Unidad 1

Administración básica de Linux

Administración de sistemas operativos

Índice

1.1. Características de Linux Server

1.2. Actualización del sistema

1.3. Administración de usuarios
- 1.3.1. Añadir usuarios
- 1.3.2. Modificar usuarios
- 1.3.3. Eliminar usuarios
- 1.3.4. Cambiar contraseñas

1.4. Configurar la zona horaria y el idioma

1.5. Administración de aplicaciones
- 1.5.1. Apt-get
- 1.5.2. Dpkg
- 1.5.3. Apt

1.6. Administración de discos
- 1.6.1. Fdisk
- 1.6.2. GNU Parted
- 1.6.3. Gparted
- 1.6.4. Discos de GNOME

1.7. Monitor de recursos

1.8. Permisos
- 1.8.1. Establecer permisos
- 1.8.2. Cambiar propietario y grupo

1.9. Gestión y control de procesos
- 1.9.1. Listar procesos
- 1.9.2. Estados de un proceso
- 1.9.3. Prioridades
- 1.9.4. Árbol de procesos
- 1.9.5. Demonios
- 1.9.6. Procesos en primer plano
- 1.9.7. Procesos en segundo plano
- 1.9.8. Cambiar procesos entre primer y segundo plano
- 1.9.9. Mostrar procesos en segundo plano o suspendidos
- 1.9.10. Terminar un proceso

1.10. Automatización de tareas
- 1.10.1. Cron
- 1.10.2. Crontab
- 1.10.3. Anacron
- 1.10.4. At

Introducción

Los sistemas Linux nunca han sido los más usados en el ámbito comercial en lo que a equipos domésticos nos referimos, pero hay otros ámbitos de la informática en los que son los principales o al menos, están a la par con sus alternativas.

Los sistemas operativos denominados servers son un tipo de sistemas que se diseñan específicamente para funcionar en el ámbito de los servidores. Realmente es un software sobre el que otros tipos de software funcionan en conjunto con el hardware de la máquina, de manera prácticamente igual a como sucede con cualquier otro tipo de sistema operativo.

Como sucede con los Sistemas Operativos de escritorio, en el caso de los servidores, tenemos sistemas de código abierto y privados. Vamos a hablar ahora de las soluciones de código abierto, en especial de Linux, pues se lleva el liderazgo de este sector.

Al finalizar esta unidad

- Sabremos realizar la configuración inicial y los primeros pasos tras la instalación de un sistema operativo Linux Server.
- Podremos actualizar el sistema y estudiar las principales vías o herramientas para la instalación de nuevo software.
- Seremos capaces de desarrollar la administración básica de usuarios y asignar permisos.
- Conoceremos como gestionar los procesos, sus estados y los procesos en primer y segundo plano y aprender a cambiar de un plano a otro.
- Podremos monitorizar los procesos y conocer los comandos básicos para su control y visionado.
- Seremos capaces de planificar las tareas para su ejecución de forma automática, bien sea a través de línea de comandos o de herramientas gráficas.

1.1. Características de Linux Server

Los sistemas Linux son de código abierto, es decir, que su código puede ser redistribuido y cambiado del modo que se desee. Además, prácticamente en su totalidad Linux es gratuito. Esto hace que, aunque estos sistemas no son los principales en el mercado, a la hora de hablar de servidores se encuentran en primera línea. Son los primeros en nivel de eficacia y si además añadimos su bajo coste, obtenemos esta clasificación.

Además de eficaces, son de los más seguros sistemas y es por eso que vamos a ver a continuación, es decir, en los siguientes apartados, sus principales características de funcionamiento.

1.2. Actualización del sistema

En los sistemas de Linux tenemos dos tipos de actualizaciones:

- Las actualizaciones de calidad y seguridad y nuevas versiones de paquetes. Este tipo de actualizaciones ayudan a que Linux siempre esté actualizado y con todos los programas al día, pero sin cambiar de versión.
- Las actualizaciones de versión, que dependiendo de que distribución de Linux se use, serán más o menos frecuentes.

Las actualizaciones de Linux en cuanto a versión de distribución se refieren, suelen ser bastante estables y eficaces, pero siempre es recomendable que se haga una copia de seguridad primero, por si acaso.

En cuanto a las actualizaciones de paquetes para que Linux tenga las versiones de estos siempre al día, también podemos actualizar el *kernel*, las ejecutamos con la siguiente secuencia de comandos:

- `sudo apt update`

```
profesor@servidor2:~$ sudo apt update
Obj:1 http://es.archive.ubuntu.com/ubuntu impish InRelease
Des:2 http://es.archive.ubuntu.com/ubuntu impish-updates InRelease [115 kB]
Des:3 http://es.archive.ubuntu.com/ubuntu impish-backports InRelease [101 kB]
Des:4 http://es.archive.ubuntu.com/ubuntu impish-security InRelease [110 kB]
Des:5 http://es.archive.ubuntu.com/ubuntu impish-updates/main amd64 Packages [306 kB]
Des:6 http://es.archive.ubuntu.com/ubuntu impish-updates/main Translation-en [93,1 kB]
Des:7 http://es.archive.ubuntu.com/ubuntu impish-updates/main amd64 c-n-f Metadata [8.040 B]
Des:8 http://es.archive.ubuntu.com/ubuntu impish-updates/restricted amd64 Packages [104 kB]
Des:9 http://es.archive.ubuntu.com/ubuntu impish-updates/restricted Translation-en [14,1 kB]
Des:10 http://es.archive.ubuntu.com/ubuntu impish-updates/restricted amd64 c-n-f Metadata [560 B]
Des:11 http://es.archive.ubuntu.com/ubuntu impish-updates/universe amd64 Packages [255 kB]
Des:12 http://es.archive.ubuntu.com/ubuntu impish-updates/universe Translation-en [77,7 kB]
Des:13 http://es.archive.ubuntu.com/ubuntu impish-updates/universe amd64 c-n-f Metadata [6.648 B]
Des:14 http://es.archive.ubuntu.com/ubuntu impish-updates/multiverse amd64 Packages [7.796 B]
Des:15 http://es.archive.ubuntu.com/ubuntu impish-updates/multiverse amd64 c-n-f Metadata [420 B]
Des:16 http://es.archive.ubuntu.com/ubuntu impish-backports/universe amd64 Packages [4.832 B]
Des:17 http://es.archive.ubuntu.com/ubuntu impish-backports/universe amd64 c-n-f Metadata [496 B]
Des:18 http://es.archive.ubuntu.com/ubuntu impish-security/main amd64 Packages [238 kB]
Des:19 http://es.archive.ubuntu.com/ubuntu impish-security/main Translation-en [71,9 kB]
Des:20 http://es.archive.ubuntu.com/ubuntu impish-security/main amd64 c-n-f Metadata [5.884 B]
Des:21 http://es.archive.ubuntu.com/ubuntu impish-security/restricted amd64 Packages [95,1 kB]
Des:22 http://es.archive.ubuntu.com/ubuntu impish-security/restricted Translation-en [12,4 kB]
Des:23 http://es.archive.ubuntu.com/ubuntu impish-security/restricted amd64 c-n-f Metadata [532 B]
Des:24 http://es.archive.ubuntu.com/ubuntu impish-security/universe amd64 Packages [207 kB]
Des:25 http://es.archive.ubuntu.com/ubuntu impish-security/universe Translation-en [61,1 kB]
Des:26 http://es.archive.ubuntu.com/ubuntu impish-security/universe amd64 c-n-f Metadata [5.396 B]
Descargados 1.902 kB en 4s (457 kB/s)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se pueden actualizar 35 paquetes. Ejecute «apt list --upgradable» para verlos.
profesor@servidor2:~$
```

Imagen 1. apt update

```
> sudo apt upgrade
```

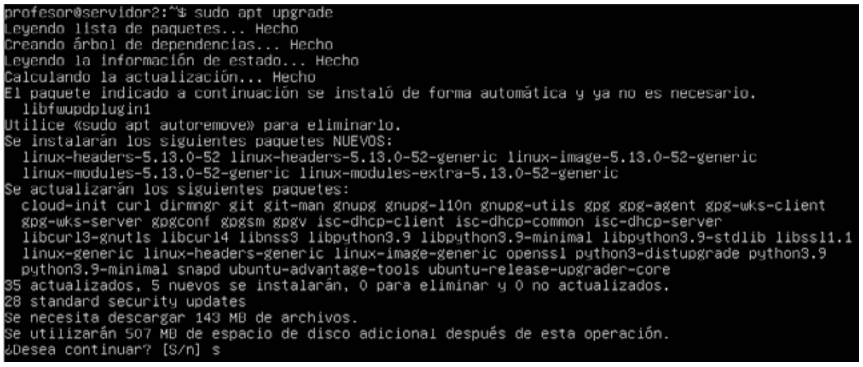


Imagen 2. apt upgrade

```
> sudo apt dist-upgrade
```

```
> sudo reboot
```

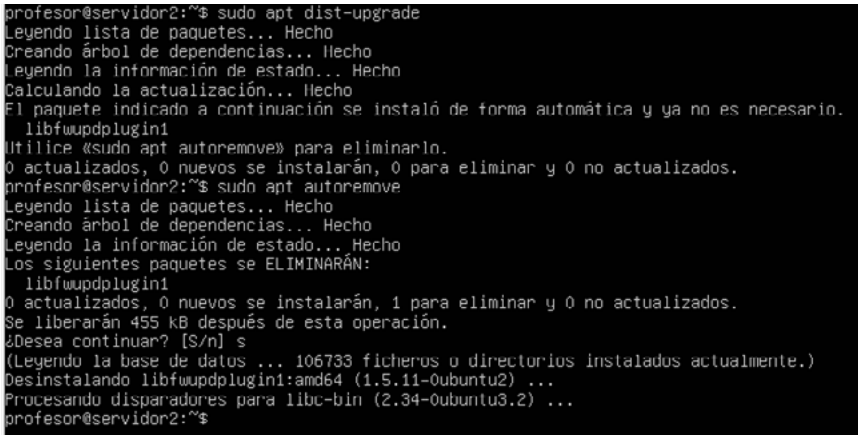


Imagen 3. apt dist-upgrade & reboot

1.3. Administración de usuarios

Los sistemas Linux basados en UNIX basan la administración de sus procesos en ficheros de configuración. La administración de los usuarios lógicamente es otro de los procesos que debemos de tener en cuenta en dicha gestión, y aunque es una labor que no se realiza a diario, cualquier administrador de sistemas debe de saber adecuadamente como se realizan estas acciones.

