloques recibidos de disco (bloques/s)
system	in: interrupciones por segundo cs: cambios de contexto por segundo
cpu	us: uso de CPU de usuario (%) sy: uso de CPU de sistema (%) id: tiempo de CPU sin trabajo (%) wa: tiempo de CPU bloqueada en operaciones de IO (de entrada y salida) (%).

Tabla 5. Columnas de vmstat. Fuente: elaboración propia.

Df

El comando `df` muestra el espacio disponible en los sistemas de ficheros montados. En la Figura 20 se muestra la salida de `df` en una instalación de Ubuntu con escritorio. El modificador `-h` imprime los valores en unidades más legibles para un usuario, como KB, MB, etc.

```
1. root@ubuntu-VirtualBox: /home/ubuntu
root@ubuntu-VirtualBox:/home/ubuntu# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            464M   0    464M   0% /dev
tmpfs           99M    1.3M  98M    2% /run
/dev/sda1       9.8G  4.2G  5.2G   45% /
tmpfs           493M   0    493M   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           493M   0    493M   0% /sys/fs/cgroup
/dev/loop0      15M   15M    0 100% /snap/gnome-logs/37
/dev/loop1      35M   35M    0 100% /snap/gtk-common-themes/319
/dev/loop2      2.4M  2.4M    0 100% /snap/gnome-calculator/180
/dev/loop4      3.8M  3.8M    0 100% /snap/gnome-system-monitor/51
/dev/loop3     141M  141M    0 100% /snap/gnome-3-26-1604/70
/dev/loop5      87M   87M    0 100% /snap/core/4917
/dev/loop6      13M   13M    0 100% /snap/gnome-characters/103
tmpfs           99M   28K   99M    1% /run/user/1000
root@ubuntu-VirtualBox:/home/ubuntu#
```

Figura 20. Comando `df -h`. Fuente. Elaboración propia.

Directorio `/proc`

En la sección del sistema de ficheros, se mencionó que el directorio `/proc` es un directorio especial, ya que contiene archivos con información del sistema operativo. De hecho, no solo se puede obtener información, sino que es posible cambiar los parámetros del núcleo de Linux editando los archivos contenidos de `/proc`, modificando así el comportamiento del sistema.

El sistema de archivos `/proc` no es un directorio real en el disco duro, sino una colección de estructuras de datos en la memoria, administrada por el núcleo, que aparece como un conjunto de directorios y archivos. El propósito de `/proc` (también llamado **sistema de archivos de procesos**) es ofrecer acceso a la información sobre el sistema operativo y sobre todos los procesos que se encuentran en ejecución.

Los archivos de `/proc` son accesibles, como cualquier otro, pero es necesario conocer el significado y la sintaxis de ellos para interpretar la información. Por ejemplo, los

comandos `cat` o `less` mostrarán el contenido de los archivos y `ls` muestra una lista con los archivos disponibles.

```

1. root@ubuntu-VirtualBox: /proc
root@ubuntu-VirtualBox:/proc# ls /proc
1      1051 1107 1242 1574 1926 245 29 410 669 79 921  buddyinfo  fs      locks  slabinfo  vmstat
10     1056 114 1289 16 2 25 30 412 678 8 932  bus        interrupts  mdstat  softirqs  zoneinfo
1003   1062 1140 13 165 20 258 32 413 684 80 946  cgroups    iomem      meminfo  stat
1011   1066 1141 1373 166 2056 26 34 434 7 81 947  cmdline    ioports    misc     swaps
1016   1067 1143 1374 167 2095 260 35 435 753 82 959  consoles    irq         modules  sys
1020   1077 1150 14 1695 21 262 376 438 758 84 963  cpuinfo     kallsyms    mounts   sysrq-trigger
1025   1081 1165 1408 17 2161 263 378 469 759 88 965  crypto      kcore       mtrr     sysvipc
1029   1084 1182 1409 1784 22 264 383 494 77 880 969  devices     key-users   net       thread-self
1033   1085 1187 1410 18 220 265 399 506 772 883 97  diskstats   keys        pagetypeinfo  timer_list
1037   1087 1194 1421 1827 2232 266 4 507 776 888 979  dma         kmsg        partitions  tty
1040   1089 12 15 188 2241 267 400 593 778 897 983  driver      kpagecgroup sched_debug  uptime
1044   1091 1210 1508 1886 23 268 402 6 78 9 992  execdomains kpagecount  schedstat   version
1046   1095 1217 152 189 24 27 405 605 782 902  acpi        fb          kpageflags  scsi        version_signature
1047   11 1229 1526 19 243 28 409 627 786 905  asound      filesystems loadavg     self        vmallocinfo
root@ubuntu-VirtualBox:/proc#

```

p

Figura 21. Listado de archivos de `/proc`. Fuente: elaboración propia.

