

# GNU/Linux, con sabor a Debian

PABLO NIKLAS

Edición Primera

© 2020 PABLO NIKLAS  
ISBN: 978-987-86-4331-1



IMPRESO EL 11 DE JUNIO DE 2020.  
COMPOSICIÓN REALIZADA CON X<sub>L</sub>A<sub>T</sub>E<sub>X</sub>.

*A Analia, por acompañarme y hacer esto posible.  
A Delfi, por iluminar mis días.  
A mis Padres y mi hermana por haber soportado mi auto-aprendizaje.*



# Antes de comenzar

```
-----  
< ¡Bienvenidos! >  
-----  
 \ ^__^  
  \  (oo)\----  
   (__)\       )\/\  
     ||----w |  
     ||     ||
```

## SOBRE ESTA PUBLICACIÓN

 IENVENIDO estimado Lector, la idea detrás de esta publicación es que los nuevos usuarios de Linux no encuentren frustrante iniciarse en este Sistema Operativo que desde hace años es ícono en el mundo informático. Si bien siempre el material está en algún lugar, a través de este trabajo, el autor, pretende acercar una compilación de información y ejemplos presentados de una forma que ayudarán en el aprendizaje sobre GNU/Linux, utilizando fuentes seleccionadas y desde su propia experiencia.

\*nix comenzó a ser utilizado mediante lo que se conoce como «la línea de comandos» desde su génesis misma, y GNU/Linux siendo un «clon» gratuito de \*nix. EL verdadero poder de estos sistemas operativos radica en la terminal de línea de comandos. Quien la domine (con todo lo que eso conlleva), ya tiene un 80% resuelto.

Se reconoce que al comienzo es muy probable que el Lector choque en cierta manera con esta forma de utilizar el sistema operativo. Le gustaría manejarse directamente con la interfaz gráfica. Pero lamentablemente, esta publicación esta orientada a «la terminal de línea de comandos».

Como el Lector probablemente haya escuchado en su infancia, para comenzar a correr, primero se debe caminar, y para multiplicar, se debe primero saber sumar.

Por esto es que primer el Lector debe conocer la línea de comandos, para poder explotar con mayor provecho este sistema operativo.

Espero que la publicación cumpla con sus expectativas y lo incentive a seguir investigando a GNU/Linux, y todas las posibilidades que presenta.

Pablo Niklas.

## RECUADROS CONCEPTUALES Y OTROS

Durante la lectura de este libro, el Lector se va a encontrar con distintas entidades pedagógicas cuya función es facilitar la lectura y asimilar mejor los distintos conceptos.

### EN EL TEXTO

- ▶ Por ejemplo, esta es **SIN** El recuadro de sintaxis y muestra, en forma abreviada y de ejemplo, la sintaxis del comando que se esté tratando en cada sección.
- ▶ Otro caso es **CMD** El recuadro de comandos, y denota que lo que se encuentra en su interior es un comando que puede escribirse en la terminal de texto tal cual esta escrito.
- ▶ Las comillas «» indican términos nuevos.

### EN LA PÁGINA

Estos recuadros tienen la característica que pueden encontrarse en cualquier parte dentro del capítulo al cual pertenecen.



Lo aquí descripto debe ser tenido en cuenta, ya sea por la importancia conceptual o por la acción a realizarse.



Contenido meramente informativo, complementa los conceptos que se estén transmitiendo.



Definiciones importantes.



Acciones que deben ser realizadas con cautela, puede causarse un daño menor.



Acciones que deben ser realizadas con extrema cautela y revisión previa, o conceptos que deben tenerse muy muy muy muy en claro, ya que puede causarse un daño mayor.



Sugerencias que puede o no ser utilizadas, pero que de serlo, enriquecen la experiencia.



# Índice general

1

## Introducción a Debian Linux

Pág. 17

1.1	GNU: El proyecto	17
1.2	Linux: El núcleo	17
1.3	Características comunes a *nix y GNU/Linux	18
	Otras características — 18 • Multiusuario — 19 • Tiempo Compartido — 19 • Multitarea — 19	
1.4	Diferencias entre *nix y GNU/Linux	19
1.5	Situación actual	21
1.6	Introducción a Debian GNU/Linux	22
1.7	Diferencias entre Rolling Release y Standard Release	22
	Standard Release — 22 • Rolling Release — 23 • ¿Cuál es mejor? — 24	
1.8	¿Donde corre Linux?	24
1.9	Instalación Máquina Virtual	25
	¿Qué es una máquina virtual? — 25 • Pasos para la instalación — 25	

2

## FHS y Comandos

Pág. 47

2.1	El estándar FHS	47
	Directorio «root» (/) — 47 • /boot — 47 • /dev — 48 • /etc — 48 • /home — 48 • /bin y /sbin — 49 • /lib — 49 • /lost+found — 49 • /mnt y /media — 49 • /usr/local — 49 • /opt — 49 • /proc y /sys — 50 • /root — 50 • /tmp, /var/tmp y /dev/shm — 50 • /usr, /usr/bin, /usr/lib y /usr/sbin — 51 • /var — 51 • Diferencias entre /sys, /proc y /dev — 51 • Subjerarquías de directorios — 52	
2.2	Tipos de archivos	52
	Archivos regulares — 52 • Directorios — 53 • Archivos de dispositivo — 53 • Pipes — 54 • Sockets — 54 • Links simbólicos — 54	
2.3	Conociendo la terminal de texto	54

2.4	Comandos	56
	man — 58 • tar — 60	
2.5	Concatenación de comandos	61
2.6	Expresiones regulares	61
	Alternación — 62 • Cuantificación — 62 • Agrupación — 63 • Carácteres especiales — 63 • Algunos ejemplos — 63	

3

**Gestión del software y Redirecciones****Pág. 67**

3.1	Introducción	67
	Funciones básicas del gestor — 68 • Descargar el software — 68 • De donde descargar — 69 • ¿Qué contiene un paquete? — 69 • apt-get install — 70 • apt-cache search — 71 • apt-get update — 72 • Formato del archivo /etc/apt/sources.list — 73	
3.2	Redireccionamiento y PIPEs	75
	> — 76 • >> — 77 • < — 77 • Redireccionamiento a stdErr — 78 • Redirección múltiple — 78 • PIPEs « » — 79	
3.3	Búsqueda y filtros	80
	Filtros — 80 • grep — 81 • find — 82 • locate y updatedb — 83	

4

**Editores****Pág. 85**

4.1	vi	85
	Modos de VI — 86 • Guía básica — 86 • Invocación de vi — 87 • Cambio de modo — 87 • Números multiplicadores — 88 • Ejemplos de manejo — 88 • Movimiento del cursor — 89 • Control de pantalla — 89 • Ingreso en modo texto — 90 • Borrar — 90 • Copiar y pegar — 90 • Búsqueda — 91 • Reemplazo — 91 • Otros — 92 • Modo «ex» o última línea — 92 • Mover — 93 • Opciones — 93	

5

**Seguridad y Accesos****Pág. 95**

5.1	Administración de usuarios	95
5.2	Tipos de usuarios	95
	Usuario root — 95 • Usuarios especiales — 95 • Usuarios normales — 96	
5.3	Archivos importantes	96
	/etc/passwd — 97 • /etc/shadow — 97 • /etc/group — 98	

5.4	Comandos principales de administración de usuarios	99
	useradd — 99 • usermod — 101 • userdel — 102 • passwd — 102	
5.5	Archivos de configuración de Bash	103
5.6	Resumen de comandos y archivos de administración de usuarios	104
	Comandos de administración y control de usuarios — 104	
5.7	Permisos	105
5.8	Permisos en formato numérico octal	107
5.9	Estableciendo los permisos con el comando chmod	108
5.10	Estableciendo permisos en modo simbólico	108
5.11	Cambiando propietario y grupo	109
5.12	Bits SUID, SGID y de persistencia «sticky bit» «setuid» — 111 • «setgid» — 112 • «sticky» — 113	110
5.13	Permisos preestablecidos con umask	114
	Máscaras simbólicas — 114 • Máscara en octal — 115 • Ejercicio de ejemplo de máscara (en octal) — 115	

6.1	Tareas y Procesos	117
	Ciclo de vida de un proceso — 117 • Columnas de ps — 118 • ¿Procesos o tareas? — 119	
6.2	Primer plano y Segundo plano	119
	Un proceso puede estar en Primer plano o en Segundo plano — 119 • Los procesos se pueden suspender — 121 • Suspender un trabajo no es lo mismo que interrumpirlo — 121	
6.3	Envío a segundo plano y eliminación de procesos	122
	Comando kill — 123 • La importancia de la planificación de procesos — 125	
6.4	Nice y Prioridad	125
	El comando nice — 126 • El comando renice — 127	
6.5	Desconectar un proceso de la terminal	127
6.6	Otros comandos relacionados	128
6.7	alias	128
	Creando un alias — 129 • Verificando los alias — 130 • Persistencia del alias — 130	

7.1	Introducción	131
	¿Qué es un filesystem? — 131 • Comparativa entre EXT4, XFS y BTRFS — 132	
7.2	Filesystems «journalizados»	133
7.3	Direccionamiento en EXT4	133
	Direccionamiento directo en sistemas de archivo EXT4 — 135 • Direccionamiento indirectos simples — 135 • Direccionamiento indirectos dobles y triples — 135 • Comandos — 136	
7.4	Incorporar un disco nuevo	137
7.5	Preparación de la VM	138
	Detección del nuevo dispositivo — 139 • Particionamiento — 142 • Creación del Filesystem — 144 • Migración del /home al nuevo filesystem — 144	
7.6	Archivo /etc/fstab	146
	Estructura del archivo /etc/fstab — 147 • Ejemplo de archivo /etc/fstab — 147	
7.7	Links	148
	Hard Links — 148 • Soft Links — 150	

8.1	systemd & init	153
	¿Que es init? — 153 • ¿Que es systemd? — 153 • Porque reemplazó a init? — 155	
8.2	Servicios	155
8.3	Pasos para dar de alta un servicio	156
8.4	Ejemplo	156
	Creación de programa de servicio — 156 • Archivo «unit» — 157 • Cargar el servicio — 157 • Iniciar el servicio — 158	
8.5	GRUB	158
	Introducción — 158 • El viejo y querido «LiLo» — 158 • Gestor de arranque — 159 • Secuencia de arranque del sistema (Resumen) — 159 • Ejecución de GNU GRUB — 160 • Notación de GRUB — 160 • Configuración de GNU GRUB — 161 • Instalación — 164 • Instalando GRUB con grub-install — 166	
8.6	GRUB II (el regreso)	167
	Nuevo diseño — 167 • Archivo grub.cfg de muestra — 168 • Directorio /etc/grub.d/ — 168 • ¿Como trabaja GRUB 2? — 170 • Agregar un nuevo script GRUB — 170 • Edición de /etc/default/grub — 171	

9.1	Una (no tan corta) introducción	173
¿Qué es una red? – 173 • Clientes y Servidores – 173 • Protocolos – 174 • La red de redes: Internet – 174 • Métodos de conmutación «switching» – 174 • El Modelo OSI – 175 • Tipos de comunicación – 179 • Elementos típicos de una red LAN – 179		
9.2	En Debian GNU/Linux	180
9.3	Preparación del entorno virtualizado	181
Configuración de VirtualBox™ – 181 • Configuración de la máquina virtual – 184		
9.4	Relevar el hardware de red detectado	184
9.5	Configurar una interfaz Ethernet	185
9.6	DHCP	186
Descubrimiento – 187 • Ofrecimiento – 187 • Solicitud de arrendamiento – 187 • Confirmación – 188		
9.7	Utilizar DHCP para configurar la interfaz automáticamente	188
9.8	Configurar la interfaz manualmente	188
9.9	Aplicando los cambios	188
Reiniciando los servicios de networking – 188 • Desactivando y Activando administrativamente la placa de red – 189		
9.10	Activar una interfaz sin dirección IP	189
9.11	Prueba	190
9.12	Cifrado	190
Introducción – 190 • Cifrado Simétrico (con clave compartida) – 191 • Cifrado Asimétrico (con clave pública y privada) – 191		
9.13	Secure Shell (SSH)	192
Descripción general del funcionamiento de SSH – 193 • Métodos de autenticación de usuarios – 194 • Configuración de OpenSSH – 196		
9.14	IP Tables	198
Tablas – 198 • Reglas – 199 • Comandos de iptables – 199 • Matches generales – 200 • Matches TCP/UDP – 200 • Targets/Jumps – 200		
9.15	Registrar las incidencias de iptables	201
9.16	Ejemplos	201
Guardar la configuración de iptables – 204		

10.1 Scripting	207
<i>¿Qué es un programa? – 207 • La cocina – 207</i>	
10.2 ¿Qué es un script?	208
10.3 Variables	209
Variables intrínsecas de bash – 209 • Variables creadas por el programador – 209	
10.4 Caracteres especiales	210
10.5 Palabras reservadas	212
10.6 El ejecutable [	212
Evaluación para archivos – 213 • Evaluación para cadenas de caracteres – 213 • Entrada/Salida – 214 • Ejemplos – 214	
10.7 Los primeros scripts	215
Script 1: Hola mundo – 215 • Script 2: Lo mismo usando una variable – 215 • Script 3: Cuando se usan más de una variable – 215 • Script 4: Si se usan caracteres especiales la cosa puede cambiar – 216	
10.8 Condicionales	216
if-then-fi – 216 • case-in-esac – 217	
10.9 Ciclos o bucles	218
while-do-done – 218 • until-do-done – 219 • for-in-done – 220	
10.10 Globales y expansiones	221
Globales – 221 • Expansiones – 221	
10.11 Aritmética de bash	223
10.12 Casos de ejemplo de scripting	225
Detector de archivos de imagen – 225 • Parseador de archivo csv – 226 • Ejemplo de case y while – 228	
10.13 Cron	229
10.14 Iniciar crond	229
10.15 Usando Cron	230
10.16 Valor o lista/incremento	233
10.17 Ejecutando Cron con múltiples usuarios, comando crontab	235
10.18 Controlando el acceso a cron	236

11.1	Comandos	237
11.2	Navegación y archivos	238
11.3	PIPE's y redirecciones	240
11.4	Permisos	240
11.5	Usuarios	241
11.6	Procesos	242

12.1	Procedimiento de Recuperación de password	245
12.2	Firewall	245
12.3	Scripting	247
12.4	Integradores	249



# 1 | Introducción a Debian Linux

## 1.1 GNU: EL PROYECTO

Corría el año 1983 y Richard Stallman, cansado de las limitaciones del software pago, creó el Proyecto GNU, con el propósito de generar un sistema operativo similar y compatible con \*nix y los estándares POSIX<sup>1</sup>. Dos años más tarde, en 1985, creó la Fundación del Software Libre «Free Software Fundation (FSF)» y desarrolló la Licencia pública general de GNU «GNU is not \*nix» (GNU GPL), para tener un marco legal que permitiera difundir libremente el software. De este modo el software de GNU fue desarrollado muy rápidamente por un conjunto importante de personas. A principios de los años 90' había bastante software disponible como para crear un sistema operativo completo. Sin embargo... todavía faltaba el «núcleo».

## 1.2 LINUX: EL NÚCLEO

En 1991, en Helsinki, «Linus Torvalds» comenzó un proyecto que más tarde llegó a ser el núcleo Linux. Esto fue al principio un emulador de terminal, al cual Torvalds solía tener acceso en los grandes servidores \*nix de la universidad. Él escribió el programa expresamente para el hardware que usaba (una Intel 80386 con 4 MB de RAM y discos IDE) e independiente de un sistema operativo. El sistema operativo que usó durante el desarrollo fue «Minix», y el compilador inicial fue: «GNU C compiler», que aún es la opción principal para compilar Linux hoy (aunque Linux puede ser compilado bajo otros compiladores, tal como el «Intel C Compiler»).

El núcleo o kernel es un conjuntos de programas, con control completo sobre todo en el sistema. El núcleo facilita las interacciones entre los componentes de hardware y software. En la mayoría de los sistemas, es uno de los primeros programas cargados al inicio (después del «bootloader» o gestor de arranque). Administra el resto de las solicitudes de inicio y de I/O del software, traduciéndolas en instrucciones de procesamiento de datos para la unidad central de procesamiento. Maneja las solicitudes de memoria y periféricos como teclados, monitores, impresoras y parlantes.[definition:kernel]

---

<sup>1</sup>«Interfaz del sistema operativo portátil (para \*nix)» es el nombre de una familia de estándares relacionados especificados por el IEEE para definir la interfaz de programación de aplicaciones (API), junto con interfaces de línea de comando (shell) y utilidades para software compatible con variantes del sistema operativo \*nix, aunque el estándar puede aplicarse a cualquier sistema operativo (inclusive Windows™).[definition:posix]

## GNU/Linux



El término GNU/Linux deviene de la unión de GNU (proyecto creado por Richard Stallman), y Linux, el núcleo o kernel del sistema operativo, (creado por Linus Torvalds).

En esta publicación, cuando se hable de GNU/Linux, se dirá Linux. Sin embargo, cuando se les pregunta a los referentes, estos dicen que no importa como se lo llame; en su núcleo, sigue siendo Linux.[[article:upfront\\_286](#)]

### 1.3 CARACTERÍSTICAS COMUNES A \*NIX Y GNU/LINUX

Los sistemas \*nix se caracterizan por varios conceptos, que son comúnmente conocidos como La Filosofía \*nix:

- ▶ El uso de archivos de texto plano para el almacenamiento de información.
- ▶ Un sistema de archivos «filesystem» jerárquico.
- ▶ Tratar a los dispositivos y a ciertos tipos de comunicación entre procesos (IPC<sup>2</sup>) como archivos.
- ▶ El uso de una gran cantidad de herramientas de software, es decir pequeños comandos que se pueden unir en el mismo intérprete de comandos, proveyendo una funcionalidad más compleja.

#### Filosofía \*nix



Surge de «La idea de que el potencial de un sistema, venga más por la relación entre los programas que de los programas por si mismos.» [[book:unixprog](#)]

#### 1.3.1 OTRAS CARACTERÍSTICAS

- ▶ Miden el tiempo como segundos desde el 1 de enero de 1970. La crisis «Y2K»<sup>3</sup> no fue un problema.

<sup>2</sup>La comunicación entre procesos (comúnmente IPC, del inglés Inter-Process Communication) es una función básica de los sistemas operativos. Los procesos pueden comunicarse entre sí, compartiendo espacios de memoria, ya sean variables compartidas o buffers, o a través de las herramientas provistas por las rutinas de IPC. IPC provee un mecanismo que permite a los procesos comunicarse y sincronizarse, normalmente a través de un sistema de bajo nivel de transferencia de mensajes que ofrece la red subyacente.[[ms:IPC](#)]

<sup>3</sup>El problema del año 2000, también conocido como efecto 2000, error del milenio, problema informático del año 2000 o por el numerónimo Y2K, es un bug o error de software causado por la costumbre que habían adoptado los programadores de omitir la centuria en el año para el almacenamiento de fechas (generalmente para economizar memoria), asumiendo que el software solo funcionaría durante los años cuyos números comenzaran con 19.[[history:y2k](#)]

- ▶ Arquitectura modular e integrada.
- ▶ Retrocompatibilidad en shell scripting, ya que los comandos y sintaxis son similares. Muchas de sus herramientas y software que son estándares en todas las distribuciones GNU/Linux, son alternativas libres a sus versiones disponibles en \*nix.

### 1.3.2 MULTIUSUARIO

Permiten el acceso concurrente de los usuarios que se hallen registrados en el sistema, otorgándoles servicio y tiempo de procesamiento en forma individual acorde los permisos necesarios.

### 1.3.3 TIEMPO COMPARTIDO

Comparten de forma concurrente un recurso computacional (tiempo de ejecución en la CPU, uso de la memoria, etc.) entre muchos usuarios del sistema operativo, permitiendo a este último acotar el tiempo de respuesta del computador y limitar el uso de la CPU por parte de un proceso dado.

### 1.3.4 MULTITAREA

La multitarea es la capacidad de realizar  $n$  tareas<sup>4</sup> simultáneamente, distribuyendo las mismas entre los núcleos «cores» que haya disponible. Una condición «sine qua non»<sup>5</sup> es que los distintos programas no pueden acceder al área de memoria de otro programa<sup>6</sup>. Todos los recursos de hardware de la computadora (entre ellos la memoria), son administrados por el kernel y solo éste (amén por las vulnerabilidad Meltdown<sup>7</sup> y Spectre<sup>8</sup>) puede acceder y controlar dicho acceso.

#### «int 80h»



En assembler, que es el lenguaje de programación que más se acerca al lenguaje de máquina, para toda operación que se quiera realizar en la computadora, **siempre se llama a la misma «puerta», int 80h**.

## 1.4 DIFERENCIAS ENTRE \*NIX Y GNU/LINUX

<sup>4</sup>Otros términos equivalentes son «programas» o «procesos».

<sup>5</sup>Expresión latina que significa «sin la cual no» y se aplica a una condición que necesariamente ha de cumplirse o es indispensable para que suceda o se cumpla algo.

<sup>6</sup>Esta es la famosa pantalla azul que rezaba «Violación de memoria».

<sup>7</sup>La vulnerabilidad Meltdown es un agujero de seguridad en el hardware que afecta a los procesadores Intel x86, a los procesadores IBM POWER, y también a algunos procesadores basados en la arquitectura ARM, y que permite que un proceso maligno pueda leer de cualquier lugar de la memoria virtual, aún sin contar con autorización para hacerlo.

<sup>8</sup>Spectre es una vulnerabilidad que afecta a los microprocesadores modernos que utilizan predicción de saltos. En la mayoría de los procesadores, la ejecución especulativa que surge de un fallo de la predicción puede dejar efectos observables colaterales que pueden revelar información privada a un atacante. Por ejemplo, si el patrón de accesos a la memoria realizados por la mencionada ejecución especulativa depende de datos privados, el estado resultante de la caché de datos constituye un canal lateral mediante el cual un atacante puede ser capaz de obtener información acerca de los datos privados empleando un ataque sincronizado.

	*NIX	GNU/LINUX
1	<p>Los desarrolladores de aplicaciones *nix saben cuales aplicaciones son las más funcionales y a cuales darles todo el soporte y optimización.</p>	<p>Por otra parte, el desarrollo para GNU/Linux es más diversificado. No existe un estándar en cuanto a las herramientas a utilizar, por tal motivo y para darle cierto nivel de coherencia, se creó la LSB (Linux Standard Base), pero aún no ha demostrado mucha efectividad.</p>
	<p>En cuanto a la arquitectura de hardware, las versiones comerciales de *nix están desarrolladas para funcionar con un pequeño puñado de arquitecturas de hardware, es decir, para modelos específicos.</p>	<p>Por su parte, GNU/Linux ha sido diseñado históricamente para ser tan compatible como sea posible. No solo se encuentra disponible para muchas arquitecturas, sino que el número de dispositivos de entrada/salida y otros externos que soporta, puede ser ilimitado. Los desarrolladores no pueden asumir que software específico va a ser instalado, por lo que no pueden optimizarlo exclusivamente.</p>
	<p>En ninguna versión de *nix el Kernel se encuentra disponible libremente, a diferencia de las distribuciones de GNU/Linux.</p>	<p>Con GNU/Linux y los demás sistemas abiertos, un parche puede ser liberado en código fuente y los usuarios finales pueden instalarlo, o incluso modificarlo si así lo desean. Generalmente, los parches así modificados tienden a no ser tan probados como los parches de *nix .</p>
	<p>Todas las versiones comerciales de *nix han evolucionado para soportar un kernel modular. Los drivers y algunas características están disponibles como componentes separados y pueden ser cargados o descargados del kernel a necesidad.</p>	<p>Ningún kernel es tan flexible como el kernel de GNU/Linux. Sin embargo, esta flexibilidad está en constante cambio.</p>
	<p>La mayoría de las versiones comerciales de *nix soportan dos, cuanto mucho tres diferentes tipos de sistemas de archivos.</p>	<p>Una de las razones por las que GNU/Linux se ha convertido en una herramienta tan importante, es la inmensa compatibilidad con otros sistemas operativos; siendo una de sus más obvias características la gran cantidad de sistemas de archivos que soporta.</p>
	<p>Si se compra una versión comercial de *nix , el vendedor proveerá de soporte técnico para asegurarse que el sistema funcione correctamente.</p>	<p>Los usuarios de GNU/Linux no pueden darse el lujo, de tener una compañía que esté revisando su sistema por ellos.</p>
	<p>*nix tiene entre un estimado de 85-120 virus a la fecha.</p>	<p>Linux tiene entre 60-100 virus estimados a la fecha. Ninguno ya se está distribuyendo.</p>

CUADRO 1.1: DIFERENCIAS ENTRE GNU/LINUX Y \*NIX.



Aunque en el archivo Léeme de Linux se indica que es un clon del sistema operativo \*nix, Linux no es un sistema operativo en sí mismo. Es un núcleo compatible con el de \*nix.

Inicialmente se publicó como un núcleo semejante a Minix y al día de hoy sigue incluyendo el mismo tipo de software: controladores, planificadores, gestores de memoria virtual y más funcionalidad habitual en un núcleo de sistema operativo.

Linux se ejecuta en el nivel con más privilegios del modo protegido, el nivel que corresponde al núcleo del sistema operativo.

La parte de un sistema operativo que se ejecuta sin privilegios o en espacio de usuario es la biblioteca<sup>º</sup> del lenguaje C. Esta biblioteca provee el entorno de tiempo de ejecución, y una serie de programas o herramientas que permiten la administración y uso del núcleo y proveer servicios al resto de programas en espacio de usuario.[[wiki:kernel](#)]

## 1.5 SITUACIÓN ACTUAL

Desde hace varios años Linux adquiere cada vez más fuerza, y su creciente popularidad impulsa a más y más usuarios a dar el «salto»[[Online:ApacheLinux](#)]. Hay que tener presente que cada distribución tiene sus propias peculiaridades y una elección correcta desde el principio puede evitar los costos de migraciones futuras.

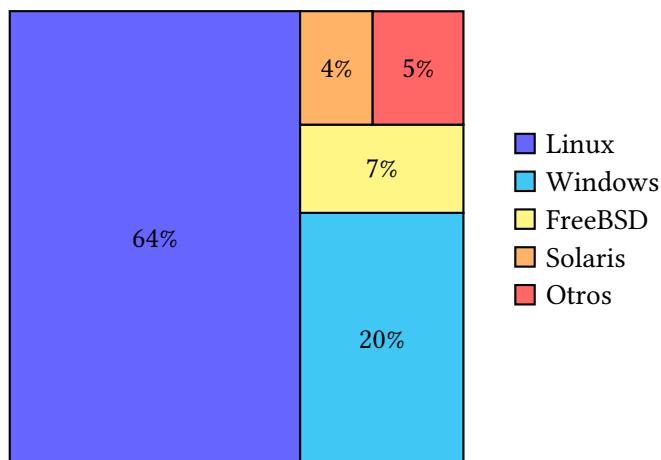


FIGURA 1.1: DISTRIBUCIÓN DE SISTEMA OPERATIVOS EN SERVIDORES.

<sup>º</sup>Muchas veces el término se informa erróneamente como «Librería», ya que en inglés «library» es biblioteca, y no «librería».

## 1.6 INTRODUCCIÓN A DEBIAN GNU/LINUX

### Distribución Linux



También llamada vulgarmente «distro», es una distribución de software basada en el núcleo Linux que incluye determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios, dando así origen a ediciones domésticas, empresariales y para servidores.

Debian GNU/Linux es un sistema operativo libre, desarrollado por miles de voluntarios alrededor del mundo, que colaboran a través de Internet.

La dedicación de Debian al software libre, su base de voluntarios, su naturaleza no comercial y su modelo de desarrollo abierto la distingue de otras distribuciones del sistema operativo GNU.

Nació en el año 1993, de la mano del proyecto Debian, con la idea de crear un sistema GNU usando Linux como núcleo ya que el proyecto Debian, organización responsable de su mantenimiento en la actualidad, también desarrolla sistemas GNU basados en otros núcleos (Debian GNU/Hurd, Debian GNU/NetBSD y Debian GNU/kFreeBSD).

Uno de sus principales objetivos es separar en sus versiones el software libre del software no libre. El modelo de desarrollo es independiente al de las empresas, es decir, creado por los propios usuarios sin depender de necesidades comerciales.

Debian no vende directamente su software, lo pone a disposición de cualquier usuario en Internet, aunque sí permite a personas o empresas distribuir comercialmente este software mientras se respete su licencia.

## 1.7 DIFERENCIAS ENTRE ROLLING RELEASE Y STANDARD RELEASE

### 1.7.1 STANDARD RELEASE

La mayoría de las distribuciones de Linux utilizan ciclos de lanzamiento estándar. Por ejemplo, Debian/Ubuntu/CentOS usan versiones estándar, que también pueden denominarse versiones puntuales o versiones estables. Algunos proyectos lanzan regularmente nuevas versiones cada seis meses. Durante el proceso de desarrollo de seis meses, toman las últimas versiones de todo el software en sus repositorios y lo empaquetan, actualizando todo el software. Luego, se freezan las versiones del software en los repositorios y se prueba durante unos meses, asegurándose de que todas las versiones de software funcionen bien juntas; y de surgir algún error se corrije.

Cuando se lanza una nueva versión del sistema operativo, se garantiza que el software ha sido probado lo suficiente para que funcione bien en su conjunto. Esta versión se mantiene congelada en el tiempo tanto como sea posible. Aún así, periódicamente, se lanzan versiones de software actualizadas para solucionar problemas de seguridad y otros errores importantes.

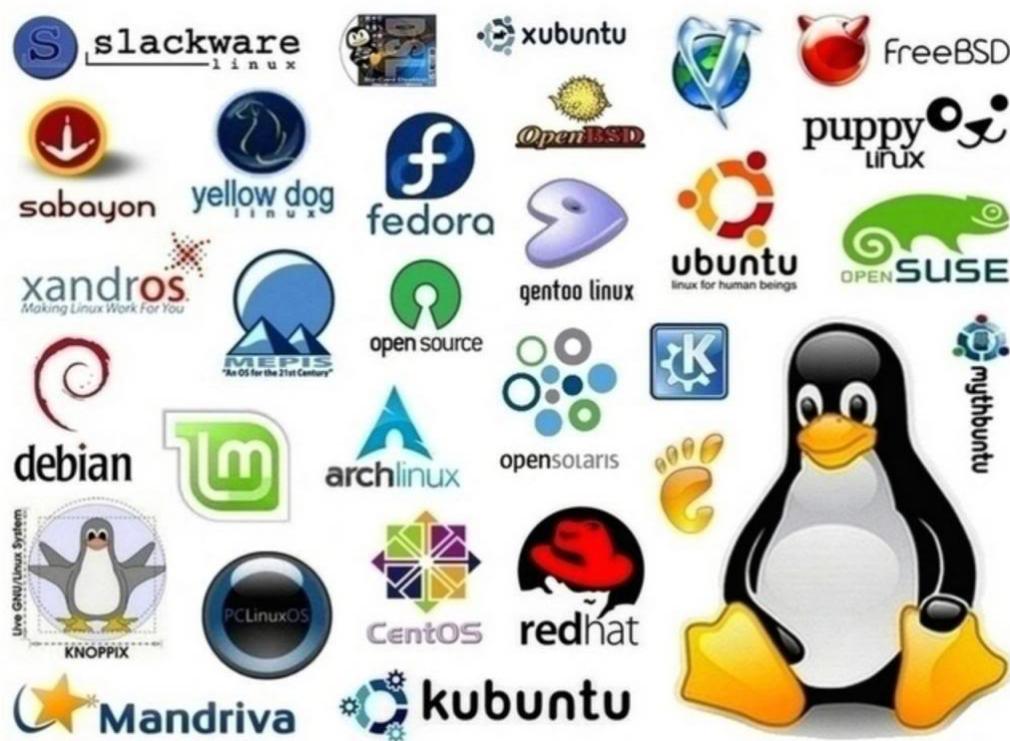


FIGURA 1.2: ALGUNAS DE LAS DISTRIBUCIONES MÁS POPULARES.

### 1.7.2 ROLLING RELEASE

Un ciclo de lanzamiento continuo prescinde de los lanzamientos regulares y estándar de distribución de Linux. Por ejemplo, Arch Linux usa un ciclo de lanzamiento rolling. No hay múltiples lanzamientos diferentes de Arch. En su lugar, solo hay una versión única de Arch. Los paquetes de software se prueban y luego se lanzan de inmediato a la versión estable de la distribución de Linux. Dependiendo de su distribución, es posible que ni siquiera vean muchas pruebas antes de que se publiquen como actualizaciones estables. Cuando se lance una nueva versión de una aplicación o utilidad del sistema, se dirigirá directamente a la distribución actual de Linux.



La distribución de una versión rolling nunca se «congela en el tiempo», sino que se actualiza de forma continua.

Como no hay versiones estándar, solo se tiene que instalar una distribución de Linux de este tipo una vez y realizar actualizaciones periódicas. Las nuevas versiones de paquetes de software llegarán gradualmente a medida que se publiquen, no se tendrán que realizar grandes actualizaciones como las de Ubuntu 13.10 a 14.04. Al instalar la distribución, se obtendrá una instantánea del software en un momento dado.

Si se necesita la última versión de un paquete, solo se debe esperar unos días y aparecerá

como una actualización para su distribución de Linux. No se tendrá que esperar seis meses hasta la próxima versión estándar de la distribución de Linux.

# 1

## 1.7.3 ¿CUÁL ES MEJOR?

Un ciclo de lanzamiento continuo es mejor si se quiere vivir en la vanguardia y tiene las últimas versiones disponibles de software; mientras que un ciclo de lanzamiento estándar es mejor si se quiere beneficiarse de una plataforma más estable con más pruebas.

Tener la última versión de todo el software suena bien, pero no es tan beneficioso como podría pensarse. Probablemente no se necesite la última versión de los servicios y utilidades del sistema de bajo nivel; ni siquiera se notaría la diferencia si se los instalara, a menos que existieran errores porque las diferentes versiones de software no se probaron juntas. La actualización de los mismos en la mitad del proceso podría hacer que el sistema se vuelva más inestable o que aparezca un error extraño. Sin embargo si se desease, la versión más reciente del software, como el de las aplicaciones de escritorio, se podría actualizar con total facilidad; incluso si se está utilizando una distribución de Linux con un ciclo de lanzamiento estándar.

Un ciclo de lanzamiento continuo hace que sea más fácil mantenerse actualizado, por supuesto, en lugar de una gran actualización de una sola vez, el software se actualiza periódicamente. Los usuarios no están usando versiones diferentes de la distribución de Linux, todos están usando la misma versión.[\[blog:rollingvsstandard\]](#)

## 1.8 ¿DONDE CORRE LINUX?

Desde celulares (con android) y modems Wifi, hasta las llamadas «supercomputadoras», Linux hoy ejecuta en un variedad de plataformas.

Principalmente, tuvo su gran empuje cuando se enfocó en:

- ▶ **Microcomputadoras o «low-range»** → También son llamados Equipos de Escritorio. La principal diferencia con los servidores, radica en el enfoque, desde el lado del hardware, de la alta disponibilidad. Como por ejemplo, la doble fuente de alimentación. El grueso de la franja de usuarios se encuentra en este segmento, siendo el que mas difusión generó.
- ▶ **Minicomputadoras o «mid-range»** → Estas computadoras se encuentran en el medio del espectro, entre las de escritorio y los mainframes. Ejemplos en este rango son aquellas con procesadores SPARC™ de Oracle™ (antes SUN), o los de la serie P de IBM™, como así también los modelos de la serie AS400 de la mencionada marca. La mayoría de los servicios de internet corren en servidores de esta gama.
- ▶ **Mainframes o «high-range»** → También llamados «big irons», son aquellas computadoras que no están orientadas a un uso mas general, sino por el contrario es específico de mundo financiero, como ser bancos, sociedades de bolsa o entidades de tarjetas de crédito. Estas computadoras están optimizadas para realizar las cuatro operaciones básicas (+, -, y ÷), pero a un volumen de procesamiento asombroso. Asimismo estas computadoras solamente se reinician cuando se debe actualizar el sistema operativo, y tienen sus componentes de hardware optimizados para usos específicos, como por ejemplo, el zIIP [**db2:zIIP**], que es un procesador para optimizar los requerimientos de carga del RDBMS<sup>10</sup> de IBM, DB2.

<sup>10</sup>«Relational Database Management System».

## 1.9 INSTALACIÓN MÁQUINA VIRTUAL

### 1.9.1 ¿QUE ES UNA MÁQUINA VIRTUAL?



Una máquina virtual, tambien conocida como «Virtual Machine», es un software que «simula» hardware.

El software llamado «hipervisor» simula todos los componentes principales de una computadora: CPU, memoria, disco, teclado, pantalla, mouse, etc.

El hipervisor distribuye los componentes físicos de la computadora donde esta instalado, llamada «host», entre las distintas simulaciones en curso, llamadas «guests».

Estas simulaciones pueden ser «mudadas» entre distintos hosts, proveyendo mas flexibilidad y redundancia.

La nube esta basada en esta arquitectura.

### 1.9.2 PASOS PARA LA INSTALACIÓN

El software de máquina virtual que se utiliza en esta publicación, es Oracle VirtualBox<sup>11</sup>. De más está decir, que se puede utilizar cualquier tipo de máquina virtual como «libvirt/kvm».

Téngase presente que puede haber ligeras variaciones entre las capturas de pantalla y el instalador actual, debido a propias actualizaciones que realiza Debian en las imágenes ISO que están disponibles para descargar.

Debe descargarse la imagen que corresponda a la arquitectura del sistema operativo host. Sea de 32 o de 64 bits.

Si se desea saber cual es la arquitectura, en el caso que sea Windows™, se deben realizar los siguientes pasos:

- 1 En el Menú Inicio→Ejecutar, escribir cmd y presionar .
- 2 En la ventana que se abre, escribir `cmd echo %PROCESSOR_ARCHITECTURE%` y presionar .
- 3 La consola indicará el tipo de arquitectura de la PC, por ejemplo «AMD64»

En el caso de un \*nix o GNU/Linux:

- 1 Abrir la consola.
- 2 Escribir `cmd uname -p` y presionar .
- 3 La consola indicará el tipo de arquitectura de la PC, por ejemplo «x86\_64»

<sup>11</sup><https://www.virtualbox.org/>

## 1

Para 64 bits, la imagen en formato ISO se debe descargar de la URL <http://debian.xfree.com.ar/debian-cd/10.3.0/amd64/iso-dvd/debian-10.3.0-amd64-DVD-1.iso>.

Para 32 bits, la imagen en formato ISO se debe descargar de la URL <http://debian.xfree.com.ar/debian-cd/10.3.0/i386/iso-dvd/debian-10.3.0-i386-DVD-1.iso>

Para una mayor tranquilidad es recomendable verificar la integridad de la imagen bajada por medio de la suma MD5. Para esto, se utiliza **CMD md5sum** en Linux o \*nix, o sino **CMD fciv** en Windows<sup>TM</sup><sup>12</sup> y cotejar el resultado con la entrada en el archivo MD5SUM correspondiente.

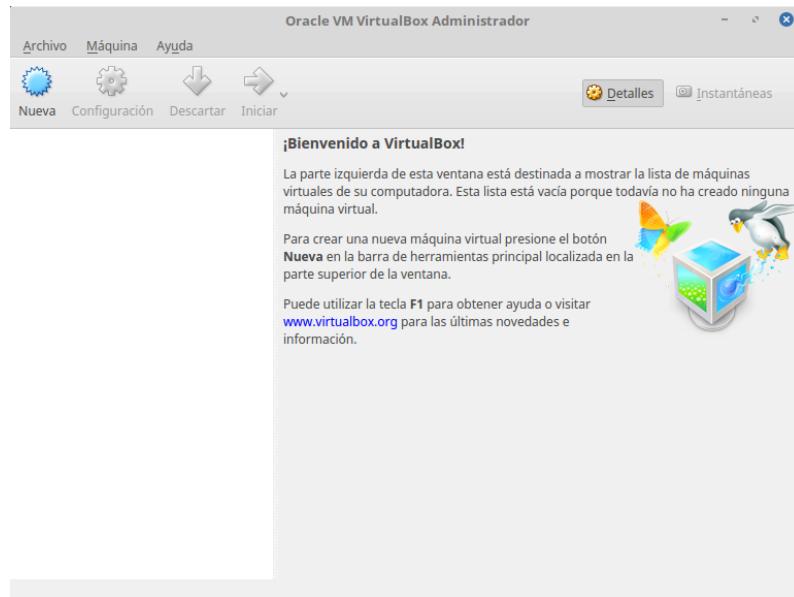


FIGURA 1.3: DIÁLOGO PRINCIPAL DE VIRTUALBOX.