Para la configuración de usuarios y grupos en Linux usaremos los archivos de configuración del sistema */etc/passwd y /etc/group* respectivamente.

Hay otros archivos de configuración que también se tratarán de manera más indirecta, como el archivo */etc/shadow* que almacena las contraseñas de los usuarios encriptadas.

Para poder trabajar con los archivos, aunque sea mediante comandos, debemos de conocer su estructura interna.

En el caso del fichero */etc/passwd*, en cada línea nueva se almacena un usuario diferente, y, además, cada línea consta de siete campos que se delimitan por el símbolo *":"* y que expresan lo siguiente:

1. **Login.** Almacena el nombre de usuario que se usa para el acceso al sistema.
2. **Password.** Alamcenamos la contraseña necesaria para que el usuario entre al sistema, vendrá marcada con "x" ya que se encuentra ubicada en el fichero /etc/shadow.
3. **UID.** El número de identificador de usuario único. El 0 corresponde al superusuario del sistema, del 1 al 99son las cuentas predeterminadas del sistema, del 100 al 999 son las cuentas administrativas del sistema y los nuevos usuarios se asociarán con un identificador a partir del 1 000.
4. **GID.** El número de identificado de grupo identifica el grupo principal del usuario.
5. **Información personal del usuario.** Cualquier información que se haya añadido como podría ser su nombre completo.
6. **Home o directorio de trabajo del usuario.** Es el directorio principal del usuario y donde se almacena por defecto toda su información. Además, es el directorio por defecto cuando el usuario accede al sistema.
7. **Shell.** Para identificar que intérprete de comandos usará el usuario del sistema.

Imagen 4. /etc/passwd.

Si por ejemplo nos fijamos en el usuario alumno de la imagen anterior, podríamos distinguir lo siguiente:

Estructura del fichero /etc/passwd						
alumno1	**x**	**1 002**	**1 002**	**,,,**	**/home/alumno1**	**/bin/bash**
Nombre de usuario	Contraseña	ID de usuario (UID)	ID de grupo (GID)	Información de usuario	Directorio home del usuario	Shell

Un usuario administrador del sistema en Linux es el que tiene privilegios sobre la gestión de este, pero no tiene necesariamente por qué ser el usuario root. Para poder otorgar privilegios se puede usar el comando sudo (hay que habilitar su uso, lógicamente) o añadir el usuario a un grupo que tenga privilegios sobre configuraciones concretas asignados.

Para administrar los privilegios en Linux son muy usados los grupos, que permiten centralizar de manera más eficiente todos estos permisos hacia usuarios.

Para configurar los grupos se usará el fichero /etc/group.

Como pasaba con los usuarios, cada una de las filas hará referencia a un grupo distinto, y constará de cuatro campos separados de nuevo por el símbolo ":", con el siguiente orden:

1. **Nombre del grupo.** Es el nombre del grupo asociado a su identificador.
2. **Contraseña.** Aunque no se suele usar, como en los usuarios, se encuentra marcada con "x" que hace referencia a que se encuentran almacenadas en el fichero /etc/gshadow
3. **Identificador de grupo.** GID o número de identificación de grupo único.
4. **Lista de usuarios.** Se listan los usuarios que pertenecen a dicho grupo como grupo secundario.

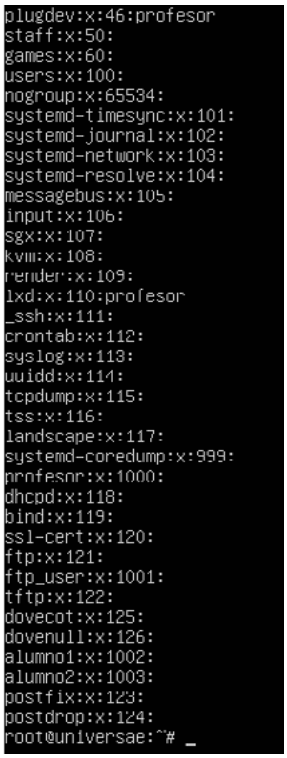


Imagen 5. /etc/group.

Si nos fijamos en el grupo alumno, de la imagen anterior:

Estructura del fichero /etc/group			
alumno	x	1000	
Nombre del grupo	Contraseña	GID	Usuarios pertenecientes al grupo.

Para identificarnos como usuario root del sistema deberíamos de ejecutar el siguiente comando en Debian:

```
sudo -i
```

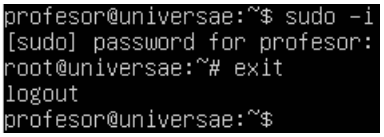


Imagen 6. Usuario root.

Y, como podemos ver, una vez que se haya establecido la contraseña, estaremos *logueados* como el superusuario del sistema y podremos ejecutar tareas administrativas. Si queremos salir de la sesión usaremos el comando **exit**.

1.3.1. Añadir usuarios

Si queremos añadir en Linux un usuario nuevo por la línea de comandos, como superusuario ejecutaremos el comando `useradd` con la siguiente estructura:

```
useradd [-g grupo] [-G grupo[, grupo …]] [-d directorio_
home [-m]] [-p contraseña_encriptada] [-s shell] login
```

Este comando añadirá una línea nueva al fichero */etc/passwd* con los datos aportados en el comando y además copiará los archivos del directorio */etc/skel*, que es el que almacena por defecto los archivos de configuración del directorio de trabajo de un usuario común. Las opciones que más usa este comando son:

- **g grupo;** se usa para asignar el grupo principal del usuario. Todos los usuarios tienen por lo menos un grupo principal al que se pertenece, y todos los demás serán grupos secundarios. Si no se especifica esta opción, habrá un grupo por defecto con el mismo nombre que el del usuario.
- **G grupo:** se listan todos los grupos secundarios, separados por comas y sin espacios.
- **d directorio_home:** se establece el directorio *home* del usuario, donde el usuario trabajará de manera normal. Si no se especifica nada, se usará el directorio */home/nombre_usuario*.
- **p contraseña_encriptada:** se especifica la contraseña de usuario que se encriptará para que no se pueda descubrir. Si no se especifica, no se podría *loguear* con este usuario al sistema.
- **m:** si no existe o no se especifica el directorio, lo crea y se copian los archivos de */etc/skel*.
- **s shell:** indica cual será el *shell* por defecto del usuario a la hora de la ejecución de los comandos.

En el siguiente ejemplo creamos un usuario *profesor_suplente* sin especificar opciones:

```
root@universae:~# useradd profesor_suplente
root@universae:~# _
```

Imagen 7. Comando useradd.

```
pepe:x:1004:1004::/home/pepe:/bin/sh
profesor_suplente:x:1005:1005::/home/profesor_suplente:/bin/sh
root@universae:~#
```

Imagen 8. Nueva línea en /etc/passwd.

Podemos ver que en el ejemplo anterior se ha añadido una línea al fichero */etc/passwd* y que su directorio *home* se ha situado en */home/profesor_suplente.*

1.3.2. Modificar usuarios

Los usuarios de Linux no solo se pueden crear por la línea de comandos, sino que también se pueden modificar usando el comando `usermod`. La sintaxis de este comando es la siguiente:

```
usermod [-c comentario] [-g grupo] [-G grupo[, grupo …]] [-d directorio_home] [-m]] [-p contraseña_encriptada] [-e fecha] [-f días] [-l nuevologin] [-L] [-U] [-s shell] login
```

Dentro de este comando hay muchas opciones que ya hemos descrito en el comando anterior, y las nuevas que se incorporan son:

- `c comentario`: establece los valores asociados al quinto campo de la línea añadida al fichero */etc/passwd*.
- `e fecha`.
- `f días`.
- `l nuevologin`: se cambia el *login* anterior por uno nuevo que se aporte.
- `L`.
- `U`.

En el siguiente ejemplo le asignamos al usuario *profesor_suplente* el directorio de trabajo */home/probando*.

```
root@universae:~# usermod profesor_suplente -d /home/probando -m
root@universae:~# _
```

Imagen 9. Comando usermod.

```
pepe:x:1004:1004::/home/pepe:/bin/sh
profesor_suplente:x:1005:1005::/home/probando:/bin/sh
root@universae:~#
```

Imagen 10. Línea modificada en /etc/passwd.

Vemos en el fichero */etc/passwd* que se ha cambiado el directorio por el que nosotros hemos especificado.

1.3.3. Eliminar usuarios

Si, por último, queremos eliminar a un usuario lo haremos con el comando `userdel` que sigue la siguiente sintaxis:

```
userdel [-r] login
```

La opción `-r` hace que también se borre el directorio *home* del usuario.

Vamos a borrar ahora el usuario *pepe*.

```
root@universae:~# userdel pepe
root@universae:~# _
```

Imagen 11. Comando userdel.

```
alumno1:x:1002:1002:,,,:/home/alumno1:/bin/bash
alumno2:x:1003:1003:,,,:/home/alumno2:/bin/bash
postfix:x:117:123::/var/spool/postfix:/usr/sbin/nologin
profesor_suplente:x:1005:1005::/home/probando:/bin/sh
root@universae:~# _
```

Imagen 12. /etc/passwd.

Y podemos ver en la imagen como este ya no aparece en el fichero */etc/passwd*.