El primer conjunto de directorios (indicado por el *flag* `d` en la salida de `ls -l /proc` o en diferente color en la Figura 21) representa los procesos que se ejecutan actualmente en su sistema. Cada directorio que corresponde a un proceso tiene el PID como nombre. Los archivos de cada directorio contienen información sobre el proceso: el binario, la línea de comandos con la que se ha ejecutado, las variables de entorno, uso de memoria, etc. La Figura 22 muestra algunos de estos ficheros para el proceso de un servidor FTP.

```

1. root@ubuntu-VirtualBox: /proc/662
root@ubuntu-VirtualBox:/proc/662# ll exe
lrwxrwxrwx 1 root root 0 May 10 07:43 exe -> /usr/sbin/vsftpd*
root@ubuntu-VirtualBox:/proc/662# cat environ ; echo
LANG=en_US.UTF-8LC_ADDRESS=es_ES.UTF-8LC_IDENTIFICATION=es_ES.UTF-8LC_MEASUREMENT=es_ES.UTF-8LC_MONETARY=es_ES.UTF-8LC_NAME=es_ES.UTF-8LC_NUMERIC=es_ES.UTF-8LC_PAPER=es_ES.UTF-8LC_TELEPHONE=es_ES.UTF-8LC_TIME=es_ES.UTF-8PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/binINVOCAATION_ID=14a737c22ab54f5eb1cc0f87a19a6905JOURNAL_STREAM=9:19683
root@ubuntu-VirtualBox:/proc/662# cat cmdline ; echo
/usr/sbin/vsftpd/etc/vsftpd.conf
root@ubuntu-VirtualBox:/proc/662#

```

Figura 22. Detalles del proceso del servidor FTP. Fuente: elaboración propia.

Otro archivo interesante es `/proc/cpuinfo`. Este contiene las características de los procesadores del equipo: fabricante, modelo, conjuntos de instrucciones, frecuencia, etc.

```
# cat /proc/cpuinfo
processor          : 0
vendor_id         : GenuineIntel
cpu family        : 6
model             : 158
model name        : Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz
stepping          : 9
cpu MHz           : 2903.998
cache size        : 8192 KB
physical id       : 0
siblings          : 1
core id           : 0
cpu cores         : 1
apicid            : 0
initial apicid    : 0
fpu               : yes
fpu_exception     : yes
cpuid level       : 22
wp               : yes
flags             : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall nx rdtscp lm constant_tsc
rep_good nopl xtopology nonstop_tsc cpuid pni pclmulqdq monitor ssse3 cx16
sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx rdrand hypervisor lahf_lm
abm 3dnowprefetch pti avx2 rdseed clflushopt
bugs              : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass
bogomips          : 5807.99
clflush size      : 64
cache_alignment   : 64
address sizes     : 39 bits physical, 48 bits virtual
power management:
```

La Tabla 6 describe algunos de los archivos y directorios de /proc.

Contenido de /proc	
/proc/acpi	Directorio con información sobre ACPI, la interfaz de control de energía.
/proc/cmdline	Línea de comandos usada para el arranque del <i>kernel</i> .
/proc/devices	Dispositivos de bloque y de carácter del sistema.
/proc/filesystems	Lista de sistemas de archivos soportados.
/proc/kcore	Imagen de la memoria física.
/proc/locks	Bloqueos (<i>locks</i>) del sistema operativo en curso.
/proc/meminfo	Información sobre la memoria física y virtual.
/proc/mounts	Lista de sistemas de archivos montados.
/proc/net	Directorio con información sobre las interfaces y los protocolos de red.
/proc/partitions	Lista de particiones.
/proc/scsi/scsi	Información de los dispositivos SCSI.
/proc/sys	Directorio con información del sistema. El contenido se puede editar para configurar parámetros específicos del sistema operativo.
/proc/uptime	Tiempo de ejecución del sistema.
/proc/version	Versión del núcleo de Linux.