Se hace click en el ícono «Nueva».



FIGURA 1.4: CONFIGURACIÓN DE LA MEMORIA DE LA MAQUINA VIRTUAL NUEVA.



FIGURA 1.5: CONFIGURACIÓN DE LA UNIDAD DE ALMACENAMIENTO.



FIGURA 1.6: CONFIGURACIÓN FORMATO DE UNIDAD DE ALMACENAMIENTO.



FIGURA 1.7: CRECIMIENTO DE LA UNIDAD DE ALMACENAMIENTO.

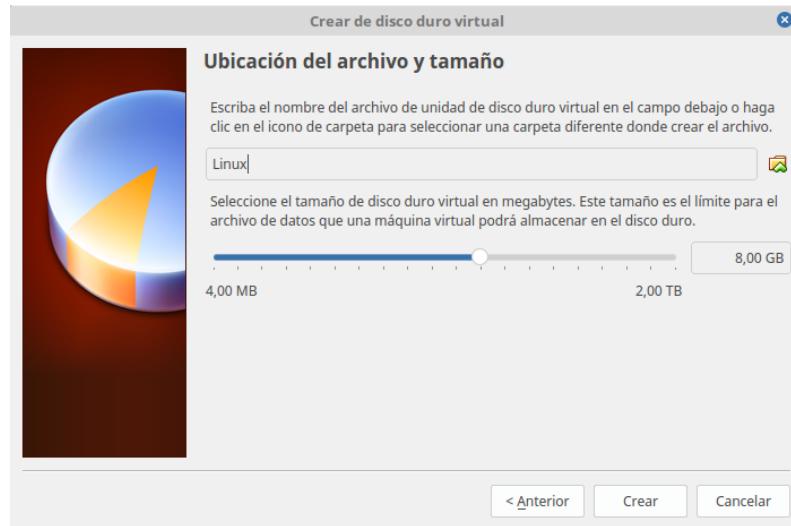


FIGURA 1.8: TAMAÑO INICIAL DE LA UNIDAD DE ALMACENAMIENTO.

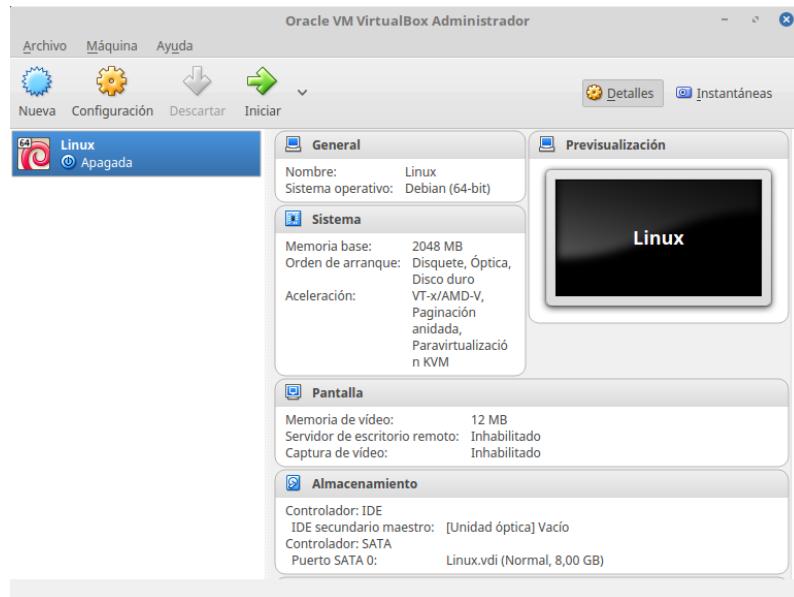


FIGURA 1.9: FINALIZACIÓN DE LA CONFIGURACIÓN DE LA VM.



FIGURA 1.10: DEFINICIÓN DE LA UNIDAD DE CD/DVDROM.

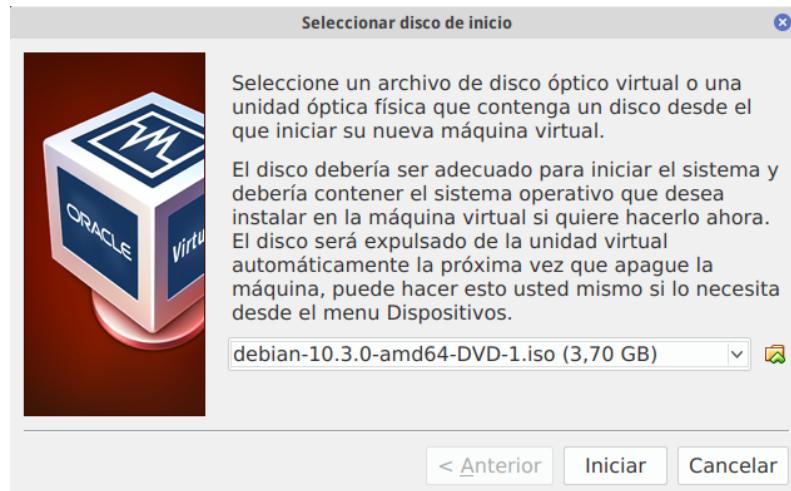


FIGURA 1.11: SELECCIÓN DEL ISO CORRESPONDIENTE.

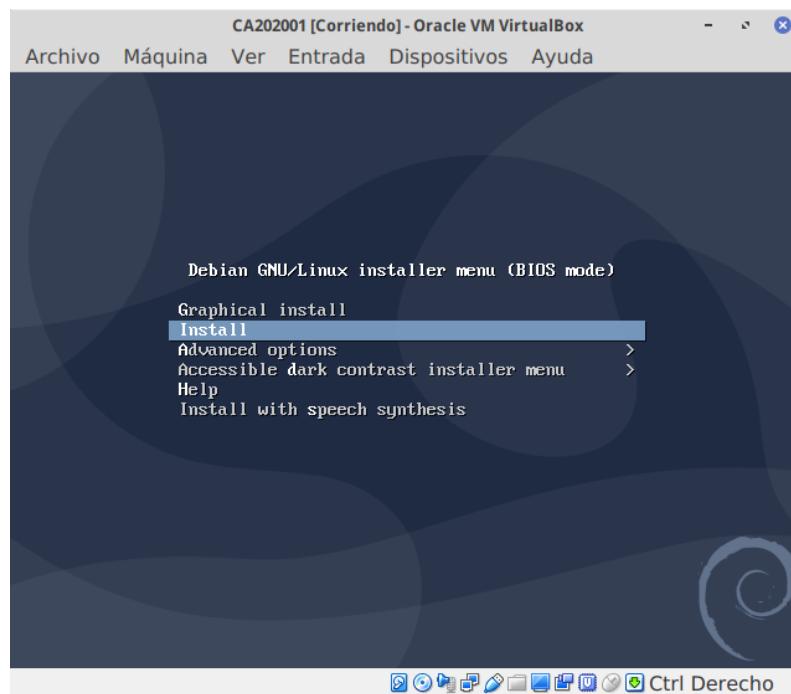


FIGURA 1.12: BOOTEZO INICIAL DE LA VM.

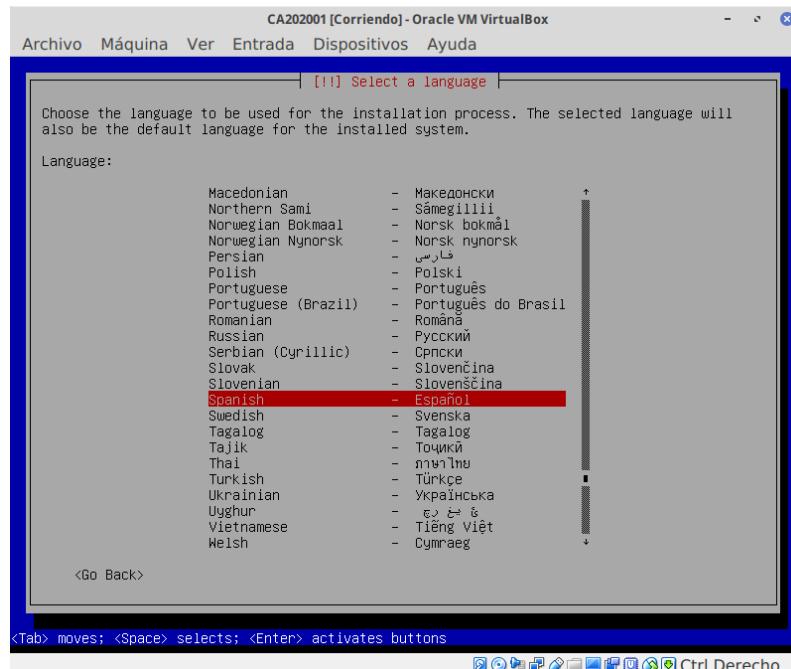


FIGURA 1.13: SE SELECCIONA DE IDIOMA DE LA INSTALACIÓN.

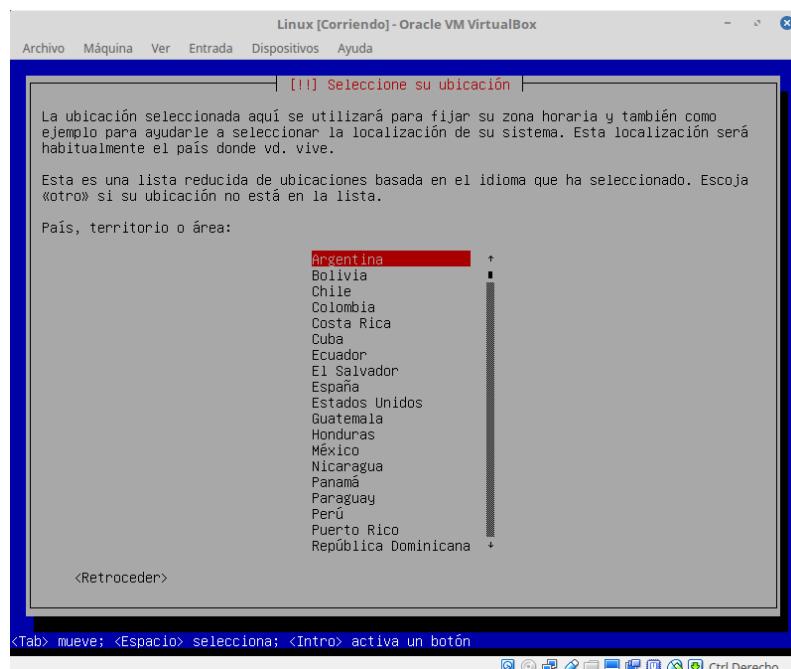


FIGURA 1.14: PAÍS DE LOCALIZACIÓN.

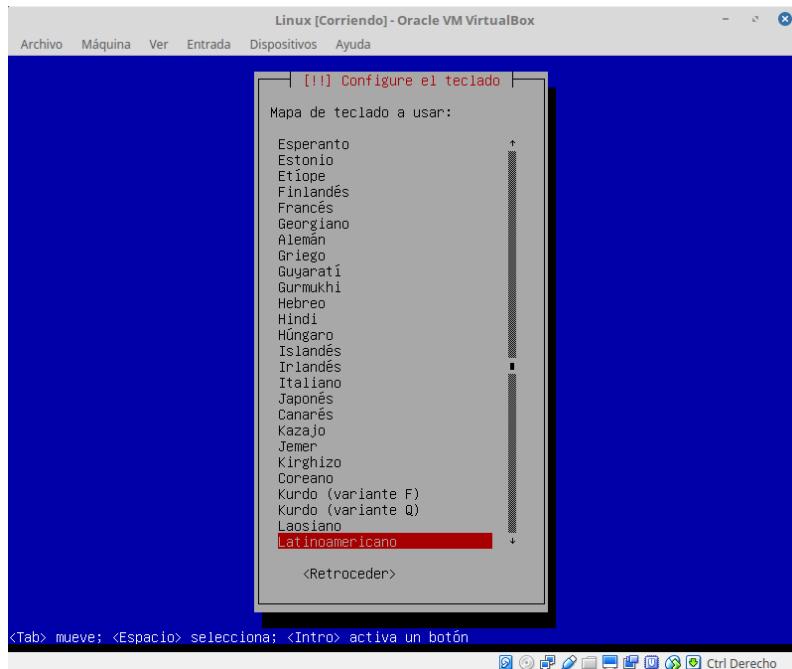


FIGURA 1.15: DISTRIBUCIÓN DEL TECLADO.

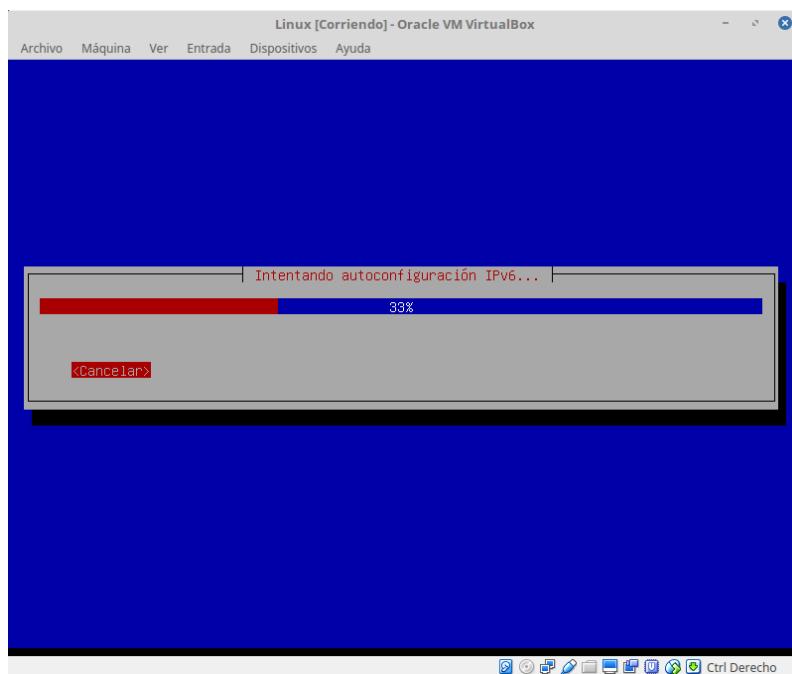


FIGURA 1.16: AUTOCONFIGURACIÓN DE LA RED.

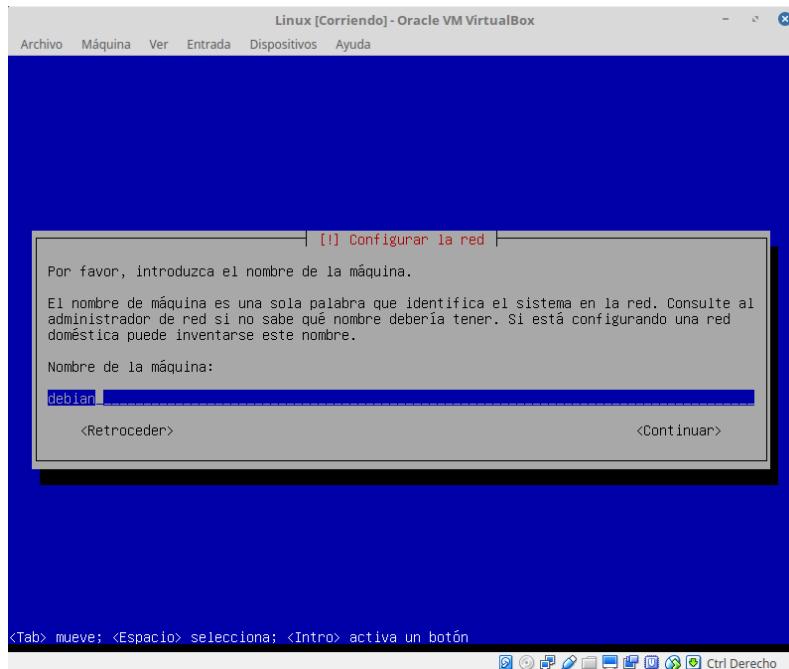


FIGURA 1.17: SE LE DA NOMBRE AL EQUIPO «HOST».

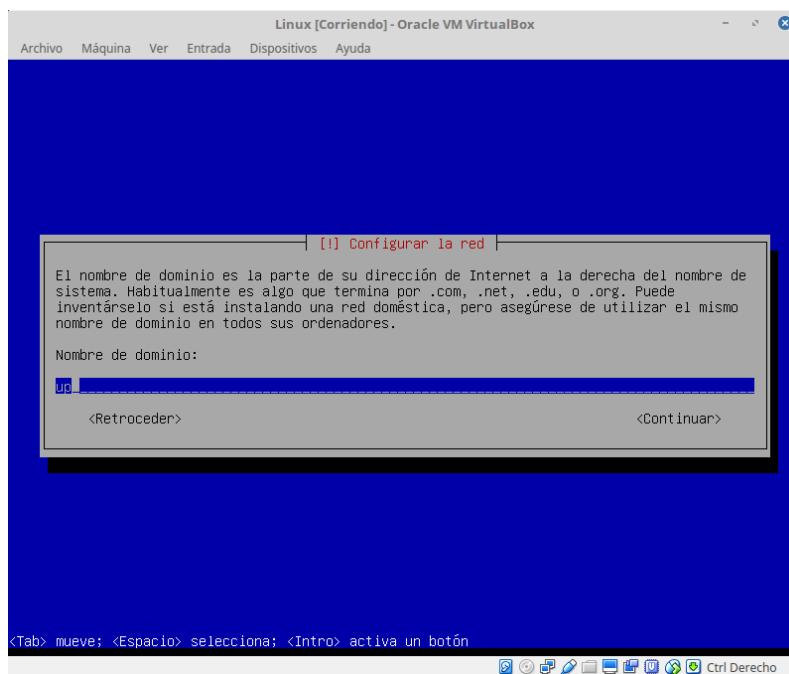


FIGURA 1.18: SE LE CONFIGURA EL DOMINIO DE LA RED.

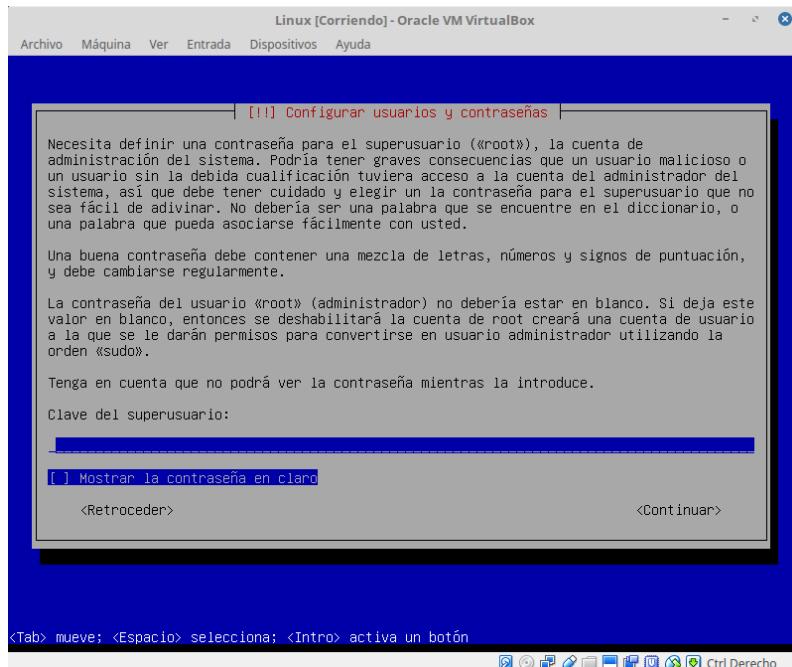


FIGURA 1.19: SE CONFIGURA LA CLAVE DEL USUARIO ROOT.

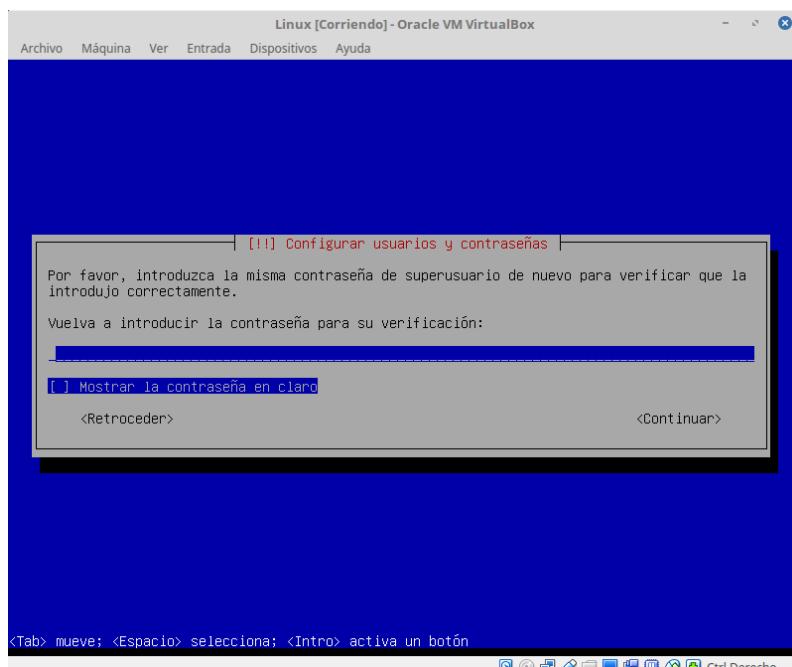


FIGURA 1.20: NUEVAMENTE SE VUELVE A INTRODUCIR LA CONTRASEÑA DE ROOT.

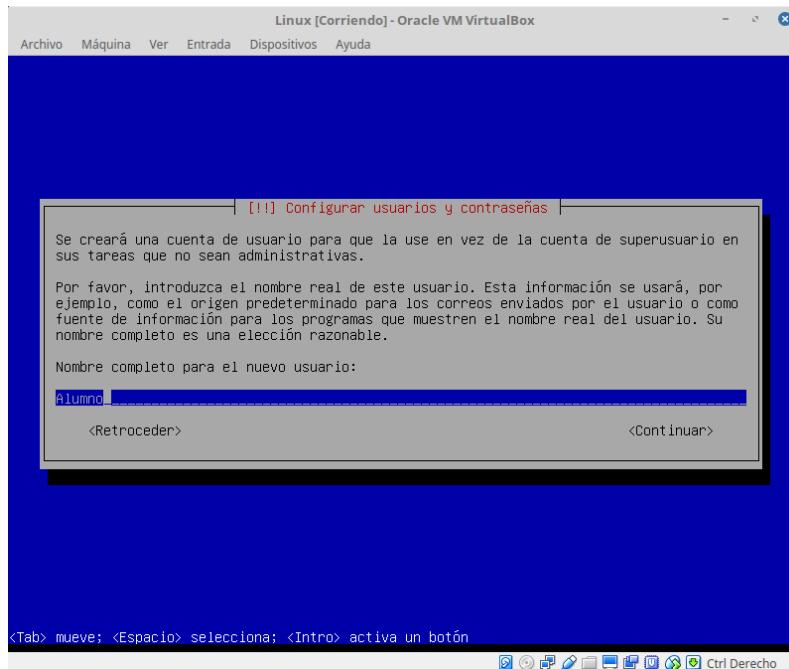


FIGURA 1.21: SE CREA UN NOMBRE DE USUARIO COMPLETO NO PRIVILEGIADO.

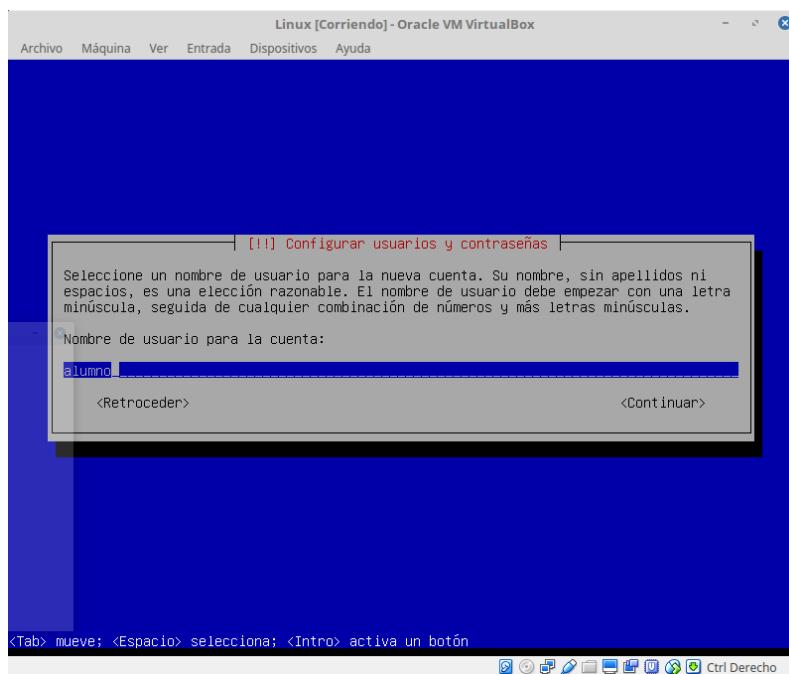


FIGURA 1.22: SE CREA UN NOMBRE DE USUARIO NO PRIVILEGIADO.

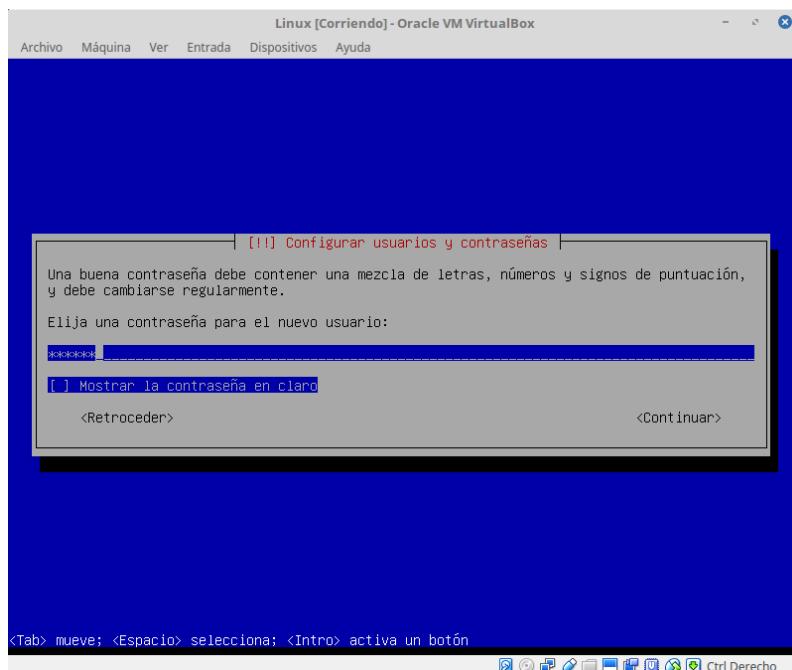


FIGURA 1.23: SE ELIJE LA CONTRASEÑA PARA ESE NUEVO USUARIO.

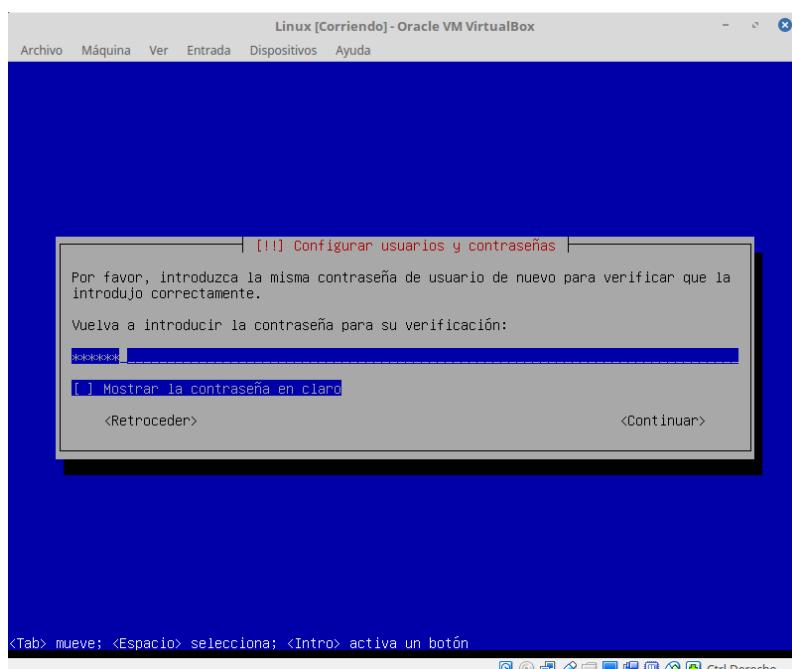


FIGURA 1.24: SE VUELVE A INTRODUCIR LA CONTRASEÑA.

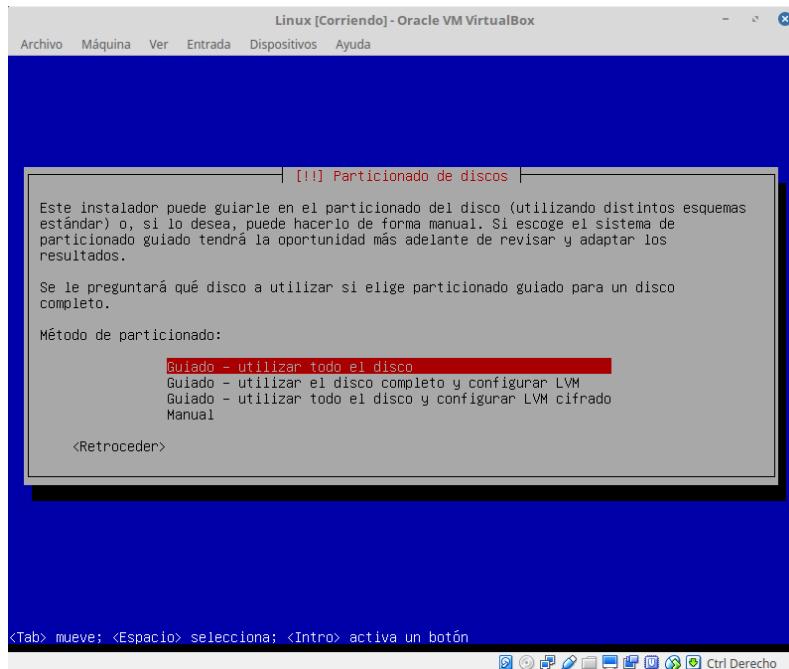


FIGURA 1.25: SE SELECCIONA TODO EL DISCO.

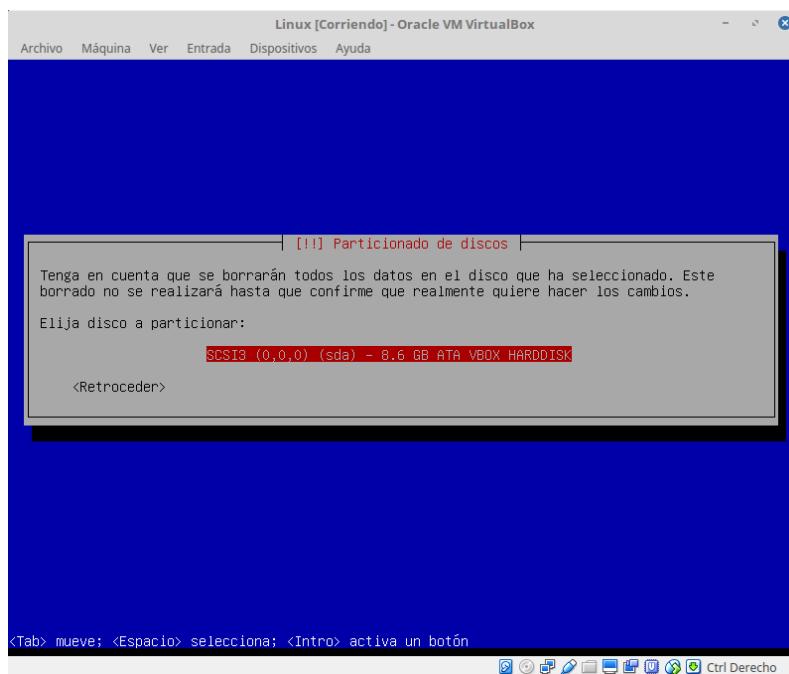


FIGURA 1.26: SE SELECCIONA EL DISCO.

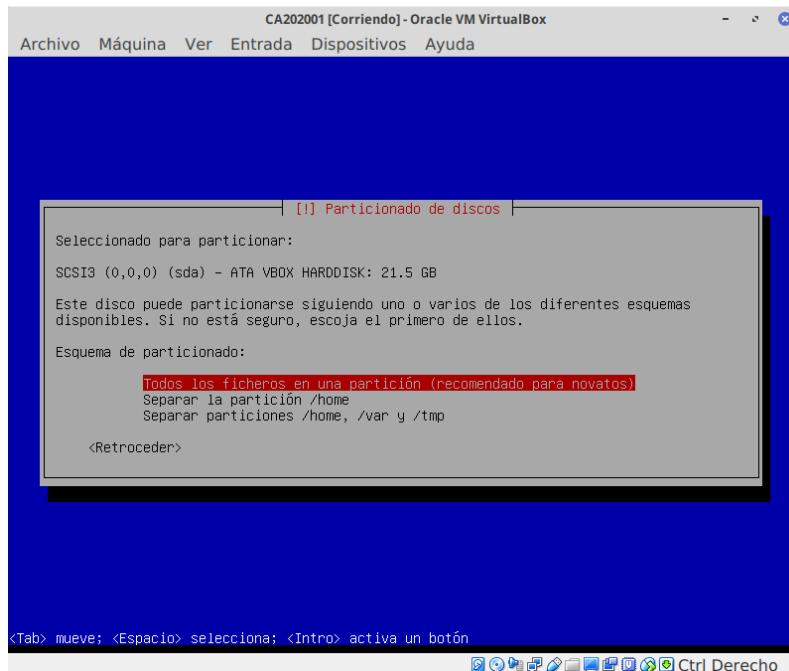


FIGURA 1.27: SE SELECCIONA PARTICIÓN SEPARADA PARA /HOME.

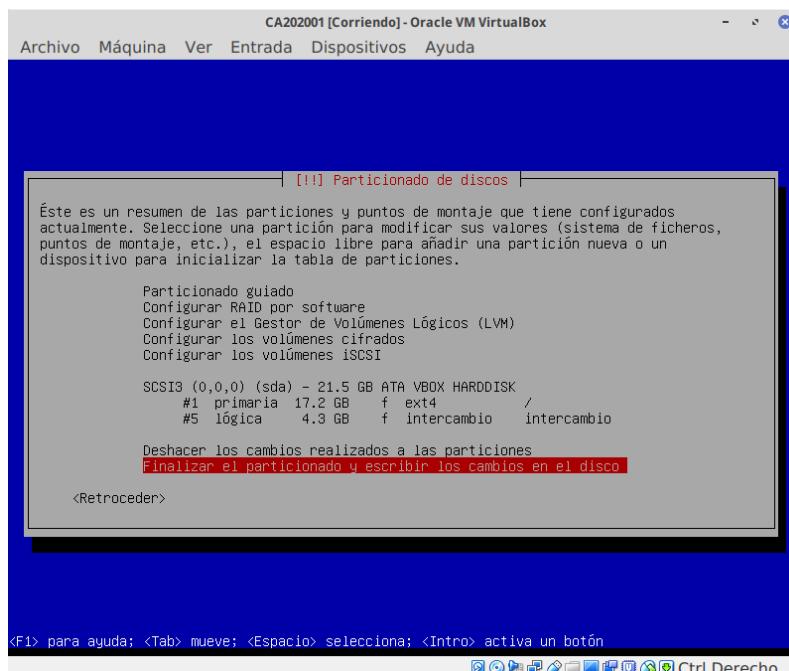


FIGURA 1.28: SE FINALIZA EL PARTICIONADO DEL DISCO.

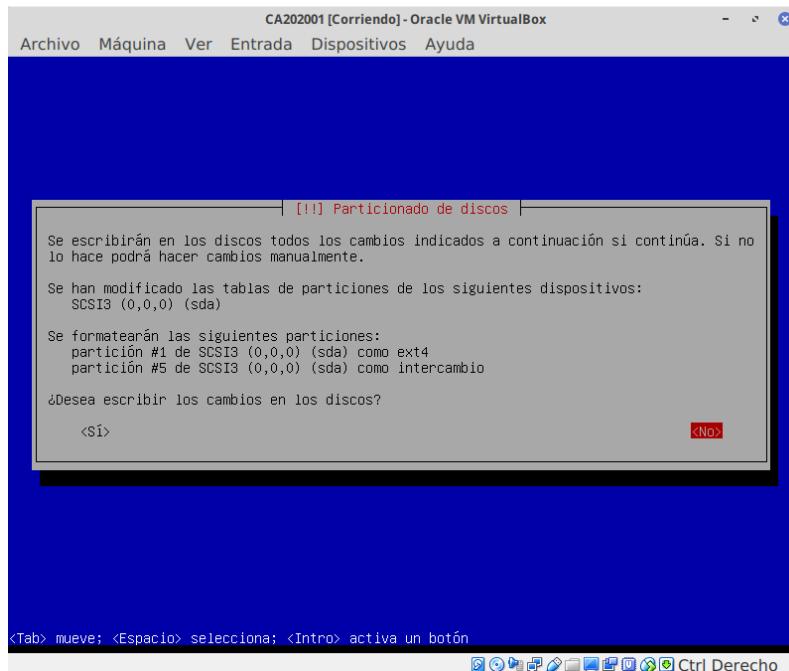


FIGURA 1.29: SE ESCRIBEN LOS CAMBIOS EN EL DISCO.

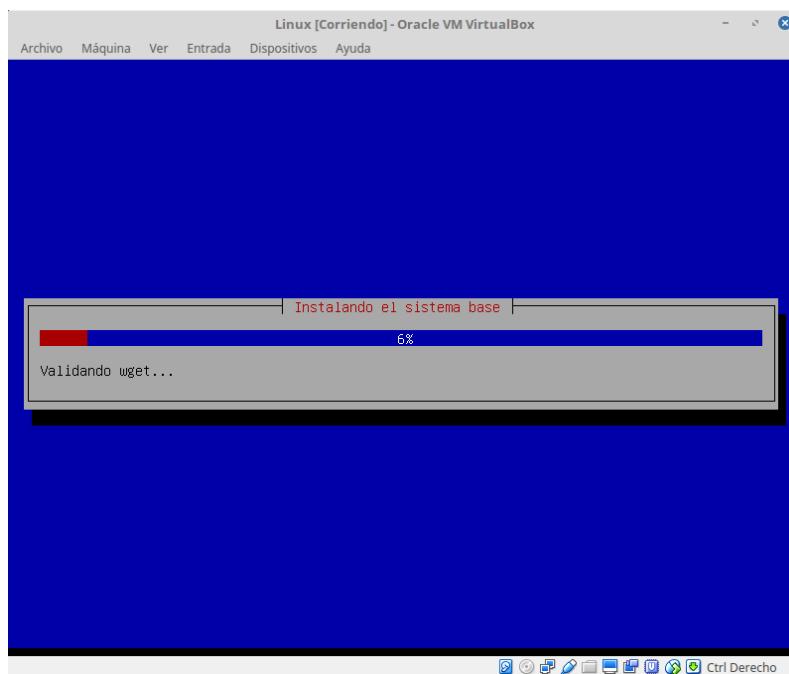


FIGURA 1.30: COMENZANDO A INSTALAR PAQUETES.

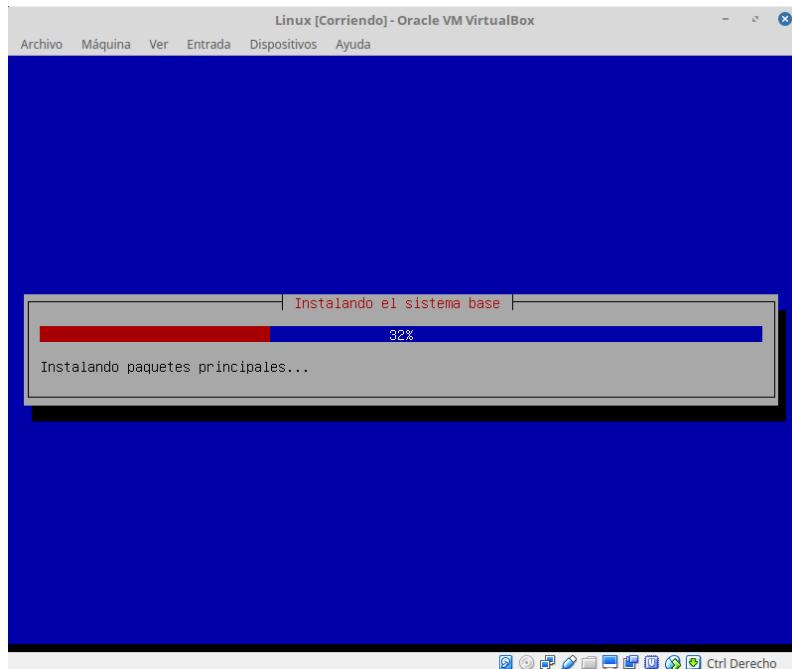


FIGURA 1.31: SIGUE INSTALANDO PAQUETES...