1.3.4. Cambiar contraseñas

Una vez que hemos creado o modificado los usuarios, para poder modificar su contraseña debemos de usar un comando especifico, el comando `password`.

En el siguiente ejemplo podemos ver como se modifica la contraseña del usuario *alumno1:*

```
root@universae:~# passwd alumno1
New password:
Retype new password:
passwd: password updated successfully
root@universae:~# _
```

Imagen 13. Comando passwd

1.4. Configurar la zona horaria y el idioma

Zona horaria

Si queremos comprobar cuál es la zona horaria del sistema además de la hora, en *Ubuntu* lanzaremos el comando `date`:

```
root@universae:~# date
jue 21 jul 2022 10:57:45 UTC
root@universae:~#
```

Imagen 14. Comando date

Lo más sencillo si queremos modificar esta zona horaria es cambiar el enlace simbólico de */etc/localtime* a */usr/share/zoneinfo/continente/ciudad*.

```
root@universae:~# cat /etc/localtime
TZif2UTCTZif2UTC
UTC0
root@universae:~#
```

Imagen 15. Contenido de */etc/localtime*

Esta acción, se puede realizar con el comando:

```
ln -sfn /usr/share/zoneinfo/continente/ciudad /etc/localtime
```

```
profesor@universae:~$ sudo ln -sfn /usr/share/zoneinfo/Europe/Madrid /etc/localtime
[sudo] password for profesor:
profesor@universae:~$ timedatectl
               Local time: jue 2022-07-21 13:02:00 CEST
         Universal time: jue 2022-07-21 11:02:00 UTC
               RTC time: jue 2022-07-21 11:02:01
              Time zone: Europe/Madrid (CEST, +0200)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
profesor@universae:~$ _
```

Imagen 16. Configuración de zona horaria

Para comprobar que la fecha o la zona horaria es la que queremos, el otro comando que podemos usar es `timedatectl`, comando que solo puede usarse en algunas distribuciones de Linux, en el caso de *Ubuntu*, lo encontramos.

```
profesor@universae:~$ timedatectl
               Local time: jue 2022-07-21 13:08:27 CEST
         Universal time: jue 2022-07-21 11:08:27 UTC
               RTC time: jue 2022-07-21 11:08:27
              Time zone: Europe/Madrid (CEST, +0200)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
profesor@universae:~$ _
```

Imagen 17. Comando timedatectl

Además, nos muestra las distintas horas, si el servidor sincroniza y la ultima hora de sincronización.

Podemos también cambiar la zona horaria con dicho comando usando las opciones:

```
timedatectl set-timezone Continente/Ciudad
```

```
profesor@universae:~$ timedatectl set-timezone Europe/Madrid
profesor@universae:~$
```

Imagen 18. Configurar zona horaria con timedatectl

Idioma

Si queremos cambiar el idioma de nuestro servidor Linux, el comando que debemos de usar es `localectl`, que, sin opciones o argumentos, nos muestra la información del idioma actualmente configurado.

```
profesor@universae:~$ localectl
   System Locale: LANG=es_ES.UTF-8
       VC Keymap: n/a
      X11 Layout: es
       X11 Model: pc105
profesor@universae:~$ _
```

Imagen 19. Comando localectl

Para ver los idiomas que tenemos disponibles para la configuración, el comando a usar es el siguiente:

```
localectl list-locales
```

```
profesor@universae:~$ localectl list-locales
C.UTF-8
es_ES.UTF-8
profesor@universae:~$
```

Imagen 20. Listado de idiomas disponibles

A la hora de añadir nuevos idiomas para su posterior configuración, debemos de usar como administradores el comando:

```
locale-gen idioma_PAÍS.codificación
```

En este comando, la parte opcional se encuentra en siglas o con las primeras letras.

```
profesor@universae:~$ sudo locale-gen en_IN.utf8
Generating locales (this might take a while)...
  en_IN.UTF-8... done
Generation complete.
profesor@universae:~$ _
```

Imagen 21. Añadiendo el inglés de Inglaterra como idioma

Una vez instalados los idiomas podemos configurarlo como idioma principal con el comando:

```
localectl set-locale LANG=idioma_PAÍS.codificación
```

```
profesor@universae:~$ localectl set-locale LANG=en_IN.utf8
==== AUTHENTICATING FOR org.freedesktop.locale1.set-locale ===
Authentication is required to set the system locale.
Authenticating as: Profesor (profesor)
Password:
==== AUTHENTICATION COMPLETE ===
profesor@universae:~$
```

Imagen 22. Configurando inglés de Inglaterra como idioma

1.5. Administración de aplicaciones

En Linux tenemos gestores de paquetes que nos permiten instalarlos, administrarlos, desinstalarlos o actualizarlos. Estos gestores, aunque se pueden usar de modo gráfico, también existen en la línea de comandos de Linux.

Realmente este método es el más usado por los administradores ya que es el más automático y sencillo. Los principales comandos son:

- `dpkg.` Este comando gestiona los paquetes **.deb** y nos permite de manera manual instalar aplicaciones de Debian.
- `apt-get.` Este comando de bajo nivel instala paquetes y dependencias automáticamente.
- `apt-cache.` Este comando que también es de bajo nivel nos permite consultar la caché de **apt**.
- `apt.` Es la versión más moderna de **apt-get**.

1.5.1. Apt-get

La principal herramienta para gestionar paquetes en Linux desde el terminal es `apt-get`. Lo debemos considerar de mayor bajo nivel en comparación con otros sistemas de gestión de paquetes. Para poder descargar dichos paquetes o sus actualizaciones, este comando se basa directamente en los repositorios que se encuentran almacenados en la ubicación */etc/apt/sources.list*.

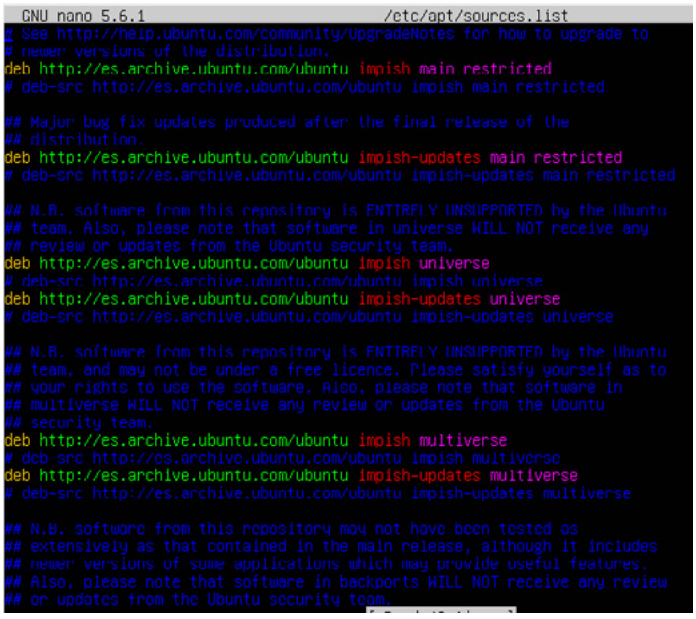


Imagen 23. Fichero */etc/apt/sources.list*

Las principales opciones de `apt-get` son:

- `apt-get update.` Se encarga de la actualización de los repositiorios de nuestro sistema.
- `apt-get install.` Instala paquetes, puede ser uno o varios.
- `apt-get upgrade.` Actualiza los paquetes y programas que tenemos instalados fijándose en los repositorios.
- `apt-get dist-upgrade.` Se encarga de actualizar todos los paquetes instalados.
- `apt-get remove.` Con este comando podemos desinstalar paquetes, pero sus archivos de configuración permanecen.
- `apt-get purge.` Al igual que el anterior, desinstalar los programas, pero también borra sus archivos de configuración.
- `apt-get autoremove.` Nos desinstala del sistema las dependencias de paquetes que eran necesarias anteriormente, pero no en la actualidad.

```
root@universae:~# apt-get remove nginx
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libgd3 libnginx-mod-http-geoip2 libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter
  libnginx-mod-mail libnginx-mod-stream libnginx-mod-stream-geoip2 libxpm4 nginx-common nginx-core
Utilice «apt autoremove» para eliminarlos.
Los siguientes paquetes se ELIMINARÁN:
  nginx
0 actualizados, 0 nuevos se instalarán, 1 para eliminar y 0 no actualizados.
Se liberarán 49,2 kB después de esta operación.
¿Desea continuar? [S/n] s
(Leyendo la base de datos ... 124910 ficheros o directorios instalados actualmente.)
Desinstalando nginx (1.18.0-6ubuntu11.1) ...
root@universae:~# apt-get purge nginx
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
El paquete «nginx» no está instalado, no se eliminará
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libgd3 libnginx-mod-http-geoip2 libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter
  libnginx-mod-mail libnginx-mod-stream libnginx-mod-stream-geoip2 libxpm4 nginx-common nginx-core
Utilice «apt autoremove» para eliminarlos.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
root@universae:~# _
```

Imagen 24. apt-get remove

1.5.2. Dpkg

Además de tener el comando `apt-get`, que nos ayuda en muchas tareas, tenemos el comando `dpkg` disponible en los servidores de Linux. Este comando se usa para instalar programas de manera manual, porque en los repositorios no lo encontramos y solo conocemos el archivo **.deb** que 2tenemos para la instalación. Su sintaxis es la siguiente:

```
dpkg -i fichero.deb
```

La opción `-i` se usa para instalar paquetes, pero podemos usar además estas otras dos:

- `-r:` borra el paquete.
- `-l:` lista el contenido del paquete.

1.5.3. Apt

Con respecto a `apt`, esta instrucción realmente es una variante de `apt-get` que nos resulta más sencilla. Auqnu8e se sigue usando su versión anterior, `apt` cada vez se usa más.

`apt-get` es un comando que tiene muchísimos usos, pero como realmente no se suelen usar casi nunca, `apt` no incluye ciertas opciones, incluyendo solo las más usadas con la intención de mejorar en sencillez.