Tabla 6. Algunos ficheros y directorios de /proc. Fuente: elaboración propia.

Programación de tareas

Ciertos programas de Linux no están pensados para ejecutarse continuamente como un servicio, pero es necesario ejecutarlos en un instante futuro dado o periódicamente. Algunas de estas tareas pueden ser copia de seguridad, análisis de *logs* o descarga de grandes ficheros (por ejemplo, ficheros de actualización o sincronización de *mirrors*) en horario nocturno.

El comando `at` programa comandos para su ejecución en un momento dado. El servicio `atd` se encarga de ejecutar los comandos programados con `at`. Estos trabajos se ejecutarán en el momento indicado, tras lo cual `at` los borrará de la lista. Los pasos para programar un trabajo son los siguientes:

- ▶ Ejecutar `at <tiempo>`. El formato para indicar el instante es muy versátil y acepta formatos como:
 - `at 21:30.`
 - `at now.`
 - `at now + 15 minutes.`
 - `at now + 4 hours.`
 - `at noon.`
 - `at now next hour` (dentro de 60 minutos).
 - `at now next day` (a la misma hora del día siguiente).
 - `at 17:00 tomorrow.`
 - `at 3:00 Jan 18, 2021.`
- ▶ Se abrirá una *shell* de `at` en la que se introducen los comandos, uno por línea.
- ▶ Para cerrar la consola de `at`, hay que pulsar `Ctrl+D`, una vez escritos los comandos.

```

1. root@ubuntu-VirtualBox: ~
root@ubuntu-VirtualBox:~#
root@ubuntu-VirtualBox:~# at 3:00 tomorrow
warning: commands will be executed using /bin/sh
at> ps aux > /tmp/$(date +%Y%m%d)-ps.log
at> uptime > /tmp/$(date +%Y%m%d)-uptime.log
at> <EOT>
job 4 at Mon May 11 03:00:00 2020
root@ubuntu-VirtualBox:~#

```

Figura 23. Programación de comandos con `at`. Fuente: elaboración propia.

La Figura 23 muestra cómo se añaden dos comandos para su ejecución a las 3:00 a. m. del día siguiente. La lista de trabajos programados se puede consultar con `atq` y se pueden borrar con `atrm <id>`.

```

$ atq
3      Mon May 11 03:00:00 2020 a root
2      Sun May 10 21:31:00 2020 a root

```

Este comando no permite, sin embargo, programar tareas que se ejecuten periódicamente de manera automática. Para ello, se puede usar el servicio `cron` y el

comando **crontab**. *Cron* no permite trabajos con más de un comando, pero siempre se puede escribir un *script* con los comandos necesarios. El servicio comprueba los trabajos disponibles cada minuto y ejecuta los que deban ejecutarse en ese momento.

Los pasos para programar un trabajo de cron son:

- ▶ Preparar el *script* o el comando que se ejecutará.
- ▶ Preparar un archivo de texto con la programación y el comando. El archivo puede contener más de una tarea, cada una con su programación.
- ▶ Ejecutar `crontab <fichero>` para confirmar la definición.
- ▶ Confirmar que el trabajo está definido con `crontab -l`.

El formato de definición de tareas es el siguiente:

```
5 0 * * * ps aux > /tmp/$(date +%Y%M%d$h%m)-ps.log
```

Las cinco primeras columnas de la línea definen la programación: cada campo representa una unidad temporal y el valor indica en qué momento se «activa» ese campo. Cada campo puede contener un número, una lista de números separadas por comas, una pareja de números con un guion representando un intervalo o un asterisco, que indica todos los valores posibles. La Tabla 7 lista la unidad y los posibles valores de cada campo.

Campos de programación de cron		
1	Minutos	0-59
2	Hora	0-23
3	Día del mes	0-31
4	Mes	1-12 o las primeras letras del mes: <i>Jan, Feb, etc.</i>
5	Día de la semana	0-6, de domingo a sábado, o las primeras letras: <i>Sun, Mon, Tue, etc.</i>

Tabla 7. Campos de la programación de cron. Fuente: elaboración propia.

Este sistema puede parecer complicado, así que la Tabla 8 tiene ejemplos de cómo interpretar la sintaxis.