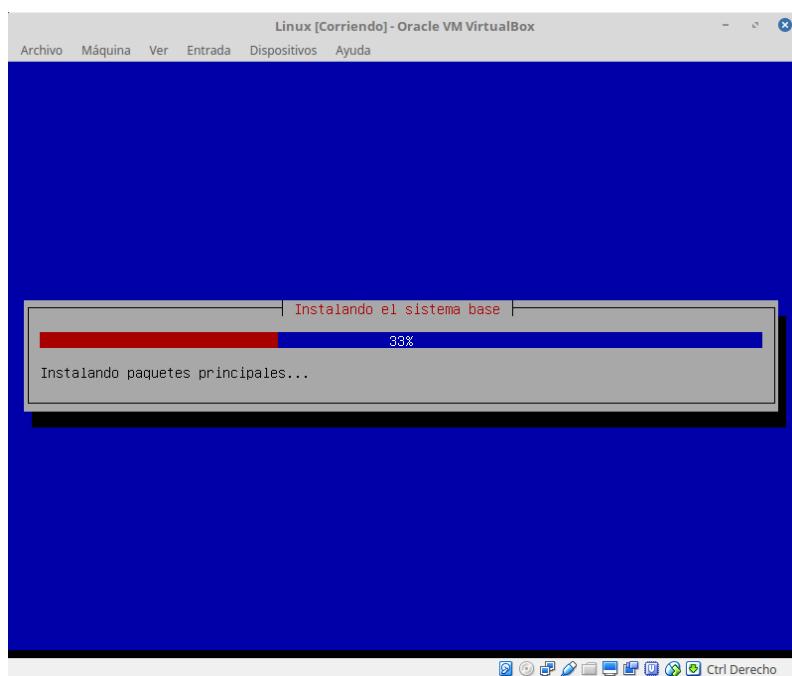


FIGURA 1.32: SIGUE INSTALANDO PAQUETES...

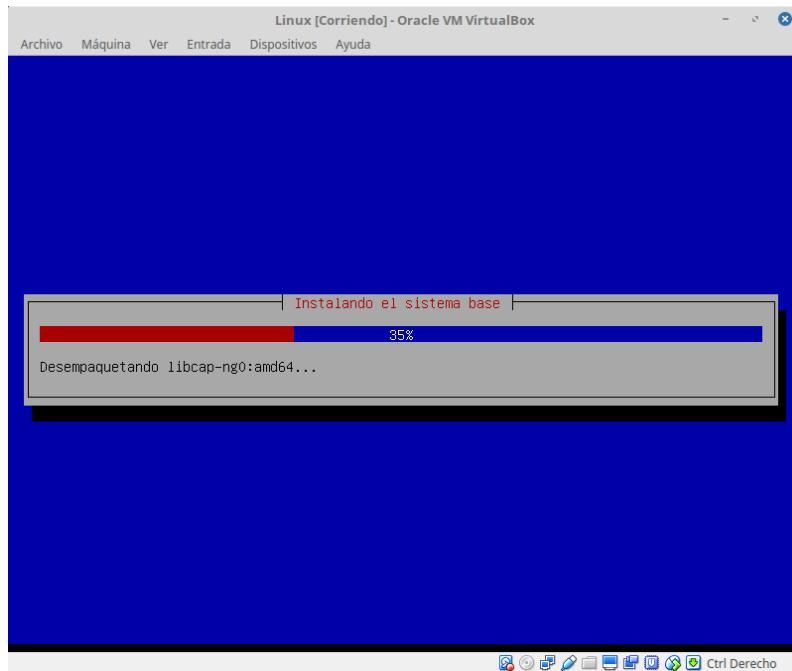


FIGURA 1.33: SIGUE INSTALANDO PAQUETES...

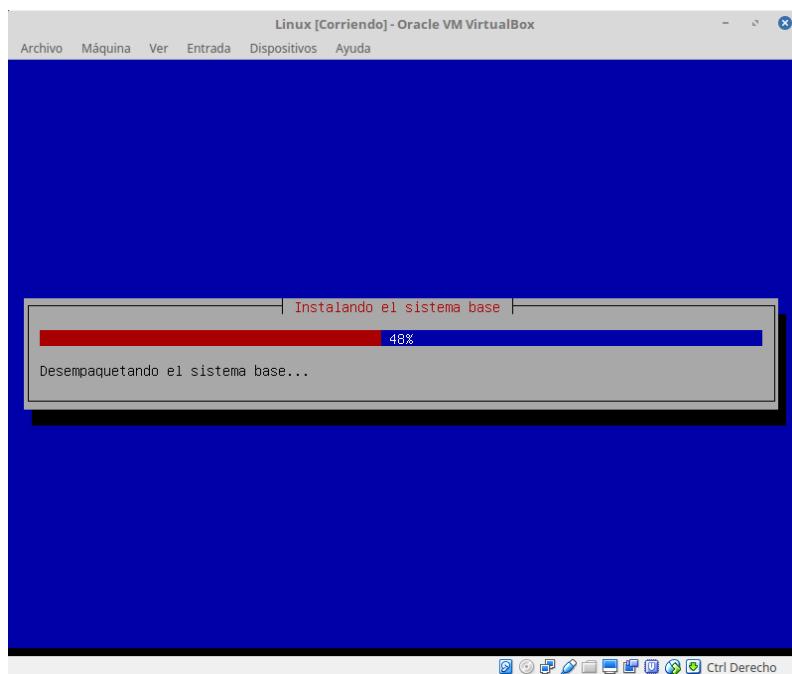


FIGURA 1.34: SIGUE INSTALANDO PAQUETES...

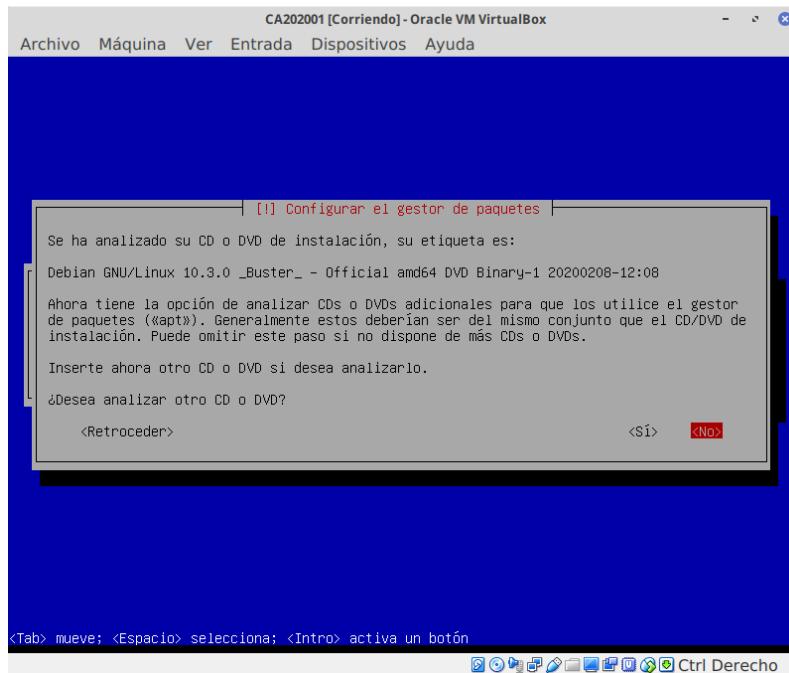


FIGURA 1.35: TERMINO DE REVISAR EL ISO1 Y SE RESPONDE QUE NO REVISE MAS.

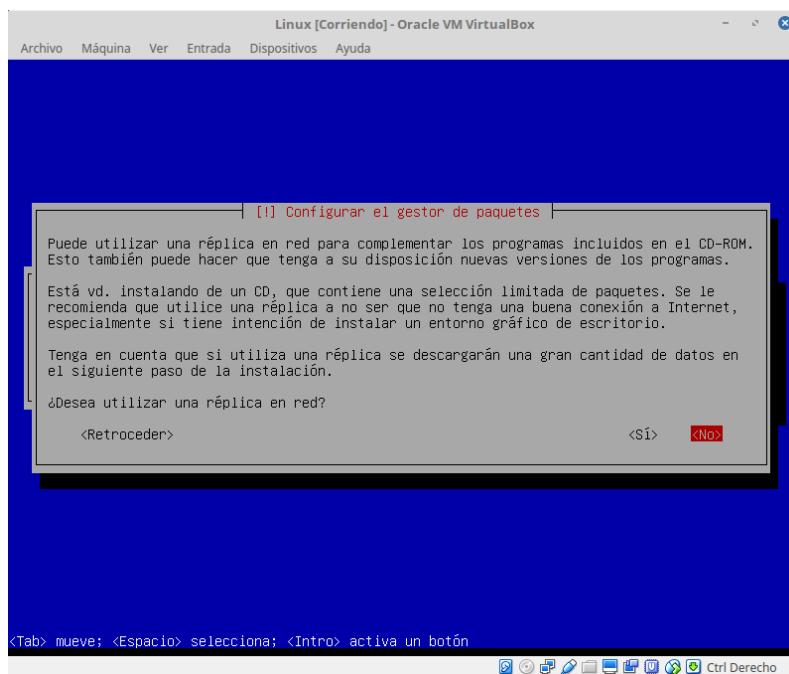


FIGURA 1.36: SE ELIJE NO UTILIZAR UNA REPLICAS DE RED.

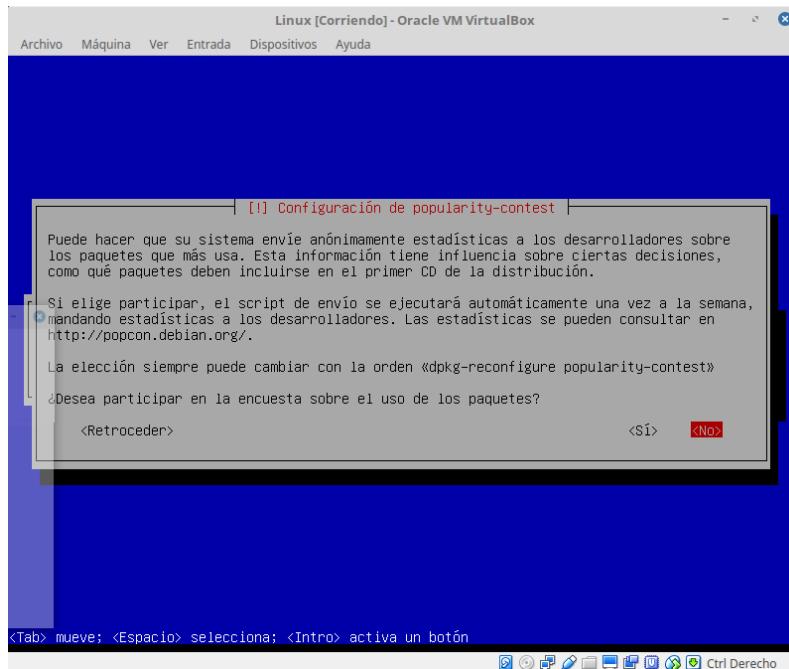


FIGURA 1.37: SE ELIJE NO PARTICIPAR EN LA ENCUESTA...

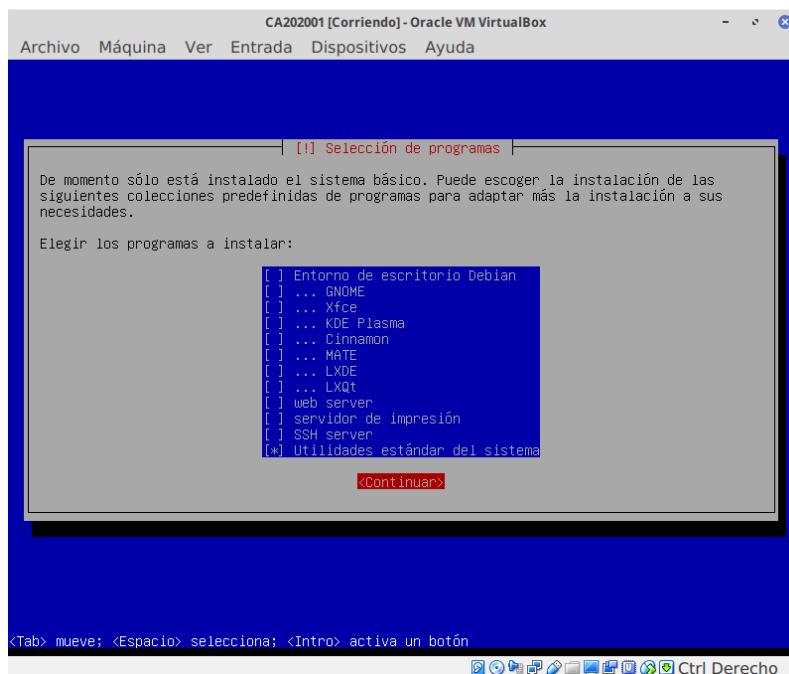


FIGURA 1.38: SE SELECCIONA ÚNICAMENTE UTILIDADES ESTÁNDAR DEL SISTEMA...

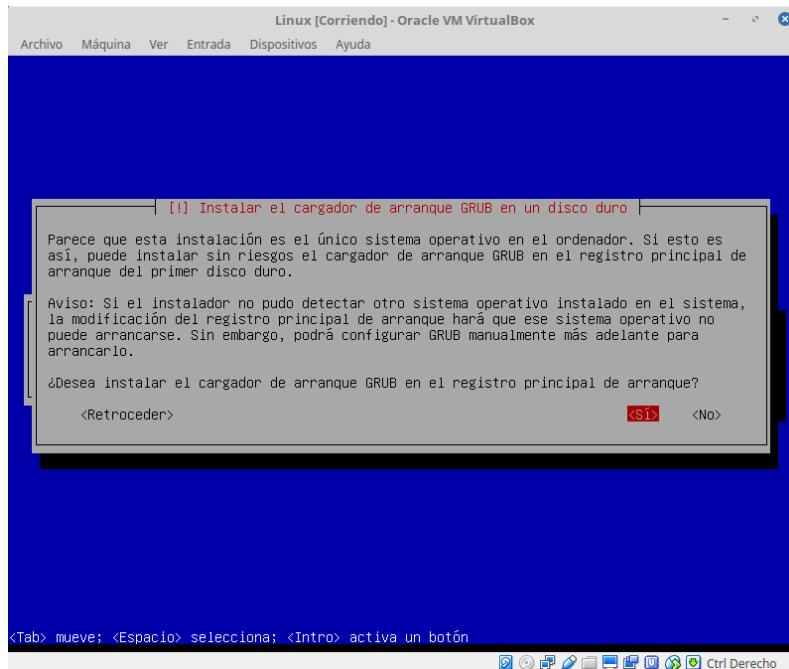


FIGURA 1.39: SE CONFIGURA INSTALAR GRUB.

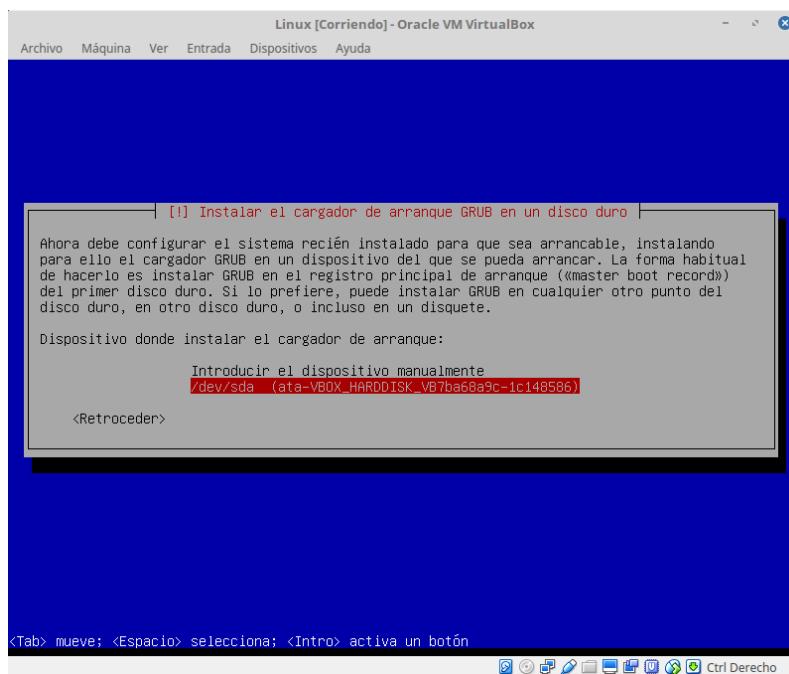


FIGURA 1.40: SE ELIGE EL DISPOSITIVO DONDE SE INSTALA GRUB.

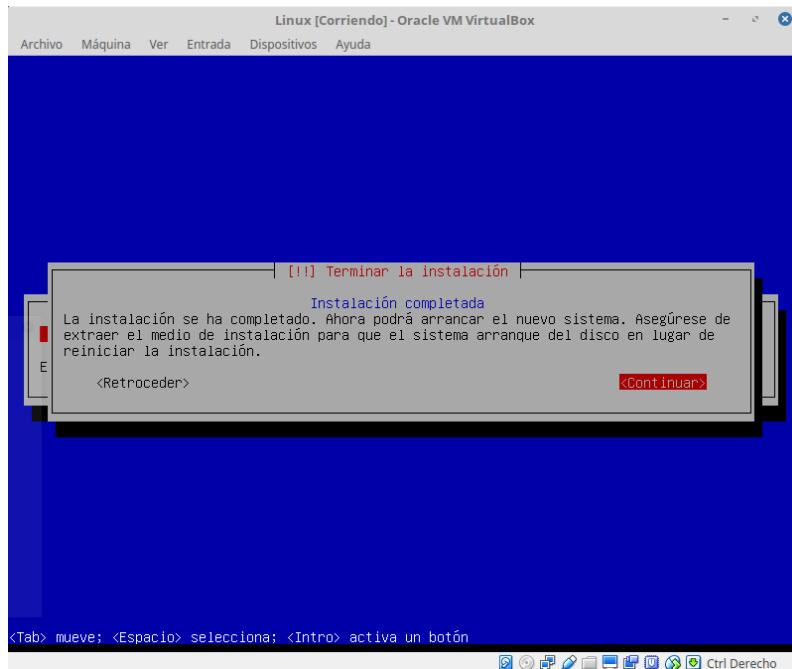


FIGURA 1.41: LA INSTALACIÓN FINALIZÓ SATISFACTORIAMENTE.

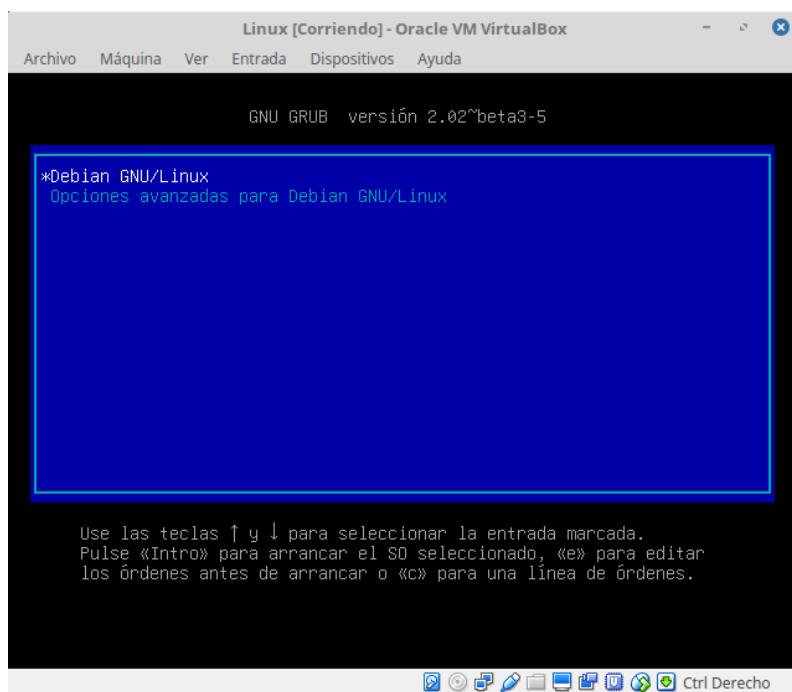


FIGURA 1.42: SE BOOTEA EL VM NUEVAMENTE Y APARECE EL MENU DE GRUB.

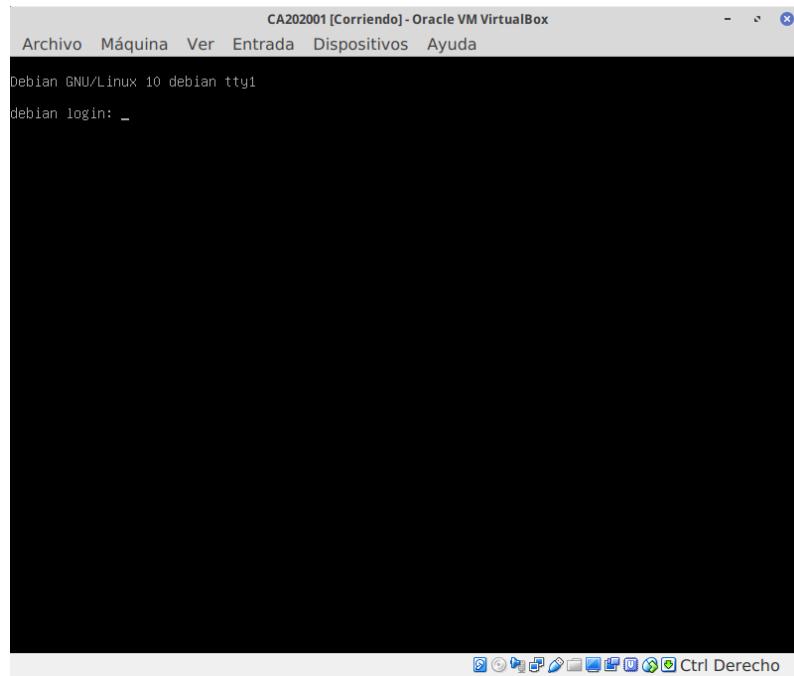


FIGURA 1.43: LA VM YA LISTA.

ensure@dollar@catcode=

## 2 FHS y Comandos

### 2.1 EL ESTÁNDAR FHS

Aunque cada distribución de Linux tiene sus propias características, la mayoría cumple el «estándar de jerarquía del sistema de archivos» (FHS). El proyecto FHS comenzó en 1993 con el objetivo de llegar a un acuerdo sobre cómo deberían organizarse los directorios, y qué archivos deberían almacenarse dónde; para que las distribuciones pudieran tener un punto de referencia único desde el cual trabajar. Muchas de las decisiones sobre la estructura de directorios se basaron en las estructuras de directorios tradicionales de \*nix con un enfoque en los servidores y con la suposición de que el espacio en disco era muy importante, por lo que las máquinas probablemente tendrían múltiples discos duros.

#### 2.1.1 DIRECTORIO «ROOT» (/)

Es el directorio de mayor importancia y del que dependen jerárquicamente el resto de los directorios.

##### El directorio /



Es uno de los directorios más críticos, ya que de llenarse, puede llevar a la caída del sistema operativo, y la necesidad de rebotarlo.

#### 2.1.2 /boot

Este directorio almacena todos los archivos del gestor de arranque (actualmente, GRUB), los archivos del kernel y los archivos «ramdisk»<sup>1</sup>. Generalmente se trata como una partición separada, de modo que el gestor de arranque puede leerlo más fácilmente. Con /boot en una partición separada, el sistema de archivos raíz puede usar funciones más sofisticadas que requieren soporte del núcleo, ya sea un sistema de archivos atípico, un cifrado de disco o una administración de volumen lógico (LVM).

<sup>1</sup>Contienen los controladores para reconocer el mínimo de hardware que se necesita para bootejar el Sistema Operativo

### 2.1.3 /dev

Aquí se encontrarán archivos de dispositivo. Los sistemas \*nix y GNU/Linux tienen el principio de «todo es un archivo», lo que significa que incluso el hardware termina siendo representado como un archivo. Este directorio contiene archivos para dispositivos en el sistema, desde discos y particiones hasta mouse y teclados. Esto es muy útil desde el punto de vista de la limitación (o mejor dicho, la falta de ella) de la cantidad de discos que se pueden conectar.

#### Límite en las cantidades de unidades de discos



Mientras que en Windows™(arrastrado por su pasado del D.O.S.) limita las unidades de disco a las letras entre la C: y la Z: (A: y B: se asignaban a las ya caducas «disketteras»), en Linux, los discos, como veremos mas adelante, se utilizan relacionándolos con archivos, y en consecuencia, no hay límite alguno.

### 2.1.4 /etc

El directorio `/etc` está destinado a almacenar archivos de configuración del sistema. Si se necesita configurar un servicio en un sistema Linux, o cambiar la red u otras configuraciones básicas, este es el primer lugar para buscar. Es un directorio pequeño que contiene la mayoría de las configuraciones que se pueden realizar en la computadora a nivel del sistema.

### 2.1.5 /home

Cada directorio en `/home` se nombra con nombre de un usuario particular y es propiedad de ese usuario (ejemplo `/home/pablo`). En un servidor, estos directorios almacenan los datos propios del usuario como puede ser el correo electrónico, sus credenciales digitales (SSH, etc). En los sistemas de escritorio, el directorio `/home` es probablemente el directorio principal con el que los usuarios interactúan. Cualquier configuración de escritorio, imágenes, medios, documentos u otros archivos que los usuarios necesitan, son almacenados en sus directorios `/home`. En un sistema de escritorio, este es el directorio más importante para realizar copias de seguridad y, a menudo, es un directorio al que se le asigna su propia partición.

#### /home en partición aparte



Al darle a `/home` su propia partición, se puede experimentar con diferentes distribuciones de Linux y volver a instalar el sistema completo en una partición / separada, y luego, al montar esta partición `/home`, todos los archivos y configuraciones están allí donde se los dejó.

### 2.1.6 /bin y /sbin

Estos directorios están diseñados para almacenar archivos ejecutables binarios. Ambos almacenan ejecutables que se consideran críticos para iniciar el sistema (como **CMD mount**). La principal diferencia entre ellos, es que el directorio **/sbin** está destinado a binarios<sup>2</sup> que los administradores usarán para administrar el sistema.

### 2.1.7 /lib

El directorio **/lib** almacena bibliotecas compartidas que los binarios esenciales en **/bin** y **/sbin** necesitan ejecutar. Este es también el directorio donde se almacenan los «módulos» (drivers o controladores) del kernel.

### 2.1.8 /lost+found

Este directorio existe en aquellos filesystems que tienen formato **ext2**, **ext3**, o **ext4**. Aquí se depositan los trozos de archivos perdidos que devienen de realizar una operación de reparación en el filesystem donde este directorio se encuentre.

### 2.1.9 /mnt y /media

Cuando se agregan sistemas de archivos adicionales a la computadora, ya sea desde una unidad USB, un montaje NFS u otras fuentes, se necesita algún lugar estándar para montarlos. Aquí es donde entran **/mnt** y **/media**. El directorio **/media** está diseñado para aquellos discos que se montan temporalmente, como CD-ROM y discos USB.

### 2.1.10 /usr/local

El directorio **/usr/local** es una versión especial de **/usr** que tiene su propia estructura interna de directorios **bin**, **lib** y **sbin**, pero **/usr/local** está diseñado para ser un lugar donde los usuarios pueden instalar su propio software fuera del software proporcionado por la distribución, sin preocuparse por sobrescribir ningún archivo de la distribución.

### 2.1.11 /opt

Los debates entre **/usr/local** y **/opt** son legendarios[**art:opt**]. Esencialmente, ambos directorios tienen el mismo propósito: proporcionar un lugar para que los usuarios instalen software fuera de sus distribuciones, pero el directorio **/opt** lo organiza de manera diferente. En lugar de almacenar binarios y bibliotecas para diferentes piezas de software juntas en un directorio compartido, como con **/usr** y **/usr/local**, el directorio **/opt** otorga a cada pieza de software su propio subdirectorio, y organiza sus archivos por debajo de lo que le plazca. La idea aquí es que, en teoría, se pueda desinstalar el software en **/opt** simplemente eliminando el directorio de ese software.

---

<sup>2</sup>También llamados «ejecutables».

## 2

## 2.1.12 /proc y /sys

Además de /dev, otros dos directorios terminan con archivos dinámicos que representan algo más que un archivo. El directorio /proc almacena archivos que representan información sobre todos los procesos en ejecución en el sistema. Se pueden usar herramientas como `ls` y `cat` para leer sobre el estado de los diferentes procesos que se ejecutan en el sistema. Este directorio a menudo también contiene archivos en /proc/sys que permiten ajustar parámetros particulares del kernel.

Si bien algunos archivos de estado del núcleo han aparecido en /proc (en particular /proc/sys), en estos días, se supone que se almacenan en /sys en su lugar. El directorio /sys está diseñado para contener todos estos archivos que le permiten interactuar con el kernel, y este directorio se llena dinámicamente con archivos que a menudo aparecen como series anidadas de enlaces simbólicos recursivos.

### ¿Para qué está /sys?



Es una medida para organizar el desorden en el que se convirtió /proc, ya que no solo contiene archivos de los procesos, sino de estados internos del kernel.

## 2.1.13 /root

El directorio /root es un directorio de usuario especial para el usuario «root» en el sistema. Es propiedad y es legible solo por el usuario root, y está diseñado para funcionar de manera similar a un directorio /home, pero solo para los archivos y configuraciones que el usuario root necesita. Actualmente, por razones de seguridad y buenas prácticas, se deshabilita al usuario root. De realizarse tareas que requieran privilegios de root, se utiliza `sudo`.

## 2.1.14 /tmp, /var/tmp y /dev/shm

Linux proporciona una serie de directorios que están diseñados para almacenar archivos temporales, según el tiempo que se deseé conservarlos. Estos directorios son ideales para los programas que necesitan almacenar algunos datos en un archivo temporalmente, es decir, aquellos datos no son necesarios que permanezcan para siempre. Lo que hace que estos directorios sean ideales para este propósito, es que se crean con permisos para que cualquier usuario pueda escribir en ellos.

- ▶ El directorio /tmp está destinado a almacenar archivos temporales. Cuando se inicia un sistema Linux, uno de los procesos de arranque iniciales borra todos los archivos en el directorio /tmp.
- ▶ El directorio /var/tmp, por otro lado, no se limpia entre reinicios, por lo que este es un buen lugar para almacenar archivos, como cachés.
- ▶ Finalmente, el directorio /dev/shm es un pequeño ramdisk, y todos los archivos que se almacenan allí residen solo en la RAM, y después de que el sistema se apaga, estos archivos se borran. Los crackers tienen preferencia por /dev/shm justamente por este motivo.

El directorio `/dev/shm` es el mejor de los tres, si se tienen archivos temporales que almacenan información confidencial como contraseñas o secretos, ya que nunca tocarán el disco; solo hay que asegurarse de dar los permisos apropiados (tema que se trata en la sección 5.7 en página 105) antes de colocar archivos secretos allí para que nadie pueda leerlos.

### 2.1.15 /usr, /usr/bin, /usr/lib y /usr/sbin

El directorio `/usr` pretende ser un directorio de solo lectura que almacena archivos que no son necesarios para iniciar el sistema. En general, cuando se instala software adicional de la distribución con sus binarios, bibliotecas y archivos de soporte, éstos son copiados en `/usr` en sus correspondientes directorios `/usr/bin`, `/usr/sbin` o `/usr/lib`, entre otros. Cuando el almacenamiento era escaso, a menudo se montaba este directorio por separado en su propio disco (de mayor tamaño), por lo que podría crecer independientemente al agregar nuevo software.

En estos días, existe una menor necesidad de tener este tipo de separación lógica, en particular porque los sistemas tienden a tener todo en una sola partición raíz grande, y los archivos que están involucrados en el procesos de booteo tienden a tener las herramientas necesarias para montar ese filesystem. Algunas distribuciones están comenzando a fusionar `/bin`, `/sbin` y `/lib` con sus correspondientes directorios `/usr` a través de enlaces simbólicos.

### 2.1.16 /var

El directorio `/var` fue diseñado para almacenar archivos que pueden variar enormemente en tamaño o pueden escribirse con frecuencia. A diferencia de `/usr`, que es de solo lectura, el directorio `/var` definitivamente debe poder escribirse, ya que dentro de él se encontrarán archivos de registro, «spool» del servidor de correo y otros archivos que pueden aparecer y desaparecer o que pueden aumentar su tamaño de maneras impredecibles.

Incluso en estos días, al menos en los servidores, si se tuviera que elegir un directorio de nivel raíz para colocarlo en su propio disco grande, el directorio `/var` sería el primero en la lista, no solo porque podría tener un tamaño bastante grande, sino también porque es posible que se deseé colocar `/var` en un disco que esté mejor optimizado para escrituras pesadas. Además, si se tiene todos sus directorios dentro de una partición raíz grande y si se queda sin espacio en el disco, el directorio `/var` es un excelente lugar para comenzar la búsqueda de archivos para eliminar.

### 2.1.17 DIFERENCIAS ENTRE /sys, /proc Y /dev

Es común notar que estos directorios generan confusión, por eso es que se citan las diferencias [[url:FHSproc](#)].

- ▶ `/dev`: Contiene los archivos de dispositivos, lo que permite el acceso de espacio de usuario al kernel. Todos los sistemas operativos POSIX lo tienen.
- ▶ `/proc`: Tiene su origen en los sistemas \*nix tipo System V, donde solo daba información sobre los procesos en ejecución. GNU/Linux amplió esto en forma considerable, agregando todo tipo de información. Este filesystem tiene archivos que se pueden escribir, cambiando en tiempo real parámetros internos del núcleo.
- ▶ `/sys`: Fue la opción más adecuada para arreglar el desorden de `/proc`. Idealmente toda aquella información que no correspondía a procesos y se encuentra en `/proc`, debería mudarse a este filesystem. Hay parámetros que están en `/sys` y no `/proc` y viceversa.

## 2

## 2.1.18 SUBJERARQUÍAS DE DIRECTORIOS

En el sistema de ficheros de \*nix y GNU/Linux (y similares), existen varias subjjerarquías de directorios que poseen múltiples y diferentes funciones de almacenamiento y organización en todo el sistema. Estos directorios pueden clasificarse en:

CLASIFICACIÓN	DESCRIPCIÓN
Estáticos	Contienen archivos que no cambian sin la intervención del administrador (root), sin embargo, pueden ser leídos por cualquier otro usuario. (/bin, /sbin, /opt, /boot, /usr/bin, etc.)
Dinámicos	Contienen archivos que son cambiantes, y pueden leerse y escribirse (algunos sólo por su respectivo usuario y el root) Contienen configuraciones, documentos, etc. Para estos directorios, es recomendable una copia de seguridad con frecuencia, o mejor aún, deberían ser montados en una partición aparte en el mismo disco, como por ejemplo, montar el directorio /home en otra partición del mismo disco, independiente de la partición principal del sistema (/); de esta forma, puede repararse el sistema sin afectar o borrar los documentos de los usuarios. (/var/mail, /var/spool, /var/run, /var/lock, /home)
Compartidos	Contienen archivos que se pueden encontrar en un ordenador y utilizarse en otro, o incluso compartirse entre usuarios.
Restringidos	Contiene ficheros que no se pueden compartir, solo son modificables por el administrador. (/etc, /boot, /var/run, /var/lock)

CUADRO 2.1: CLASIFICACIÓN DE DIRECTORIOS-

## 2.2 TIPOS DE ARCHIVOS

Los sistemas operativos de la familia de \*nix, como GNU/Linux, poseen seis tipos de archivos distintos.

## 2.2.1 ARCHIVOS REGULARES

Un archivo regular es un archivo en el sistema que contiene datos, texto o instrucciones de programa.

- ▶ Se utiliza para almacenar su información, como un texto que ha escrito o una imagen que ha dibujado. Este es el tipo de archivo con el que normalmente se trabaja.
- ▶ Siempre ubicado dentro / debajo de un archivo de directorio.
- ▶ No contiene otros archivos.
- ▶ En la salida de formato largo de **CMD ls -l**, este tipo de archivo se especifica mediante el símbolo «-».

## 2.2.2 DIRECTORIOS

Los directorios almacenan tanto archivos especiales como ordinarios. Para usuarios familiarizados con Windows o Mac OS, los directorios de \*nix y GNU/Linux son equivalentes a las carpetas. Un archivo de directorio contiene una entrada para cada archivo y subdirectorio que alberga. Si tiene 10 archivos en un directorio, habrá 10 entradas en el directorio. Cada entrada tiene dos componentes:

- 1 El nombre del archivo.
- 2 Un número de identificación único para el archivo o directorio (llamado número de «índice»)

Características de los directorios:

- ▶ Son puntos de ramificación en el árbol jerárquico.
- ▶ Se utiliza para organizar grupos de archivos.
- ▶ Pueden contener archivos ordinarios, archivos especiales u otros directorios.
- ▶ Nunca contienen información «real» con la que se trabajaría (como el texto).
- ▶ Solo se utilizan para organizar archivos.
- ▶ Todos los archivos son descendientes del directorio raíz, (llamado `/`) ubicado en la parte superior del árbol.
- ▶ En la salida de formato largo de `CMD ls -l`, este tipo de archivo se especifica mediante el símbolo «d».

## 2.2.3 ARCHIVOS DE DISPOSITIVO

Se utilizan para representar un dispositivo físico real, como una impresora, una interfaz de red, o un terminal, y se usan para operaciones de entrada/salida (I/O). Aparecen en el sistema de archivos como un archivo ordinario o un directorio. En los sistemas \*nix y GNU/Linux hay dos tipos de archivos especiales para cada dispositivo: archivos especiales de carácter y archivos especiales de bloques.

Para los dispositivos de carácter, la información se transmite de un carácter a la vez. Sin embargo, para los dispositivos de disco, el acceso sin formato significa leer o escribir en bloques enteros de bloques de datos, que son nativos del disco.

- ▶ Cuando se utiliza un archivo especial de carácter para la entrada/salida (E/S) del dispositivo, los datos se transfieren un carácter a la vez. Este tipo de acceso se llama acceso de dispositivo de carácter o sin formato.
- ▶ En la salida de formato largo de `CMD ls -l`, los archivos especiales de carácter están marcados con el símbolo «c».
- ▶ Cuando se utiliza un archivo especial de bloque para la entrada/salida (E/S) del dispositivo, los datos se transfieren en bloques grandes de tamaño fijo. Este tipo de acceso se denomina acceso de dispositivo de bloque.

- ▶ En la salida de formato largo de `CMD ls -l`, los archivos especiales de bloques están marcados con el símbolo «b».

#### 2.2.4 PIPES

\*nix y GNU/Linux permiten vincular comandos mediante una canalización. La canalización actúa como un archivo temporal, que solo existe para almacenar datos de un comando hasta que otro lo lea. Una tubería proporciona un flujo de datos de una sola vía. La salida o el resultado de la primera secuencia de comandos se utiliza como entrada a la segunda secuencia de comandos. Se verá en mayor profundidad en la sección 3.2 en página 75.

En la salida de formato largo de `CMD ls -l`, las tuberías con nombre están marcadas con el símbolo «p».

#### 2.2.5 SOCKETS

Un «socket» de \*nix y GNU/Linux (o socket de comunicación entre procesos) es un archivo especial que permite la comunicación avanzada entre procesos. Se utiliza en un marco de aplicación cliente-servidor. En esencia, es un flujo de datos, muy similar al flujo de red (y sockets de red), pero todas las transacciones son locales al sistema de archivos.

En la salida de formato largo de `CMD ls -l`, los sockets están marcados con el símbolo s.

#### 2.2.6 LINKS SIMBÓLICOS

El enlace simbólico se usa para hacer referencia a otro archivo del sistema de archivos. El enlace simbólico también se conoce como enlace suave o blando<sup>3</sup>. Contiene una forma de texto de la ruta al archivo al que hace referencia. Para un usuario final, el enlace simbólico parecerá tener su propio nombre, pero cuando intente leer o escribir datos en este archivo, en cambio, hará referencia a estas operaciones con el archivo al que apunta. Si eliminamos el enlace simbólico, el archivo de datos todavía estaría allí. Si eliminamos el archivo de origen o lo movemos a una ubicación diferente, el archivo simbólico no funcionará correctamente. Tiene un comportamiento similar a los «accesos directos» de Windows.