1.6. Administración de discos

Una de las principales labores de un administrador de sistemas es manejar los discos en Linux, es decir, la creación y administración de las particiones de los discos duros donde alojemos el sistema. Tenemos buenas herramientas para realizar estas acciones tanto de manera gráfica como mediante comandos.

De igual modo, siempre hay que tener en cuenta que *Ubuntu Server* no cuenta con ninguna GUI, por lo que en dicho sistema solo se puede trabajar por terminal.

1.6.1. Fdisk

Una de las principales herramientas que nos encontramos en la terminal de *Ubuntu* es `fdisk`. Este comando nos permite crear y administrar tablas de particiones en distintos formatos. Su interfaz es bastante sencilla, aunque basada en texto y que nos muestra distintos menús dependiendo de las opciones que vayamos eligiendo.

Para acceder a `fdisk` debemos de ejecutar el comando del siguiente modo:

```
fdisk disco
```

```
profesor@universae:~$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.36.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x3272ac74.

Command (m for help):
```

Imagen 25. fdisk

Una vez que nos encontramos en `fdisk`, si lanzamos la opción `m` se nos muestra la ayuda, con distintas opciones que realmente nos indican si queremos mostrar, crear, redimensionar, eliminar, modificar, copiar o mover las particiones en la que se divide el disco seleccionado.

```
DOS (MBR)
 a   toggle a bootable flag
 b   edit nested BSD disklabel
 c   toggle the dos compatibility flag

Generic
 d   delete a partition
 F   list free unpartitioned space
 l   list known partition types
 n   add a new partition
 p   print the partition table
 t   change a partition type
 v   verify the partition table
 i   print information about a partition

Misc
 m   print this menu
 u   change display/entry units
 x   extra functionality (experts only)

Script
 I   load disk layout from sfdisk script file
 O   dump disk layout to sfdisk script file

Save & Exit
 w   write table to disk and exit
 q   quit without saving changes

Create a new label
 g   create a new empty GPT partition table
 G   create a new empty SGI (IRIX) partition table
 o   create a new empty DOS partition table
 s   create a new empty Sun partition table
```

Imagen 26. Opciones de fdisk

1.6.2. GNU Parted

Otra de las herramientas que nos brinda Linux para poder administrar las particiones en las que dividimos los discos es **parted**. Al igual que el anterior comando, se puede usar con distintos formatos de tablas de particiones. Esta herramienta nos permite agregar, eliminar, y modificar el tamaño de las distintas particiones al mismo tiempo que trabaja con los sistemas de ficheros que podemos ubicar en dichas particiones.

Para iniciar la herramienta, el comando a usar con permisos de administrador es **parted**, y tiene el siguiente aspecto:

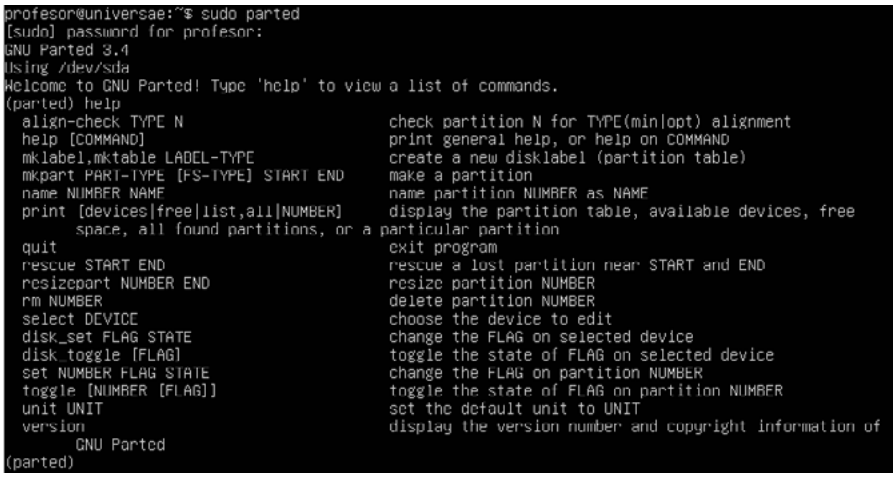


Imagen 27. Parted

1.6.3. Gparted

Como herramienta gráfica de los sistemas Linux para la administración de las particiones podemos contar con **Gparted** que es bastante avanzado y funciona en prácticamente cualquier tipo de sistema operativo.

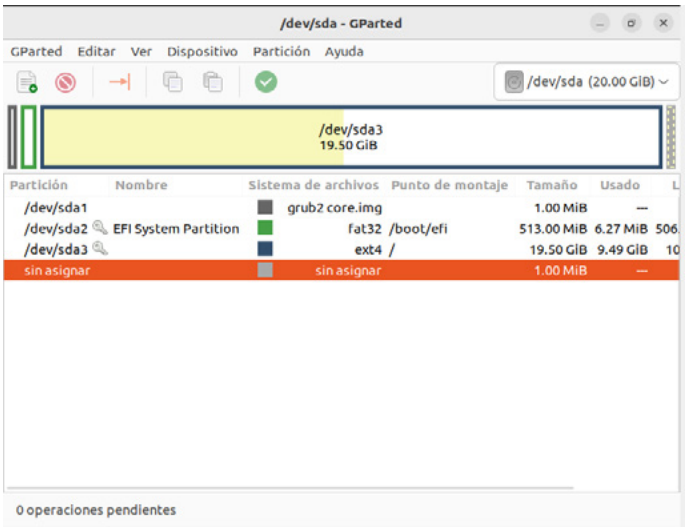


Imagen 28. *Gparted*

1.6.4. Discos de GNOME

Podemos crear particiones y formatearlas en distintas unidades para después montarlas y desmontarlas con otra utilidad gráfica, *Discos de GNOME*. Esta es la herramienta que por defecto llevan todos los entornos gráficos *GNOME*.

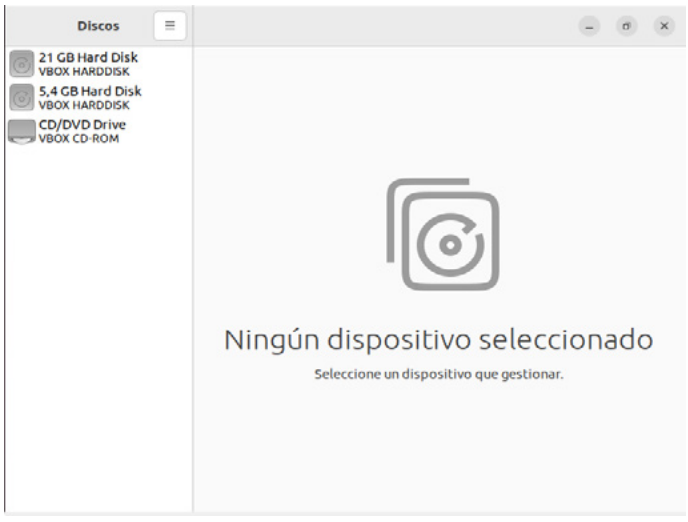


Imagen 29. *Discos de GNOME*

1.7. Monitor de recursos

Para los sistemas como *Ubuntu* contamos una de las mejores herramientas que se pueden usar para monitorizar el rendimiento del equipo, **top**. Este comando es muy práctico y nos permite ver el rendimiento de nuestro sistema en tiempo real y sin interfaz gráfica.

Cualquier distribución de Linux lleva incorporado **top**, pero hay versiones aún más sofisticadas de dicho comando que se deben instalar manualmente.

Para lanzar el comando debemos de escribir en la terminal: **top**.

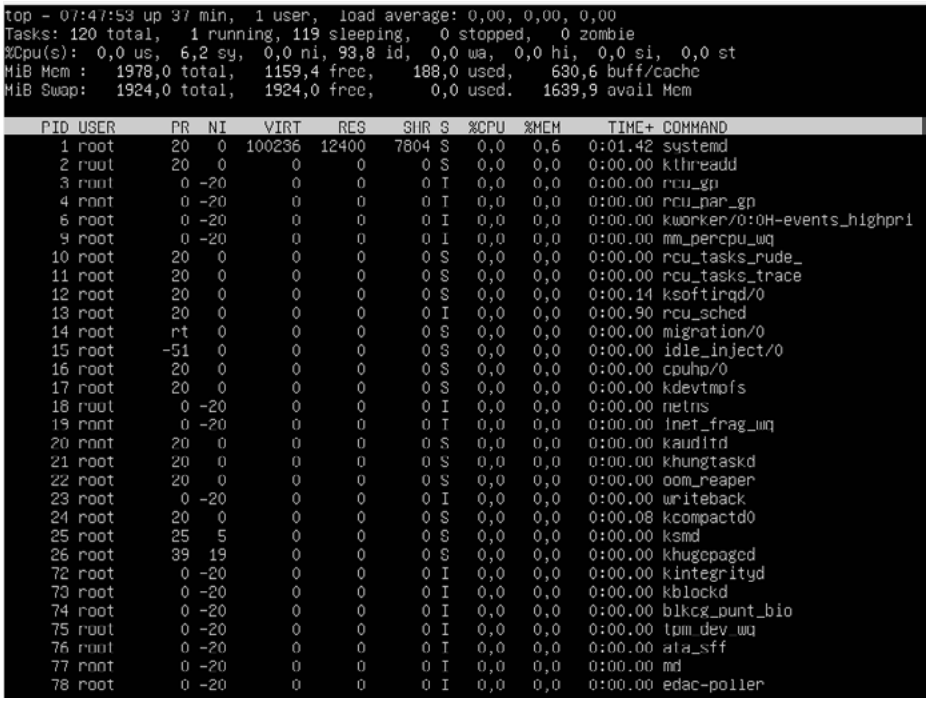


Imagen 30. top

Como podemos ver en la anterior imagen, este comando nos muestra los usuarios, el tiempo de ejecución, la cantidad de tareas y su estado. Además, nos muestra el porcentaje de recursos que cada proceso está consumiendo, su *PID* para su administración, y demás cosas.