Ejemplos de programación de cron	
5 0 * * *	Todos los días a las 00:05
0,15,30,45 * * * *	Cada 15 minutos
0 3 1 * *	Mensualmente, el día 1 de cada mes, a las 3:00
* * * * *	Cada minuto
10 0 * * 6	Los sábados a las 00:10

Tabla 8. Ejemplos de cron. Fuente: elaboración propia.

Cuando un usuario ejecuta `crontab -l`, el comando solo muestra los trabajos programados por él mismo. Por ejemplo, si el usuario `root` define el trabajo del ejemplo, su listado de trabajo sería el siguiente:

```
$ crontab -l
5 0 * * * ps aux > /tmp/$(date +%Y%M%d$h%m)-ps.log
```

Es decir, exactamente el contenido del fichero con el que se ha definido el trabajo. Sin embargo, el fichero `/etc/crontab` contiene la programación de trabajos de

sistema. Por ejemplo, en un equipo Ubuntu, el fichero `/etc/crontab` por defecto contiene lo siguiente:

```
$ cat /etc/crontab
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.monthly )
```

Además de unas variables de entorno que se aplicarán a cada uno de los trabajos, el fichero contiene cuatro trabajos, que corresponden con tareas horarias, diarias, semanales y mensuales. El comando `run-parts` ejecutará los *scripts* contenidos en cada una de las carpetas `/etc/cron.*`, siguiendo la programación del fichero `crontab`. Los servicios de sistema pueden aprovechar esta funcionalidad para añadir *scripts* recurrentes en estas carpetas y así delegar la ejecución periódica a `cron`. Por ejemplo, en la Figura 24, se muestran las tareas diarias. Una de ellas, `/etc/cron.daily/passwd`, crea copias de seguridad de los ficheros de las cuentas de usuario y grupo en la carpeta `/var/backup`.

```
1. root@ubuntu-VirtualBox: ~
root@ubuntu-VirtualBox:~# ll /etc/cron.daily/
total 76
drwxr-xr-x  2 root root  4096 Apr 30 09:53 ./
drwxr-xr-x 126 root root 12288 May 10 08:18 ../
-rw-r--r--  1 root root   102 Nov 16 2017 .placeholder
-rwxr-xr-x  1 root root   311 May 29 2017 0anacron*
-rwxr-xr-x  1 root root   376 Nov 20 2017 apport*
-rwxr-xr-x  1 root root  1478 Apr 20 2018 apt-compat*
-rwxr-xr-x  1 root root   355 Dec 29 2017 bsdmaintils*
-rwxr-xr-x  1 root root   384 Dec 12 2012 cracklib-runtime*
-rwxr-xr-x  1 root root  1176 Nov  2 2017 dpkg*
-rwxr-xr-x  1 root root   372 Aug 21 2017 logrotate*
-rwxr-xr-x  1 root root  1065 Apr  7 2018 man-db*
-rwxr-xr-x  1 root root   538 Mar  1 2018 mlocate*
-rwxr-xr-x  1 root root  1387 Dec 13 2017 ntp*
-rwxr-xr-x  1 root root   249 Jan 25 2018 passwd*
-rwxr-xr-x  1 root root  3477 Feb 21 2018 popularity-contest*
-rwxr-xr-x  1 root root   246 Mar 21 2018 ubuntu-advantage-tools*
-rwxr-xr-x  1 root root   214 Jun 27 2018 update-notifier-common*
root@ubuntu-VirtualBox:~# cat /etc/cron.daily/passwd
#!/bin/sh

cd /var/backups || exit 0

for FILE in passwd group shadow gshadow; do
    test -f /etc/$FILE || continue
    cmp -s $FILE.bak /etc/$FILE && continue
    cp -p /etc/$FILE $FILE.bak && chmod 600 $FILE.bak
done
root@ubuntu-VirtualBox:~#
```

Figura 24. Tareas diarias en /etc/cron.daily. Fuente: elaboración propia.

5.6. Referencias bibliográficas

Frampton, S. (s. f.). 6.6. Linux Password & Shadow File Formats. En *Linux Administration Made Easy*.

<https://www.tldp.org/LDP/lame/LAME/linux-admin-made-easy/shadow-file-formats.html>

Matotek, D., Turnbull, J. y Lieverdink, P. (2017). *Pro Linux System Administration: Learn to Build Systems for Your Business Using Free and Open Source Software* (2.ª ed.). Apress.