En la salida de formato largo de `CMD ls -l`, el enlace simbólico está marcado con el símbolo «l» (que es una L minúscula).

En el en el capítulo 7 en página 131, se trata mas en profundidad

### 2.3 CONOCIENDO LA TERMINAL DE TEXTO

Todo lo que se puede hacer con la interfaz gráfica, se puede hacer con linea de comandos. Es mas, por el contrario, se pueden realizar mayor cantidad de tareas via linea de comando que por medio de la interfaz gráfica. Para un mejor entendimiento, primero se hablará del «prompt». El prompt es el texto que usualmente tiene la forma descripta en la figura 2.1 de la siguiente página.

Se puede apreciar que el prompt de la figura termina con «~». Ese carácter especial representa la ubicación de inicio del usuario (C:\Users\...> en Windows).

<sup>3</sup>Depende de la traducción de «soft», en contraposición a «hard», otro tipo de link.

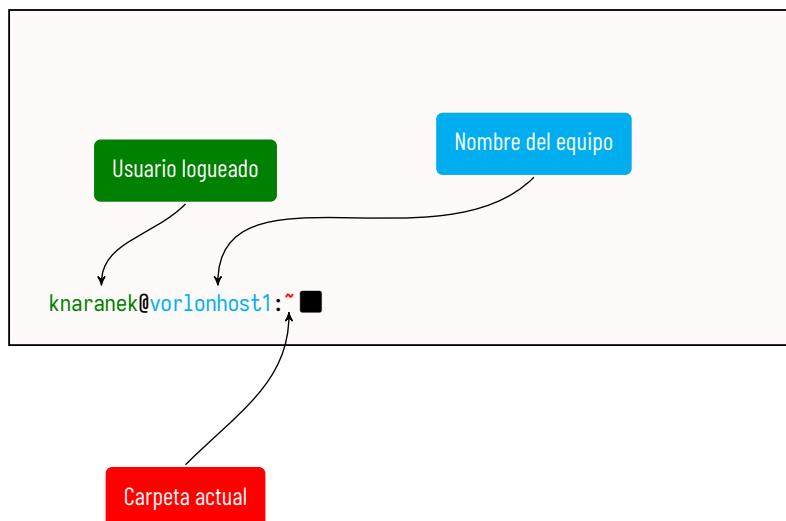


FIGURA 2.1: ELEMENTOS DE UNA TERMINAL DE COMANDOS

La terminal de comandos también está en Windows, y también viene con prompt, pero con características propias de ese sistema operativo. La forma que tiene el prompt en Windows es C:\>.

La terminal de comandos existe desde que \*nix y Linux nacieron, tiene distintos «sabores» según el tipo de interfaz que se utilice y hay para muchas variantes. Desde la de texto puro como getty capaz de generar una terminal vía un enlace serial, o las más utilizadas actualmente, que son las terminales de comandos gráficas que poseen muchas características que hacen al ingreso de comandos una tarea más amena.

**Terminal de comandos**

≡

La terminal de comandos es el aplicativo o herramienta que permite al usuario por medio de comandos, interactuar con el sistema operativo.

Al programa que se está ejecutando en la terminal, que recibe los comandos que se le ingresan, se le denomina «shell» o «interprete de comandos».

Las funciones del shell son entre otras:

- ▶ Recibir los comandos escritos en línea de comandos.
- ▶ Ejecutar los comandos recibidos y/o imprimir los errores que haya durante la ejecución.
- ▶ Interpretar los archivos de shell script.
- ▶ Configurar de forma sencilla el entorno de ejecución.

El shell asignado por defecto al usuario es **bash**. Hay otros shells como «csh», «zsh», «sh», etc.

Si bien hay mas combinaciones de teclas, para comenzar a trabajar, son suficientes las listadas a continuación:

- ▶ **Ctrl** + **I** → Limpia la pantalla.
- ▶ **↑** → Último comando ingresado. Subsecuentes pulsaciones navegan hacia atrás en los comandos ingresados.
- ▶ **↓** → Navega hacia adelante en los comandos ingresados.
- ▶ **Ctrl** + **b** → Busca lo que se esta tipeando, en el histórico de comandos.
- ▶ **Ctrl** + **d** → Cierra y sale de la sesión.
- ▶ **Tab** → Completa el comando que se esta escribiendo. **Muy utilizada.**

## 2.4 COMANDOS

Aquí se presentan los comandos que son utilizados mayoritariamente. Las opciones aquí indicadas son solo algunas de las disponibles en cada comando. Es altamente recomendable utilizar las páginas **man**, que tienen documentados los comandos del sistema operativo.

### Mensajes



Téngase presente que en la mayoría de los casos (generalmente en los filtros), los comandos escritos en la terminal no devuelven ningún mensaje durante su ejecución ni al finalizar la misma, excepto que se les indique lo contrario.

COMANDO	DESCRIPCIÓN
<b>CMD</b> apropos <b>CMD</b> cat	Busca entre las páginas de manual y las descripciones. Abre el archivo (para verlo de a poco se tienen los paginadores. <b>CMD</b> more y <b>CMD</b> less) o sino con <b>CMD</b> > archivo, crea archivo y lo rellena con ese contenido.
<b>CMD</b> cd <b>CMD</b> cp <b>CMD</b> df	Cambia de directorio. Copia archivos. Indica espacio disponible en los «filesystems» (particiones). Con la opción -h informa ese espacio en términos humanos.
<b>CMD</b> diff	Muestra las diferencias entre archivos. La salida generada en caso de ser códigos de fuentes, es utilizada por <b>CMD</b> patch.
<b>CMD</b> du	Resume el uso de espacio de disco para un conjunto de archivos, o en forma recursiva para directorios.
<b>CMD</b> exit <b>CMD</b> find	Produce la terminación normal de un programa. Busca archivos en tiempo de ejecución sobre el filesystem indicado, por medio de criterios tan específicos como se necesite, permitiendo ademas invocar algún comando específico con cada archivo que cumpla el criterio.

Cuadro 2.2 – Continuación

COMANDO	DESCRIPCIÓN
 <b>grep</b>  <b>head</b>  <b>history</b>  <b>locate</b>	Filtre un archivo e imprime las líneas que concuerdan con el patrón dado. Muestra las primeras diez (por defecto) líneas del comienzo de un archivo. Lista el histórico de comandos. <sup>4</sup>
 <b>logout</b>	Busca archivos, consultando la base de datos, generada por medio de  <b>updatedb</b> . Sale de la sesión. La ayuda del comando  <b>logout</b> se encuentra en la página de manual del  <b>bash</b> , por ser un comando interno del mismo.
 <b>ls</b>	Listar el contenido de directorios. ► <b>-a</b> → Muestra archivos ocultos. ► <b>-l</b> → Muestra el detalle de los archivos. Lista los archivos con mucho más detalle, especificando para cada archivo sus permisos, el número de enlaces rígidos, el nombre del propietario, el grupo al que pertenece, el tamaño en bytes <sup>5</sup> y la fecha de la última modificación.
 <b>mkdir</b>	Crea directorios. ► <b>-p</b> → Crea directorios en cascada, es decir crea los directorios padres hasta llegar al que se desea en última instancia.
 <b>more</b>	Es un paginador, muestra el contenido de un archivo de texto, paginando por pantallas. Se avanza mediante  .
 <b>mv</b>  <b>pwd</b>  <b>rm</b>  <b>rmdir</b>  <b>shutdown</b>	Mueve archivos o directorios. También renombra. «print working directory» (muestra el directorio actual). Borra archivos o directorios. Borra directorios vacíos únicamente. Las tareas que en común realiza son: detiene cada uno de los servicios, sincronizando <sup>6</sup> los filesystems, y desmontandolos. Ahora bien, mediante parámetros, se indica que tipo de apagado o reinicio se desea: ► <b>-H/halt</b> → Deja la máquina eléctricamente encendida, pero el sistema operativo esta bajo, el kernel detuvo los CPUs. ► <b>-r/reboot</b> → Reset del equipo. ► <b>-p/poweroff</b> → Este es el valor por default. Inicia el apagado ACPI <sup>7</sup> del equipo.
 <b>tail</b>	Muestra las primeras diez (por defecto) líneas del final de un archivo.

<sup>4</sup>Tener presente que de haber más de una sesión abierta, solo mostrará el histórico de comandos previo a la dicha sesión y los tipeados luego.

<sup>5</sup>Deviene del inglés «bite» morder, es la cantidad de bits que simultáneamente pueden ser procesados por la computadora. Actualmente, 8 bits.

<sup>6</sup>Vaciando los buffers y cache de disco, escribiendo los cambios efectivamente, mediante el llamado «sync» del kernel.

<sup>7</sup>ACPI es el acrónimo inglés de «Advanced Configuration and Power Interface» (Interfaz Avanzada de Configuración y Energía). Es un estándar resultado de la actualización de APM a nivel de hardware, que controla el funcionamiento del BIOS y proporciona mecanismos avanzados para la gestión y ahorro de la energía. Va más allá de las posibilidades de APM. Así, por ejemplo, convierte la pulsación del botón de apagado en un simple evento, de tal forma que el sistema operativo puede detectarlo y le permite efectuar un apagado ordenado de la máquina, sin riesgo para el hardware de ésta como ocurría anteriormente.

Cuadro 2.2 – Continuación

COMANDO	DESCRIPCIÓN
<b>CMD touch</b>	Modifica fecha de último acceso. Muy utilizado para crear archivos vacíos.
<b>CMD uname</b>	Imprime información sobre el sistema.
<b>CMD updatedb</b>	Explora los filesystems y genera una base de datos con los archivos y su ubicación, para ser consultada luego por <b>CMD locate</b> .
<b>CMD wc</b>	Muestra la cantidad de líneas, palabras y bytes de un archivo (en ese orden).
<b>CMD whatis</b>	Imprime la descripción breve de la página de manual.
<b>CMD whereis</b>	Busca e informa la ubicación y el archivo de la página <b>CMD man</b> sobre un comando determinado.
<b>CMD which</b>	Busca un comando, entre los paths (directorios) que se encuentran mencionados en la variable de entorno «PATH».

CUADRO 2.2: COMANDOS

**Atención con los espacios**

En la línea de comando, los espacios son separadores de campo y argumentos (lo define la variable de entorno IFS). Los espacios deben ser tenidos en cuenta ya que si no, se pueden producir resultados no deseados, más aún con los comandos «destructivos», como **rm**.

**2.4.1 man**

**CMD man** es una herramienta que se utiliza para documentar y aprender sobre comandos, archivos, llamadas de sistema, etc.; disponible en sistemas operativos \*nix y GNU/Linux.

La mayoría de las aplicaciones aportan documentación de manual accesible desde el mismo comando:

**SIN man <comando>**

Con **[q]** o **[Ctrl]+[C]** se sale de las páginas man y se vuelve al prompt.

Por ejemplo, para saber todo lo relacionado sobre el comando **CMD man**, se escribe el comando **CMD man man**:

```
ttyS9@vorlonhost:~$ man man
MAN(1)          Útiles de Páginas de Manual          MAN(1)
NOMBRE
man - una interfaz de los manuales de referencia electrónicos
```

**SINOPSIS**

```
man [-c|-w|-tZT dispositivo] [-adhu7V] [-m sistema[,...]] [-L locale]
[-p cadena] [-M ruta] [-P paginador] [-r prompt] [-S lista] [-e
extension] [[sección] pagina ...] ...
man -l [-7] [-tZT dispositivo] [-p cadena] [-P paginador] [-r prompt]
fichero ...
man -k [-M ruta] palabra_clave ...
man -f [-M ruta] pagina ...
:q
tty$9@vorlonhost:~$ █
```

En ocasiones el mismo programa posee diversas secciones de manual, cuando esto ocurre suele mostrar un mensaje indicando la sección entre paréntesis, como «man(1)» y «man(7)», o «exit(1)» y «exit(3)».

En el cuadro 2.3 de la siguiente página, se muestran las secciones disponibles. Para acceder a estas secciones basta con indicarlo de la siguiente forma:

**SIN** `man [sección] <comando>`

Ejemplo:



```
tty$9@vorlonhost:~$ man 3 printf
PRINTF(3)           Linux Programmer's Manual          PRINTF(3)

NAME
printf, fprintf, dprintf, sprintf, snprintf, vprintf, vfprintf,
vdprintf, vsprintf, vsnprintf - formatted output conversion

SYNOPSIS
#include <stdio.h>
:q
tty$9@vorlonhost:~$ █
```

### Manual en «PDF»



Si se quiere generar alguna página man en formato pdf (por ejemplo, de «bash»), basta con ejecutar el siguiente comando:

**CMD** `man -t bash | ps2pdf - > bash.pdf`

Sección	Descripción
0	Archivos de cabecera de la biblioteca estándar de C.
1	Comandos Generales. (default)
2	Llamadas al sistema. <sup>8</sup>
3	Biblioteca C de funciones.
4	Archivos especiales (normalmente dispositivos, que se pueden encontrar en /dev) y drivers.
5	Formatos de fichero y convenciones.
6	Juegos y salvapantallas.
7	Miscelánea.
8	Comandos de administración del sistema y servicios del sistema operativo (daemons).
9	Rutinas del Kernel.
n	La librería gráfica Tcl/Tk.
x	El subsistema gráfico «X Window».

CUADRO 2.3: SECCIONES DE LAS PÁGINAS MAN.

#### 2.4.2 tar

**CMD** **tar** «Tape ARchive» →Se usa para empaquetar archivos en un solo archivo (por sí solo empaqueta, no comprime). Con las opciones de compresión invoca las librerías de los algoritmos de compresión «gzip» (-z) y «bzip2» (-j). Es la opción por defecto para hacer backups.

##### PARA REALIZAR UN «BACKUP»

Su sintaxis es la siguiente:

```
SIN tar cvzf <archivo_backup>.tar.gz <directorio_a_backuper>
```

Un ejemplo de uso es, si se quisiera realizar un backup del servidor web apache, y del motor de base de datos MySQL<sup>910</sup>:

```
CMD tar czf mysql.tar.gz /var/lib/mysql
```

```
CMD tar czf apache.tar.gz /var/www
```

##### PARA REALIZAR UN «RESTORE»

Su sintaxis es la siguiente:

```
SIN tar xvzf <archivo_backup>.tar.gz -C <donde se dejan los archivos restaurados>
```

Y un ejemplo de restauración en base al ejemplo anterior, sería:

```
CMD tar xzf mysql.tar.gz -C /var/lib
```

```
CMD tar xzf apache.tar.gz -C /var
```

<sup>9</sup>Hoy MariaDB.

<sup>10</sup>Tener presente que debe detenerse el servicio de MySQL previamente a la realización del backup.

## 2.5 CONCATENACIÓN DE COMANDOS

Todos los comandos que se escriben en el shell «BASH», se pueden encadenar y se ejecutan en función de su código de retorno; **si es cero, la operación fue exitosa, caso contrario, no lo fue.** Como se puede apreciar en el cuadro respectivo, estos operadores guardan relación con los operadores lógicos del lenguaje C.[book:C]

Operador	Expresión	Descripción
	cmd1    cmd2	cmd2 se ejecuta únicamente, si cmd1 <b>no</b> finalizó satisfactoriamente.
&&	cmd1 && cmd2	cmd2 se ejecuta únicamente, si cmd1 finalizó satisfactoriamente.
;	cmd1 ; cmd2	cmd2 se ejecuta una vez que cmd1 finaliza su ejecución sin importar si fue satisfactoria o no.

CUADRO 2.4: CONCATENACIÓN DE COMANDOS EN BASH.

## 2.6 EXPRESIONES REGULARES

### Expresiones Regulares



También conocida como «regex», «regexp» o«expresión racional», es una secuencia de caracteres que forma un patrón de búsqueda, principalmente utilizada para la búsqueda de patrones de cadenas de caracteres u operaciones de sustituciones.

Las expresiones regulares se usan habitualmente para varias operaciones:

- ▶ **Validaciones de datos** → Verificar si una cadena está bien formada.
- ▶ **Data Scraping** → Buscar páginas web que tengan un conjunto específico de palabras, eventualmente en un orden determinado.-
- ▶ **Data Wrangling** → Transformar datos crudos, a otro formato.
- ▶ **Parseo de cadenas** → Por ejemplo obtener todos los parametros de un verbo GET.
- ▶ **Otros** → Coloreado de sintaxis, packet sniffing, etc.

Se las puede clasificar en:

- ▶ Alternación
- ▶ Cuantificación
- ▶ Agrupación

- ▶ Grupos de caracteres
- ▶ Negación
- ▶ Carácteres especiales
- ▶ Escapes

### 2.6.1 ALTERNACIÓN

Expresiones del tipo «a o b» → a|b, donde el carácter «|» hace del operador OR.

#### EJEMPLOS

- /a|b/ coincide con el carácter a o el b.  
 /pepa|e/ coincide con pepe o pepa.

### 2.6.2 CUANTIFICACIÓN

Expresiones del tipo aaaa → a{5}.

- . Un carácter únicamente.
- ? Que el carácter precedente, concuerda una vez o ninguna.
- \* Que el carácter precedente, concuerde ninguna o más veces.
- + Que el carácter precedente, concuerde una o más veces.
- {n} Que el carácter precedente concuerde exactamente  $n$  veces.
- {n,m} El carácter precedente concuerda por lo menos  $n$  veces, y no más que  $m$  veces.

#### Compatibilidad



Se debe tener presente que no todas las expresiones regulares funcionan de la misma manera en todos los lenguajes y/o versiones de los comandos.[book:regex]

#### EJEMPLOS

- /ab+c/ coincide con abc y abbc, pero no con ac.  
 /ab?c/ coincide con ac y abc, pero no con abbc.  
 /pepe\*/ coincide con pep, pepe y pepeeeee  
 /a{1,2}/ coincide con a y aa, pero no con aaa.  
 /a{2,}/ coincide con aa y aaaaa, pero no aa.

### 2.6.3 AGRUPACIÓN

Los parentésis permiten agrupar una expresión regular, para trabajarla como subexpresiones.

#### EJEMPLOS

$(abc)^+$  coincide al menos una vez con toda la expresión abc.

$((ab)^+1)^+$  coincide con abababab1ab1.

Dentro de los corchetes «[]» se puede poner un grupo de caracteres que se desea encontrar.

#### EJEMPLOS

[aeiou] coincide con cualquier vocal.

[a-z0-9] se permiten rangos, esta expresión coincide con cualquier letra de la «a» a la «z» y con cualquier número.

[^aeiou] el símbolo ^ dentro de un grupo es la negación, esta expresión coincide con cualquier carácter que no sea una vocal.

#### Recursos externos



Hay varios recursos en internet sobre este tópico. Uno muy ilustrativo es <https://extendsclass.com/regex-tester.html> o también <https://regex101.com>.

### 2.6.4 CARÁCTERES ESPECIALES

- . Cualquiera carácter (casi cualquiera).

\n Nueva línea.

\t Tabular.

^ Concuerda con el comienzo de línea.

\$ Concuerda con el fin de línea.

### 2.6.5 ALGUNOS EJEMPLOS

#### INTRODUCCIÓN

Estos ejemplos pueden ser probados por medio de los comandos **CMD egrep** o **CMD grep -E**, que es lo mismo.

La forma de uso es:

```
SIN echo '<expresion a verificar>' | egrep '<expresion regular>'
```

Por ejemplo, una regexp que solo deje pasar expresiones numéricas Figure 2.2.

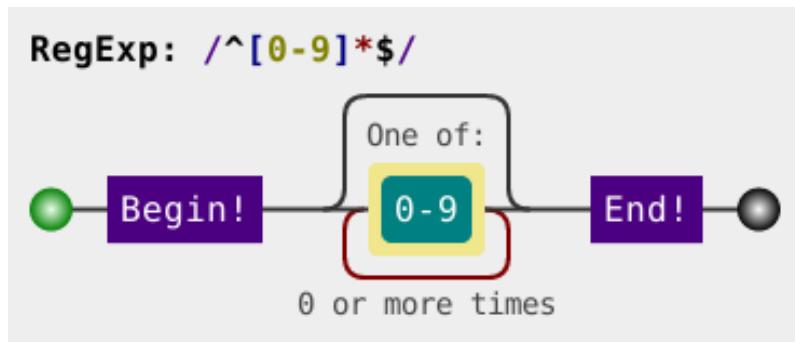


FIGURA 2.2: REGEXP: SOLO NÚMEROS.

```
ttyS9@vorlonhost:~$ echo 999|egrep '^[0-9]*$'
999

ttyS9@vorlonhost:~$ echo 999a|egrep '^[0-9]*$'
ttyS9@vorlonhost:~$ █
```

#### VALIDADOR DE MAIL

Un validador de mail (figura 2.3) acorde a la RFC<sup>11</sup> 821 del año 1982.

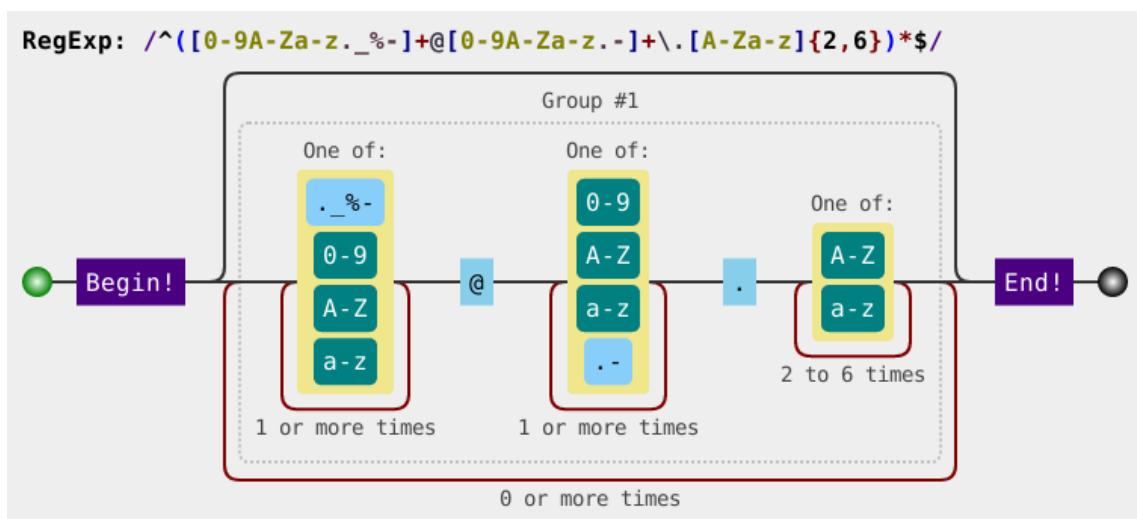


FIGURA 2.3: REGEXP: VALIDADOR DE MAIL.

---

<sup>11</sup>«Los Request for Comments», más conocidos por sus siglas RFC, son una serie de publicaciones del grupo de trabajo de ingeniería de internet que describen diversos aspectos del funcionamiento de Internet y otras redes de computadoras, como protocolos, procedimientos, etc. y comentarios e ideas sobre estos.

```
ttyS9@vorlonhost:~$ echo pepe@hola.com | egrep '^([a-zA-Z0-9._%-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6})*$'
pepe@hola.com

ttyS9@vorlonhost:~$ echo pepe@hola | egrep '^([a-zA-Z0-9._%-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6})*$'
ttyS9@vorlonhost:~$ █
```

#### VALIDADOR DE HORA EN FORMATO 24HS

Un validador de hora en formato de 24 Hs se puede analizar en la figura 2.4

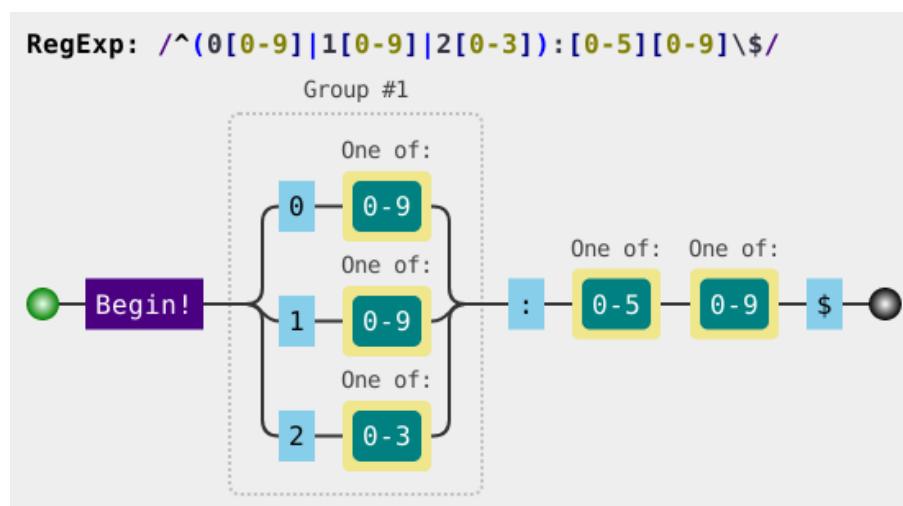


FIGURA 2.4: REGEXP: VALIDADOR DE HORA EN FORMATO 24HS

```
ttyS9@vorlonhost:~$ echo '12:34' |egrep '^(0[0-9]|1[0-9]|2[0-3]):[0-5][0-9]$'
12:43

ttyS9@vorlonhost:~$ echo '12:66' |egrep '^(0[0-9]|1[0-9]|2[0-3]):[0-5][0-9]$'
ttyS9@vorlonhost:~$ █
```

#### CASO DE USO REAL

Para un shell script que el autor desarolló para la creación de exámenes<sup>12</sup>, en una parte, se utilizaron expresiones regulares (figura 2.5 de la siguiente página) para buscar cierta instrucción

<sup>12</sup><https://github.com/pabloniklas/aegs>

del lenguaje L<sup>A</sup>T<sub>E</sub>X:

2

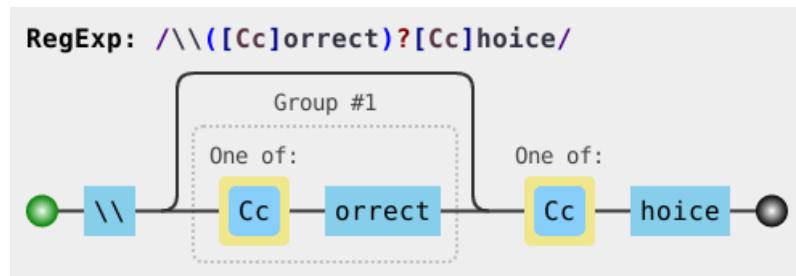


FIGURA 2.5: REGEXP: BÚSQUEDA DE EXPRESIÓN.

Se propone como tarea para el lector, ¿que instrucción (con sus variantes), dejaría pasar esta regexp?

```
$ echo ----- |grep -vE '\\([Cc]orrect)?[Cc]hoice'
```

■

# 3

## Gestión del software y Redirecciones

### 3.1 INTRODUCCIÓN

#### Gestores de paquetes



Los gestores de paquetes se utilizan para automatizar el proceso de instalación, actualización, configuración y eliminación de programas

Hoy en día hay muchos gestores de paquetes para sistemas basados en \*nix/Linux. Los gestores de paquetes también se utilizan para instalar y gestionar módulos para lenguajes como phar (PHP), pip (Python), npm (JS Node), entre otros.

El flujo de trabajo general comienza cuando el usuario solicita un paquete utilizando el administrador de paquetes disponible en el sistema «debianlike»<sup>1</sup> (`CMD apt`). El gestor luego encuentra el paquete solicitado desde una ubicación conocida y lo descarga. Luego, el gestor instala el paquete y aconseja sobre los pasos manuales que considere necesarios.

\*nix comenzó siendo un sistema operativo de y para programadores. Esto significa que cada vez que se escribía un nuevo programa tenía que compilarse<sup>2</sup>, linkeditarse<sup>3</sup> y ejecutarse.

\*nix tiene la capacidad de usar bibliotecas («objetos compartidos»). Para facilitar la tarea de construir software más complicado, se desarrolló la herramienta `CMD make`. El código fuente se enviaba con un *Makefile* (el archivo que usa `CMD make`). Igual era una tarea laboriosa ya que el desarrollador tenía que ocuparse de las dependencias (es decir, las bibliotecas compartidas).

Las distribuciones de Linux actuales contienen miles de paquetes. Esto se debe a su diseño modular, reutilización de código y creación de código colaborativo. Sin embargo, existe una compensación entre la reutilización del código y las dependencias incompatibles. Los administradores

<sup>1</sup>El término «...like» es para indicar el «tipo» de distribución de GNU/Linux, en este caso, cualquier distribución derivada de Debian

<sup>2</sup>En informática, un compilador es un tipo de traductor que transforma un programa entero de un lenguaje de programación (llamado código fuente) a otro. Usualmente el lenguaje objetivo es código máquina, aunque, como Java, también puede ser traducido a un código intermedio («bytecode») o a texto. A diferencia de los intérpretes, los compiladores reúnen diversos elementos o fragmentos en una misma unidad (un programa ejecutable o una biblioteca), que puede ser almacenada y reutilizada. Este proceso de traducción se conoce como compilación

<sup>3</sup>Un enlazador (en inglés, «linker») es un programa que toma los objetos generados en los primeros pasos del proceso de compilación, la información de todos los recursos necesarios (biblioteca), quita aquellos recursos que no necesita, y enlaza el código objeto con su(s) biblioteca(s) con lo que finalmente produce un fichero ejecutable o una biblioteca. En el caso de los programas enlazados dinámicamente, el enlace entre el programa ejecutable y las bibliotecas se realiza en tiempo de carga o ejecución del programa.

# 3

de paquetes resuelven esta complejidad agilizando el proceso.

Durante la instalación hay un momento donde el sistema pide que se seleccione una ubicación http o ftp en internet. Generalmente (no siempre), es conveniente elegir servidores espejos (mirrors) locales.

## Paquete .deb



Un paquete es simplemente un archivo que contiene archivos ejecutables (sean binarios o no), archivos de configuración e información sobre los requerimientos de dependencias.

Esto sirve para configurar una herramienta llamada APT («Advanced Packaging Tool»), es un sistema de gestión de paquetes creado por el proyecto Debian. APT ayuda en gran medida en la instalación y eliminación de programas en los sistemas tipo Debian GNU/Linux. El subsistema en cuestión se encarga de bajar el paquete, conjuntamente con sus dependencias y además realiza todas las tareas previas y posteriores, a la copia de los archivos del paquete.[\[apt:articulo\]](#)

### 3.1.1 FUNCIONES BÁSICAS DEL GESTOR

- ▶ Trabajar con archivadores de archivos para extraer archivos de paquetes.
- ▶ Asegurar la integridad y autenticidad del paquete verificando sus certificados digitales y sumas de verificación.
- ▶ Buscar, descargar, instalar o actualizar software existente desde un repositorio de software o tienda de aplicaciones.
- ▶ Agrupar paquetes por función para reducir la confusión del usuario.
- ▶ Administrar dependencias para garantizar que un paquete esté instalado con todos los paquetes que requiere, evitando así el “infierno de dependencias”.

La interfaz de usuario de un gestor puede ser una línea de comando, una interfaz gráfica o ambas. A menudo, los usuarios pueden buscar paquetes por nombre o categoría. Algunos incluso muestran reseñas de usuarios o calificaciones de paquetes. La instalación por lotes también es posible con este gestor. Algunos pueden admitir la «actualización segura» (conservar las versiones existentes) o «retener» (bloquear el paquete a una versión específica).

### 3.1.2 DESCARGAR EL SOFTWARE

Los paquetes se descargan de los repositorios de software, a menudo simplemente llamados «repositorios». Los términos alternativos incluyen «sources» y/o «feeds». Estos repositorios están disponibles en línea, en ubicaciones bien definidas y sirven como un punto de distribución central para los paquetes.

## Repositorios espejados



Para un mejor rendimiento y mayor redundancia, estos repositorios pueden ser espejados por muchas otras ubicaciones en todo el mundo. Los repositorios locales también pueden espejar repositorios oficiales remotos, para ahorrar ancho de banda y obtener mayor privacidad.

Si bien la mayoría de los desarrolladores usarán estos repositorios para descargar paquetes, los desarrolladores avanzados también pueden contribuir o cargar paquetes para ser alojados en estos repositorios. Todos los «repos» informan el proceso que los desarrolladores deben seguir para cargar paquetes. Los repositorios oficiales tienen un estricto proceso de revisión y aprobación. Los repositorios gestionados por la comunidad pueden tener un proceso más relajado. En todos los casos, los repositorios están destinados a ser libres de malware.

### 3.1.3 DE DONDE DESCARGAR

Cada administrador de paquetes tiene archivos de configuración asociados que apuntan a ubicaciones de repositorio. Por ejemplo, en Debian, `/etc/apt/sources.list` contiene las ubicaciones de los repositorios. Esto incluye los repositorios oficiales, pero los usuarios también pueden actualizar este archivo para obtener paquetes de otros repositorios. Del mismo modo, la configuración para las distribuciones de Fedora y CentOS se encuentra en `/etc/yum.conf` para `CMD yum` y `/etc/dnf/dnf.conf` para `CMD dnf`. En Arch Linux, está en `/etc/pacman.conf` cuando se usa `CMD pacman`.

## Cuidado al agregar «repos»



Al agregar repositorios de terceros a un administrador de paquetes, los usuarios deben asegurarse de verificar que esos repositorios sean confiables y no conflictuen con las definiciones de paquetes de otros repositorios (sobre todo los oficiales).

Los repositorios deben ser escogidos cuidadosamente, para que no se termine con un malware que infecte el sistema. De hecho, este es uno de los problemas resueltos por repositorios de confianza. En lugar de descargar software de un sitio web de un tercero, descargarlo a través del administrador de paquetes desde un repositorio de confianza, es una práctica segura. Adicionalmente, los paquetes que se descargan/installan están firmados digitalmente, con la firma pública registrada en el sistema operativo.

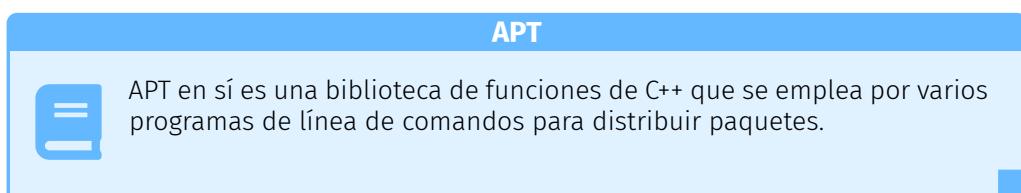
### 3.1.4 ¿QUE CONTIENE UN PAQUETE?

Un paquete incluye el software en cuestión, que puede ser una aplicación o una biblioteca compartida. Si se trata de un paquete de desarrollo, incluirá archivos de encabezado o «header» para construir el software que depende de una biblioteca. Los paquetes están destinados a distribuciones específicas y, por lo tanto, las rutas de instalación, la integración de escritorio y los

scripts de inicio se configuran para la distribución de destino. Los formatos de paquete pueden incluir \*.tgz (para archivos de código fuente), \*.deb (para Debian) o \*.rpm (para Red Hat).

## 3

Los paquetes también incluyen metadatos. Esto incluirá resumen, descripción, lista de archivos, versión, autoría, arquitectura específica, sumas de verificación de archivos, licencias y paquetes dependientes. Estos metadatos son esenciales para que el gestor de paquetes haga su trabajo correctamente.



### 3.1.5 apt-get install

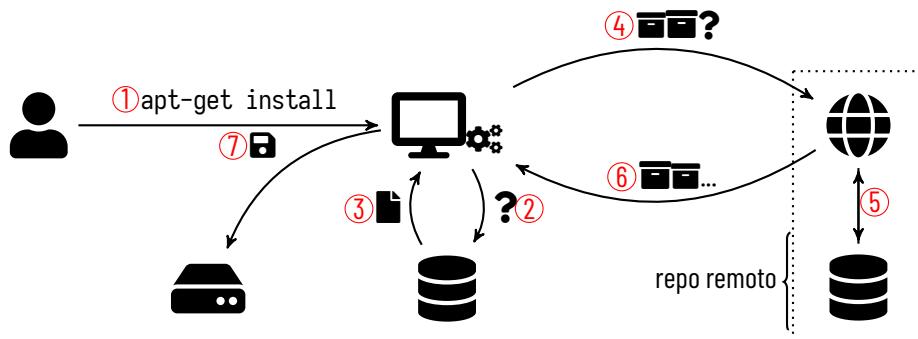


FIGURA 3.1: PROCESO DE INSTALACIÓN DE SOFTWARE CON APT.

Esta acción debe realizarse con privilegios de root. En la figura 3.1, se muestra el proceso de instalación de software. El cual contempla las siguientes etapas:

- 1 El usuario escribe el comando `sudo apt-get install <paquete a instalar>`.
- 2 El gestor de paquetes consulta en el índice local la disponibilidad del paquete en cuestión y sus dependencias.
- 3 Se obtiene el listado de archivos de paquetes (\*.deb), que serán descargados.
- 4 El gestor de paquetes solicita paquetes respectivos a los servidores definidos en el archivo /etc/apt/sources.list.
- 5 El servidor busca los paquetes y los presenta para su descarga,
- 6 Se descargan los paquetes.
- 7 Se instalan los paquetes, lo cual implica:
  - ▶ Ejecución de acciones previas a la copia de archivos.
  - ▶ Copia de archivos.

- ▶ Ejecución de acciones posteriores a la copia de archivos.

Por ejemplo, si se quiere instalar el paquete «ccze», que es un colorizador de logs.

```
ttyS9@vorlonhost:~$ sudo apt-get install ccze
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
ccze
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 75,1 kB de archivos.
Se utilizarán 308 kB de espacio de disco adicional después de esta operación.
0% [Esperando las cabeceras]
Des:1 http://archive.ubuntu.com/ubuntu bionic/universe amd64 ccze amd64 0.2.1-4
  ↳[75,1 kB]
Descargados 75,1 kB en 1s (57,7 kB/s)
Seleccionando el paquete ccze previamente no seleccionado.
(Leyendo la base de datos ... 607288 ficheros o directorios instalados
  ↳actualmente.)
Preparando para desempaquetar .../ccze_0.2.1-4_amd64.deb ...
Desempaquetando ccze (0.2.1-4) ...
Configurando ccze (0.2.1-4) ...
Procesando disparadores para man-db (2.8.3-2ubuntu0.1) ...
ttyS9@vorlonhost:~$
```

### 3.1.6 apt-cache search

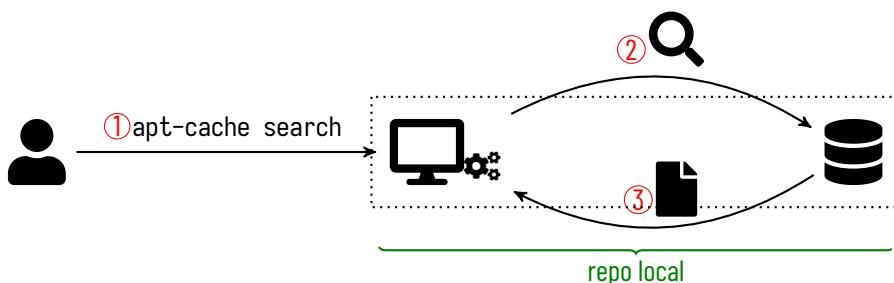


FIGURA 3.2: ESQUEMA DE FUNCIONAMIENTO DE apt-cache search.

Con el argumento [search <palabra clave> ] se busca (Figura 3.2) la palabra clave en el índice local de aplicaciones. Esta información es una especie de caché, ya que se obtiene de las diferentes fuentes definidas en el archivo sources.list.

**SIN** apt-cache search <paquete>

Ejemplo:

```
ttyS9@vorlonhost:~$ apt-cache search mame
backup21 - herramienta de bajo mantenimiento respaldar/restaurar
libboost-python-dev - archivos de desarrollo de la biblioteca Boost.Python
↳(versión predeterminada)
libboost-python1.62-dev - Boost.Python Library development files
libboost-python1.62.0 - Boost.Python Library
libboost-python1.65-dev - Boost.Python Library development files
gnome-video-arcade - Simple MAME frontend
mame - Multiple Arcade Machine Emulator (MAME)
mame-data - Multiple Arcade Machine Emulator (MAME) -- data files
mame-doc - Documentation for MAME
mame-tools - Tools for MAME
mame-extra - Additional files for the Multiple Arcade Machine Emulator (MAME)
ttyS9@vorlonhost:~$ █
```

### OTRAS BÚSQUEDAS

Se pueden buscar información de otras formas, en función de que si un archivo se encuentra instalado o no:

- **Si esta instalado** → Para buscar algun archivo específico y saber en que paquete se encuentra, debian provee un sitio: [https://www.debian.org/distrib/packages#search\\_contents](https://www.debian.org/distrib/packages#search_contents). Muy útil para buscar bibliotecas (mal llamadas librerías).
- **Si no esta instalado** → Como apt es una cáscara para **CMD dpkg**, se puede utilizar el mismo para, por ejemplo, buscar la pertenencia de un archivo a un paquete. Por ejemplo, si se quiere saber a que paquete pertenece un archivo, se debe tipar: **CMD dpkg -S <archivo>**. El parámetro «archivo» debe ser con el path completo.