Es interesante destacar que **top** tiene múltiples opciones entre las que debemos destacar es que da la opción a ordenar la salida por *PID, %CPU, %MEM*, tiempo de ejecución etc.

Otra de las opciones que no da esta herramienta es la opción **-d**, para indicar cada cuantos segundos queremos que se actualice el estado de los procesos. Si no indicamos nada, se deja por defecto, que sería cada 3 segundos.

Los principales estados de los procesos en Linux, a través de **top** son:

Estados de un proceso	
Estados	**Descripción**
Running (R)	Procesos en ejecución
Sleeping (S)	Procesos que se encuentran esperando a que llegue su turno de ejecución.
Stopped (D)	Procesos que esperan a que finalicen operaciones de entrada/salida.
Zombie (Z)	Procesos que han terminado pero su proceso padre aún no ha detectado dicho fin.

Con respecto al entorno gráfico, tenemos muchísimas herramientas para poder evaluar cómo se encuentra nuestro sistema de manera más amigable, o por lo menos más visual.

Hay herramientas, como el *Monitor del sistema de Ubuntu*, que viene por defecto, pero existen otras que muestran más información y que tendrán que ser instaladas dependiendo de lo que necesitemos.

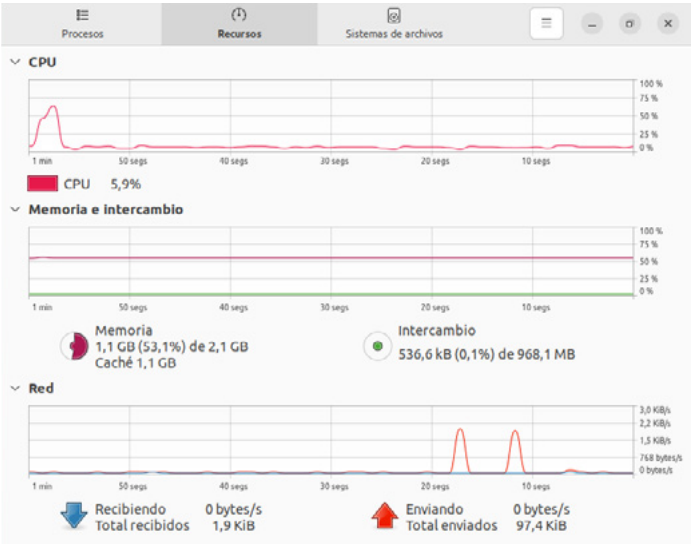


Imagen 31. Monitor del sistema de Ubuntu

1.8. Permisos

En Linux tenemos tres niveles de permisos de manera principal, que son los siguientes:

- Permisos del propietario
- Permisos del grupo
- Permisos del resto de usuarios

Los permisos del propietario son los permisos que tiene el creador del archivo o directorio en primera instancia cuando se crean, estos también pueden cambiar ya sea porque cambiemos el propietario o porque se cambien los permisos de este.

En Linux es común que los usuarios pertenezcan a distintos grupos de trabajo ya que por defecto se añade un nuevo usuario a ciertos grupos. Estos permisos se aplican a todos los usuarios de un grupo predefinido para el archivo. Si no se pone nada, el grupo que se pone por defecto es el de por defecto del usuario propietario.

Estos son los permisos que afectan al resto de usuarios que no forman parte del grupo en cuestión antes mencionado ni son el propietario.

Para ver los permisos por ejemplo de los directorios del home de un usuario lanzamos el comando **ls -l**.

```
profesor@universae:/home$ ls -l
total 16
drwxr-x--- 4 alumno1  alumno1  4096 jul 15 12:19 alumno1
drwxr-x--- 2 alumno2  alumno2  4096 jul 15 07:35 alumno2
drwxr-x--- 2 ftp_user ftp_user 4096 jul  7 09:39 ftp_user
-rw-r--r-- 1 root     root        0 jul 26 09:02 probando
drwxr-x--- 7 profesor profesor 4096 jul 26 07:47 profesor
profesor@universae:/home$
```

Imagen 32. ls -l

Como podemos ver en la imagen de más arriba los permisos se dividen en grupos de tres en el mismo orden que se ha establecido al principio: propietarios, grupo, otros.

Además, vemos que hay hasta tres letras, *r, w y x:*

- R: es el permiso de lectura que permite que se lea el contenido del archivo en cuestión. No es suficiente para que se pueda ver el contenido de un directorio.
- W: es el permiso de escritura que nos permite hacer cambios dentro de un directorio o de un archivo en concreto.
- X: es el permiso de ejecutar que nos permite ejecutar programas o script de Linux. Además, es necesario que se acompañe junto al permiso de lectura si se quiere ver el contenido de un directorio.

También podemos ver que salen dos nombres justo después, el primero hace referencia al usuario propietario y el segundo hace referencia al grupo propietario.

1.8.1. Establecer permisos

Para cambiar los permisos de usuario en Linux, usaremos el comando `chmod`, pero tenemos dos vertientes, una usando letras y otra dotación numérica.

Con letras, tenemos claro que los permisos que cambiaremos son los de los tres grupos de arriba nombrados.

Para esto necesitaremos saber que se identifican del siguiente modo: *u, dueño; g, grupo y o, otros.*

Ahora que ya tenemos en cuenta esto, vamos a cambiar los permisos del archivo prueba de modo que el grupo y otros usuarios también puedan modificar.

1. Abrimos un terminal y *logueamos* como root.
2. Lanzamos el siguiente comando:

   `chmod u/g/o+-r/w/x archivo/directorio`

Imagen 33. Cambiar permisos de Linux usando letras

3. Como podemos ver, se pone todo de manera consecutiva, por ejemplo, hemos puesto *go* para hacer referencia al grupo y a otros usuarios y *+w* para que quede claro que se añade el permiso de escritura.
4. Cuando hemos lanzado la opción *ls -l*, vemos que los permisos han sido cambiados.
5. Si, por otro lado, quisiéramos quitar de nuevo estos permisos, el comando sería:

   `chmod go-w prueba`

Ahora vamos a ver como se cambian los permisos en Linux con el método numérico. Para esto debemos de saber lo primero que cada permiso se asocia con un número:

- R → 4
- W → 2
- X → 1

El comando total queda del siguiente modo:

`chmod n1n2n3 archivo,`

Donde n1 son los permisos para el propietario, n2 para el grupo y n3 para los otros usuarios.

Cada uno de los números lo sacaremos de la suma de cada uno de los números de los permisos, por eso, el máximo permiso que se puede otorgar es el *777*.

El ejemplo anterior en esta numeración sería:

`chmod 666 prueba`

1.8.2. Cambiar propietario y grupo

Además de poder cambiar los permisos de los archivos, también podemos interactuar sobre los propietarios de los ficheros.

Cada fichero tiene dos propietarios, un usuario y un grupo.

Si lo que queremos es cambiar el propietario de un fichero. El comando que debemos de usar es `chown`. Este comando funciona con la siguiente sintaxis:

```
chown usuario fichero
```

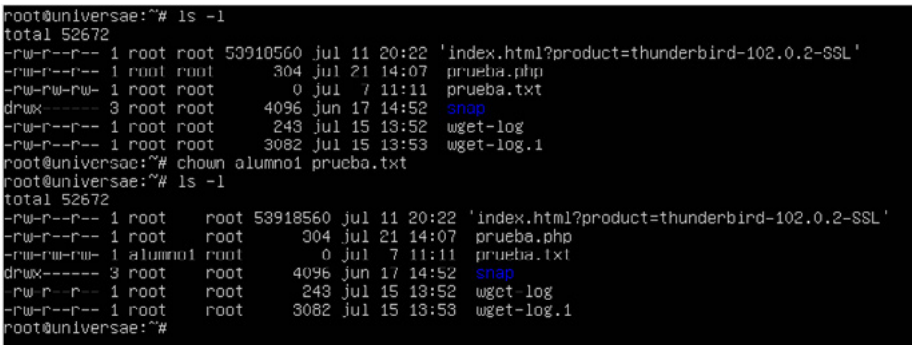


Imagen 34. Funcionamiento de chown

Por otro lado, si lo que queremos es cambiar el grupo propietario de un fichero en Linux lo que haremos será ejecutar el comando `chgrp` con la siguiente sintaxis:

```
chgrp grupo fichero
```

Imagen 35. Funcionamiento de chgrp

Para ambos comandos podemos indicar varios ficheros a los que queremos cambiarles las propiedades.

También para los dos podemos usar la opción `-R` para la recursividad, es decir, que se apliquen los cambios a anteriores directorios cuando nombramos una ruta completa.

1.9. Gestión y control de procesos

Para gestionar todos los procesos, el sistema operativo recurre a ciertas operaciones de creación, comunicación, compartición y finalización de estos. El módulo del sistema que se encarga de esto es el planificador de procesos.

Los procesos pasan por distintos estados antes de finalizar y es el planificador de procesos el que se encarga de la gestión de estos y de que se modifique su estado mediante un algoritmo de planificación.

Cuando usamos un sistema operativo multiproceso en el que tenemos una serie de procesos ejecutándose al mismo tiempo en un único procesador es muy probable que muchos de ellos estén bloqueados por falta de recursos o porque simplemente no haya ocurrido la operación necesaria para que el proceso se reactive Los procesos tienen distintos estados que se suelen explicar mediante un diagrama de proceso que muestra cómo van cambiando los procesos según las necesidades que el Sistema operativo impere.