Página de Upstart (<http://upstart.ubuntu.com/>).

Van Vugt, S. (2015). *Beginning the Linux Command Line* (2.ª ed.). Apress.

Rethinking PID 1

Pid Eins. (s. f.). *Rethinking PID 1*. <http://0pointer.de/blog/projects/systemd.html>

Este artículo ofrece una crítica al porqué de Systemd. Es muy técnico, pero la lectura abarca conceptos complejos con un enfoque manejable, incluso para administradores con poca experiencia en Linux.

Kill Signals and Commands

Linux.org. (s. f.). *Kill Signals and Commands*. <https://www.linux.org/threads/kill-signals-and-commands-revised.11625/>

Este corto artículo repasa las señales disponibles en Linux, su significado y las diferentes formas de utilizarlas. Es más una guía de uso que un artículo, ya que es eminentemente práctico.

Filesystem Hierarchy Standard

LSB Workgroup, The Linux Foundation. (2015). *Filesystem Hierarchy Standard*. https://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html

El estándar del árbol de directorio de Linux, editado por un grupo de trabajo específico, es una lectura densa y solo recomendable para desarrolladores del *kernel* y afines. No obstante, los capítulos de introducción sirven de pistas para entender el porqué de esta arquitectura, antes de entrar en los detalles profundos del diseño y la implementación.

1. ¿Qué comando sirve para solicitar una terminación programada de un proceso?
 - A. `kill -15 <pid>`.
 - B. `kill -9 <pid>`.
 - C. `kill -15 <nombre>`.
 - D. Ninguna de las anteriores.

2. ¿Qué tipos de permisos tendrá un archivo normal tras configurarlo con `chmod 777 filename`?
 - A. Lectura y escritura para el usuario propietario, nada más.
 - B. Lectura, escritura y ejecución para cualquier usuario del sistema.
 - C. Lectura para cualquier usuario, escritura y ejecución para el propietario.
 - D. Ninguna de las anteriores.

3. ¿Qué comando permiten averiguar el PID de un proceso?
 - A. `pidof`.
 - B. `ps`.
 - C. `top`.
 - D. Todos los anteriores.

4. ¿Qué relación hay entre usuarios y grupos?
 - A. Los usuarios pueden pertenecer a un único grupo.
 - B. Los usuarios pertenecen al menos a un grupo principal y a cero o más grupos suplementarios.
 - C. Los usuarios pueden no pertenecer a un grupo o pueden pertenecer a uno o más grupos.
 - D. Los grupos puede agrupar otros grupos, pero no usuarios.

5. ¿Qué opción del comando `systemctl` reinicia un servicio?
- A. `restart`.
 - B. `stop`.
 - C. `status`.
 - D. `start`.
6. ¿Qué fichero aloja las contraseñas de los usuarios en texto plano?
- A. `/etc/passwd`.
 - B. `/etc/shadow`.
 - C. Todos los anteriores.
 - D. Ninguno de los anteriores.
7. ¿Cuál es el resultado de ejecutar `ln -s /usr/bin/python3.6 ~/python`?
- A. Aparece un enlace simbólico al binario de Python en la *home* del usuario con el nombre `python`.
 - B. Aparece un enlace simbólico al binario de Python en la *home* del usuario con el nombre `python3.6`.
 - C. Aparece un enlace fuerte al binario de Python en la *home* del usuario con el nombre `python`.
 - D. Aparece un enlace simbólico al binario de Python en la raíz del árbol de directorios con el nombre `python`.
8. ¿Qué comandos permiten crear, modificar y borrar usuarios?
- A. `usercreate`, `usermodify` y `userdelete`.
 - B. `usercreate`, `usermod` y `deluser`.
 - C. `useradd`, `moduser` y `rmuser`.
 - D. `useradd`, `usermod` y `userdel`.

9. ¿Por qué visudo ha de ejecutarse con sudo?
- A. Por convenio.
 - B. No es necesario.
 - C. Porque es un comando privilegiado.
 - D. Puede no usarse si no se edita el fichero.

10. ¿Qué *flag* denota un archivo oculto?
- A. d.
 - B. l.
 - C. Ninguno.
 - D. s.