#### 3.1.7 apt-get update

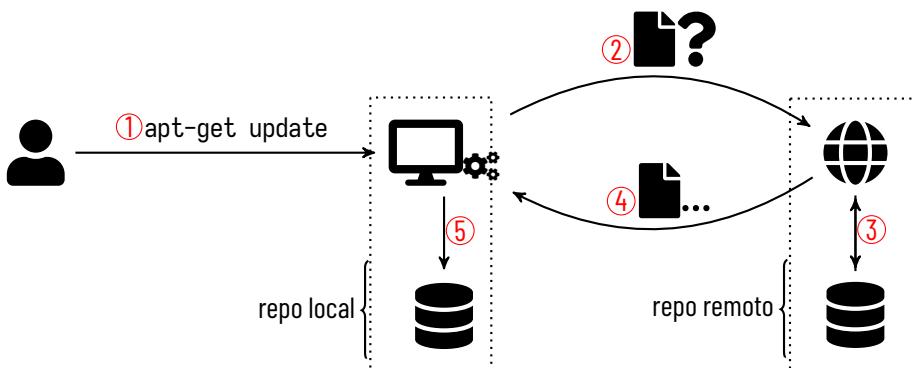


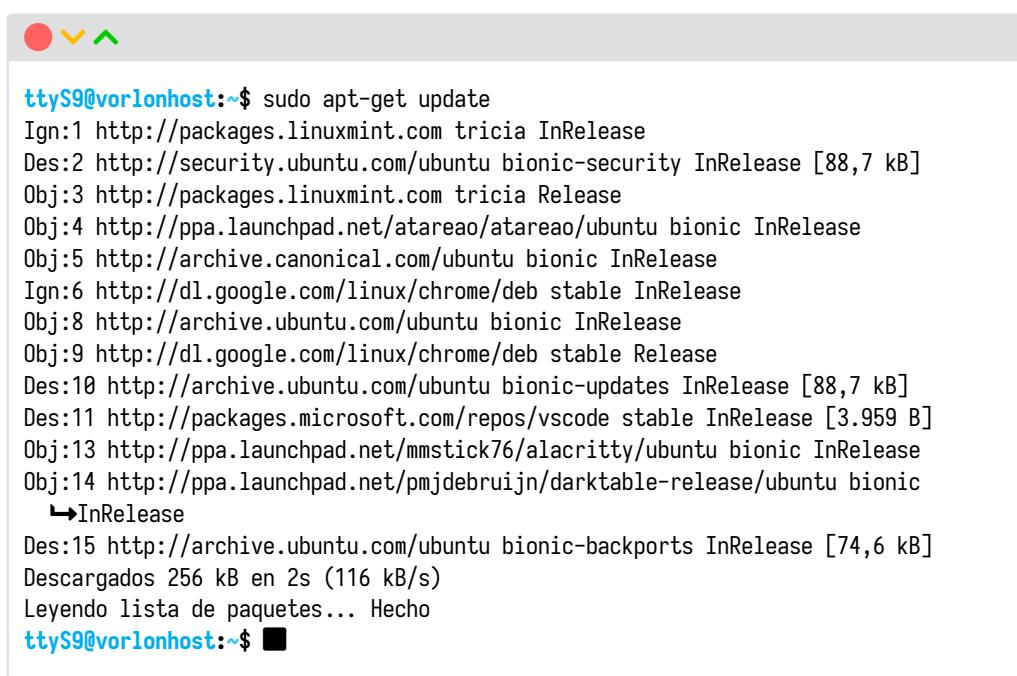
FIGURA 3.3: ESQUEMA DE FUNCIONAMIENTO DE `apt-get update`.

Debe realizarse con privilegios de root. Cuando se ejecuta el comando **CMD apt update**, se actualizan los índices locales con los que se encuentran en los repositorios indicados en el archivo

/etc/apt/sources.list.

Los pasos que se encuentran gráficos en Figura 3.3 en la anterior página, son los que a continuación se detallan:

- 1 El usuario ejecuta con privilegios de root, el comando **CMD apt-get update**.
- 2 El gestor de paquetes se conecta a los servidores definidos en /etc/apt/sources.list.
- 3 En el equipo remoto, se obtienen los indices del/los repositorio/s.
- 4 Los indices generados son recuperados.
- 5 El gestor de paquetes actualiza los índices locales de paquetes, con los obtenidos de internet.



```
ttyS9@vorlonhost:~$ sudo apt-get update
Ign:1 http://packages.linuxmint.com tricia InRelease
Des:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [88,7 kB]
Obj:3 http://packages.linuxmint.com tricia Release
Obj:4 http://ppa.launchpad.net/atareao/atareao/ubuntu bionic InRelease
Obj:5 http://archive.canonical.com/ubuntu bionic InRelease
Ign:6 http://dl.google.com/linux/chrome/deb stable InRelease
Obj:8 http://archive.ubuntu.com/ubuntu bionic InRelease
Obj:9 http://dl.google.com/linux/chrome/deb stable Release
Des:10 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88,7 kB]
Des:11 http://packages.microsoft.com/repos/vscode stable InRelease [3.959 B]
Obj:13 http://ppa.launchpad.net/mmstick76/alacritty/ubuntu bionic InRelease
Obj:14 http://ppa.launchpad.net/pmjdebruijn/darktable-release/ubuntu bionic
     InRelease
Des:15 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74,6 kB]
Descargados 256 kB en 2s (116 kB/s)
Leyendo lista de paquetes... Hecho
ttyS9@vorlonhost:~$
```

### 3.1.8 FORMATO DEL ARCHIVO /etc/apt/sources.list

Cada línea del archivo sigue el siguiente formato:

**SIN** deb [ opción1=valor1 opción2=valor2 ] uri distro [componente1] [componente2] [...]

- **deb** → La primer palabra en cada línea, «deb» o «deb-src», indica el tipo de archivo; deb indica que el archivo contiene paquetes binarios (deb), los paquetes precompilados que normalmente se usan, deb-src indica los paquetes fuente, que son las fuentes originales del programa más el «archivo de control de debian» (.dsc) y el archivo «diff.gz» que contiene los cambios necesarios para empaquetar el programa.
- **uri** → URL del repositorio desde el que desea descargar los paquetes. La lista principal de los espejos del repositorio de Debian se encuentra en <https://www.debian.org/mirror/list>.

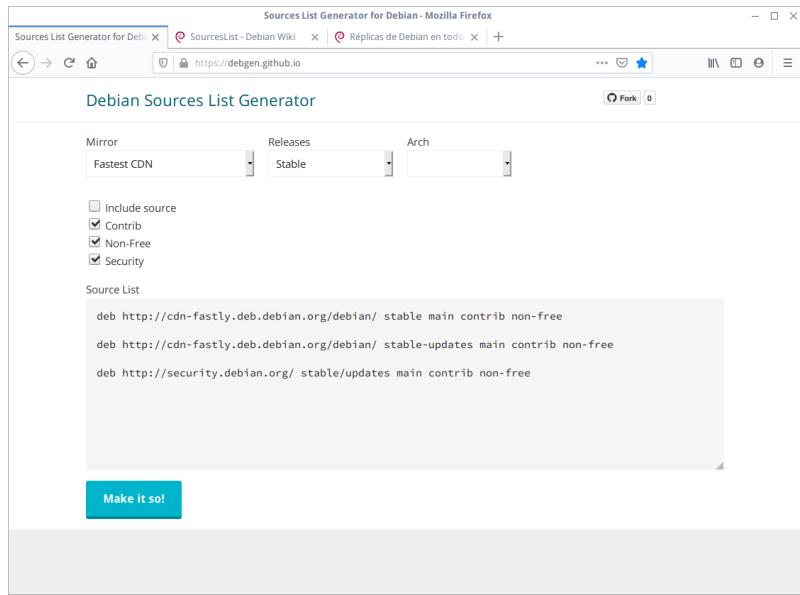


FIGURA 3.4: HTTPS://DEBGEN.GITHUB.IO/

- ▶ **distro** → Puede ser el nombre/alias en código del lanzamiento (jessie, stretch, buster, sid) o la clase de lanzamiento (oldstable, stable, testing, instable) respectivamente. Si se quiere rastrear una «clase» de lanzamiento, se debe usar el nombre de la clase, si desea rastrear una distro, use el nombre del código.
- ▶ **componente** → Aquí pueden estar en forma excluyente las palabras «main», «contrib» o «non-free».
  - ▶ **main** → Consiste en paquetes compatibles con la DFSG<sup>4</sup>, que no dependen de software fuera de esta área para operar. Estos son los únicos paquetes considerados parte de la distribución de Debian.
  - ▶ **contrib** → Contiene software compatible con DFSG, pero tienen dependencias que no están en «main» (posiblemente empaquetados para Debian en versiones no gratuitas).
  - ▶ **non-free** → Contiene software que no cumple con el DFSG.

#### No utilizar «stable» como distro



Se debe evitar el uso de «stable» en `sources.list`, ya que da lugar a sorpresas desagradables y a roturas de sistemas cuando se actualiza a la próxima versión.

Este es un ejemplo de archivo `/etc/apt/sources.list`

```
/etc/apt/sources.list

1 deb http://ftp.ccc.uba.ar/pub/linux/debian/debian/ jessie main
2 deb http://security.debian.org/ jessie/updates main
3 deb http://ftp.ccc.uba.ar/pub/linux/debian/debian/ jessie-updates main
```

Este archivo se genera durante la instalación si es que se indica que busque otras fuentes de instalación.

Sin embargo, es posible modificarlo de ser necesario, por ejemplo si los servidores a los cuales se conectan, tienen problemas de conectividad o simplemente dejaron de existir.

Hay sitios en internet, que de manera online generan un archivo sources.list. Uno de es <https://debsources.github.io/>, figura 3.4 en la anterior página.

## 3.2 REDIRECCIONAMIENTO Y PIPES

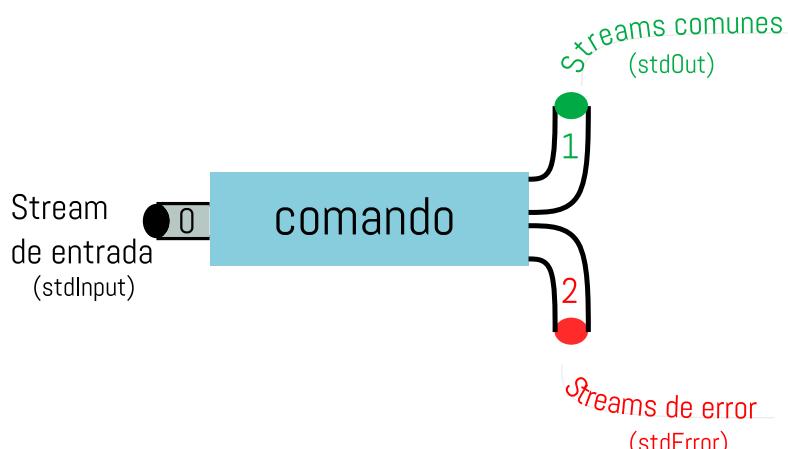


FIGURA 3.5: ESTRUCTURA DE LOS STREAMS DE UN COMANDO.

Cuando un comando se ejecuta, se le proveen de tres «canales» para su comunicación (figura 3.5):

- ▶ **stdin o «Standard Input»** → Por este canal, identificado por el número «0», el comando recibe un flujo «stream» de caracteres para ser procesado por éste.
- ▶ **stdout o «Standard Output»** → Por este canal, identificado por el número «1», el comando emite mensajes o streams propios del procesamiento.
- ▶ **stderr o «Standard Error»** → Por este canal, identificado por el número «2», el comando emite mensajes o streams correspondientes a una situación de error.

El intérprete de comandos o «shell», otorga a cada stream un dispositivo por defecto. A estos dispositivos ya no se los va a denominar canales, sino «descriptores de archivo».

Estos dispositivos son normalmente: la pantalla (para stdOut y stdErr) y el teclado (stdIn), sin embargo, éstos pueden ser sustituidos por otros. A este aspecto de cambiar la stdIn y stdOut por otros dispositivos, se lo conoce como «redirección de entradas y de salidas».

A los operadores `>` y `>>` se les debe anteponer el número correspondiente al descriptor de archivo que va a ser redirigido. Es decir que:

## 3

- ▶ Para `stdOut` sería `SIN comando 1> archivo` o `SIN comando 1>> archivo`.
- ▶ Para `stdErr` sería `SIN comando 2> archivo` o `SIN comando 2>> archivo`.



Si bien hay que anteponer el número de descriptor que se desea redireccionar, **para el caso de `stdOut` puede omitirse**, ya que el valor por defecto para el redireccionamiento es el correspondiente para `stdOut`, es decir, 1.

Existen cuatro tipos de redireccionamientos:

- ▶ `>` → Mayor.
- ▶ `>>` → Doble mayor.
- ▶ `<` → Menor.
- ▶ `|` → Pipe.

### 3.2.1 >

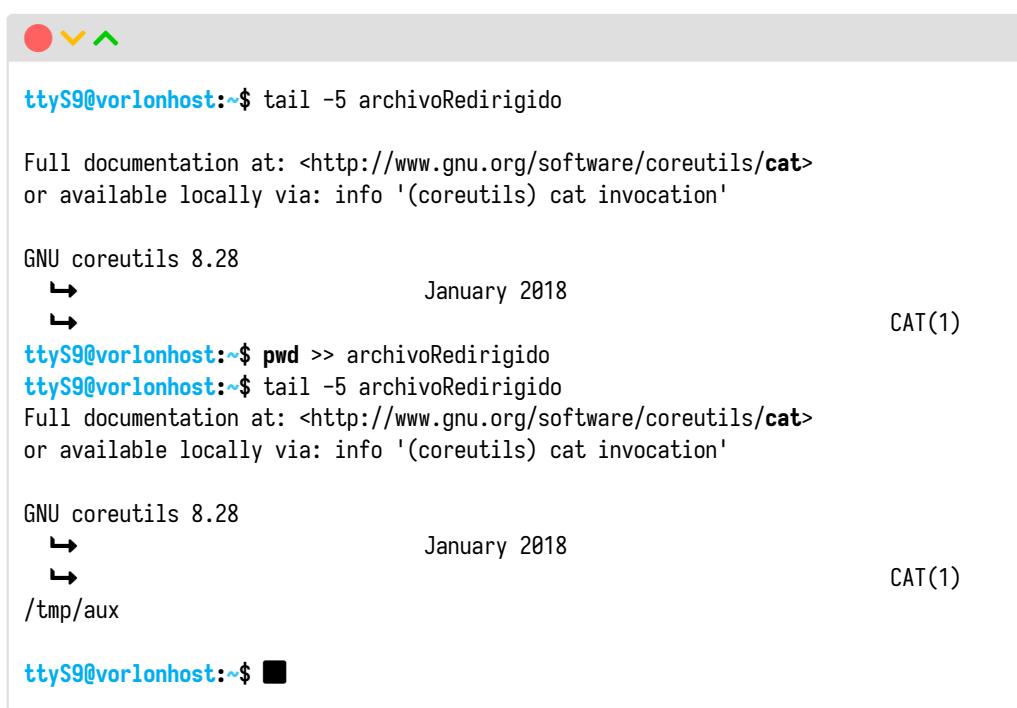
El primero, (`>`) permite que en vez de que el resultado de un comando se muestre en pantalla, se pueda almacenar en un archivo u otro dispositivo. Ejemplo:

```
ttyS9@vorlonhost:~$ ls -l archivoRedirigido
ls: no se puede acceder a 'archivoRedirigido': No existe el archivo o el
  directorio
ttyp9@vorlonhost:~$ man cat > archivoRedirigido
ttyp9@vorlonhost:~$ ls -l archivoRedirigido
-rw-r--r-- 1 ttys9 ttys9 2247 ene 29 15:14 archivoRedirigido
ttyp9@vorlonhost:~$ head -5 archivoRedirigido
CAT(1)
↑                         User Commands
↑
ttyp9@vorlonhost:~$ █
```

Como se puede apreciar, en vez de escribir la información acerca del `CMD cat` en pantalla, el comando crea un archivo totalmente nuevo en el directorio actual.

### 3.2.2 >>

El segundo > permite añadir el contenido de la salida a un archivo ya existente (o crearlo en caso de no existir). Ejemplo:



```

ttyS9@vorlonhost:~$ tail -5 archivoRedirigido
Full documentation at: <http://www.gnu.org/software/coreutils/cat>
or available locally via: info '(coreutils) cat invocation'

GNU coreutils 8.28
 ↵          January 2018
 ↵
ttyS9@vorlonhost:~$ pwd >> archivoRedirigido
ttyS9@vorlonhost:~$ tail -5 archivoRedirigido
Full documentation at: <http://www.gnu.org/software/coreutils/cat>
or available locally via: info '(coreutils) cat invocation'

GNU coreutils 8.28
 ↵          January 2018
 ↵
/tmp/aux

ttyS9@vorlonhost:~$ █

```

Como se puede observar, el comando añadirá la ruta del directorio actual de trabajo al final del archivo, en vez de escribir la información del directorio actual en pantalla.

#### Diferencias entre > y >>

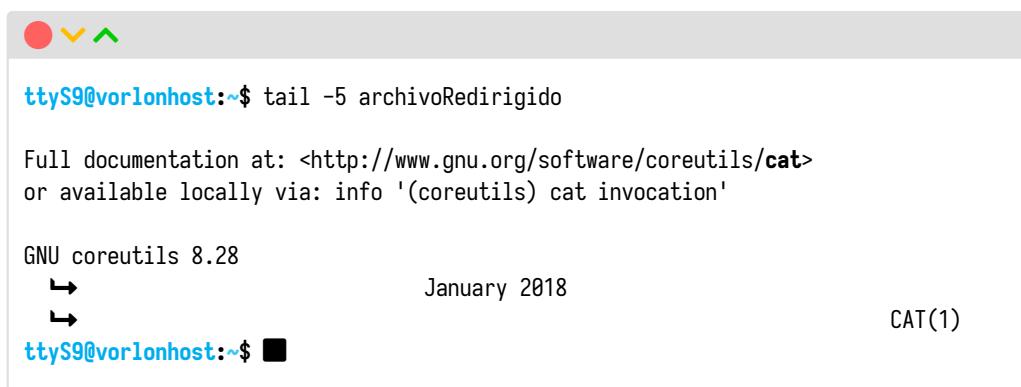


Si bien ambos símbolos (> y >>), crean el archivo destino si no existe, es fundamental la diferencia entre > y >>: si el archivo ya existía previamente. > **destruye** el contenido existente, mientras que >>, agrega contenido al existente, es decir, **no destruye**.

### 3.2.3 <

El redireccionamiento de stdIn le permite al comando leer la información a procesar desde un fichero u otro dispositivo en lugar de leer los datos directamente desde el teclado. Para redireccionar la stdIn, se utiliza el operador < seguido del nombre de fichero o dispositivo de donde tomará la información. Ejemplo:

## 3



```
ttyS9@vorlonhost:~$ tail -5 archivoRedirigido
Full documentation at: <http://www.gnu.org/software/coreutils/cat>
or available locally via: info '(coreutils) cat invocation'

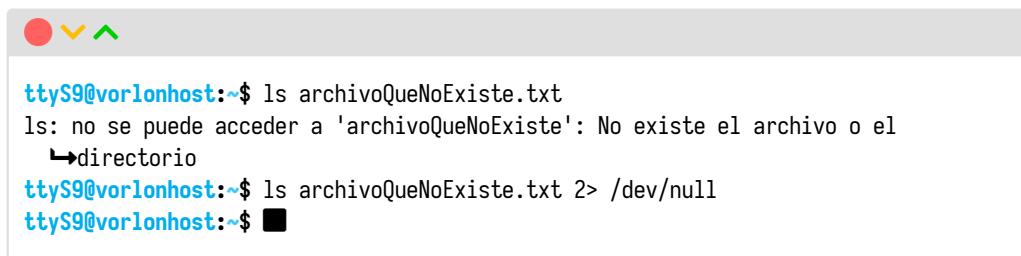
GNU coreutils 8.28
                         January 2018
                         CAT(1)
ttxS9@vorlonhost:~$
```



```
ttyS9@vorlonhost:~$ grep GNU < archivoRedirigido
GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Copyright © 2017 Free Software Foundation, Inc. License GPLv3+: GNU GPL version
 ↲3 or later <http://gnu.org/licenses/gpl.html>.
GNU coreutils 8.28
ttxS9@vorlonhost:~$
```

### 3.2.4 REDIRECCIONAMIENTO A stdErr

El redireccionamiento de la stdErr, se hace indicando el descriptor que le corresponde (#2), previo al >. Por ejemplo:



```
ttyS9@vorlonhost:~$ ls archivoQueNoExiste.txt
ls: no se puede acceder a 'archivoQueNoExiste': No existe el archivo o el
 ↲directorio
ttxS9@vorlonhost:~$ ls archivoQueNoExiste.txt 2> /dev/null
ttxS9@vorlonhost:~$
```

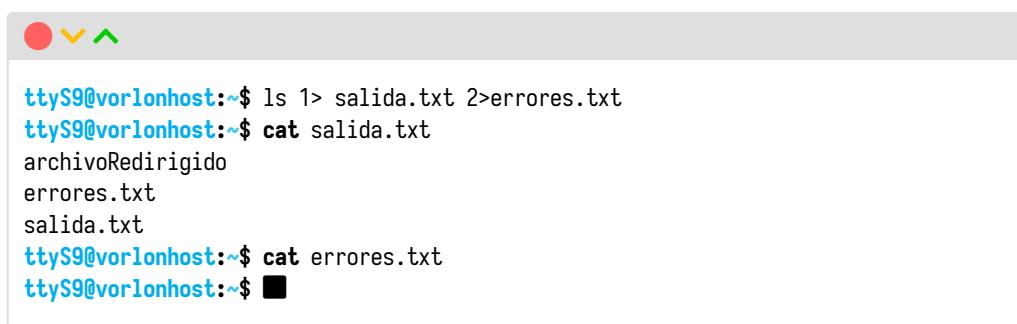
En este ejemplo, se redireccionaron los errores al dispositivo `/dev/null`, que cumple las funciones de ser un «agujero negro».

### 3.2.5 REDIRECCIÓN MÚLTIPLE

Es posible redirigir multiples streams en la misma invocación, es decir:

SIN <comando> >> salidaStdOut 2> salidaStdErr

Por ejemplo:



```
ttyS9@vorlonhost:~$ ls 1> salida.txt 2> errores.txt
ttyS9@vorlonhost:~$ cat salida.txt
archivoRedirigido
errores.txt
salida.txt
ttyS9@vorlonhost:~$ cat errores.txt
ttyS9@vorlonhost:~$ █
```

### 3.2.6 PIPES «|»

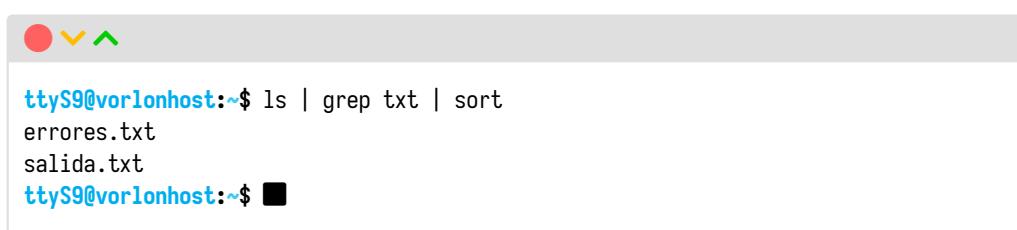
#### pipeline



Una pipeline es una secuencia de procesos encadenados juntos por sus streams standards.

De esa manera, la salida de cada comando o proceso (`std0ut`), alimenta directamente la entrada (`stdIn`) del siguiente. Figura 3.6 de la siguiente página.

Es una de las principales características de este tipo de Sistemas Operativos, por ejemplo:



```
ttyS9@vorlonhost:~$ ls | grep txt | sort
errores.txt
salida.txt
ttyS9@vorlonhost:~$ █
```

Es casi posible realizar programas en solo una línea de comandos. En la jerga de internet, a eso se lo llama «bashoneliners».<sup>5</sup>

<sup>5</sup><http://www.bashoneliners.com/>

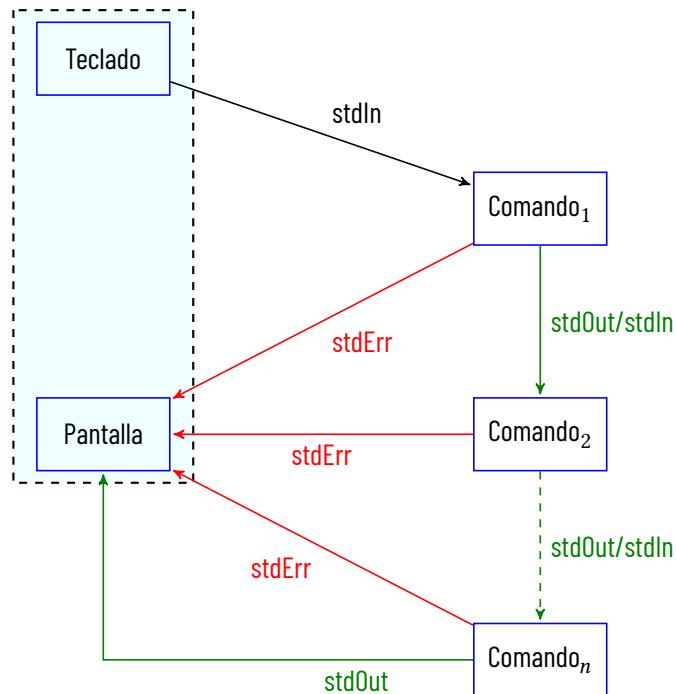


FIGURA 3.6: UNA PIPELINE DE PROGRAMAS EJECUTÁNDOSE DESDE UNA TERMINAL.



Incorporar a la edición, el texto correspondiente a la salida de ejecución de un comando (vía `stdOut`)

`CMD vi <(uptime)`

### 3.3 BÚSQUEDA Y FILTROS

#### 3.3.1 FILTROS

##### Filtros



Los filtros son comandos que, procesando un stream, retornan cadenas de texto que son relativas al stream procesado. Por ejemplo, `CMD grep`, etc.

Se listan en el cuadro 3.1 de la siguiente página algunos comandos que funcionan como filtros, muchos de éstos de uso frecuente:

COMANDO	DESCRIPCIÓN
CMD cat	Concatena archivos e imprime en la std0ut.
CMD sort	Ordena las líneas de un archivo de texto.
CMD head	Devuelve las primeras líneas de un archivo.
CMD tail	Devuelve las últimas líneas de un archivo.
CMD wc	Imprime el número de bytes, líneas y palabras de una archivo.
CMD more	Pagina el contenido de una archivo en pantalla.
CMD strings	Muestra los caracteres «imprimibles» de un archivo.
CMD sed	Editor de «streaming».
CMD split	Divide un archivo en porciones acorde a un criterio.
CMD tr	Borra o traduce caracteres.
CMD nl	Numera las líneas de un archivo.
CMD grep	Imprime aquellas líneas que respetan un parámetro.

CUADRO 3.1: FILTROS

### 3.3.2 grep

«grep» permite buscar, dentro de archivos, las líneas que concuerdan con un patrón, y la muestra.

Opciones más usadas:

OPCIÓN	DESCRIPCIÓN
-c	En lugar de imprimir las líneas que coinciden, muestra el número de líneas coincidentes.
-e PATRON	Permite especificar varios patrones de búsqueda o proteger aquellos patrones de búsqueda que comienzan con el signo «-».
-r	Busca recursivamente dentro de todos los subdirectorios del directorio actual.
-v	Muestra las líneas que no coinciden con el patrón buscado.
-i	Ignora la distinción entre mayúsculas y minúsculas.
-n	Numera las líneas en la salida.
-E	Permite usar expresiones regulares <sup>6</sup> . Es equivalente a usar CMD egrep .
-o	Indica a CMD grep que muestre solo la parte de la línea que coincide con el patrón.
-f ARCHIVO	Extrae los patrones del archivo que se especifique. Los patrones del archivo deben ir uno por línea.
-H	Imprime el nombre del archivo con cada coincidencia.
--color	En color la coincidencia.
-l	Lista los nombre de los archivos donde haya coincidencia.
-w	Busca la palabra exacta.

CUADRO 3.2: OPCIONES USUALES DE grep.

<sup>6</sup>Más sobre este tema en la sección 2.6 en página 61.

## 3

## 3.3.3 find

**find**



El comando **CMD find**, permite buscar (y listar) archivos acorde a un criterio determinado.

Por defecto, **CMD find**, sólo lista, pero también, por medio de parámetros, puede lograrse que ejecute una acción específica teniendo como parámetro al mismo archivo encontrado.



**CMD find** es muy útil para iterar una acción determinada en una lista de archivo que se genera por el criterio de **find**.

EJEMPLOS DEL USO DE **find**

Busca y lista archivos de tipo directorio (-type d) de nombre script, y comienza su búsqueda sobre el directorio raíz (/), suprimiendo los errores de acceso (2>/dev/null).

```
CMD find / -type d -name "script" 2>/dev/null
```

Busca e imprime (-print) archivos de tipo regular (-type f), con una fecha de modificación menor a 7 días (-mtime -7).

```
CMD find Descargas/ -type f -mtime -7 -print
```

Si se desea listar los que se modificaron los últimos 30 minutos.

```
CMD find Descargas/ -type f -mmin -30 -print -ls
```

Si se le agrega el argumento -exec ejecuta una acción por cada archivo que cumpla el criterio de búsqueda que le indicamos.

Si se quiere obtener el «hash MD5» de cada archivo en /usr/bin, el criterio es que se busquen archivos regulares (-type f), a partir de /usr/bin, luego que ejecute (-exec) el **CMD md5sum**. Este último requiere a continuación el nombre de archivo cuyo MD5 se desea calcular. El par de llaves, «{}» es reemplazado por cada nombre de archivo en cada iteración y por último, se debería finalizar con «;», pero como también es un carácter reservado para bash, se utiliza el carácter \ para que bash no lo interprete.

```
ttyS9@vorlonhost:~$ find /usr/bin -type f -exec md5sum {} \;
b7b0b3cc16ac2c0d93a30849155cba93  /usr/bin/cinnamon-slideshow
7ba9ca0c29a037ad08eec0c8eba0c2d2  /usr/bin/gsettings-schema-convert
f4ab4c079ffd1d606872b66774a47744  /usr/bin/lslocks
6758a958ee98cc115987eab15245d099  /usr/bin/lrfviewer
ed77f834fd2c8c12a86189015990a20d  /usr/bin/pgrep
5ef14c033d5a3887a4f4b008548a3203  /usr/bin/lslogins
5cd6ca7fe77c0f8a91e3b854d6aa9c7f  /usr/bin/lsusb
cd8465fd3e041be3a9ea5d80eee20dd5  /usr/bin/btmgmt
ttyS9@vorlonhost:~$
```

También es útil para saber que archivos son los mas grandes, y así analizar su eliminación y liberar espacio en disco.

**CMD** `find Descargas/ -type f -size +1G -print`

Si bien `find` es muy flexible con los criterios de búsqueda y posterior ejecución de comandos, tiene una sola desventaja: es lento, pero con la tecnología de los discos de estado sólido, esto dejará de ser un problema.

### 3.3.4 locate y updatedb

Por otro lado, hay una forma mas rápida de buscar un archivo, pero limitado a solo el nombre:

**CMD** `updatedb` y **CMD** `locate`.

La sintaxis de `locate` es **SIN** `locate <archivo a buscar>`

**CMD** `updatedb` recorre todos los filesystems (por default incluye filesystems remotos, pero en la configuración que se incorpora en el paquete de instalación, los excluye). Al finalizar la ejecución se tiene una base de datos con todos los archivos que se encuentran en los filesystems. La base de datos generada, se puede consultar con **CMD** `locate`.

```
ttyS9@vorlonhost:~$ locate php.ini
/etc/php/7.2/cli/php.ini
/usr/lib/php/7.2/php.ini-development
/usr/lib/php/7.2/php.ini-production
/usr/lib/php/7.2/php.ini-production.cli
ttyS9@vorlonhost:~$
```



# 4 Editores

## 4.1 VI

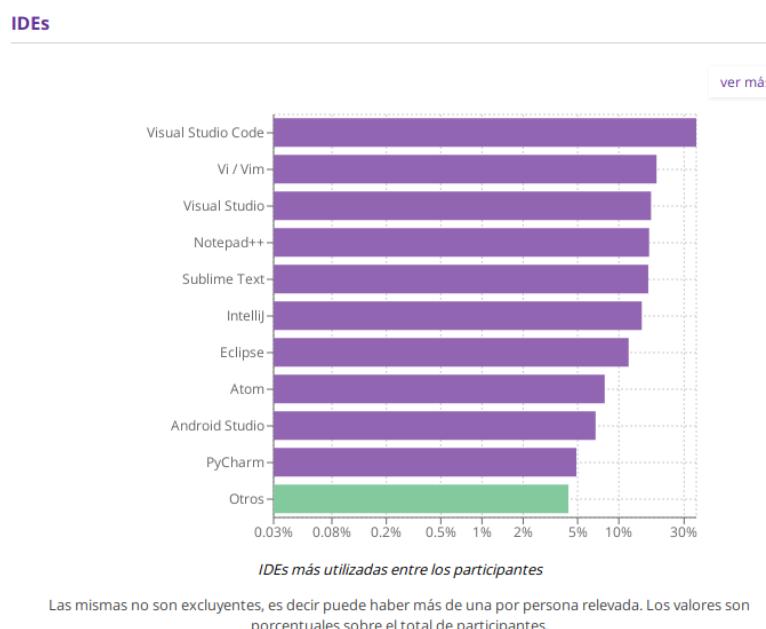


FIGURA 4.1: RANKING DE IDES.[article:encuesta]

- ▶ Es un editor de texto de pantalla completa que maneja en memoria el texto entero de un archivo.
- ▶ Es el editor clásico de \*nix; está en todas las versiones.
- ▶ Puede usarse en cualquier tipo de terminal con un mínimo de teclas; esto lo hace difícil de usar al principio.
- ▶ Como la mayoría de las configuraciones en \*nix se manejan editando archivos, disponer de la capacidad de uso de vi es esencial en la administración de un sistema.

Existen en \*nix otros editores, como **CMD emacs**, que provee un ambiente de trabajo completo; también versiones fáciles de manejar como **CMD jove** o **CMD pico**, o aún mínimas e inmediatas como **CMD ae**. En ambiente X-Windows hay muchos editores amigables, fáciles de usar y con múltiples capacidades («visual studio code» por ejemplo). No obstante, **CMD vi** está en todos los \*nix, y como requiere pocos recursos se usa mucho en administración, para programar y en situaciones de emergencia. En casos de roturas de discos, corrupción de sistemas de archivos, errores en el arranque y otras catástrofes, puede ser el único editor disponible.

**vim**



Existe un editor vi ampliado llamado vim «Vi-IMproved» que contiene facilidades adicionales, así como diversas versiones del vi original. En todos los casos, el conjunto de comandos básicos es el mismo.

#### 4.1.1 MODOS DE VI

Existen tres modos o estados en **CMD vi**, figura 4.2.

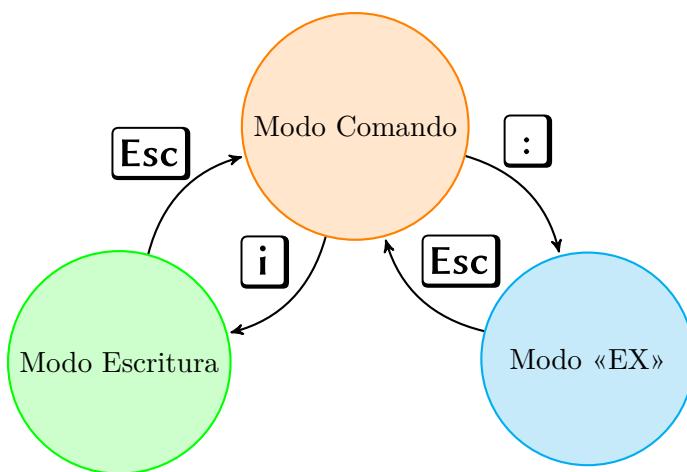


FIGURA 4.2: MODOS DE OPERACIÓN DEL vi/vim.

- ▶ **Modo comando:** las teclas ejecutan acciones que permiten desplazar el cursor, recorrer el archivo, ejecutar comandos de manejo del texto y salir del editor. Es el modo inicial de **CMD vi**.
- ▶ **Modo texto o modo inserción:** las teclas ingresan caracteres en el texto.
- ▶ **Modo última línea o «ex»:** las teclas se usan para escribir comandos en la última línea al final de la pantalla.

#### 4.1.2 GUÍA BÁSICA

Con unos pocos comandos básicos se puede trabajar en **CMD vi** editando y salvando texto:

COMANDO	DESCRIPCIÓN
<b>CMD vi arch1</b>	Arranca en modo comando editando el archivo arch1.
<b>i</b>	Inserta texto a la izquierda del cursor.
<b>a</b>	Agrega texto a la derecha del cursor.
<b>Esc</b>	Vuelve a modo comando.
<b>x</b>	Borra el carácter bajo el cursor.
<b>d</b>	Borra una línea.
<b>h</b> o <b>←</b>	Mueve el cursor un carácter a la izquierda.

Cuadro 4.1 – CONTINUACIÓN...

COMANDO	DESCRIPCIÓN
[j] o [↓]	Mueve el cursor una línea hacia abajo.
[k] o [↑]	Mueve el cursor una línea hacia arriba.
[l] o [→]	Mueve el cursor un carácter a la derecha.
[:] + [w]	Salva el archivo (graba en disco).
[:] + [q]	Sale del editor (debe salvarse primero).

CUADRO 4.1: COMANDOS BÁSICOS DE vi.

#### 4.1.3 INVOCACIÓN DE vi

**CMD vi** se puede invocar de varia formas, las cuales son explicadas en el cuadro 4.2.

COMANDO	DESCRIPCIÓN
CMD vi	Abre la ventana de edición sin abrir ningún archivo.
CMD vi arch1	Edita el archivo arch1 si existe; si no, lo crea.
CMD vi arch1 arch2	Edita sucesivamente los archivos arch1 y luego arch2.
CMD vi +45 arch1	Edita el archivo arch1 posicionando el cursor en la línea 45.
CMD vi +\$ arch1	Edita el archivo arch1 posicionando el cursor al final del archivo.
CMD vi +/Habia arch1	Edita el archivo arch1 en la primera ocurrencia de la palabra «Habia».

CUADRO 4.2: FORMAS DE INVOCAR A vi.

#### vi es «case sensitive»



El editor vi, al igual que todo \*nix, diferencia mayúsculas y minúsculas. Confundir un comando en minúscula digitando uno en mayúscula suele tener consecuencias catastróficas. Se aconseja evitar sistemáticamente el uso del «caps-lock»; siempre mantener el teclado en minúsculas.

#### 4.1.4 CAMBIO DE MODO

En la figura 4.2 en la anterior página se muestra un grafo de los tres estados de **CMD vi** y como se conmutan entre ellos.

- ▶ Comando a Texto: [i] [I] [a] [A] [o] [O]. Sobreescritura [R].
- ▶ Texto a Comando: [Esc]
- ▶ Comando a Última línea: [/] [?]
- ▶ Última línea a Comando: [←] (al finalizar el comando) o [Esc] (interrumpe el comando)

Confundir un modo con otro es la de mayor dificultad para el manejo de vi. Como ayuda se puede activar un indicador de modo, escribiendo «:set showmode», tal cual se muestra en la figura 4.3.

```

root@vectorsigma:~ - □ ×
~ VIM - VI Mejorado
~ versión 8.0.1453
~ por Bram Moolenaar et al.
~ Modificado por pkg-vim-maintainers@lists.alioth.debian.org
~ Vim es código abierto y se puede distribuir libremente
~ ;Patrocine el desarrollo de Vim!
~ escriba <:help sponsor<Intro>> para más información
~ escriba <:q<Intro>> para salir
~ escriba <:help<Intro>> o <F1> para obtener ayuda
~ escriba <:help version8<Intro>> para información de la versión
~ :set showmode 0,0-1 Todo

```

FIGURA 4.3: COMANDO «SET SHOWMODE».