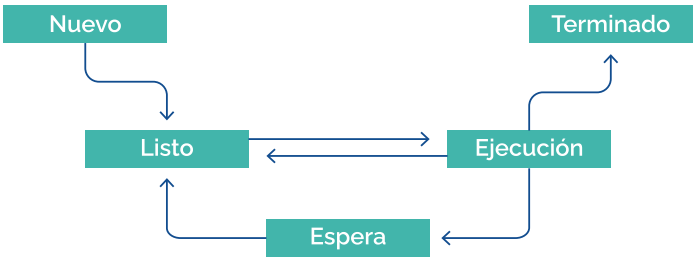


Imagen 36. Estados de un proceso.

Como se ve en el diagrama anterior, un proceso puede pasar por múltiples estados:

- **Nuevo.** Este estado es el primer estado que puede tener un proceso, y se acaba de crear ya sea porque se ha abierto un programa o porque se ha abierto un fichero. Un proceso nuevo no es válido por sí solo ya que carece de los recursos necesarios.
- **Listo o preparado (ready).** Este estado es cuando el proceso ya está listo, pero a falta de que el procesador comience su ejecución. Cuando un proceso está en este estado puede que haya dos vertientes, o que se ponga en ejecución, o que directamente termine porque al final no sea necesario su ejecución o porque sea necesaria su finalización para que otro proceso suceda.
- **Ejecución (run).** En estado, el proceso se encuentra trabajando en la CPU y ejecutando instrucciones. Hay tres vertientes: el proceso puede ejecutar todas y cada una de las instrucciones que se le han dado hasta que termine y finalice; también puede que el proceso se bloquee por necesidades del usuario o del propio sistema; por último, puede que el proceso vuelva directamente a estar en preparado porque su tiempo de ejecución se ha excedido y no ha empezado.

IMPORTANTE

Es muy importante que un administrador del sistema conozca la política de planificación de procesos del sistema que administra pues esta hará que unos procesos se ejecuten antes o después y esto puede o bien beneficiar al sistema o perjudicarlo.

- **Espera o bloqueado (wait).** El proceso se encuentra a la espera de que suceda cualquier cambio que lo haga o volver a estar preparado, o entrar en ejecución directamente e incluso finalizar si fuese necesario.
- **Terminado.** Este es el estado en el cual se encuentra el proceso una vez que ha pasado por los demás estados y es cuando desaparece, ya no tiene retorno.

El sistema operativo gestiona los procesos mediante colas, pero cada una es diferente debido a que puede haber una por ejemplo para los procesos nuevos y otra para los que están en espera.

Es tarea del planificador del procesador el revisar estas colas y decidir qué proceso se ejecuta además de cuándo, cómo, con qué recursos y dónde.

1.9.1. Listar procesos

Los procesos se identifican gracias a un identificador único denominado *PID, Identificador de proceso*. El *PCB* de cada proceso almacena información acerca de este, principalmente:

- El PID del proceso.
- Identificación del proceso padre, PPID.
- Usuario propietario.
- Valores del estado del proceso en el momento de producirse el cambio de contexto.
- Estado.
- Valores de referencia de memoria RAM.
- Ficheros abiertos.
- Buffers de memoria usados.

Para obtener información acerca de los procesos del sistema usaremos el comando:

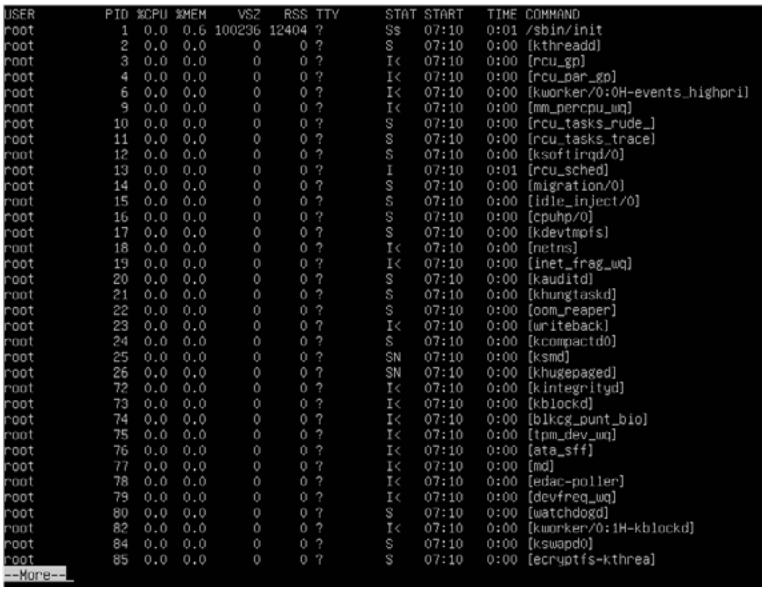
```
ps [modificadores]
```

Este comando tiene un sinfín de opciones y una potencia mayor aún, por lo que para poder verlos todos debemos de acceder a su manual de ayuda con el comando:

```
man ps.
```

No obstante, los principales modificadores son:

- Para obtener información de todos los procesos del sistema:
  - ps aux
  - ps -ef
- Para imprimir información junto a un árbol de procesos.
  - ps axjf



```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.6 100236 12404 ?        Ss   07:10   0:01 /sbin/init
root           2  0.0  0.0      0     0 ?        S    07:10   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   07:10   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   07:10   0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<   07:10   0:00 [kworker/0:0H-events_highpri]
root           9  0.0  0.0      0     0 ?        I<   07:10   0:00 [mm_percpu_wq]
root          10  0.0  0.0      0     0 ?        S    07:10   0:00 [rcu_tasks_rude_]
root          11  0.0  0.0      0     0 ?        S    07:10   0:00 [rcu_tasks_trace]
root          12  0.0  0.0      0     0 ?        S    07:10   0:00 [ksoftirqd/0]
root          13  0.0  0.0      0     0 ?        I    07:10   0:01 [rcu_sched]
root          14  0.0  0.0      0     0 ?        S    07:10   0:00 [migration/0]
root          15  0.0  0.0      0     0 ?        S    07:10   0:00 [idle_inject/0]
root          16  0.0  0.0      0     0 ?        S    07:10   0:00 [cpuhp/0]
root          17  0.0  0.0      0     0 ?        S    07:10   0:00 [kdevtmpfs]
root          18  0.0  0.0      0     0 ?        I<   07:10   0:00 [netns]
root          19  0.0  0.0      0     0 ?        I<   07:10   0:00 [inet_frag_wq]
root          20  0.0  0.0      0     0 ?        S    07:10   0:00 [kauditd]
root          21  0.0  0.0      0     0 ?        S    07:10   0:00 [khungtaskd]
root          22  0.0  0.0      0     0 ?        S    07:10   0:00 [oom_reaper]
root          23  0.0  0.0      0     0 ?        I<   07:10   0:00 [writeback]
root          24  0.0  0.0      0     0 ?        S    07:10   0:00 [kcompactd0]
root          25  0.0  0.0      0     0 ?        SN   07:10   0:00 [ksmd]
root          26  0.0  0.0      0     0 ?        SN   07:10   0:00 [khugepaged]
root          72  0.0  0.0      0     0 ?        I<   07:10   0:00 [kintegrityd]
root          73  0.0  0.0      0     0 ?        I<   07:10   0:00 [kblockd]
root          74  0.0  0.0      0     0 ?        I<   07:10   0:00 [blkcg_punt_bio]
root          75  0.0  0.0      0     0 ?        I<   07:10   0:00 [tpm_dev_wq]
root          76  0.0  0.0      0     0 ?        I<   07:10   0:00 [ata_sff]
root          77  0.0  0.0      0     0 ?        I<   07:10   0:00 [md]
root          78  0.0  0.0      0     0 ?        I<   07:10   0:00 [edac-poller]
root          79  0.0  0.0      0     0 ?        I<   07:10   0:00 [devfreq_wq]
root          80  0.0  0.0      0     0 ?        S    07:10   0:00 [watchdogd]
root          82  0.0  0.0      0     0 ?        I<   07:10   0:00 [kworker/0:1H-kblockd]
root          84  0.0  0.0      0     0 ?        S    07:10   0:00 [kswapd0]
root          85  0.0  0.0      0     0 ?        S    07:10   0:00 [ecryptfs-kthrea]
--More--
```

Imagen 36. Salida comando ps aux

El comando `ps aux` muestra una serie de información que se establece en la cabecera:

- Usuario propietario del proceso.
- PID del proceso.
- CPU consumida en porcentaje.
- Memoria RAM consumida en porcentaje.
- Tamaño del proceso en la memoria virtual en KB.
- Tamaño de la memoria residente de proceso en KB.
- Terminal de lanzamiento.
- Estado del proceso.
- Tiempo de inicio del proceso.
- Tiempo de CPU consumido.
- Comando que lo ejecuta.

1.9.2. Estados de un proceso

Los estados de un proceso en el comando `ps aux` pueden ser los siguientes:

Estados de un proceso	
Estado	**Descripción**
R	Ejecutándose o listo para ser ejecutado. (*Runnable*)
S	Bloqueado o durmiendo (*Sleeping*).
T	Parado (*Trace*).
Z	Zombi (proceso muerto pero el proceso padre no ha detectado su final).
I	Inactivo en creación (*idle*)
N	Con prioridad menor de lo normal (*NICE*).
<	Con prioridad mayor de lo normal.
+	Se encuentra en el grupo de procesos en primer plano.
s	Proceso líder de sesión.
L	Proceso multihilo.

1.9.3. Prioridades

A la hora de gestionar los recursos del sistema, Linux se basa en el administrador de procesos para determinar que procesos deben de ir ejecutándose uno tras otro, es decir, cual debe de estar dentro del procesador o no. Esto se decide dependiendo del cálculo de prioridades.

Cada uno de los procesos de Linux se ejecutan con una prioridad que tienen establecida y que es un número entero. En prácticamente todos los sistemas de Linux, las prioridades o el valor de prioridad, va de -20 hasta 19, con -20 como lo más favorable o alta y 19 la menos favorable o la más baja.