Esto hace aparecer una leyenda que indica si se está en modo comando o inserción.

#### 4.1.5 NÚMEROS MULTIPLICADORES

Muchos comandos aceptan un número multiplicador antes del comando. La acción es idéntica a invocar el comando tantas veces como indica el multiplicador.

Ejemplos:

- ▶ **[1] [0] [j]** en modo comando avanza 10 líneas.
- ▶ **[5] [Y]** copia 5 líneas y las retiene para luego pegar.

#### 4.1.6 EJEMPLOS DE MANEJO

Los ejemplos del cuadro 4.3 de la siguiente página, asumen que el editor se encuentra en modo comando.

SECUENCIA	DESCRIPCIÓN
<b>[←] [↑] [↓] [→]</b>	Mueven el cursor (si el terminal lo permite).
<b>[h] [j] [k] [l]</b>	Mueven el cursor (igual que las flechas).
<b>[i] texto [Esc]</b>	Inserta la palabra «texto» y vuelve a comando.
<b>[x]</b>	Borra el carácter sobre el cursor.
<b>[d] [w]</b>	Borra una palabra.

Cuadro 4.3 – CONTINUACIÓN...

SECUENCIA	DESCRIPCIÓN
<b>[d] [d]</b>	Borra una línea.
<b>[3] [d] [d]</b>	Borra las 3 líneas siguientes.
<b>[u]</b>	Deshace último cambio.
<b>[Z] [Z]</b>	Graba cambios y sale de vi.
<b>[:] [q] [←]</b>	Sale de vi sin grabar cambios.
<b>expresión [←]</b>	Busca la expresión indicada.
<b>[3] [Y]</b>	Copia 3 líneas para luego pegar.
<b>[:] [6] [r] arch3</b>	Inserta debajo de la línea 6 el archivo arch3.

CUADRO 4.3: EJEMPLOS DE MANEJO DE vi.

#### 4.1.7 MOVIMIENTO DEL CURSOR

En el cuadro 4.4 se listan distintas formas de moverse en **CMD vi**.

SECUENCIA	DESCRIPCIÓN
<b>[←] [↑] [↓] [→]</b>	Mover en distintas direcciones.
<b>[h] o [←]</b>	Una posición hacia la izquierda.
<b>[l] o [→]</b>	Una posición hacia la derecha.
<b>[k] o</b>	Una línea hacia arriba.
<b>[j] o</b>	Una línea hacia abajo.
<b>[\\$]</b>	Fin de línea.
<b>[0]</b>	Principio de línea.
<b>[1] [G]</b>	Comienzo del archivo.
<b>[G]</b>	Fin del archivo.
<b>[1] [8] [G]</b>	Línea número 18.
<b>[Ctrl] + [G]</b>	Mostrar número de línea actual.
<b>[w]</b>	Comienzo de la palabra siguiente.
<b>[e]</b>	Fin de la palabra siguiente.
<b>[E]</b>	Fin de la palabra siguiente antes de espacio.
<b>[b]</b>	Principio de la palabra anterior.
<b>[^]</b>	Primera palabra de la línea.
<b>[%]</b>	Hasta el paréntesis que aparezca.
<b>[H]</b>	Parte superior de la pantalla.
<b>[L]</b>	Parte inferior de la pantalla.
<b>[M]</b>	Al medio de la pantalla.
<b>[ ]</b>	Cursor a la columna 23.

CUADRO 4.4: MOVIMIENTOS DEL CURSOR EN vi.

#### 4.1.8 CONTROL DE PANTALLA

Las secuencias de teclas para el gobierno de la pantalla, se halla descriptas en el cuadro 4.5 de la siguiente página.

SECUENCIA	DESCRIPCIÓN
<b>[Ctrl] + [f]</b>	Una pantalla adelante.
<b>[Ctrl] + [b]</b>	Una pantalla atrás.
<b>[Ctrl] + [l]</b>	Redibujar la pantalla.
<b>[Ctrl] + [d]</b>	Media pantalla adelante.
<b>[Ctrl] + [u]</b>	Media pantalla atrás.

CUADRO 4.5: CONTROL DE PANTALLA EN vi.

#### 4.1.9 INGRESO EN MODO TEXTO

En el cuadro 4.6, se muestran algunas de las secuencias que permiten ingresar al modo texto.

SECUENCIA	DESCRIPCIÓN
<b>i</b>	Insertar antes del cursor.
<b>I</b>	Insertar al principio de la línea.
<b>a</b>	Insertar después del cursor.
<b>A</b>	Insertar al final de la línea.
<b>o</b>	Abrir línea debajo de la actual.
<b>O</b>	Abrir línea encima de la actual.
<b>R</b>	Sobreescribir (cambiar) texto.

CUADRO 4.6: COMANDOS PARA INGRESO DE TEXTO EN vi.

#### 4.1.10 BORRAR

Las secuencias de teclas para el borrado, se encuentran descriptas en el cuadro 4.7.

SECUENCIA	DESCRIPCIÓN
<b>x</b>	Borrar carácter bajo el cursor.
<b>d</b> <b>d</b>	Borrar línea.
<b>D</b>	Borrar desde cursor hasta el fin de línea.
<b>d</b> <b>w</b>	Borrar desde cursor hasta el fin de palabra.
<b>d</b> <b>\$</b>	Borrar desde cursor hasta el fin de línea.
<b>d</b> <b>0</b>	Borrar desde cursor hasta el principio de línea.

CUADRO 4.7: COMANDOS PARA BORRADO DE TEXTO EN vi

#### 4.1.11 COPIAR Y PEGAR

En el cuadro 4.8 de la siguiente página, figuran las secuencias de teclas para realizar distintas operaciones de «Copiar & Pegar».

SECUENCIA	DESCRIPCIÓN

<b>[Y] o [y] [y]</b>	Copiar línea.
<b>[P]</b>	Pegar antes del cursor.
<b>[P]</b>	Pegar después del cursor.
<b>[y] [w]</b>	Copiar palabra.
<b>[y] [\$]</b>	Copiar de cursor a fin de línea.
<b>0 [0] [y] [a] [y] [o] 0 [0] [a] [Y]</b>	Copiar línea en buffer llamado «a».
<b>[a] 0 [0] [a] [y] [w]</b>	Copiar palabra en buffer llamado «a».
<b>0 [0] [a] [p]</b>	Pegar desde buffer «a», a la derecha del cursor.
<b>0 [0] [a] [P]</b>	Pegar desde buffer «a», a la izquierda del cursor.
<b>0 [0] [b] [d] [d]</b>	Borrar línea y guardar en buffer «b».
<b>0 [0] [b] [d] [w]</b>	Borrar palabra y guardar en buffer «b».

CUADRO 4.8: COPIAR Y PEGAR EN vi

#### 4.1.12 BÚSQUEDA

En el cuadro 4.9, se listan distintas secuencias de teclas para realizar búsquedas.

SECUENCIA	DESCRIPCIÓN
/str	Buscar hacia adelante cadena de caracteres «str».
?str	Buscar hacia atrás cadena de caracteres «str».
[n]	Repetir último comando o .
[N]	Repetir último comando o para el otro lado.
[f]c	Buscar el siguiente carácter «c» en la línea.
[F]c	Buscar el anterior carácter «c» en la línea.
[t]c	Ir al carácter anterior al siguiente «c».
[T]c	Ir al carácter posterior al precedente «c».
;	Repetir el último comando [f], [F], [t], o [T].
,	Último comando [f], [F], [t], o [T] para el otro lado.

CUADRO 4.9: BÚSQUEDA EN vi.

#### Tener presente que



- ▶ La cadena a buscar con / o ? puede ser una expresión regular.
- ▶ La acción de [f], [F], [t], u [T] alcanza sólo a la línea actual; si el carácter buscado no está en esa línea el cursor no se mueve.

#### 4.1.13 REEMPLAZO

Los comandos listados en el cuadro 4.4 de la siguiente página, admiten multiplicadores: un número delante del comando. Al escribir un comando de reemplazo, el editor coloca un símbolo «\$» en donde termina el pedido de reemplazo. Se escribe normalmente, sobreescribiendo, hasta donde se necesite, y se sale con **Esc**.

Secuencia	Descripción
<code>c</code>	Reemplaza caracteres.
<code>c w</code>	Reemplaza palabras.
<code>C o keysc \$</code>	Reemplaza hasta el fin de línea.
<code>c 0</code>	Reemplaza desde el comienzo de línea.

FIGURA 4.4: REEMPLAZO EN VI.

Secuencia	Descripción
<code>J</code>	Unir dos líneas en una.
<code>u</code>	Deshacer última acción.
<code>U</code>	Deshacer todos los cambios en una línea.

CUADRO 4.10: OTROS COMANDOS EN VI.

La sintaxis del comando de búsqueda y reemplazo es la siguiente:

SIN :<desde>, <hasta>s/<buscar>/<reemplazar>/g

#### 4.1.14 OTROS

Comandos misceláneos, listados en el cuadro 4.10.

#### 4.1.15 MODO «EX» O ÚLTIMA LÍNEA

Se listan en el cuadro 4.11, algunos de los comandos que se pueden tipear en la última línea del `CMD vi`.

Secuencia	Descripción
<code>: + q</code>	Salir si no hubo cambios.
<code>: + q +</code>	salir sin guardar cambios.
<code>: + w</code>	Guardar cambios.
<code>: + w arch1</code>	Guardar cambios en archivo arch1.
<code>: + w + q</code>	Guardar cambios y salir.
<code>: + x</code>	Guardar cambios y salir.
<code>: + r arch2</code>	Insertar un archivo.
<code>: + e arch2</code>	Editar un nuevo archivo.
<code>: + e + arch2</code>	Idem sin salvar anterior.
<code>: + r + comando</code>	Insertar salida de comando.
<code>: + shell</code>	Salir al shell (vuelve con <code>CMD exit</code> ).

CUADRO 4.11: MODO «EX» O ÚLTIMA LÍNEA.

Secuencia	Descripción
:1	Mueve a línea 1.
:15	Mueve a línea 15.
:\$	Mueve a última línea.

CUADRO 4.12: SALTAR A UNA LÍNEA ESPECÍFICA EN vi.

Secuencia	Descripción
:set	Cambio de opciones.
:set nu	Mostrar números de línea.
:set nonu	No mostrar números de línea.
:set showmode	Mostrar modo actual de <b>CMD vi</b> .
:set nowrapmode	No mostrar modo actual de <b>CMD vi</b> .

CUADRO 4.13: ALGUNAS CONFIGURACIONES EN vi.

#### 4.1.16 MOVER

En el cuadro 4.12, se listan algunos ejemplos de comandos para ir a una línea específica.

#### 4.1.17 OPCIONES

En el cuadro 4.13, se listan comando para realizar algunas configuraciones mínimas.



Si se desea conocer configuraciones más interesantes de vim, y hasta donde este se puede extender, se busca en cualquier buscador de internet la cadena «vim like an ide», y se configura el buscador para que muestre «solo imágenes».

A los fines de adquirir velocidad, hay recursos en internet para realizar prácticas de **CMD vim**. Por ejemplo <https://vim-adventures.com>. Figura 4.5 de la siguiente página.

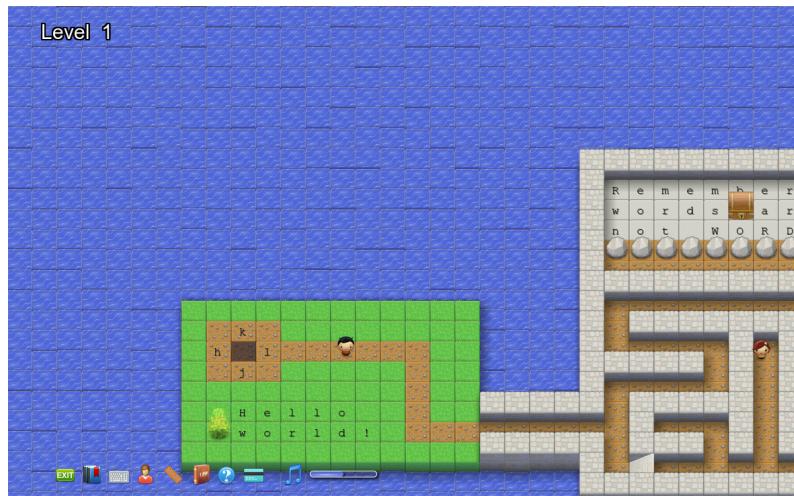


FIGURA 4.5: VIM-ADVENTURES

## 5 | Seguridad y Accesos

### 5.1 ADMINISTRACIÓN DE USUARIOS

GNU/Linux es un sistema multi-usuario, por lo tanto, la tarea de añadir, modificar, eliminar y en general administrar usuarios se convierte en algo no solo de rutina, sino se suma importancia, además de ser un elemento de seguridad que mal administrado o tomado sin la consideración adecuada, puede convertirse en una vulnerabilidad.

### 5.2 TIPOS DE USUARIOS

Los usuarios en \*nix/Linux se identifican por un identificador único de usuario, «User ID», UID. Y pertenecen a un grupo principal de usuario, identificado también por un identificador único de grupo, «Group ID», GID. El usuario puede pertenecer a más grupos además del principal.

Es posible identificar tres tipos de usuarios distintos en GNU/Linux:

#### 5.2.1 USUARIO root

- ▶ También llamado «superusuario» o administrador.
- ▶ Su UID es 0 (cero).
- ▶ Es la única cuenta de usuario con privilegios sobre todo el sistema.
- ▶ Acceso total a todos los archivos y directorios con independencia de propietarios y permisos.
- ▶ Controla la administración de cuentas de usuarios.
- ▶ Ejecuta tareas de mantenimiento del sistema.
- ▶ Puede detener el sistema.
- ▶ Instala software en el sistema.
- ▶ Puede modificar o reconfigurar el kernel, controladores, etc.

#### 5.2.2 USUARIOS ESPECIALES

- ▶ Ejemplos: bin, daemon, adm, lp, sync, shutdown, mail, operator, squid, apache, etc.
- ▶ Se les llama también cuentas del sistema.

- ▶ No tienen todos los privilegios del usuario root, pero dependiendo de la cuenta, asumen distintos privilegios de root. Esto es para proteger al sistema de posibles vulnerabilidades.
- ▶ No tienen contraseñas pues son cuentas que no están diseñadas para iniciar sesiones con ellas.
- ▶ También se los conoce como cuentas de «no inicio de sesión» (nologin).
- ▶ Se crean (generalmente) automáticamente al momento de la instalación de Linux o de un servicio (web, mail, ftp, etc.)
- ▶ Generalmente se les asigna un UID entre 1 y 100 (definido en /etc/login.defs)

### 5.2.3 USUARIOS NORMALES

- ▶ Se usan para usuarios individuales.
- ▶ Cada usuario dispone de un directorio de trabajo, ubicado generalmente en /home.
- ▶ Cada usuario puede personalizar su entorno de trabajo.
- ▶ Tienen solo privilegios completos en su directorio de trabajo o HOME.
- ▶ Por seguridad, es siempre mejor trabajar como un usuario normal en vez del usuario root, y cuando se requiera hacer uso de comandos solo de root, utilizar el comando **CMD su** o **CMD sudo**.
- ▶ En las distros actuales de Linux se les asigna generalmente un UID superior a 500.

## 5.3 ARCHIVOS IMPORTANTES

Los tres archivos más importantes están muy relacionados entre sí; passwd, shadow y groups, tal cual se indica en la figura 5.1, de la misma manera que se relacionan las tablas de un sistema de base de datos relacional. En cierta forma, son tablas, cuyos registros están separados por CR<sup>1</sup> «Carrier Return», y los campos por dos puntos «::».

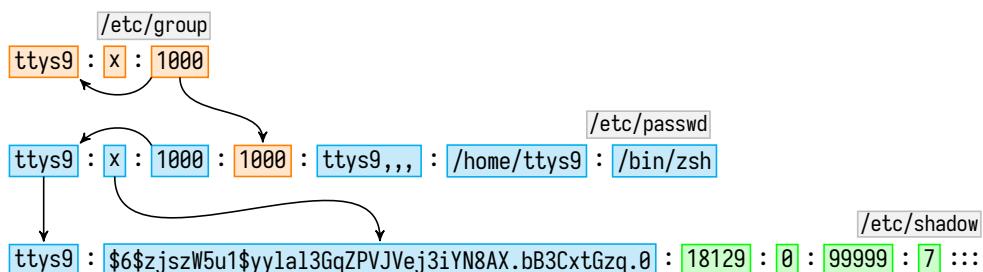


FIGURA 5.1: RELACIÓN ENTRE LOS ARCHIVOS shadow, passwd Y groups

Se puede observar que el campo #3 del archivo /etc/passwd, esta relacionado con el campo #1 del mismo archivo. A su vez, el campo #1 del archivo /etc/passwd, esta relacionado con el campo #1 del archivo /etc/group. Y por último, el campo #4 del archivo /etc/passwd, esta relacionado con el campo #3 del archivo /etc/group.

<sup>1</sup>Es el carácter que se inserta en Linux o \*nix, cuando se presiona **↵**. En Windows son CR y LF «Line Feed»

### 5.3.1 /etc/passwd

Cualquiera que sea el tipo de usuario, todas las cuentas se encuentran definidas en el archivo de configuración `/etc/passwd`. Este archivo es de texto tipo ASCII, se crea al momento de la instalación, conjuntamente con el usuario `root` y las cuentas especiales, más las cuentas de usuarios normales que se hayan indicado al momento de la instalación.

El archivo `/etc/passwd` contiene una línea para cada usuario, similar a las siguientes:

#### Fragmento de `/etc/passwd`

```
1 root:x:0:0:root:/root:/bin/bash
2 erik:x:501:500:Erik Lehnsherr:/home/erik:/bin/bash
```

La información de cada usuario está dividida en siete campos delimitados cada uno por «`:`» dos puntos.

CAMPO	DESCRIPCIÓN
1	Es el nombre del usuario, identificador de inicio de sesión ( <code>login</code> ). Tiene que ser único.
2	La « <code>x</code> » indica la contraseña cifrada del usuario, además también indica que se está haciendo uso del archivo <code>/etc/shadow</code> ; si no se hace uso de este archivo, este campo se vería algo así como: « <code>ghy675gjuXCc12r5gt78uuu6R</code> ».
3	Número de identificación del usuario ( <code>UID</code> ). Tiene que ser único. 0 para <code>root</code> , generalmente las cuentas o usuarios especiales se numeran del 1 al 100 y las de usuario normal del 101 en adelante, en las distribuciones más recientes esta numeración comienza a partir del 500.
4	Numeración de identificación del grupo ( <code>GID</code> ). El que aparece es el número de grupo principal del usuario, pero puede pertenecer a otros, esto se configura en <code>/etc/groups</code> .
5	Comentarios o el nombre completo del usuario.
6	Directorio de trabajo establecido para el usuario (generalmente se lo sitúa allí ni bien ingresa al sistema operativo.)
7	Shell que va a utilizar el usuario de forma predeterminada.

CUADRO 5.1: CAMPOS EN `/etc/passwd`

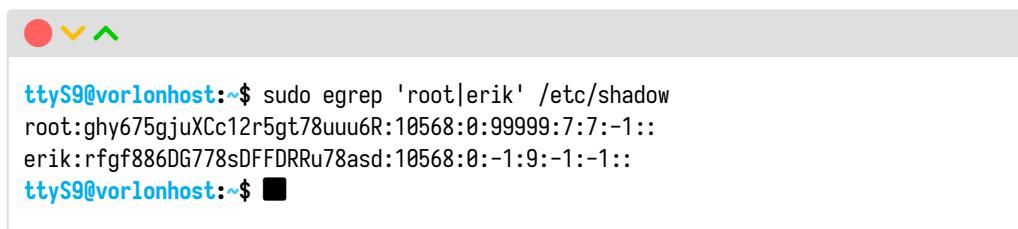
### 5.3.2 /etc/shadow

Anteriormente (en sistemas `*nix™`) las contraseñas cifradas se almacenaban en el mismo `/etc/passwd`. El problema es que `passwd` es un archivo que puede ser leído por cualquier usuario del sistema, aunque sólo puede ser modificado por `root`. Con cualquier computadora potente de hoy en día, con un buen programa de descifrado de contraseñas y paciencia, es posible tener éxito en contraseñas débiles con un ataque por fuerza bruta (por eso la conveniencia de cambiar periódicamente la contraseña de `root` y de otras cuentas importantes). El archivo `shadow`, resuelve el problema ya que solo puede ser leído por `root`. Considérese a `shadow` como una extensión de `passwd` ya que no solo almacena la contraseña cifrada, sino que tiene otros campos de control de

## 5

contraseñas.

El archivo /etc/shadow contiene una línea para cada usuario, similar a las siguientes:



```
ttyS9@vorlonhost:~$ sudo egrep 'root|erik' /etc/shadow
root:ghy675gjuXCC12r5gt78uuu6R:10568:0:99999:7:7:-1::
erik:rfgf886DG778sDFFDRRu78asd:10568:0:-1:9:-1:-1::
ttyS9@vorlonhost:~$
```

La información de cada usuario está dividida en nueve campos delimitados cada uno por «::» dos puntos.

CAMPO	DESCRIPCIÓN
1	Nombre de la cuenta del usuario.
2	Contraseña cifrada, un «*» indica cuenta de nologin.
3	Días transcurridos desde el 1/ene/1970 hasta la fecha en que la contraseña fue cambiada por última vez.
4	Número de días que deben transcurrir hasta que la contraseña se pueda volver a cambiar.
5	Número de días tras los cuales hay que cambiar la contraseña. (-1 significa nunca). A partir de este dato se obtiene la fecha de expiración de la contraseña.
6	Número de días antes de la expiración de la contraseña, en que se le avisará al usuario al inicio de la sesión.
7	Días después de la expiración en que la contraseña procederá a inhabilitarse, si es que no se cambió.
8	Fecha de caducidad de la cuenta. Se expresa en días transcurridos desde el 1/Enero/1970 <sup>2</sup> .
9	Reservado.

CUADRO 5.2: CAMPOS EN /etc/shadow

### 5.3.3 /etc/group

Este archivo guarda la relación de los grupos a los que pertenecen los usuarios del sistema, contiene una línea para cada usuario con tres o cuatro campos por usuario:

<sup>2</sup>A este período de tiempo, se lo conoce como «epoch».

```
ttyS9@vorlonhost:~$ egrep 'root|ana|admin|ventas' /etc/group
root:x:0:root
ana:x:501:
admin:x:502:erik
ventas:x:503:jane,erik
ttyS9@vorlonhost:~$ █
```

CAMPO	DESCRIPCIÓN
1	Nombre de la cuenta del usuario.
2	«x» indica la contraseña del grupo, que no existe, si hubiera se mostraría un «hash» cifrado.
3	Es el Group ID (GID) o identificación del grupo.
4	Es opcional e indica la lista de grupos a los que pertenece el usuario.

CUADRO 5.3: CAMPOS EN /etc/group

Actualmente al crear al usuario con **CMD useradd** se crea también automáticamente su grupo principal de trabajo GID, con el mismo nombre del usuario. Es decir, si se añade el usuario erik también se crea el /etc/group el grupo erik. Aún así, existen comandos de administración de grupos que se explicarán más adelante.

## 5.4 COMANDOS PRINCIPALES DE ADMINISTRACIÓN DE USUARIOS

### 5.4.1 useradd

**CMD useradd** o **CMD adduser** permiten añadir nuevos usuarios al sistema desde la línea de comandos. Las opciones más comunes se encuentran descriptas en el cuadro 5.4 de la siguiente página.

ARGUMENTO	DESCRIPCIÓN
-c	Añade un comentario al momento de crear al usuario, campo #5 de /etc/passwd.
-d	Directorio de trabajo o home del usuario, campo 6 de /etc/passwd.
-e	Fecha de expiración de la cuenta, formato AAAA-MM-DD, campo #8 de /etc/shadow.
-g	Número de grupo principal del usuario (GID), campo #4 de /etc/passwd.
-G	Otros grupos a los que puede pertenecer el usuario, separados por comas.

Cuadro 5.4 – CONTINUACIÓN...

ARGUMENTO	DESCRIPCIÓN
-r	Crea una cuenta del sistema (también llamada especial), su UID será menor al definido en /etc/login.defs en la variable UID_MIN, además no se crea el directorio de inicio.
-s	Shell por defecto del usuario cuando ingrese al sistema. Si no se especifica, bash, es el que queda establecido.
-u	El UID del usuario, si no se indica esta opción, automáticamente se establece el siguiente número disponible a partir del último usuario creado.

CUADRO 5.4: ARGUMENTOS DE useradd

Ahora bien, realmente no hay necesidad de indicar ninguna opción ya que si se hace lo siguiente:

**CMD** sudo useradd juan

Se creará el usuario y su grupo, así como las entradas correspondientes en /etc/passwd, /etc/shadow y /etc/group. También se creará el directorio de inicio o de trabajo: /home/juan y los archivos de configuración que van dentro de este directorio y que más adelante se detallan.

Las fechas de expiración de contraseñas, entre otros campos, quedan lo más generosas posibles para que no haya problemas de caducidad, así que prácticamente lo único que faltaría hacer, sería añadir la contraseña del usuario y algún comentario o identificación de la cuenta. Viendo las opciones, con -c es posible establecer el comentario, (campo #5 de /etc/passwd):

**CMD** sudo useradd -c "Pablo Niklas" pablo

Siempre el nombre del usuario es el último parámetro del comando. Si se quiere salir del default, se puede establecer algo como lo siguiente:

**CMD** useradd -d /usr/juan -s /bin/csh -u 800 -c "Pablo Niklas" pablo

Con lo anterior se está cambiando su directorio de inicio, su shell por default será csh y su UID será el 800 en vez de que el sistema tome el siguiente número disponible.

### adduser y addgroup



Añaden usuarios y grupos al sistema de acuerdo a las opciones de la línea de órdenes y a la configuración en /etc/adduser.conf. Ofrecen una interfaz más sencilla para programas de bajo nivel como **CMD** useradd, **CMD** groupadd y **CMD** usermod, seleccionando valores para el identificador de usuario (UID) e identificador de grupo de usuarios (GID) conforme a las normas de Debian. También crean el directorio personal de la forma /home/USUARIO con la configuración predeterminada, ejecutan un script personalizado y otras funcionalidades.

### 5.4.2 usermod

**CMD** `usermod` permite modificar o actualizar un usuario o cuenta ya existente. Sus opciones más comunes se encuentran descriptas en el cuadro 5.5.

ARGUMENTO	DESCRIPCIÓN
<code>-c</code>	Añade o modifica el comentario, campo #5 de <code>/etc/passwd</code> .
<code>-d</code>	Modifica el directorio de trabajo o home del usuario, campo #6 de <code>/etc/passwd</code> .
<code>-e</code>	Cambia o establece la fecha de expiración de la cuenta, formato AAAA-MM-DD, campo #8 de <code>/etc/shadow</code> .
<code>-g</code>	Cambia el número de grupo principal del usuario (GID), campo #4 de <code>/etc/passwd</code> .
<code>-G</code>	Establece otros grupos a los que puede pertenecer el usuario, separados por comas.
<code>-l</code>	Cambia el login o nombre del usuario, campo #1 de <code>/etc/passwd</code> y de <code>/etc/shadow</code> .
<code>-L</code>	Bloquea la cuenta del usuario, no permitiéndole que ingrese al sistema. No borra ni cambia nada del usuario, solo lo deshabilita.
<code>-s</code>	Cambia el shell por defecto del usuario cuando ingrese al sistema.
<code>-u</code>	Cambia el UID del usuario.
<code>-U</code>	Desbloquea una cuenta previamente bloqueada con la opción <code>-L</code> .

CUADRO 5.5: ARGUMENTOS DE usermod

Si se quiere cambiar el nombre de usuario de «erik» a «eric»:

**CMD** `usermod -l erik eric`

También cambiará el nombre del directorio de inicio o «home» en `/home`, pero si no fuera así, entonces:

**CMD** `usermod -d /home/erik eric`

Otros cambios o modificaciones en la misma cuenta:

**CMD** `usermod -c "supervisor de area" -s /bin/ksh -g 505 eric`

Lo anterior modifica el comentario de la cuenta, su shell por defecto que ahora sera «Korn» y su grupo principal de usuario queda establecido al GID 505 y todo esto se aplicará al usuario eric que como se observa debe ser el último argumento del comando.

El usuario eric salió de vacaciones y hay que asegurarse de que nadie use su cuenta:

**CMD** `usermod -L eric`

## 5

## 5.4.3 userdel

Como su nombre lo indica, **CMD userdel** elimina una cuenta del sistema. Puede ser invocado de tres maneras:

**CMD userdel eric**

Sin opciones elimina la cuenta del usuario de `/etc/passwd` y de `/etc/shadow`, pero no elimina su directorio de trabajo ni archivos contenidos en el mismo, esta es la mejor opción, ya que elimina la cuenta pero no la información de la misma.

**CMD userdel -r eric**

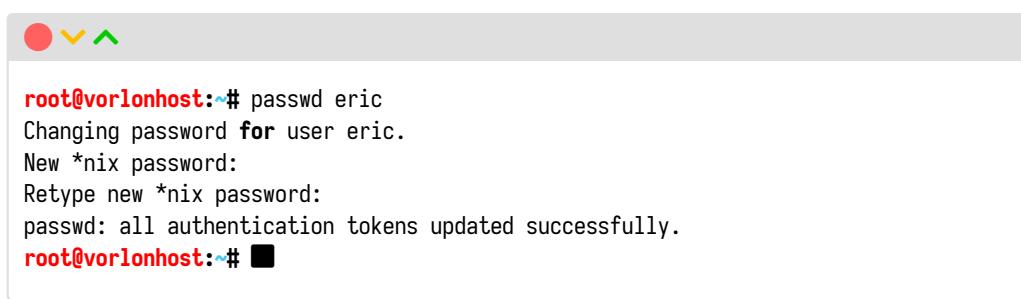
Al igual que lo anterior elimina la cuenta totalmente, pero con la opción «`-r`» además elimina su directorio de trabajo y archivos y directorios contenidos en el mismo, así como su buzón de correo en `/var/spool/mail`, si es que estuvieran configuradas las opciones de correo. La cuenta no se podrá eliminar si el usuario está logueado o en el sistema al momento de ejecutar el comando.

**CMD userdel -f eric**

La opción «`-f`» es igual que la opción «`-r`», elimina todo lo del usuario, cuenta, directorios y archivos del usuario, pero además lo hace sin importar si el usuario está actualmente en el sistema trabajando. Es una opción muy radical, además de que podría causar inestabilidad en el sistema, así que hay que usarla solo en casos muy extremos.

## 5.4.4 passwd

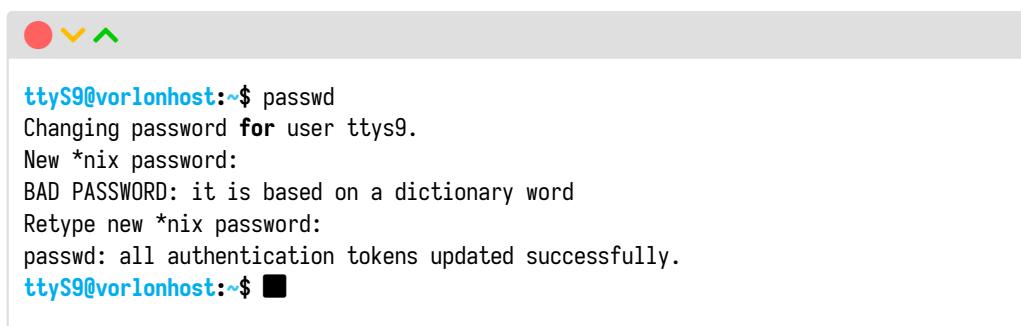
Crear al usuario con **CMD useradd** es el primer paso, el segundo es asignarle una contraseña a ese usuario, lo cual se logra con el comando **CMD passwd** que permitirá ingresar la contraseña y su verificación:



```
root@vorlonhost:~# passwd eric
Changing password for user eric.
New *nix password:
Retype new *nix password:
passwd: all authentication tokens updated successfully.
root@vorlonhost:~#
```

El usuario `root` es el único que puede cambiar las contraseñas de cualquier usuario. Los usuarios normales pueden cambiar su contraseña en cualquier momento con tan solo invocar **CMD passwd** sin argumentos, y se podrá de esta manera cambiar la contraseña cuantas veces se lo requiera.

**CMD passwd** tiene integrado la validación de contraseñas que sean comunes, cortas, de diccionario, etc. así que por ejemplo se intentó como usuario normal cambiar la contraseña a «`qwerty`» el sistema mostrará lo siguiente:



```
ttyS9@vorlonhost:~$ passwd
Changing password for user ttys9.
New *nix password:
BAD PASSWORD: it is based on a dictionary word
Retype new *nix password:
passwd: all authentication tokens updated successfully.
ttyS9@vorlonhost:~$
```

Nótese que al ingresar «qwerty» como contraseña se detectó que es una secuencia ya conocida como contraseña y se informa mediante la advertencia: «BAD PASSWORD: it is based on a dictionary word», sin embargo se permite continuar, al ingresar la verificación. Es decir, **CMD** `passwd` avisa de contraseñas malas o débiles pero permite establecerlas si realmente se desea (solo siendo root).

Resumiendo entonces, se podría decir que todo se reduce a dos líneas de comandos para crear y dejar listo para trabajar a un usuario en Linux: **CMD** `adduser` y **CMD** `passwd`; **CMD** `adduser` hace todo el trabajo de establecer el shell, directorio de inicio, copiar archivos iniciales de configuración de la cuenta, etc. y después **CMD** `passwd` establece la contraseña.

**CMD** `passwd` tiene varias opciones que permiten bloquear la cuenta -1, desbloquearla -u, y varias opciones más que controlan la vigencia de la contraseña, es decir, es otro modo de establecer los valores de la cuenta en `/etc/shadow`. Para más información se debe consultar las páginas del manual.

## 5.5 ARCHIVOS DE CONFIGURACIÓN DE BASH

Tanto root como los usuarios normales, en sus directorios de inicio, tienen varios archivos que comienzan con un punto (.) es decir están ocultos. Varían mucho dependiendo de la distribución de Linux que se tenga, pero seguramente se pueden encontrar los enunciados en el cuadro 5.6 de la siguiente página.

ARCHIVO	DESCRIPCIÓN
<code>.bash_profile</code>	Aquí se podrá indicar alias, variables, configuración del entorno, etc. que se definen al principio de la sesión.
<code>.bash_logout</code>	Aquí se podrá indicar acciones, programas, scripts, etc., que se deseé ejecutar al salir de la sesión.
<code>.bashrc</code>	Es igual que <code>.bash_profile</code> , se ejecuta al principio de la sesión, tradicionalmente en este archivo se indican los programas o scripts a ejecutar, a diferencia de <code>.bash_profile</code> que configura el entorno.

Cuadro 5.6 – CONTINUACIÓN...

ARCHIVO	DESCRIPCIÓN
.profile	Es leído por la mayoría de los shells en el inicio, incluyendo bash. Sin embargo, .bash_.profile se utiliza para configuraciones específicas para bash. El código de inicialización general, debe ir en .profile. Si es específico para bash, usar .bash_profile. Éste no está realmente diseñado para bash específicamente, en cambio, .bash_profile si. (.profile es para Bourne y otros shells similares, en los que bash está basado) Bash procesará .profile si no se encuentra .bash_profile.
.bash_login	Es un respaldo para .bash_profile, si no se encuentra. Generalmente es mejor usar .bash_profile o .profile en su lugar.

CUADRO 5.6: ARCHIVOS DE CONFIGURACIÓN DE BASH.

## 5.6 RESUMEN DE COMANDOS Y ARCHIVOS DE ADMINISTRACIÓN DE USUARIOS

Existen varios comandos más que se usan muy poco en la administración de usuarios, que sin embargo permiten administrar aún más a detalle a tus usuarios de Linux. Algunos de estos comandos permiten hacer lo mismo que los comandos previamente vistos, solo que de otra manera, y otros como **CMD chpasswd** y **CMD newusers** resultan muy útiles y prácticos cuando de dar de alta a múltiples usuarios se trata.

A continuación se presenta un resumen de los comandos y archivos vistos más otros que un poco de investigación:

### 5.6.1 COMANDOS DE ADMINISTRACIÓN Y CONTROL DE USUARIOS

En el cuadro 5.7 de la siguiente página se ilustran los comandos para la administración de los usuarios.

COMANDO	DESCRIPCIÓN
adduser	Idem useradd, con las salvedades mencionadas anteriormente.
chage	Permite cambiar o establecer parámetros de las fechas de control de la contraseña.
chpasswd	Actualiza o establece contraseñas en modo batch, múltiples usuarios a la vez. (se usa junto con newusers)
id	Muestra la identidad del usuario (UID) y los grupos a los que pertenece.
gpasswd	Administra las contraseñas de grupos (/etc/group y /etc/gshadow).
groupadd	Añade grupos al sistema (/etc/group).
groupdel	Elimina grupos del sistema.
groupmod	Modifica grupos del sistema.

Cuadro 5.7 – CONTINUACIÓN...

COMANDO	DESCRIPCIÓN
groups	Muestra los grupos a los que pertenece el usuario.
newusers	Actualiza o crea usuarios en modo batch, múltiples usuarios a la vez. (se usa junto chpasswd)
pwconv	Establece la protección shadow (/etc/shadow) al archivo /etc/passwd.
pwunconv	Elimina la protección shadow (/etc/shadow) al archivo /etc/passwd.
useradd	Añade usuarios al sistema (/etc/passwd).
userdel	Elimina usuarios del sistema.
usermod	Modifica usuarios.

CUADRO 5.7: COMANDOS DE ADMINISTRACIÓN Y CONTROL DE USUARIOS.

## 5.7 PERMISOS

Se ha visto esa combinación de rwx y - cuando se lista un directorio, ya se sabe que son los permisos pero, ¿cómo se usan y cómo funcionan?

### Niveles de permisos de acceso



En Linux, todo archivo y directorio tiene tres niveles de permisos de acceso: los que se aplican al **propietario** del archivo, los que se aplican al **grupo** que tiene el archivo y los que se aplican **al resto** de los usuarios.

Se pueden ver los permisos listando un directorio con **CMD ls -l**:

```

● ▼ ^

tys9@vorlonhost:~$ ls -l
-rwxrwxr-- 1 eric ventas 9090 sep  9 14:10 presentacion
-rw-rw-r-- 1 eric eric 2825990 sep  7 16:36 reporte1
drwxr-xr-x 2 eric eric   4096 ago 27 11:41 videos
tys9@vorlonhost:~$ █

```

Mejor es ver por partes el listado, se toma como ejemplo la primera línea:

1. La primera columna (-rwxrwxr--) es el tipo de archivo y sus permisos.
2. La siguiente columna es el número de enlaces al archivo.
3. La tercera columna (eric) representa al propietario del archivo.

## 5

4. La cuarta columna (ventas) representa el grupo al que pertenece el archivo.

Las siguientes son el tamaño, la fecha y hora de última modificación y por último el nombre del archivo o directorio.

El primer carácter al extremo izquierdo, representa el tipo de archivo, los posibles valores para esta posición son los siguientes:

COMANDO	DESCRIPCIÓN
-	«archivo regular» - un guión representa un archivo común (de texto, html, .mp3, .jpg, etc.)
d	«directorio» representa un directorio.
l	«link», es decir un enlace o acceso directo.
b	«archivos de bloque», archivos que representan hardware. Usualmente en /dev.
c	«archivos de caracter», proveen un stream serial para entrada o salida. Las terminales son ejemplo. Se hallan en /dev.
p	«archivo de pipe», archivo FIFO o «named pipes», o tuberías, no ocupan lugar en el espacio y son como un punto de referencia en el filesystem al solo efecto de que dos procesos intercambien información. Se crean con  mkfifo. <sup>3</sup>

CUADRO 5.8: TIPOS DE ARCHIVO

Los siguientes nueve restantes, representan los permisos del archivo y deben verse en grupos de tres.

PERMISOS	VALOR OCTAL	DESCRIPCIÓN
- - -	0	Sin permisos alguno.
- - x	1	Solo permiso de ejecución.
- w -	2	Solo permiso de escritura.
- w x	3	Permisos de escritura y ejecución.
r - -	4	Solo permiso de lectura.
r - x	5	Permisos de lectura y ejecución.
r w -	6	Permisos de lectura y escritura.
r w x	7	Todos los permisos establecidos, lectura, escritura y ejecución.