En el comando `top` podíamos ver las propiedades de los procesos, pero con el comando `ps` y la opción `-l` también se pueden ver:

```
profesor@universae:~$ ps
    PID TTY          TIME CMD
   1797 tty1     00:00:00 bash
   2827 tty1     00:00:00 ps
profesor@universae:~$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000    1797     743  0  80   0 -  2841 do_wai tty1     00:00:00 bash
0 R  1000    2828    1797  0  80   0 -  3107 -      tty1     00:00:00 ps
profesor@universae:~$
```

Imagen 37. Salidas de ps & ps -l

La prioridad la vemos descrita en la columna *NI*.

Podemos deducir viendo la salida del comando anterior que para los procesos que inician los usuarios regulares, por lo general la prioridad predefinida es 0.

En tema de prioridades podemos usar el comando `nice` que se usa para ver el valor de prioridades predefinidas. También podemos usar dicho comando añadiéndole la opción `-n` para ejecutar un proceso con una prioridad diferente a la que tiene establecida. Si usamos valores positivos, sumamos a la prioridad, y si usamos valores negativos, restamos a la prioridad.

También puede darse el caso en el que haya procesos que se están ejecutando ya y que necesitan cambiarse de prioridad. Para poder cambiar prioridades con el proceso ya iniciado, usamos el comando `renice`. Aquí no indicamos cuanto deben os de mover la prioridad, sino directamente que prioridad queremos que tenga.

1.9.4. Árbol de procesos

En cualquier sistema Unix, los procesos se generan en base a un proceso "padre" ya creado mediante un mecanismo de clonación, generándose un proceso "hijo". Es decir, los procesos en Linux surgen de predecesores que generan otros procesos parta realizar una tarea distinta.

Dicha estructura de procesos desemboca en que la estructura que se forma en los procesos es la de un árbol jerárquico, donde podemos ver que hay procesos que generan otros. El comando más usado para ver dicha estructura de árbol es `pstree`.

Imagen 38. pstree

1.9.5. Demonios

Los demonios del sistema son distintos *scripts* que se encuentran como procesos cargados en memoria hasta que se les indique que deben ser ejecutados. Esto se traduce en que, al estar cargados en memoria, no ocupan capacidad de la CPU, por lo que sin importar los que tengamos podremos seguir trabajando con total normalidad.

Estos procesos se ejecutan en segundo plano o *background* y suelen tener asociado un *Shell script* que se almacena en el directorio */etc/init.d.* A través de dicho *script*, podemos iniciar, parar o ver el estado del demonio.

```
root@universae:~# cd /etc/init.d/
root@universae:/etc/init.d# ls
apache2                 dbus              keyboard-setup.sh  open-vm-tools   ssh
apache-htcacheclean     dovecot           kmod               plymouth        tftpd-hpa
apparmor                grub-common       lvm2               plymouth-log    udev
apport                  hwclock.sh        lvm2-lvmpolld      postfix         ufw
console-setup.sh        inetutils-inetd   multipath-tools    procps          unattended-upgrades
cron                    irqbalance        named              rsync           uuidd
cryptdisks              isc-dhcp-server   nginx              rsyslog         vsftpd
cryptdisks-early        iscsid            open-iscsi         screen-cleanup  x11-common
root@universae:/etc/init.d# _
```

Imagen 39. */etc/init.d*

Si queremos iniciar un demonio podemos usar el comando `systemctl` con las siguientes opciones:

- Para iniciar un demonio tenemos que usar la opción `start`.
- Para parar un demonio tendremos que usar el comando `stop`.
- Para reiniciar el demonio usaremos el comando `restart`. Reiniciar significa volver a cargar los ficheros de configuración, lo que hace ideal usar esta opción cuando se cambian los parámetros de configuración.

Los demonios son ideales a la hora de gestionar programas que son independientes a las sesiones de usuario, como procesos que queremos mantener durante la sesión o que queremos que se inicien al arrancar el sistema sin necesidad de ejecutarlos manualmente.

1.9.6. Procesos en primer plano

En los sistemas Linux, cualquier proceso que ejecutemos de manera manual se iniciará en primer plano. La entrada de la información vendrá del teclado y la salida la proporciona por pantalla. Con distintos comandos de redirección de ficheros podríamos redireccionar las salidas o las entradas de los procesos, pero si no se indica otra cosa, la salida es mediante el terminal o la *Shell*.

Si ejecutamos el comando `sleep`, esperaremos unos segundos, los que indiquemos hasta que se nos devuelva el control del terminal, porque debemos esperar a que termine el proceso.

```
root@universae:/etc/init.d# sleep 10
```

Imagen 40. Comando sleep con 10 segundos de espera

El *Shell* de Linux nos permite que no tengamos que esperar hasta que termine un proceso para poder empezar otro, lo que se conoce como proceso en ejecución en segundo plano o *background*.

1.9.7. Procesos en segundo plano

Los procesos en segundo plano se ejecutan sin estar conectados al terminal y si necesita que añadamos información de manera manual, el proceso se detendrá.

La principal ventaja del uso de procesos en segundo plano es que al mismo tiempo que el proceso se encuentra en ejecución, podemos lanzar otros procesos en primer plano o también en segundo sin necesidad de esperar a que termine el primero lanzado.

En las siguientes imágenes vemos un ejemplo de los procesos en segundo plano. Mandamos el comando `sleep 10` a segundo plano al añadirle al final `&`.

```
profesor@universae:~$ sleep 10 &
[1] 1657
profesor@universae:~$ _
```

Imagen 41. Ejecución del proceso sleep 10 en segundo plano

En este momento se nos muestra el número de proceso asignado para dicho proceso y además nos dice en que número se está ejecutando, en nuestro caso es el primer proceso en segundo plano. Nos da el control de la terminal de manera inmediata.

Cuando se termina el proceso, bien porque finalice, bien porque lo paremos, el *Shell* nos lo indica con la palabra *Done*.

```
profesor@universae:~$ sleep 10 &
[1] 1657
profesor@universae:~$ ^C
[1]+  Done                    sleep 10
profesor@universae:~$ _
```

Imagen 42. Fin del proceso en segundo plano

1.9.8. Cambiar procesos entre primer y segundo plano

Si queremos enviar un proceso en primer plano a segundo plano, lanzamos el comando `bg`.

```
profesor@universae:~$ sleep 10
^Z
[3]+  Stopped                 sleep 10
profesor@universae:~$ bg
[3]+ sleep 10 &
profesor@universae:~$
```

Imagen 43. Mover procesos a segundo plano

Podemos fijarnos en que primero hemos tenido que lanzar un *Ctrl+Z* para que nos deje escribir en la terminal, ya que suspende el proceso.

En este momento el proceso lazado se encuentra en segundo plano. Este comando, `bg`, se encarga de mover a segundo plano el proceso más reciente que hayamos suspendido. Si tenemos diferentes procesos suspendidos, debemos de usar el número trabajo del *Shell*, que es el número entre corchetes para poder diferenciarlos, quedando el comando del siguiente modo:

```
bg número_trabajo
```

```
profesor@universae:~$ sleep 100
^Z
[1]+  Stopped                 sleep 100
profesor@universae:~$ sleep 1000
^Z
[2]+  Stopped                 sleep 1000
profesor@universae:~$ bg 1
[1]- sleep 100 &
profesor@universae:~$ bg 2
[2]+ sleep 1000 &
profesor@universae:~$
```

Imagen 44. Mover varios procesos a segundo plano

Si queremos realizar la acción, al contrario, sería del mismo modo, pero con el comando `fg`.

```
profesor@universae:~$ fg 2
sleep 1000
```

Imagen 45. Mover procesos a primer plano

1.9.9. Mostrar procesos en segundo plano o suspendidos

Si hemos mandado varios procesos a segundo plano y no sabemos su número de trabajo, su estado o su nombre, podemos listarlos con el comando `jobs` de modo que se nos muestra dicha información por si la necesitamos.

```
profesor@universae:~$ jobs
[2]+  Stopped                 sleep 1000
[3]-  Running                 sleep 1000 &
profesor@universae:~$ _
```

Imagen 46. Comando jobs

1.9.10. Terminar un proceso

Dependiendo del tipo del sistema en que nos encontremos, es común cerrar por la fuerza aplicaciones o finalizar procesos. Al igual que tenemos procesos en segundo plano, podemos finalizar o terminar procesos que ya no necesitemos.

Tenemos tres comandos distintos para finalizar los procesos en la terminal de Linux:

- `kill`. Cuando queremos terminar un proceso que haya creado alguno de los usuarios usamos este comando. Su sintaxis es la siguiente:

  ```
  kill -9 PID
  ```

```
profesor@universae:~$ kill -9 1652
[3]-  Killed                  sleep 1000
profesor@universae:~$
```

Imagen 47. Comando kill

Este comando tiene varias opciones, pero de momento nos basta con conocer esta, la opción `-9`.

- `pkill`. Este comando se encarga de matar el proceso indicando su nombre, sin necesidad de indicar el *PID*. La sintaxis es:

  ```
  pkill proceso
  ```

- `killall`. Se sigue la misma sintaxis que en el anterior, pero la diferencia se encuentra en que además de finalizar el proceso indicado, finaliza también los procesos que dependen del que indicamos.

1.10. Automatización de tareas

Los sistemas Linux tienen la suerte de que las tareas se pueden almacenar y planificar a lo largo del tiempo o para que se ejecuten de manera recurrente gracias a *cron*. La utilidad *cron*, que cuenta con comandos asociados al terminal, se basa en un demonio que fijándose en ficheros de configuración del sistema preestablecidos ejecuta ciertas acciones sobre el mismo en periodos de tiempo previamente dictaminados

1.10.1. Cron

Hay varios archivos en los que se fija *cron,* pero el principal es */etc/crontab*.