CUADRO 5.9: ESQUEMA DE PERMISOS

Los tres primeros representan los permisos para el propietario del archivo. Los tres siguientes son los permisos para el grupo del archivo y los tres últimos son los permisos para el resto del mundo u otros.

En cuanto a las letras de la figura 5.2 de la siguiente página, su significado son los siguientes:

<sup>3</sup>Queda fuera del alcance de esta publicación.

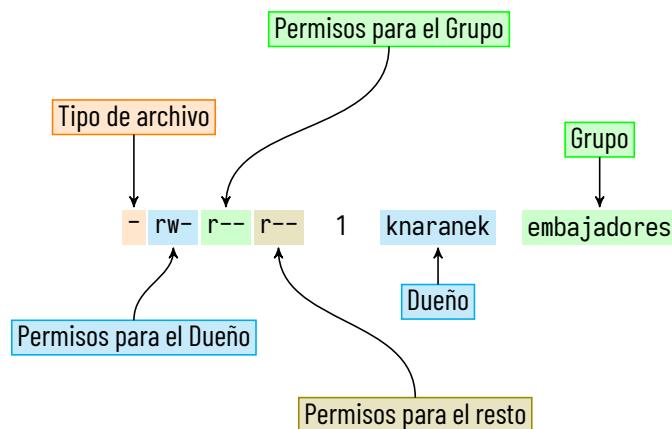


FIGURA 5.2: DISTRIBUCIÓN DEL ESQUEMA DE PERMISOS

- ▶ **r** → «read» lectura.
- ▶ **w** → «write» escritura (en archivos: permiso de modificar, en directorios: permiso de crear archivos en el dir.)
- ▶ **x** → «execution» ejecución (en archivos: ejecutarlo, en directorios: poder entrar).

Las nueve posiciones de permisos son en realidad un bit que o está encendido (mostrado con su letra correspondiente) o está apagado (mostrado con un guión -), así que, por ejemplo, permisos como **rw-rw-r--**, indicaría que los permisos al propietario (rwx) permiten leer, escribir y ejecutar el archivo, al grupo (o sea los usuarios que estén en mismo grupo del archivo) (rw-) podrá leer y escribir pero no ejecutar el archivo, y a cualquier otro usuario del sistema (r--), sólo podrá leer el archivo, ya que los otros dos bits de lectura y ejecución no se encuentran encendidos o activados.

## 5.8 PERMISOS EN FORMATO NUMÉRICO OCTAL

La combinación de valores de cada grupo de los usuarios forma un número octal, el bit x es 20 es decir 1, el bit w es 21 es decir 2, el bit r es 22 es decir 4, entonces:

- ▶ **r = 4**
- ▶ **w = 2**
- ▶ **x = 1**

La combinación de bits encendidos o apagados en cada grupo da ocho posibles combinaciones de valores ( $2^3 = 8$ ), es decir la suma de los bits encendidos, como se aprecia en la tabla 5.9 en la anterior página.

Cuando se combinan los permisos del usuario, grupo y otros, se obtienen un número de tres cifras que conforman los permisos del archivo o del directorio. Esto es más fácil visualizarlo con algunos ejemplos en la tabla 5.10 de la siguiente página.

Permisos	Valor	Descripción
rw-----	600	El propietario tiene permisos de lectura y escritura.
rwx--x--x	711	El propietario tiene permisos de lectura, escritura y ejecución, el grupo y otros sólo ejecución.
rwxr-xr-x	755	El propietario tiene permisos de lectura, escritura y ejecución, el grupo y otros pueden leer y ejecutar el archivo.
rwxrwxrwx	777	El archivo puede ser leído, escrito y ejecutado por quien sea.
r-----	400	Solo el propietario puede leer el archivo, pero se puede modificar ni ejecutar, ni por él ni por nadie.
rw-r----	640	El usuario propietario puede leer y escribir, el grupo puede leer el archivo y otros no pueden hacer nada.

CUADRO 5.10: EQUIVALENCIAS DE PERMISOS DE ARCHIVOS ENTRE SIMBÓLICO Y OCTAL.

## 5.9 ESTABLECIENDO LOS PERMISOS CON EL COMANDO chmod

Habiendo entendido lo anterior, es ahora fácil cambiar los permisos de cualquier archivo o directorio, usando el comando **chmod** «change mode», cuya sintaxis es la siguiente:

**SIN** `chmod [opciones] permisos archivo[s]`

algunos ejemplos:

**CMD** `chmod 755 reporte1` **CMD** `chmod 511 respaldo.sh` **CMD** `chmod 700 pablo*` **CMD** `chmod 644 *`

Los ejemplos anteriores establecen los permisos correspondientes que el usuario propietario desea establecer, el tercer ejemplo (**CMD** `chmod 700 pablo *`) cambiará los permisos a todos los archivos que empiecen con «pablo» es decir pablo01, pablo02, pablo\_respaldo, etc. debido al carácter «\*» que es parte de las expresiones regulares que el shell acepta, e indica «lo que sea». El último ejemplo por lo tanto cambiará los permisos a los archivos dentro del directorio actual.

Una opción común cuando se desea cambiar todo un árbol de directorios, es decir, varios directorios anidados y sus archivos correspondientes, es usar la opción «-R», de recursividad:

**CMD** `chmod -R 755 respaldos/*`

Esto cambiará los permisos a 755 (rwxr-xr-x) del directorio respaldos y de todos los subdirectorios y archivos que estén contenidos en de éste.

## 5.10 ESTABLECIENDO PERMISOS EN MODO SIMBÓLICO

Otra manera de establecer los permisos de un archivo o directorio es a través de identificadores del bit (r,w, o x) de los permisos, como ya se vió anteriormente, pero ahora identificando además lo siguiente:

**SIN** `chmod augo[+|-]rwx[,...] archivo[s]`

- ▶ al usuario con la letra «u»

- ▶ al grupo con la letra «g»
- ▶ a otros usuarios con la letra «o»
- ▶ y cuando nos referimos a todos (usuario, grupo, otros) con la letra «a» (all, «todos» en inglés)
- ▶ el signo «+» para establecer el permiso
- ▶ el signo «-» para eliminar o quitar el permiso

La sintaxis es muy simple, por ejemplo, si se quiere que otros tengan permiso de escritura el comando es **SIN chmod o+w <archivo>**. Por otro lado, si se quiere que todos los usuarios posean permisos de ejecución, **CMD chmod a+x archivo**. En este modo de establecer permisos, sólo hay que tomar en cuenta que partiendo de los permisos ya establecidos se agregan o se quitan a los ya existentes. Se puede notar con ejemplos su manera de trabajar, en la tabla 5.11.

ANTES	SE OTORGA	DESPUÉS	DESCRIPCIÓN
rwx-----	a+x	rwx--x--x	Agregar a todos (all) permisos de escritura.
rwx--x--x	go-x	rwx-----	Se eliminan permiso de ejecución para grupo y otros.
rwxr-xr-x	u-x,go-r	rwx---x--x	Al usuario se le quita ejecución, al grupo y otros se les quita lectura.
rwxrwxrwx	u-x,go-rwx	rwx-----	Al usuario se le elimina ejecución, al grupo y otros se les eliminan todos los permisos.
r-----	a+r,u+w	rw-r--r--	A todos se les agrega lectura, al usuario se le agrega escritura.
rw-r-----	u-rw,g+w,o+x	---rw---x	Al usuario se le eliminan lectura y escritura, al grupo se le agrega lectura y a otros se les agrega ejecución.

CUADRO 5.11: EJEMPLOS DE OTORGAMIENTO DE PERMISOS DE ARCHIVOS.

## 5.11 CAMBIANDO PROPIETARIO Y GRUPO

Volviendo a mostrar el listado al inicio de este artículo:

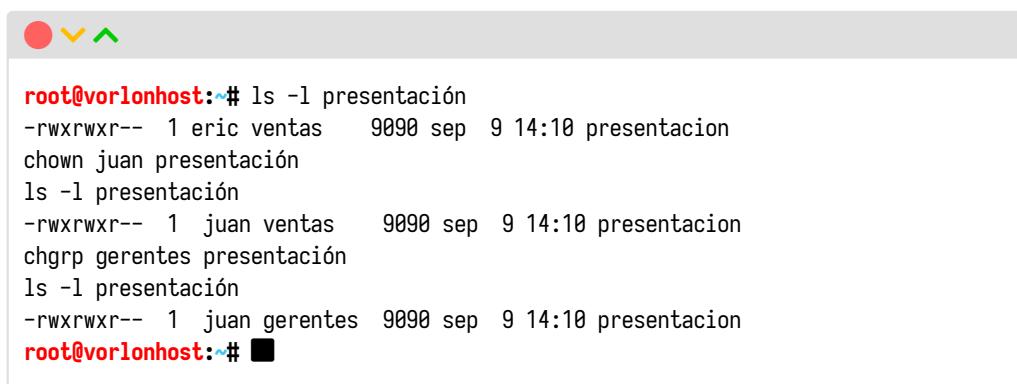


```
ttyS9@vorlonhost:~$ ls -l
-rwxrwxr-- 1 eric ventas 9090 sep  9 14:10 presentacion
-rw-rw-r-- 1 eric eric 2825990 sep  7 16:36 reporte1
drwxr-xr-x 2 eric eric   4096 ago 27 11:41 videos
ttyS9@vorlonhost:~$ █
```

Se denota en la tercera y cuarta columna al usuario propietario del archivo y al grupo al que pertenece. Es posible cambiar estos valores a través de los comandos **CMD chown** («change owner», cambiar propietario) y **CMD chgrp** («change group», cambiar grupo). La sintaxis es muy sencilla:

**SIN** `chown usuario archivo[s]` ó **SIN** `chgrp grupo archivo[s]`

Además al igual que con `chmod`, también es posible utilizar la opción «-R» para recursividad.



```

root@vorlonhost:~# ls -l presentación
-rwxrwxr-- 1 eric ventas 9090 sep 9 14:10 presentacion
chown juan presentación
ls -l presentación
-rwxrwxr-- 1 juan ventas 9090 sep 9 14:10 presentacion
chgrp gerentes presentación
ls -l presentación
-rwxrwxr-- 1 juan gerentes 9090 sep 9 14:10 presentacion
root@vorlonhost:~# █

```

 Solo el usuario `root` puede cambiar usuarios y grupos a su voluntad sobre cualquier usuario, queda claro que habiendo ingresado al sistema como usuario normal, solo podrá hacer cambios de grupos, y eso solo a los que pertenezca.

Una manera rápida para el usuario `root` de cambiar usuario y grupo al mismo tiempo, es con el mismo comando **CMD chown** de la siguiente manera:

**CMD** `chown pablo.gerentes presentación`

**CMD** `chown pablo:gerentes presentación`

Así, se cambiará el `usuario.grupo` en una sola instrucción.

## 5.12 BITS SUID, SGID Y DE PERSISTENCIA «STICKY BIT»

En \*nix y GNU/Linux existen tres bits de permisos especiales que pueden ser asignados a directorios y/o ficheros ejecutables, «setuid» «set user information», «setgid» «set group information» y «sticky».

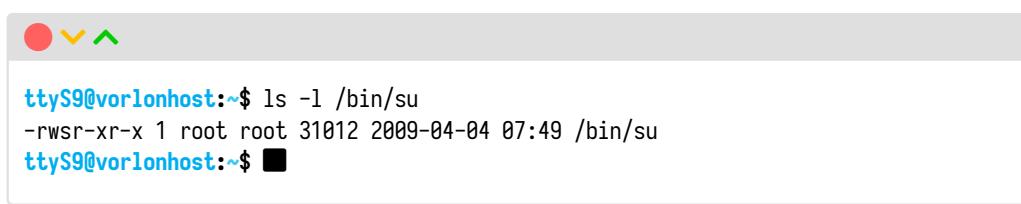
Los posibles valores serían los que figuran en el cuadro 5.10 en página 108.

### 5.12.1 «SETUID»

El bit «setuid» es assignable a ficheros ejecutables, y permite que cuando un usuario ejecute dicho fichero, el proceso adquiera los permisos del propietario del fichero ejecutado. El ejemplo más claro de fichero ejecutable y con el bit setuid es `CMD su`.

`CMD su` sirve para ejecutar un shell con identificadores de grupo y de usuario distintos al usuario que inició que la sesión, por ello ha de tener este bit y así permitir adquirir al usuario temporalmente permisos administrativos para poder hacer el cambio que se necesite.

Se puede ver que el bit está asignado: «s» haciendo un `CMD ls`:



```
ttyS9@vorlonhost:~$ ls -l /bin/su
-rwsr-xr-x 1 root root 31012 2009-04-04 07:49 /bin/su
ttyS9@vorlonhost:~$ █
```

Para asignar este bit a un fichero:

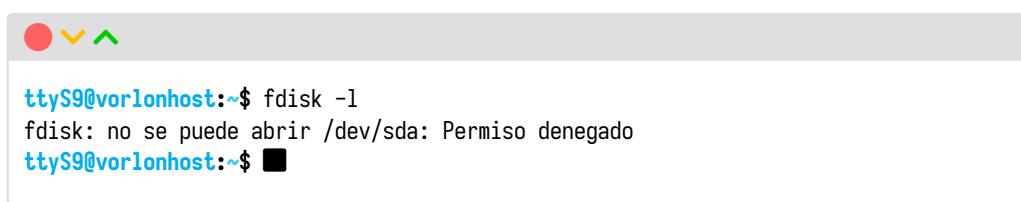
`CMD chmod u+s /bin/su`

Y para quitarlo:

`CMD chmod u-s /bin/su`

Para que se vea un ejemplo de lo que se podría hacer.

El usuario eric no tiene permisos para ejecutar correctamente un `CMD fdisk -l`:



```
ttyS9@vorlonhost:~$ fdisk -l
fdisk: no se puede abrir /dev/sda: Permiso denegado
ttyS9@vorlonhost:~$ █
```

Pero si se asigna el bit «setuid» al binario de `CMD fdisk`



```
root@vorlonhost:~# chmod u+s /sbin/fdisk
root@vorlonhost:~# ls -l /sbin/fdisk
-rwsr-xr-x 1 root root 93048 2009-02-18 20:43 /sbin/fdisk
root@vorlonhost:~# █
```

## 5

Se prueba de nuevo ejecutar **CMD fdisk** con el usuario eric y se obtienen privilegios de root:

```
root@vorlonhost:~# fdisk -l

Disco /dev/sda: 160.0 GB, 160041885696 bytes
255 cabezas, 63 sectores/pista, 19457 cilindros
Unidades = cilindros de 16065 * 512 = 8225280 bytes
Identificador de disco: 0x000c3c51

Dispositivo Inicio     Comienzo      Fin      Bloques  Id Sistema
/dev/sda1    *          1        3232    25959424    7  HPFS/NTFS
/dev/sda2        3233       9683    51817657+   83  Linux
/dev/sda3       9684       9855    1381590    82  Linux swap / Solaris
/dev/sda4       9856      19457    77128065   83  Linux
root@vorlonhost:~#
```

Luego hay que volver a quitarlo:

**CMD chmod u-s /sbin/fdisk**

### SetUID no aplica en scripts



Debido a la alta probabilidad de fallas de seguridad, muchos sistemas operativos ignoran el atributo setUID cuando se aplican a «shell scripts». **Esta prohibición esta dada a nivel del kernel.[suid:shellscripts]**

#### 5.12.2 «SETGID»

Si el bit «SetUID» permitía que el proceso adquiriera los permisos del propietario del fichero ejecutado, «SetGID» hace lo mismo pero adquiriendo los privilegios del grupo asignado al fichero, o al directorio. Este bit entonces, será muy útil cuando varios usuarios de un mismo grupo necesiten trabajar con recursos dentro de un mismo directorio.

En el siguiente ejemplo, se asigna el bit «setgid» la carpeta /compartido y se le asignan permisos totales para el propietario y el grupo (770) y el bit «segid» (2):

**CMD mkdir compartido && chmod 2770 /compartido**

Con **CMD ls** se visualiza el bit asignado:



```
ttyS9@vorlonhost:~$ ls -l
drwxrws--- 2 eric eric      4096 2011-04-24 21:27 compartido
ttyS9@vorlonhost:~$ █
```

Se va a hacer la prueba de crear un fichero dentro del directorio. Se lo va a crear estando autenticado como root, pero al tener el bit «setgid» le asignará el grupo eric en lugar de root.



```
ttyS9@vorlonhost:~$ su

root@vorlonhost:~# touch test
root@vorlonhost:~# ls -l
total 0
-rw-r--r-- 1 root eric 0 2011-04-24 21:28 test
root@vorlonhost:~# whoami
root
root@vorlonhost:~# █
```

En lugar de modo octal se puede asignar también los permisos «setgid» del siguiente modo:

**CMD** chmod g+s /compartido

Y quitarlo:

**CMD** chmod g-s /compartido

### SUID y SGID



Algo sumamente delicado y que se tiene que tomar muy en cuenta es lo que se decida establecer con permisos de bit SUID y SGID, ya que debe recordarse que al establecerlos de esta manera, **cualquier usuario podrá ejecutarlos como si fuera el propietario original de ese programa**. Esto puede tener consecuencias de seguridad severas en el sistema.

#### 5.12.3 «STICKY»

Este bit suele asignarse en directorios a los que todos los usuarios tienen acceso, y permite evitar que un usuario pueda borrar ficheros/directorios de otro usuario dentro de ese directorio, ya que todos tienen permiso de escritura. Este bit se asigna siempre en /tmp y /var/tmp.

## 5

`tmp` tiene permisos 777, el bit «sticky» se asignaría del siguiente modo:

**CMD** `chmod 1777 /tmp`

También así:

**CMD** `chmod o+t /tmp`

Y para quitarlo:

**CMD** `chmod o-t /tmp`

Con **CMD** `ls` se verá la «t» asignada:

```
● ▼ ▲
root@vorlonhost:~# ls -l
drwxrwxrwt 13 root root 4096 2011-04-24 20:55 tmp
root@vorlonhost:~# █
```

### Mejor es sudo



Siempre se debe considerar y reconsiderar si conviene que un usuario normal ejecute aplicaciones propias de `root` a través del cambio de bits SUID o SGID. Mejores alternativas son que la ejecución se dirima vía los comandos **CMD** `sudo` y **CMD** `su`.

## 5.13 PERMISOS PREESTABLECIDOS CON umask

**CMD** `umask` (abreviatura de «user mask», máscara de usuario) es una orden (y una función en entornos POSIX) que establece los permisos por defecto para los nuevos archivos y directorios.

### 5.13.1 MÁSCARAS SIMBÓLICAS

Una máscara establecida a `u=rwx,g=rwx,o=` implica que los nuevos archivos tendrán los permisos `rwxrwx---`, y los nuevos directorios tendrán los permisos `rwxrwx---`.

```
● ▼ ▲
root@vorlonhost:~# umask u=rwx,g=rwx,o=
root@vorlonhost:~# mkdir foo
root@vorlonhost:~# touch bar
root@vorlonhost:~# ls -l
drwxrwx--- 2 dave 512 Sep 1 20:59 foo
```

```
-rw-rw---- 1 dave 0 Sep 1 20:59 bar
root@vorlonhost:~#
```

### 5.13.2 MÁSCARA EN OCTAL

Las máscaras en octal se calculan realizando el AND binario del complemento unario del argumento (utilizando el NOT binario) y los permisos completos.

#### Persistencia de los cambios



Los cambios tendrán efecto únicamente durante la sesión actual. Es decir, no quedan persistentes, una vez que se sale de la sesión.

El modo de acceso completo es 666 en el caso de archivos, y 777 en el caso de directorios. La mayoría de los shells de \*nix proporcionan una orden `umask` que afecta a todos los procesos hijos ejecutados en ese shell.

#### Máscara 022



Una máscara común es 022. Esto es enmascarando los permisos de escritura para el grupo y para otros, lo que asegura que los nuevos archivos sólo pueden modificarse por el propietario (es decir, el usuario que los creó).

Otro valor de máscara común es 002, que deja permisos de escritura para los miembros del grupo al que pertenece el archivo. Esto puede utilizarse en archivos en áreas de trabajo compartidas, en las que varios usuarios trabajan con los mismos archivos.

### 5.13.3 EJERCICIO DE EJEMPLO DE MÁSCARA (EN OCTAL)

Se asume que, para este ejercicio, la máscara tiene el valor 174. El  $1_8$  previene que se establezca el bit de ejecución, el  $7_8$  previene que se establezca cualquier bit del grupo, y el  $4_8$  previene que se establezca el bit de lectura para otros.

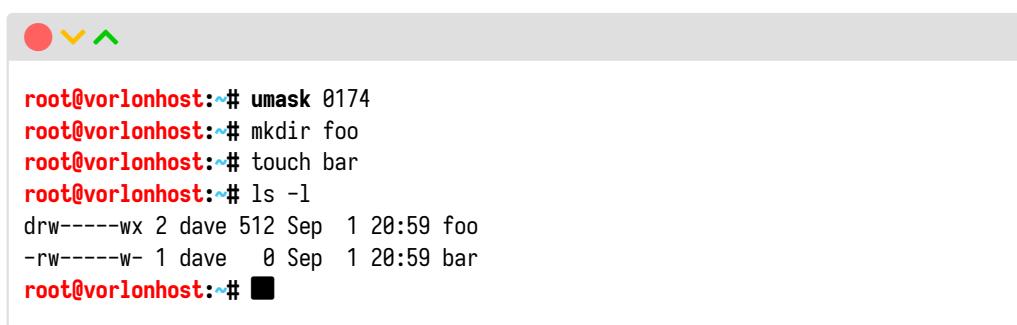
Cualquier nuevo archivo se creará con los permisos 602 y cualquier nuevo directorio tendrá permisos 603. ¿Cómo se verifica?

- 1 El modo para los archivos por defecto es 666, y se pasa al sistema binario para determinar que permisos implica  $\rightarrow 666_8 = (110\ 110\ 110)_2$

## 5

- 2 Se necesita aplicar la máscara del correspondiente binario de 174, ergo, hay que pasarlo a binario  $\rightarrow 174_8 = (001\ 111\ 100)_2$
- 3 Ahora se aplica la máscara 174, para saber que tipo de permisos efectivos le queda a cada archivo creado. Al efectuar la operación, se verá que los permisos para cada archivo nuevo, es 602.  $\rightarrow 666_8 \wedge \neg 174_8 = 602$ ,  $\neg(001\ 111\ 100)_2 = (110\ 000\ 011)_2$  y  $(110\ 110\ 110)_2 \wedge (110\ 000\ 011)_2 = (110\ 000\ 010)_2$ .
- 4 Para los directorios, el modo por defecto es 777, nuevamente se pasa a binario  $\rightarrow 777_8 = (111\ 111\ 111)_2$  y  $174_8 = (001\ 111\ 100)_2$ .
- 5 Repitiendo la operación, se resuelve que el modo asignado es 603  $\rightarrow 777_8 \wedge \neg(174)_8 = 603_8$ .
- 6 Se niega en binario a 174  $\rightarrow \neg(001\ 111\ 100)_2 = (110\ 000\ 011)_2$
- 7 Se realiza la operación y se obtiene 603 en binario  $\rightarrow (111\ 111\ 111)_2 \wedge (110\ 000\ 011)_2 = (110\ 000\ 011)_2$

Probándolo en la terminal:



```

root@vorlonhost:~# umask 0174
root@vorlonhost:~# mkdir foo
root@vorlonhost:~# touch bar
root@vorlonhost:~# ls -l
drw----wx 2 dave 512 Sep  1 20:59 foo
-rw----w- 1 dave    0 Sep  1 20:59 bar
root@vorlonhost:~# █

```

**6****Gestión de Procesos y alias****6.1 TAREAS Y PROCESOS**

Cada vez que se ejecuta un programa, se lanza lo que se conoce como «proceso», que es simplemente el nombre que se le da a un programa cuando se está ejecutando en memoria, es decir, una instancia de un programa.

**6.1.1 CICLO DE VIDA DE UN PROCESO**

Al igual que muchos asuntos en la vida, el proceso también tiene un ciclo de vida, comienza, pasa por diferentes estados y culmina.

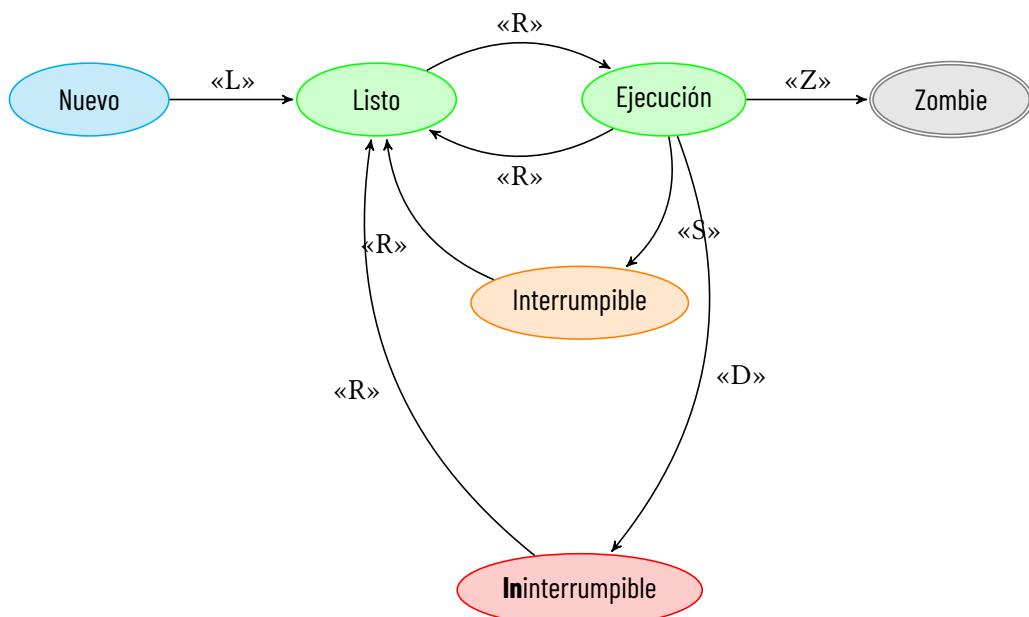


FIGURA 6.1: ESTADOS DE LOS PROCESOS EN LINUX.

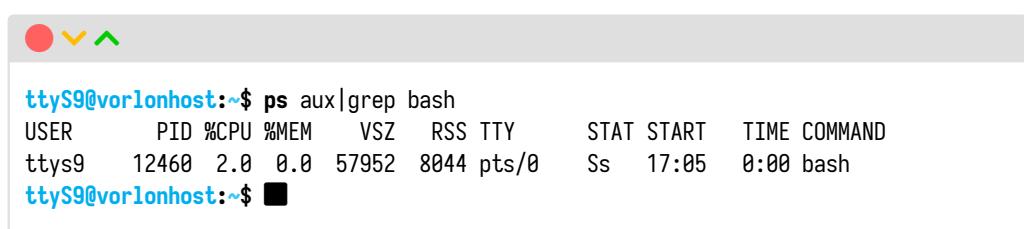
Se puede obtener una lista detallada de procesos con `ps`. En la columna STAT, se obtiene el estado actual del proceso:

Estos estados descritos, pueden a su vez estar acompañados por otros códigos, brindando mas información:

Por ejemplo:

Estado	en ps	Descripción
Bloqueo ininterrumpible	D	En este estado el proceso no puede ser eliminado, y si se intenta realizar otra operatoria sobre el recurso I/O que tiene a este proceso asignado, el otro comando también va a quedar en estado «D», haciendo imposible la cancelación. Si el recurso I/O no se libera, la única manera de recuperar el control es reiniciando la computadora.
En ejecución o ejecutable	R	El proceso se encuentra en la cola de ejecución, esperando que se le asigne tiempo de ejecución, para ser ejecutado, o ya se encuentra siendo ejecutado.
Bloqueo interrumpible	S	El proceso se encuentra esperando un evento del usuario. Este tipo de procesos pueden cancelarse con <b>[Ctrl]+[C]</b> .
Suspendido	T	El proceso está detenido por una señal de administración de tareas ( <b>[Ctrl]+[Z]</b> ).
En debugging	t	El proceso se encuentra detenido por un debugger durante el debugging.
Proceso finalizado	X	El proceso se encuentra finalizado o muerto (difícil que llegue a verse este estado).
Zombie	Z	Los procesos que se encuentran en este estado, no están realizando tarea alguna, sin embargo consume recursos computacionales. Esta terminado, pero su proceso padre no le avisó al kernel, para que libere los recursos asignados a éste.

CUADRO 6.1: DESCRIPCIÓN DE LOS ESTADOS DE LOS PROCESOS EN LINUX.



```

ttyS9@vorlonhost:~$ ps aux|grep bash
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
ttsys9   12460  2.0  0.0  57952  8044 pts/0    Ss   17:05  0:00 bash
ttyS9@vorlonhost:~$ █

```

Analizando solo la columna STAT, se ve que el shell asignado esta esperando un evento «S» (el teclado), y que es líder en su sesión «s», lo cual es cierto porque es el shell.

### 6.1.2 COLUMNAS DE ps

En la tabla 6.3 en página 120, se describen las columnas que usualmente aparecen<sup>1</sup> en la salida del comando **CMD ps aux**.

<sup>1</sup>Mediante la opción «-o» se puede configurar que aparezcan mas columnas.

Estado	Descripción
«<>	Alta prioridad.
«N»	Baja prioridad.
«L»	Tiene páginas bloqueadas en memoria.
«S»	Proceso líder en la sesión.
«l»	Es multithread.
«+»	Es un proceso en el grupo de primer plano.

CUADRO 6.2: CÓDIGOS ADICIONALES A LOS ESTADOS DE LOS PROCESOS EN LINUX.

### 6.1.3 ¿PROCESOS O TAREAS?

Un proceso que esta corriendo se denomina tarea para el shell. Los términos «proceso» y «tarea», son intercambiables. Sin embargo, se suele denominar «tarea» a un «proceso», cuando se usa en conjunción con control de tareas, que es un rasgo del shell el cual permite cambiar entre distintas tareas.

## 6.2 PRIMER PLANO Y SEGUNDO PLANO

En muchos casos, los usuarios solo ejecutan un trabajo a la vez, que es el último comando que escribieron desde el shell. Sin embargo, se podrán ejecutar diferentes tareas al mismo tiempo, cambiando entre cada uno de ellos conforme se necesite. ¿Cuán beneficioso puede llegar a ser esto? Se supone que se está con un procesador de textos, y de repente se necesita parar y realizar otra tarea. Se podrá suspender temporalmente el editor, y volver al shell para realizar cualquier otra tarea, y luego regresar al editor como si no se lo hubiese dejado nunca.

### 6.2.1 UN PROCESO PUEDE ESTAR EN PRIMER PLANO O EN SEGUNDO PLANO



Solo puede haber un proceso en primer plano al mismo tiempo, en la misma sesión de usuario.

El proceso que está en primer plano, es el que interactúa con el usuario, recibe entradas de teclado, y envía las salidas al monitor (salvo, por supuesto, que haya redirigido la entrada o la salida). El proceso en segundo plano, no recibe ninguna señal desde el teclado y por lo general se ejecuta en silencio sin necesidad de interacción.

Algunos programas necesitan mucho tiempo para terminar. Compilar programas es una de estas tareas, así como comprimir un fichero grande. No tiene sentido que el usuario se siente y se quede viendo mientras estos procesos terminan. En estos casos es mejor lanzarlos en segundo plano, para dejar la consola en condiciones de ejecutar otro programa.

Estado	Descripción
USER	Usuario dueño del proceso. El proceso heredará los permisos del usuario que lo ejecutó.
PID	«Identificador único de proceso». Es un número único por el cual los procesos se identifican.
%CPU	Utilización del CPU del proceso. Actualmente, es el tiempo de CPU utilizado, dividido por el tiempo que el proceso ha estado ejecutándose, (relación tiempo de cpu / tiempo real), expresado como un porcentaje.
%MEM	Relación entre el tamaño del conjunto residente del proceso y la memoria física en la máquina, expresada como un porcentaje.
VSZ	Tamaño de memoria virtual <sup>2</sup> del proceso en KiB (unidades de 1024 bytes). Las asignaciones de dispositivos están actualmente excluidas; Esto puede cambiar.
RSS	Tamaño residente, la memoria física no intercambiada (es decir no páginada a disco) que un proceso está utilizando (en kiloBytes).
TTY	Terminal desde donde se lanzó.
STAT	Estado del proceso.
START	Hora en que se lanzó el comando. Si el proceso se inició hace menos de 24 horas, el formato de salida es «HH:MM:SS», de lo contrario es «Mmm dd» (donde «Mmm» es un nombre de mes de tres letras).
TIME	Tiempo acumulado de CPU, en formato «[DD-]HH:MM:SS».
COMMAND	Nombre del comando (solo el nombre del ejecutable). Un proceso marcado «defunct» está parcialmente muerto, esperando ser completamente destruido por su padre. Cuando se especifica en último lugar, esta columna se extenderá hasta el borde de la pantalla. Si <code>CMD ps</code> no puede determinar el ancho de visualización, como cuando la salida se redirige (canaliza) a un archivo u otro comando, el ancho de salida es indefinido (puede ser 80, ilimitado, determinado por la variable TERM, y así sucesivamente). La variable de entorno «COLUMNS» o la opción « --cols» se pueden usar para determinar exactamente el ancho en este caso.

CUADRO 6.3: COLUMNAS DE ps.

## 6.2.2 LOS PROCESOS SE PUEDEN SUSPENDER

Un proceso suspendido es aquel que no se está ejecutando actualmente, sino que está temporalmente parado. Después de suspender una tarea, se puede indicar a la misma que continúe, en primer plano o en segundo, según se necesite. Retomar una tarea suspendida no cambia en nada el estado de la misma, la tarea continuará ejecutándose justo donde se dejó.

## 6.2.3 SUSPENDER UN TRABAJO NO ES LO MISMO QUE INTERRUMPIRLO

Cuando se interrumpe un proceso (generalmente con la pulsación de **Ctrl**+**C**), el proceso muere, deja de estar en memoria y utilizar recursos del equipo. Una vez eliminado, el proceso no puede continuar ejecutándose, y deberá ser lanzado otra vez para volver a realizar sus tareas.

También se puede dar el caso de que algunos procesos capturan la interrupción, de modo que pulsando **Ctrl**+**c** no se detienen inmediatamente. Esto se hace para permitir al proceso realizar operaciones necesarias de limpieza antes de terminar. De hecho, algunos procesos simplemente no se dejan matar por ninguna interrupción.

Una tarea detenida es una tarea que no se está ejecutando, es decir, que no usa tiempo de CPU, y que no está haciendo ningún trabajo (la tarea aún ocupa un lugar en memoria, aunque puede ser volcada a disco). Una tarea en segundo plano, se está ejecutando, y usando memoria, a la vez que completando alguna acción mientras se hace otro trabajo. Sin embargo, una tarea en segundo plano puede intentar mostrar texto en la terminal, lo que puede resultar molesto si se está intentando hacer otra cosa.

Por ejemplo, si se usó **STI yes**:

**CMD** **yes &**

Sin redirigir stdOut a /dev/null, una cadena de yes se mostraran en el monitor, sin modo alguno de interrumpirlo (no se puede hacer uso de **Ctrl**+**c** para interrumpir tareas en segundo plano). Para poder parar esas interminables yes, se tendría que usar el comando **CMD fg** para pasar la tarea a primer plano, y entonces usar **Ctrl**+**c** para matarla, o escribir ciegamente «**CMD killall yes**».



No se pueden usar los ID de proceso con **CMD fg** o **CMD bg**.

Otra observación: normalmente, **CMD fg** y **CMD bg** actúann sobre el último proceso parado (indicado por un «+» junto al número de tarea cuando se usa **CMD jobs**). Si se tienen varios procesos corriendo a la vez, se podrá mandar a primer o segundo plano una tarea específica indicando el ID de tarea como argumento de **CMD fg** o **CMD bg**, como en:

**CMD fg %2**

para la tarea de primer plano número 2, o

## 6

**CMD** bg %3

para la tarea de segundo plano número 3.



Recordar que el uso de control de tareas es una utilidad del shell. Los comandos **CMD fg**, **CMD bg** y **CMD jobs** son internos del shell. Si por algún motivo se utiliza un shell que no soporta control de tareas, existe la probabilidad de que no se disponga de estos comandos.

### 6.3 ENVÍO A SEGUNDO PLANO Y ELIMINACIÓN DE PROCESOS

El comando **CMD yes** es un comando aparentemente sin mucha utilidad que envía una serie interminable de «yes» a la salida estándar. (Realmente es muy útil; si se utiliza un «pipe» para unir la salida de **CMD yes** con otro comando que haga preguntas del tipo si/no, la serie de «yes» confirmará todas las preguntas).

Por ejemplo:

```
ttyS9@vorlonhost:~$ yes
y
y
y
y
```

La serie de «yes» continuará hasta el infinito, a no ser que se la elimine, pulsando la tecla de interrupción, generalmente **Ctrl**+**C**. También uno se puede deshacer de esta serie de «yes» redirigiendo la salida estándar de **CMD yes** hacia `/dev/null`.

**CMD yes > /dev/null**

La terminal no se llena de «yes», pero el prompt de la shell no retorna. Esto es porque **CMD yes** sigue ejecutándose y enviando esos yes a `/dev/null`. Para recuperarlo, se debe pulsar la tecla de interrupción<sup>3</sup>.

Se supone ahora que se quiere dejar que el comando **CMD yes** siga ejecutándose, y volver al mismo tiempo al shell para trabajar en otras cosas. Para ello se enviará a **CMD yes** a segundo plano, lo que permitirá ejecutarlo, pero sin necesidad de interacción. Una forma de mandar procesos a segundo plano es añadiendo un carácter **&** al final de cada comando.

<sup>3</sup>Sobre todo aquellos que tienen un operación de I/O pendiente.



```
ttyS9@vorlonhost:~$ yes > /dev/null &
[1] 164
ttyS9@vorlonhost:~$ █
```

Como se podrá ver, se dispone nuevamente del prompt. ¿Pero qué es «[1] 164»?, ¿Se está ejecutando realmente el comando **CMD yes**?

«[1]» representa el número de tarea del proceso **CMD yes**. El shell asigna un número a cada tarea que se esté ejecutando. Como **CMD yes** es el único comando que se está ejecutando, se le asigna el número de tarea 1. El número 164 es el número de identificación del proceso, o PID, que es el número que el sistema le asigna al proceso. Ambos números pueden usarse para referirse a la tarea como se verá mas adelante.

Ahora se tiene el proceso **CMD yes** corriendo en segundo plano, y enviando constantemente la cadena «yes» hacia el dispositivo `/dev/null`. Para chequear el estado del proceso, se utiliza el comando interno de la shell **CMD jobs**:



```
ttyS9@vorlonhost:~$ jobs
[1]+ Running yes >/dev/null &
ttyS9@vorlonhost:~$ █
```

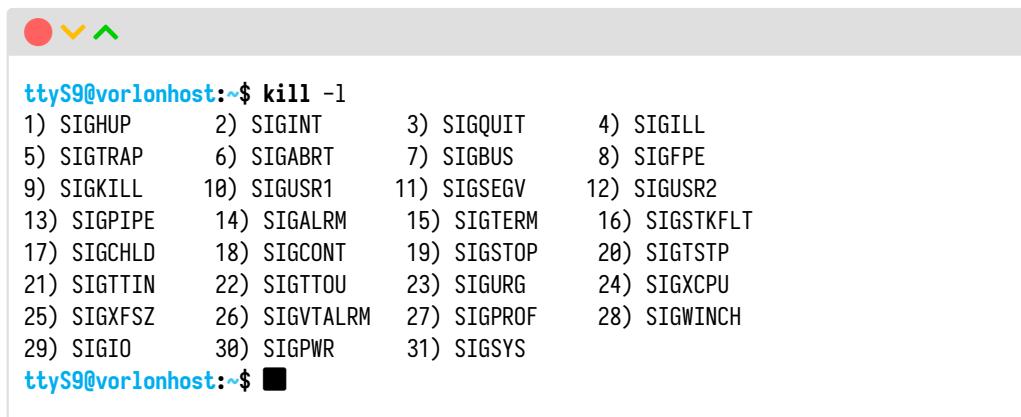
También se puede usar el comando **CMD ps**, como se indicó antes, para comprobar el estado de la tarea.