Para poder trabajar con *cron* debemos de realizar lo siguiente:

1. Revisar si está instalado en nuestro sistema (Algunas distribuciones no lo llevan por defecto, pero se encuentra disponible en todas). Esto se realiza con el siguiente comando:

   `dpkg -l cron`

```
root@universae:~# dpkg -l cron
Deseado=desconocido(U)/Instalar/eliminaR/Purgar/retener(H)
| Estado-No/Inst/ficheros-Conf/desempaqUetado/medio-conF/medio-inst(H)/espera-disparo(W)/pendienTe-
|/ Err?-(ninguno)/requiere-Reinst (Estado,Err: mayúsc.-malo)
||/ Nombre         Versión          Arquitectura Descripción
+++-==============-================-============-=================================
ii  cron           3.0pl1-137ubuntu2 amd64        process scheduling daemon
lines 1-6/6 (END)
```

Imagen 48. Comando dpkg -l cron

2. Si no está instalado, lo instalaremos.

3. Si está instalado, comprobamos que se encuentra activo con el comando:

   `systemctl status cron`

```
root@universae:~# systemctl status cron
• cron.service - Regular background program processing daemon
     Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-07-26 13:51:49 CEST; 1h 1min ago
       Docs: man:cron(8)
   Main PID: 699 (cron)
      Tasks: 1 (limit: 2209)
     Memory: 448.0K
        CPU: 15ms
     CGroup: /system.slice/cron.service
             └─699 /usr/sbin/cron -f -P

jul 26 13:51:49 universae.lan cron[699]: (CRON) INFO (pidfile fd = 3)
jul 26 13:51:49 universae.lan cron[699]: (CRON) INFO (Running @reboot jobs)
jul 26 14:09:01 universae.lan CRON[1695]: pam_unix(cron:session): session opened for user root by (>
jul 26 14:09:01 universae.lan CRON[1696]: (root) CMD (   [ -x /usr/lib/php/sessionclean ] && if [ ! >
jul 26 14:09:01 universae.lan CRON[1695]: pam_unix(cron:session): session closed for user root
jul 26 14:17:01 universae.lan CRON[1756]: pam_unix(cron:session): session opened for user root by (>
jul 26 14:17:01 universae.lan CRON[1756]: pam_unix(cron:session): session closed for user root
jul 26 14:39:01 universae.lan CRON[1784]: pam_unix(cron:session): session opened for user root by (>
jul 26 14:39:01 universae.lan CRON[1785]: (root) CMD (   [ -x /usr/lib/php/sessionclean ] && if [ ! >
jul 26 14:39:01 universae.lan CRON[1784]: pam_unix(cron:session): session closed for user root
lines 1-21/21 (END)
```

Imagen 49. Comando systemctl status cron.

4. Si no está activo, lo activamos con el comando:

   ```
   systemctl start cron
   ```

5. Si sí que está activo, lo reiniciamos para empezar de cero con el comando:

   ```
   systemctl restart cron
   ```

6. Ya podemos empezar a trabajar con esta utilidad.

```
root@universae:~# systemctl restart cron
root@universae:~#
```

Imagen 50. Comando systemctl restart cron

El fichero */ect/crontab* solo puede ser ejecutado por el superusuario *root* y cuenta con la siguiente estructura.

```
root@universae:~# cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
# You can also override PATH, but by default, newer versions inherit it from the environment
#PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
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
)
#
root@universae:~#
```

Imagen 51. Fichero /etc/crontab.

Podemos ver que la estructura que se sigue para que se ejecute una orden esa la siguiente:

```
Minutos(0 – 59) Hora(0 – 23) Día del mes(1 – 31) Mes(1 – 12) Día de la semana(0 – 6 | día) usuario comando
```

Si no se especifica nada, y se pone en su lugar "*", esto quiere decir que recoge todos los valores.

1.10.2. Crontab

Si queremos editar el fichero */etc/crontab*, lo mejor es usar el comando `crontab -e`. Que abrirá un editor que nos permitirá añadir opciones siguiendo la sintaxis de los ficheros.

```
root@universae:~# crontab -e
no crontab for root - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/vim.tiny
  4. /bin/ed

Choose 1-4 [1]: _
```

Imagen 52. Comando crontab -e.

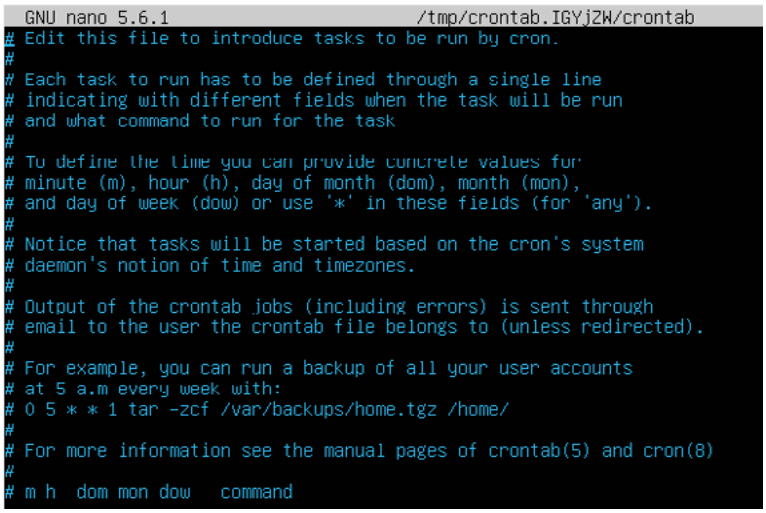


Imagen 53. Editor nano.

Si queremos ver el *crontab* del usuario en cuestión usaremos el comando `crontab -l`.

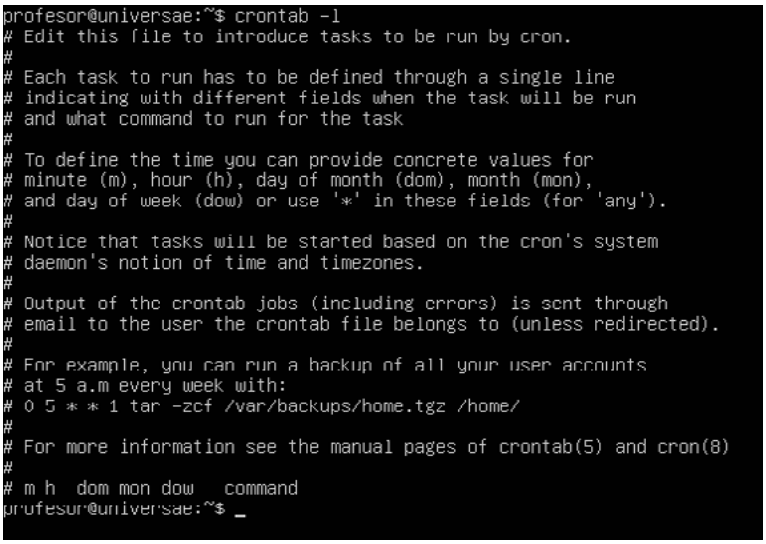


Imagen 54. Comando crontab -l.

Y si queremos eliminarlo usaremos el comando `crontab -r`.

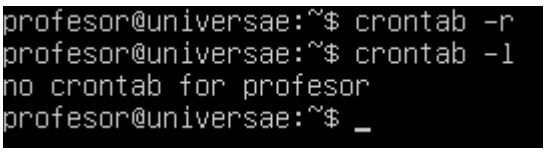


Imagen 55. Comando crontab -r.

1.10.3. Anacron

Como hemos visto en apartados anteriores, con **cron** programamos ciertas tareas que queremos que se lleven a cabo, pero esto no siempre funciona, pues necesitamos que el sistema se encuentre en funcionamiento. Si coincide que una tarea no se realiza porque el sistema está apagado, entonces no se llevaría a cabo. Para que se puedan ejecutar ciertas tareas sin necesidad de que el sistema esté en funcionamiento constante tenemos disponible el programador de tareas llamado **anacron**.

Este planificador inicia su demonio a la misma vez que iniciamos el sistema y hace un recorrido por las tareas programadas seleccionando las que n han sido llevadas a cabo y realizándolas. Estas tareas suelen estar alojadas en los directorios/ficheros */etc/cron.**.

```
profesor@universae:/etc$ ls cron.*
cron.d:
e2scrub_all  php

cron.daily:
apache2  apport  apt-compat  dpkg  logrotate  man-db

cron.hourly:

cron.monthly:

cron.weekly:
man-db
profesor@universae:/etc$
```

Imagen 56. Ficheros /etc/cron.*

El comando para instalar dicha utilidad es:

```
apt install anacron
```

1.10.4. At

Cuando usamos `cron` se programan tareas periódicas específicas que se ejecutan en intervalos de tiempo definidos por nosotros.

Es interesante también que conozcamos herramientas que nos ayuden a ejecutar ciertas tareas simplemente en momentos puntuales, aunque también programadas, pero que solo se ejecutan una vez. La utilidad que disponen para esto los sistemas Linux es `at`.

Para instalar esta herramienta necesitamos lanzar el comando:

```
apt install at
```

Una vez que tenemos la utilidad instalada, tenemos dos ficheros con los que trabajar:

- */etc/at.allow.* Si existe dicho fichero, solo los usuarios que se contemplen en este pueden ejecutar `at`.
- */etc/at.deny.* Si existe este fichero, todos los usuarios menos los que se incluyen en este fichero pueden ejecutar `at`.
- Si no existe ninguno de los ficheros, solo *root* puede ejecutar el comando.

```
root@universae:/etc# ls at.*
at.deny
root@universae:/etc#
```

Imagen 57. Ficheros relacionados con at

Podemos ver que en nuestro caso existe el segundo de los ficheros.

Para usar el comando, la sintaxis que debemos de seguir es:

```
at HH[:]MM[am|pm] [Mes día] script
```

```
root@universae:~# at 2am tomorrow < prueba.php
warning: commands will be executed using /bin/sh
job 1 at Thu Jul 28 02:00:00 2022
root@universae:~# _
```

Imagen 58. Ejecución del comando **at**

www.universae.com