### 6.3.1 COMANDO kill

**CMD kill**, que literalmente quiere decir matar, sirve no sólo para matar o terminar procesos sino principalmente para enviar señales (signals) a los procesos. La señal por default, es decir, cuando no se indica ninguna, es terminar o matar amablemente el proceso, y la sintaxis es:

**SIN kill <PID>**

Así por ejemplo, es posible enviar, no sólo la señal de finalización SIGTERM, sino también una señal de STOP al proceso y se suspenderá su ejecución, después cuando se quiera mandar una señal de CONTinuar, el proceso continuara desde donde se quedó.



```
ttyS9@vorlonhost:~$ kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL
 5) SIGTRAP     6) SIGABRT     7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1    11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD    18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN    22) SIGTTOU    23) SIGURG      24) SIGXCPU
25) SIGXFSZ    26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGIO      30) SIGPWR      31) SIGSYS
ttyS9@vorlonhost:~$
```

Las señales más comunes son la «19» y «20» que detienen momentáneamente la ejecución de un proceso o programa, «18» la continua, «1» que es la señal de «hang up» que obliga al proceso a releer sus archivos de configuración estando en ejecución y «9» que termina rotundamente un proceso.

### El comportamiento es programado (casi...)



Tener presente que a excepción de la señal SIGKILL (-9), que la trata el kernel, el comportamiento de los procesos respecto a las señales es **programado previamente**, tanto sea por quien desarrolló las bibliotecas de C, ej. SIGTERM, o por el programador del proceso.

Para una forma amable de detención (**[Ctrl]+[Z]**) del proceso, se usa la señal SIGTSTP.<sup>4</sup>

**SIN** `kill -TSTP [pid]`

Para una forma más rotunda de detención (ya que no puede ser ignorada), se usa la señal SIGSTOP.

**SIN** `kill -STOP [pid]`

Para continuar con la ejecución del proceso, se envía la señal SIGCONT.

**SIN** `kill -CONT [pid]`

Este comando toma como argumento **un número de tarea o un número de ID de un proceso**. Esta era la tarea 1, así que se usa:

**CMD** `kill %1`

Cuando se identifica la tarea con el número de tarea, se debe preceder el número con el carácter de porcentaje %. Ahora que ya se ha matado la tarea, se puede usar **CMD** `jobs` de nuevo

---

<sup>4</sup>Es más usual que algunos utilitarios tengan rutinas específicas para el tratamiento de esta señal. Por ejemplo SCP

para comprobarlo:



```
ttyS9@vorlonhost:~$ jobs
[1]+ Terminated yes >/dev/null
ttyS9@vorlonhost:~$ █
```

La tarea está, en efecto, muerta, y si de utilizarse **jobs** de nuevo, no mostrará nada. También se podrá matar la tarea usando el número de ID de proceso (PID), el cuál se muestra conjuntamente con el ID de tarea cuando arranca la misma. En el ejemplo el ID de proceso es 164, así que el comando:



```
ttyS9@vorlonhost:~$ kill 164
Terminated.
ttyS9@vorlonhost:~$ █
```

### 6.3.2 LA IMPORTANCIA DE LA PLANIFICACIÓN DE PROCESOS

A continuación Carta de Margaret H. Hamilton [[wiki:margaret](#)], Directora de la programación de la computadora de vuelo del Apolo. Laboratorio MIT, Cambridge, Massachusetts [[article:margaret\\_apollo](#)]:

“ La computadora (o mejor dicho, el software en ésta), fue lo suficientemente inteligente para reconocer que se le estaba pidiendo realizar más tareas que las que debería realizar. Envío una alarma, que le informaba al astronauta: «estoy sobrecargada con más tareas de las que podría realizar ahora, ergo, solo me voy a quedar con las más importantes, por ejemplo, las de aterrizaje». En realidad, la computadora fue programada para hacer más que reconocer condiciones de error. Se incorporó un completo conjunto de programas de recuperación. La acción del software en este caso, fue eliminar las tareas de baja prioridad y reactivar las de alta. Si la computadora no hubiera reconocido este problema y tomado las acciones de recuperación en su momento, dudo que el Apolo XI, haya podido haber aterrizado satisfactoriamente. ”

## 6.4 NICE Y PRIORIDAD

Antes tenemos que aclarar que el kernel o núcleo del sistema operativo, organiza la memoria en dos grandes partes. Una es conocida como el espacio de kernel o «kernel space», y la otra como

espacio de usuario o «userspace». En el espacio de kernel, sin entrar en detalle, se puede asegurar que viven los procesos que son nativos del kernel. En el espacio de usuario, por el contrario, viven todos aquellos procesos que son de algún usuario del sistema operativo (sea root o cualquier otro)

### Prioridad es «niceness»



Al opuesto de lo que se puede suponer como usuarios, lo que se entiende por prioridad del proceso, en GNU/Linux, es el nivel de «niceness».

El valor «nice» es de espacio de usuario y la prioridad PR es la prioridad real del proceso que utiliza el kernel de Linux. En GNU/Linux, las prioridades del sistema son de 0 a 139. De 0 a 99 para tiempo real y de 100 a 139 para usuarios. El intervalo de valores «nice» es de -20 a +19, donde -20 es el más alto, 0 predeterminado y +19 es el más bajo. La relación entre «nice» y prioridad es:  $Pr = 20 + Ni$ , lo que se traduce en  $Pr = 20 + Ni, Ni \in [-20, +19]$ , esto da una abanico de 39 niveles de prioridad. Como el kernel, reserva los primeros niveles de prioridad para los sistemas de tiempo real (de baja latencia), el nivel de prioridad en realidad es  $Pr = 120 + Ni, Ni \in [-20, +19]$  [procesos:nicert].

#### 6.4.1 EL COMANDO nice

Sin argumentos, devuelve la prioridad por defecto:

```
ttyS9@vorlonhost:~$ nice
0
ttyS9@vorlonhost:~$ █
```

Inicia el comando con una prioridad de -5, lo que le da más tiempo de CPU:

**SIN** `nice -n -5 <comando>`

### Prioridades de «tiempo real»



Las prioridades reales desde la 0 a la 99 son prioridades de **tiempo real** (RT), no pueden ser modificadas con herramientas de usuario. En **CMD top** o **CMD htop** se las verá con prioridad «RT».

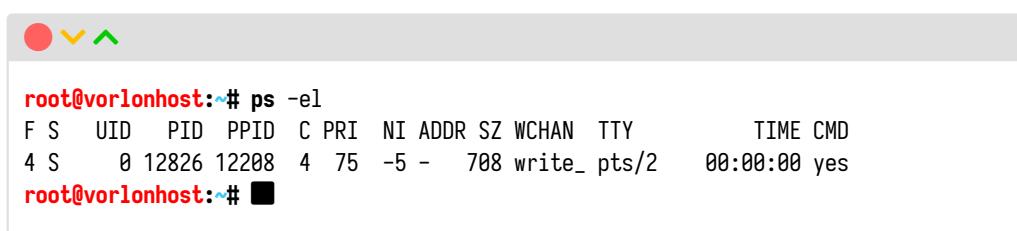
### 6.4.2 EL COMANDO renice

Así como `nice` establece la prioridad de un proceso cuando se inicia su ejecución, `renice` permite alterarla en tiempo real, sin necesidad de detener el proceso.

Se ejecuta el programa `yes` con prioridad -5

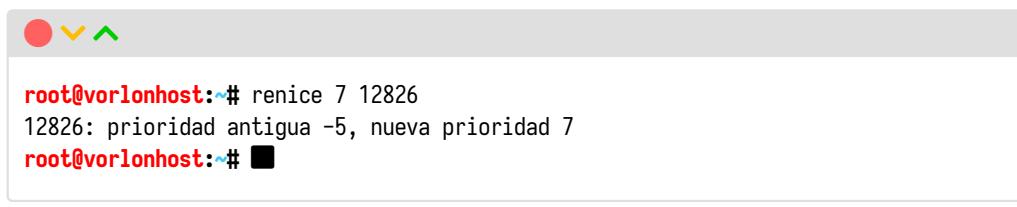
```
CMD nice -n -5 yes >/dev/null &
```

Se deja ejecutando `yes` y en otra terminal se analiza con `ps`



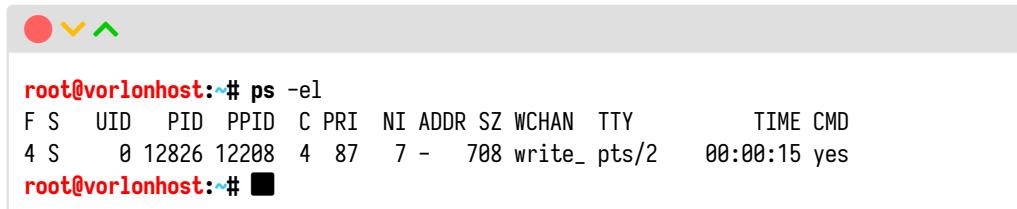
```
root@vorlonhost:~# ps -el
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN TTY          TIME CMD
4 S     0 12826 12208  4  75   -5 -    708 write_ pts/2    00:00:00 yes
root@vorlonhost:~#
```

Se reprioriza el proceso.



```
root@vorlonhost:~# renice 7 12826
12826: prioridad antigua -5, nueva prioridad 7
root@vorlonhost:~#
```

Se verifica



```
root@vorlonhost:~# ps -el
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN TTY          TIME CMD
4 S     0 12826 12208  4  87   7 -    708 write_ pts/2    00:00:15 yes
root@vorlonhost:~#
```

### 6.5 DESCONECTAR UN PROCESO DE LA TERMINAL

Supongase que se inicia una sesión remota de terminal, y se ejecuta un proceso que llevará un tiempo largo, ej. compresión de archivo, copia de archivos de tamaño considerable, borrado seguro, etc. Puede ocurrir que se corte la conexión, dando un resultado incierto al proceso, ya que puede o no cancelar el proceso lanzado, es aquí donde entra el comando `disown`.

**disown**



**CMD** **disown** permite que el comando lanzado continúe su ejecución, sin importar de que se cierre la terminal.



```
ttyS9@vorlonhost:~$ gzip archivomuymuygrande.txt &
[1] 31937
ttyS9@vorlonhost:~$ disown %1
ttyS9@vorlonhost:~$ █
```

## 6.6 OTROS COMANDOS RELACIONADOS

En el cuadro 6.4, se listan algunos comandos relacionados con el manejo de procesos.

Comando/s	Descripción
top	Muestra los procesos que más recursos del sistema utilizan. Se actualiza periódicamente.
pstree	Muestra los procesos mediante un árbol, ilustrando las relaciones padre-hijo.
skill / killall <sup>5</sup> / pkill	Igual que kill pero para un grupo de procesos.
fuser	Identifica qué procesos (PID) utilizan un archivo determinado.
pgrep	Listado selectivo de procesos, basado en los nombres de los mismos y en otros tipos de atributos.

CUADRO 6.4: OTROS COMANDOS RELACIONADOS

## 6.7 alias

**alias**



Como su nombre lo indica, es útil para reemplazar un comando o serie de comandos con otro más corto y sencillo

Un ejemplo práctico, si se desea ver los «logs» del sistema, sería usar **CMD colorizer** o **CMD ccze**, los cuales se encargan de colorear el resultado en consola. La línea podría ser:

```
ttyS9@vorlonhost:~$ sudo tail -n 2 /var/log/syslog | ccze -A
Jan 28 08:07:15 vectorsigma anacron[12391]: Job 'cron.daily' terminated
Jan 28 08:07:15 vectorsigma anacron[12391]: Normal exit (1 job run)
ttyS9@vorlonhost:~$ █
```

Sin embargo sería más sencillo, en vez de escribir toda esa línea, que se escribiera en la consola por ejemplo, «**CMD syslog**».

### 6.7.1 CREANDO UN ALIAS

Crear un alias es sencillo. La sintaxis es:

**SIN** alias palabra\_corta='comando o palabras a reemplazar'

**CMD** alias syslog='sudo tail -n 5 /var/log/syslog | ccze -A'

si se toma el ejemplo anterior, es:

```
ttyS9@vorlonhost:~$ alias syslog='sudo tail -n 5 /var/log/syslog | ccze -A'
ttyS9@vorlonhost:~$ syslog
[sudo] contraseña para ttys9:
Jan 28 19:27:31 vectorsigma wpa_supplicant[1809]: wlp8s0: CTRL-EVENT-CONNECTED -
    ↳Connection to 84:b2:62:66:98:02 completed [id=0 id_str=]
Jan 28 19:27:31 vectorsigma wpa_supplicant[1809]: wlp8s0:
    ↳CTRL-EVENT-SIGNAL-CHANGE above=0 signal=-49 noise=9999 txrate=11000
Jan 28 19:27:31 vectorsigma NetworkManager[1786]: <info>  [1580250451.3893]
    ↳device (wlp8s0): supplicant interface state: associating -> completed
Jan 28 19:27:31 vectorsigma wpa_supplicant[1809]: wlp8s0:
    ↳CTRL-EVENT-SIGNAL-CHANGE above=1 signal=-50 noise=9999 txrate=72200
Jan 28 19:27:31 vectorsigma kernel: [14248.508679] wlp8s0: Limiting TX power to 2
    ↳dBm as advertised by 84:b2:62:66:98:02
ttyS9@vorlonhost:~$ █
```

El comando va escrito entre comillas simples.

## Duración de un alias



Si se define directamente en la consola, la definición durará hasta que se salga de la sesión. Para una mayor persistencia, ver la sección 6.7.3.

### 6.7.2 VERIFICANDO LOS ALIAS

Para verificar un solo alias, se realiza por medio del mismo comando de la siguiente manera:

**SIN** alias <nombre del alias> ó **SIN** alias | grep <nombre del alias>

por ejemplo

**CMD** alias syslog ó **CMD** alias | grep syslog

Para listar todos los comandos, se debe escribir únicamente el comando sin ningún otro parámetro adicional:

**CMD** alias

### 6.7.3 PERSISTENCIA DEL ALIAS

Ahora, si se quiere de forma permanente, se escribe lo mismo dentro del archivo `~/.bashrc` el cual está en el directorio del usuario. Si no está, se lo crea (siempre con el punto delante).

Se le añade la linea tal cual se escribe en la terminal: `alias syslog='sudo tail -n 5 /var/log/syslog | ccze -A'`

#### .bashrc

```
1 alias ll='ls -alF'
2 alias la='ls -A'
3 alias l='ls -CF'
4 alias syslog='sudo tail -n 5 /var/log/syslog | ccze -A'
```

Cuando ya se tenga añadida la línea del alias en este archivo, simplemente se escribe en consola:

**CMD** source /.bashrc

Y se podrá verificar, sin salir de la sesión, si quedó bien configurado el alias.

# 7

# Manejo de Filesystems y Links

## 7.1 INTRODUCCIÓN

### «filesystem» o sistema de archivos



El sistema de archivos o sistema de ficheros es el componente del sistema operativo encargado de administrar y facilitar el uso de las memorias periféricas, ya sean secundarias o terciarias.

Las principales funciones de un filesystem son:

- ▶ Asignar espacio a los archivos.
- ▶ Administrar el espacio libre y el acceso a los datos resguardados.
- ▶ Estructurar la información guardada en un dispositivo de almacenamiento de datos (normalmente el disco duro de una computadora)



La mayoría de los sistemas operativos manejan su propio sistema de archivos.

Los sistemas de archivos proveen métodos para crear, mover, renombrar y eliminar tanto archivos como directorios.

### 7.1.1 ¿QUÉ ES UN FILESYSTEM?

Adicionalmente a las definiciones ya conocidas, se puede definir un filesystem como la manera en que la información se estructura u organiza, en un unidad de almacenamiento (un disco por ejemplo).

Si bien los filesystems varían en como direccionan la información en su interior, una analogía algo exacta (ya que es muy parecida a la usa XFS) sería imaginarse una cinta, con divisiones de

igual tamaño; se supone un tamaño arbitrario de cuatro espacios para caracteres de cualquier tipo y dos para números. Cada división tiene una «dirección» que es única. Figura 7.1.

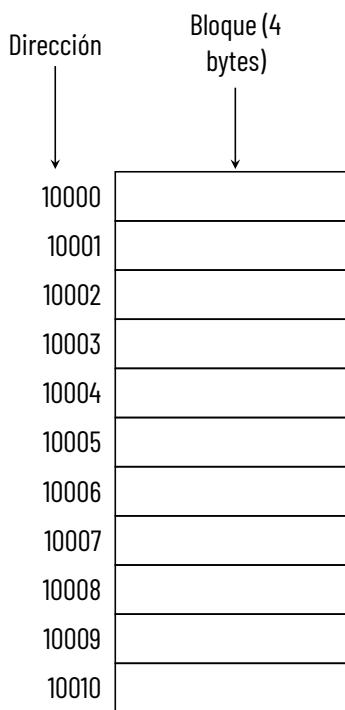


FIGURA 7.1: ESTRUCTURA DE UN FILESYSTEM CONCEPTUAL VACÍO.

Si se quiere guardar un archivo con la palabra «Hola», en principio no habría problema, porque la palabra cabe en el bloque. Figura 7.2 en página 134.

Ahora bien, si se desea almacenar la cadena «Buen Día», se estaría en un problema porque no entraría en el bloque por superar el tamaño de cuatro caracteres. Es entonces que se procede de la siguiente manera: Se guardan en el primer bloque, en la dirección 10000, los primeros cuatro caracteres, y en la dirección del siguiente bloque los restantes cuatro caracteres, en este ejemplo 10003, ya que no tiene porque ser consecutivo<sup>1</sup>. Figura 7.3 en página 134. De esta manera es que se almacenan los archivos en los filesystems «conceptualmente».

### 7.1.2 COMPARATIVA ENTRE EXT4, XFS Y BTRFS

En la tabla 7.1 de la siguiente página, hay se describe una comparación entre los filesystems más utilizados actualmente. EXT4 es la versión 4 del venerable filesystem EXT2, XFS es un filesystem muy probado, creado en los 90s por SGI, una empresa que se encargaba de realizar efectos especiales, ergo XFS es muy estable, creado para archivos de gran tamaño, y de mucha performance. Por otro lado BTRFS es un sistema operativo nuevo, no tan probado pero con características muy útiles como snapshots, compresión, etc.

CARACTERÍSTICA	EXT4	XFS	BTRFS
Arquitectura	BTree Hasheado	BTree+ <sup>2</sup>	Basado en extensiones

<sup>1</sup>Esto se denomina fragmentación de los archivos, y es uno de los problemas que aqueja a los usuarios de Windows.

<sup>2</sup>Es un tipo de estructura de datos de árbol, representa una colección de datos ordenados de manera que se permite una inserción y borrado eficientes de elementos. Es un índice, multinivel, dinámico, con un límite máximo y

	2006	1994	2009
Introducido el			
Tamaño máximo de volumen	1 Ebyte	8 EBytes	16 Ebytes
Tamaño máximo de archivo	16 Tbytes	8 EBytes	16 Ebytes
Máximo número de archivos	4 billones	$2^{64}$	$2^{64}$
Longitud máxima de nombre de archivo	255 bytes	255 bytes	255 bytes
Atributos	Si	Si	Si
Compresión transparente	No	No	Si
Cifrado transparente	Si	No	Planificado
Copy-on-write (COW) <sup>3</sup>	No	Planificado	Si
Snapshots	No	Planificado	Si

CUADRO 7.1: DIFERENCIAS ENTRE LOS DISTINTOS FILESYSTEMS ACTUALES.

## 7.2 FILESYSTEMS «JOURNALIZADOS»

Los filesystems journalizados guardan en forma anticipada los cambios que se van a realizar. Si falla el sistema operativo, o se corta la luz, etc., justo en el instante en que se estaba realizando el cambio, el sistema operativo utiliza la información del journal<sup>4</sup> para actualizar el sistema operativo y reproducir de nuevo el cambio, lo cual sucede generalmente cuando se remonta el filesystem o cuando se verifica la consistencia del mismo (con **CMD** `fsck`).

Filesystems journalizados son por ejemplo: JFS (1990), XFS (1993), EXT3 (2001) y EXT4 (2006) entre otros.

## 7.3 DIRECCIONAMIENTO EN EXT4

En la sección 5.7 en página 105, se vió un modelo de filesystem, a los efectos de introducirse en el concepto de direccionamiento. En esta sección se explicará el direccionamiento con algo más de detalle y tomando en cuenta un filesystem real como EXT4. El ínodo es como un índice, que informa en qué bloque se guarda el contenido del archivo. Ahora bien, el tamaño de un inodo es fijo, lo cual trae como consecuencia directa una limitación en el tamaño máximo de archivo. Debido a esta limitación, se implementaron distintos tipos de direccionamiento a los efectos de maximizar el uso de los ínodos.

mínimo en el número de claves por nodo. Un árbol B+ es una variación de un árbol B.

<sup>3</sup>«copy-on-write» es una política de optimización utilizada en programación. Si múltiples procesos piden recursos que inicialmente son indistinguibles (iguales), se les devuelven punteros al mismo recurso; en el momento en que un proceso intenta modificar su "copia" del recurso, se crea una copia auténtica para prevenir que los cambios producidos por dicho proceso sean visibles por todos los demás. Todo ocurre de forma transparente para los procesos. La principal ventaja de este método es que no se crea ninguna copia adicional del recurso si ningún proceso llega a realizar modificaciones.

<sup>4</sup>Tambien conocido como registro o bitácora.

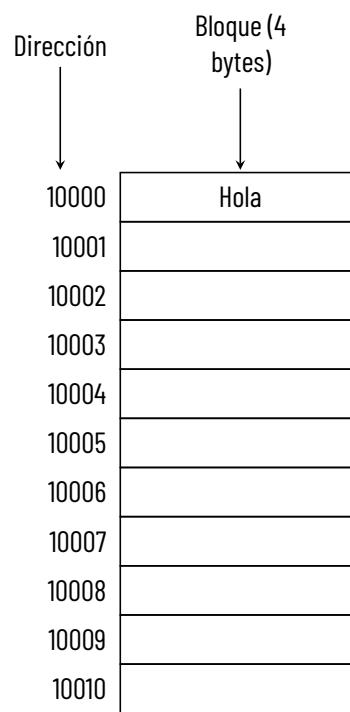


FIGURA 7.2: ESTRUCTURA DE UN FILESYSTEM CONCEPTUAL CON «HOLA».

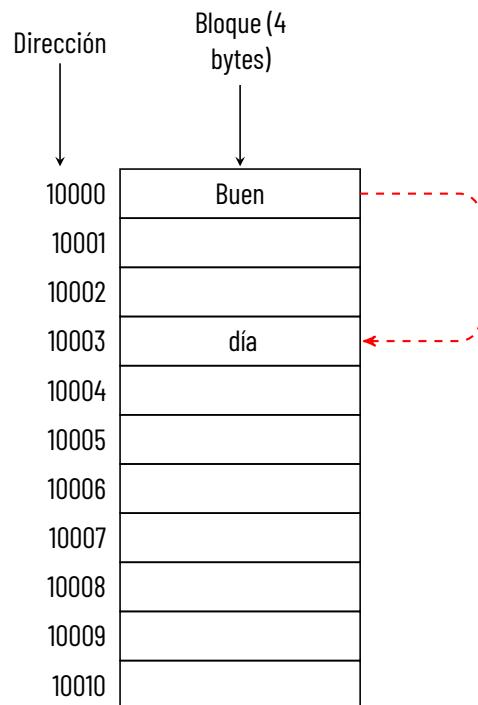


FIGURA 7.3: ESTRUCTURA DE UN FILESYSTEM CONCEPTUAL CON «BUEN DÍA».

### 7.3.1 DIRECCIONAMIENTO DIRECTO EN SISTEMAS DE ARCHIVO EXT4

#### Direccionamiento directo



Un inodo tiene una tabla de direccionamiento de quince entradas. Doce de las quince entradas permiten un **direccionamiento directo** a un bloque de datos del disco duro.

Si por su tamaño, un archivo se almacena en siete bloques, un inodo puede direccionar de forma directa el contenido total del archivo.

### 7.3.2 DIRECCIONAMIENTO INDIRECTOS SIMPLES

Al finalizar el espacio para las doce entradas directas se tendrá que usar un direccionamiento indirecto.

La posición trece de la tabla de direccionamiento del inodo apuntará a un bloque de datos que contendrá una nueva tabla de direcciones hacia los bloques que almacenan el contenido del archivo.

### 7.3.3 DIRECCIONAMIENTO INDIRECTOS DOBLES Y TRIPLES

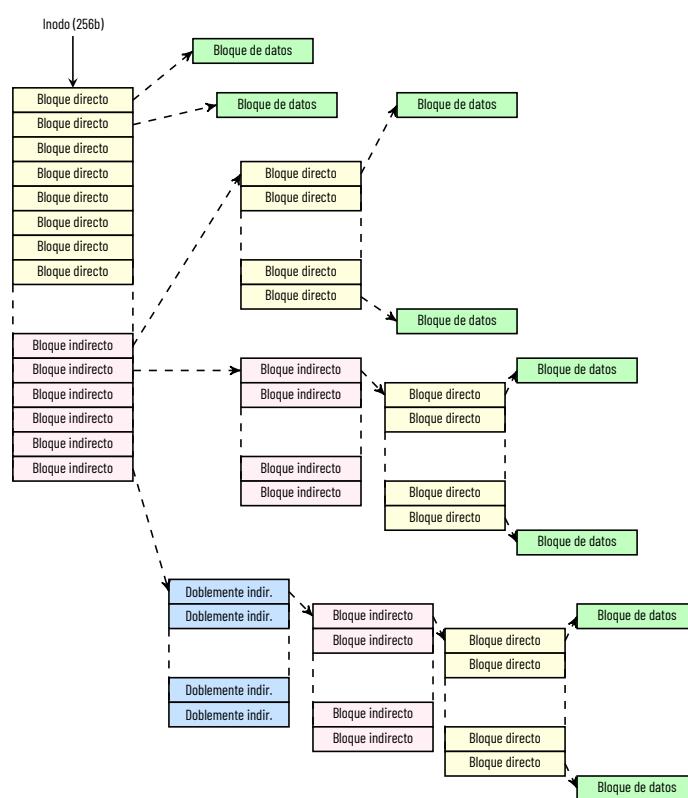


FIGURA 7.4: DIRECCIONAMIENTO EN FILESYSTEMS DE ÍNODOS

Del mismo modo que se hacen direccionamientos indirectos simples, también se pueden hacer direccionamientos indirectos dobles y triples con las entradas 14 y 15, como se muestra en la figura 7.4 en la anterior página.

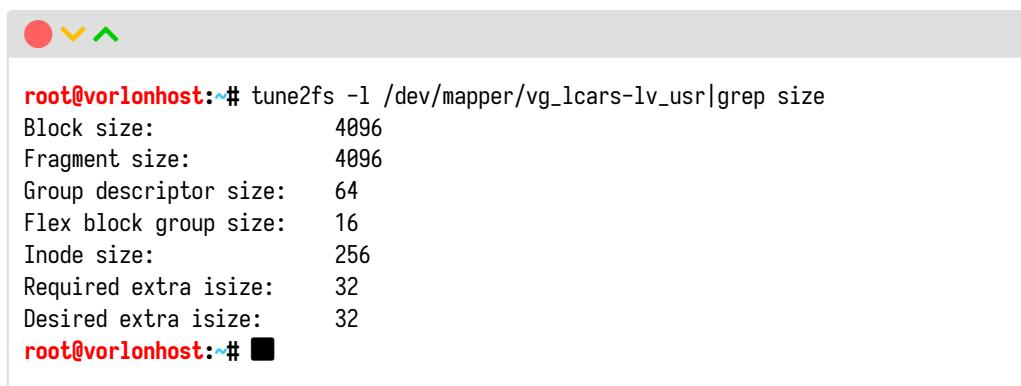
Direccionando de esta forma, se logra conseguir archivos con un tamaño de hasta 16TB en sistemas de ficheros EXT4. Como desventaja, el acceso con direccionamiento doble y triple, vuelve más lento el acceso a los archivos.

### 7.3.4 COMANDOS

#### OBTENER EL TAMAÑO DE INODO Y BLOQUE

Para lograr esto se debe utilizar **tune2fs** que permite obtener (y modificar) parametría interna del filesystem tipo «extN».

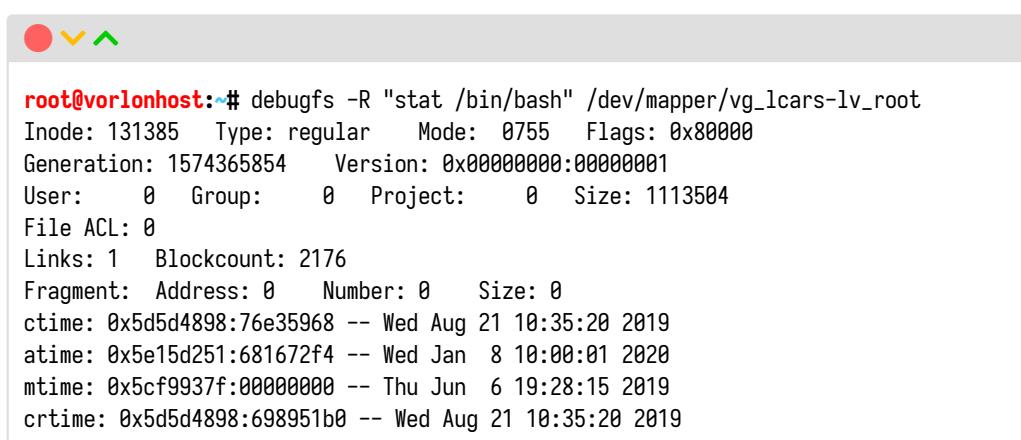
En la salida de este comando, se debe prestar atención a los campos «Block size» y «Inode size».



```
root@vorlonhost:~# tune2fs -l /dev/mapper/vg_lcars-lv_usr|grep size
Block size:          4096
Fragment size:      4096
Group descriptor size: 64
Flex block group size: 16
Inode size:          256
Required extra isize: 32
Desired extra isize: 32
root@vorlonhost:~#
```

#### QUE BLOQUES OCUPA EL ARCHIVO /bin/bash (Y CUANTOS)

Aquí se debe prestar atención a los campos «Inode», que informan en qué inodo se encuentran, y los «EXTENTS», que informa la cantidad de bloques de datos.



```
root@vorlonhost:~# debugfs -R "stat /bin/bash" /dev/mapper/vg_lcars-lv_root
Inode: 131385  Type: regular  Mode: 0755  Flags: 0x80000
Generation: 1574365854  Version: 0x00000000:00000001
User: 0  Group: 0  Project: 0  Size: 1113504
File ACL: 0
Links: 1  Blockcount: 2176
Fragment: Address: 0  Number: 0  Size: 0
ctime: 0x5d5d4898:76e35968 -- Wed Aug 21 10:35:20 2019
atime: 0x5e15d251:681672f4 -- Wed Jan 8 10:00:01 2020
mtime: 0x5cf9937f:00000000 -- Thu Jun 6 19:28:15 2019
crtim: 0x5d5d4898:698951b0 -- Wed Aug 21 10:35:20 2019
```

```
Size of extra inode fields: 32
Inode checksum: 0x1d747c1a
EXTENTS:
(0-271):76288-76559
root@vorlonhost:~#
```

De esta salida se puede obtener el número de ínodo para /bin/bash, es #131385 y que el archivo /bin/bash, ocupa 271 bloques (desde el #76288 al #76559).

#### 7.4 INCORPORAR UN DISCO NUEVO

Montar el dispositivo, asegura que la computadora reconozca el formato del mismo; Si la computadora no puede reconocer el formato, el dispositivo no se puede montar.

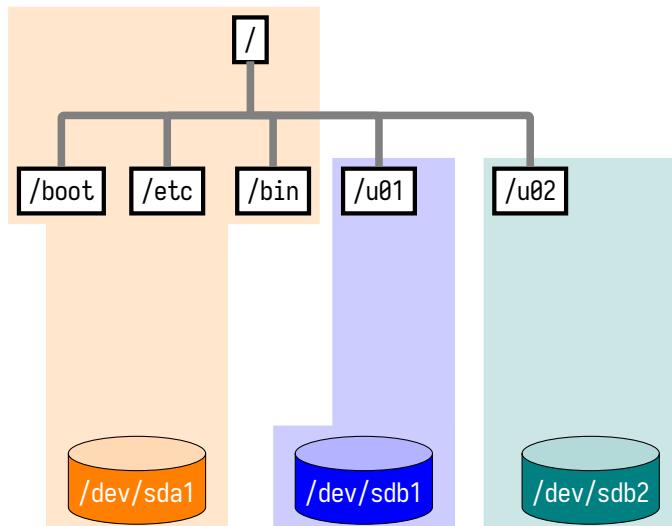


FIGURA 7.5: TRES DISPOSITIVOS MONTADOS.

#### ¿Qué es montar un disco?

 La computadora incorpora el sistema de archivos de los medios a su sistema de archivos local y crea un punto de montaje, un enlace disponible localmente a través del cual se accede al dispositivo. En Windows, el punto de montaje está representado por una letra como «C:»; en \*nix o Linux, el punto de montaje es un directorio.

Generalmente, cuando se desea más espacio, se agrega un disco a un computadora ya operativa. Para poder utilizarlo, «en general» se realizan los siguientes pasos:

- 1 Se lo inicializa y se lo partitiona, es decir, crea la tabla de particiones a fin de cargarle por

lo menos una entrada a la misma. Toda esa información se almacena en el sector cero del disco (los primeros 512 bytes del mismo).<sup>5</sup>

- 2 Se procede a formatear la partición creada, o sea, se crea el filesystem.
- 3 Por último se «monta», es decir, se conecta en forma lógica al Sistema Operativo, y a partir de ese momento, ya es reconocido y puede utilizarse.
- 4 Si se desea que el filesystem este disponible cada vez que el sistema operativo reboote, se debe agregar la entrada correspondiente al archivo /etc/fstab.

## 7.5 PREPARACIÓN DE LA VM

Para poder continuar, primero se debe configurar la máquina virtual, agregando un «disco», y así poder migrar el filesystem /home, al nuevo disco recién agregado.

Los pasos para agregar el disco, a la máquina virtual existente son:

- 1 Diálogo de VirtualBox. Se selecciona la configuración. Figura 7.6.
- 2 Almacenamiento. Figura 7.7 de la siguiente página.
- 3 Se confirma la configuración. Figura 7.8 de la siguiente página.
- 4 Se elige tipo de unidad. Figura 7.9 en página 140.
- 5 Tamaño de la unidad. Figura 7.10 en página 140.
- 6 Tamaño inicial. Figura 7.11 en página 141.
- 7 Configuración finalizada. Figura 7.12 en página 141.

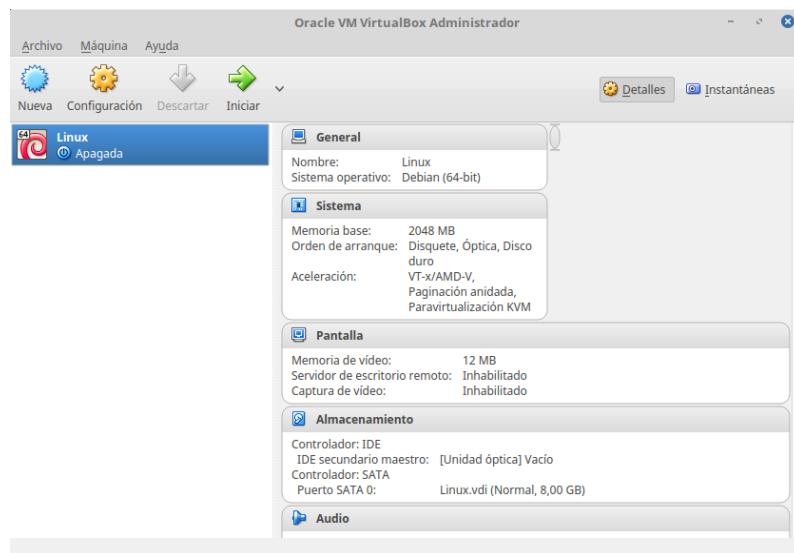


FIGURA 7.6: DIÁLOGO DE VIRTUALBOX. CONFIGURACIÓN.

<sup>5</sup>En un SSD, cada sector es de 2048 bytes.

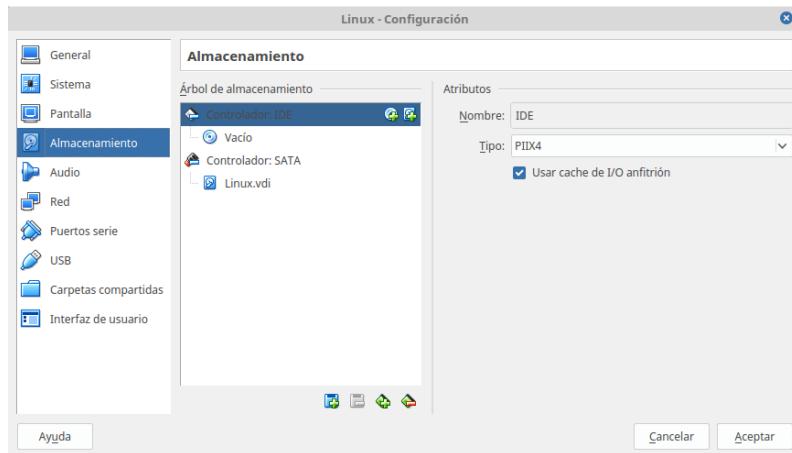


FIGURA 7.7: ALMACENAMIENTO.



FIGURA 7.8: CONFIRMACIÓN.

### 7.5.1 DETECCIÓN DEL NUEVO DISPOSITIVO

Hay dos formas de verificar si el nuevo dispositivo de almacenamiento fue detectado:

- 1 Analizando los logs del kernel y buscando la entrada que indica dicha detección
- 2 Por medio del comando `CMD lsblk`, que lista los dispositivos de bloque que el kernel ve.

#### ANÁLISIS LOS LOGS DEL KERNEL

Para comprender que es lo que se busca, una muy breve introducción (con algo de historia).

Hay varias clases de discos. En este caso, tiene que ver con la manera en la cual el CPU «habla» con el disco:

- ▶ **IDE** → «Integrated Device Electronics» Es un estándar de interfaces para la conexión de dispositivos de almacenamiento masivo de datos (de aquí vienen P •ATA y S •ATA).<sup>6</sup>
- ▶ **SCSI** → «Small Computer System Interface». Es un viejo protocolo de comunicación (1986) de alta performance para discos duros. Sigue vigente actualmente, siendo la interfaz SAS 4.0 su revisión más moderna.

<sup>6</sup>`CMD hdparm` permite modificar ciertos parámetros, aumentando la performance.



FIGURA 7.9: SE ELIGE TIPO DE UNIDAD.



FIGURA 7.10: TAMAÑO DE LA UNIDAD.

Históricamente el driver de discos SCSI siempre fue mas performante que el IDE. Es por eso, que cuando surgieron los discos SATA, se utilizó el driver SCSI.

Se supone que el equipo informático del lector posee discos SATA, entonces, para saber el nombre del dispositivo, se busca en los logs la cadena SCSI.

```
ttyS9@vorlonhost:~$ dmesg|grep SCSI
[    0.524373] Block layer SCSI generic (bsg) driver version 0.4      loaded
[    0.557240] ➞(major 250) SCSI subsystem initialized
[    1.651730] sd 3:0:0:0: [sdb] Attached SCSI disk
[    1.654292] sd 2:0:0:0: [sda] Attached SCSI disk
ttyS9@vorlonhost:~$ █
```

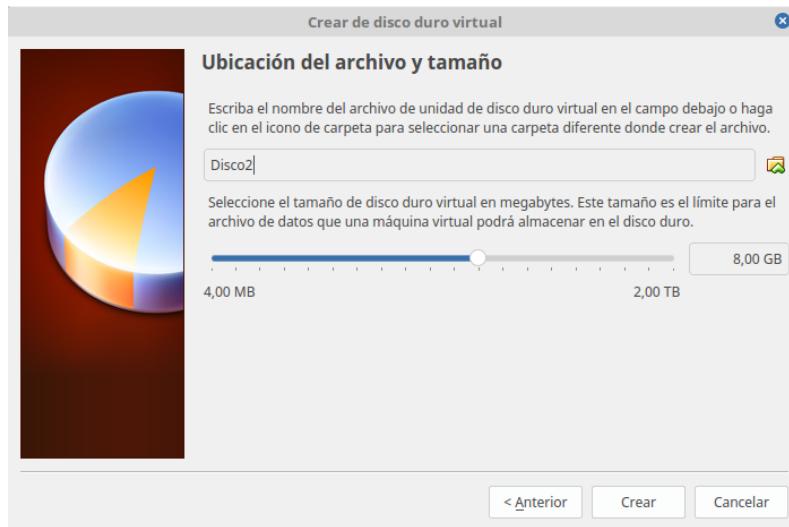


FIGURA 7.11: TAMAÑO INICIAL.

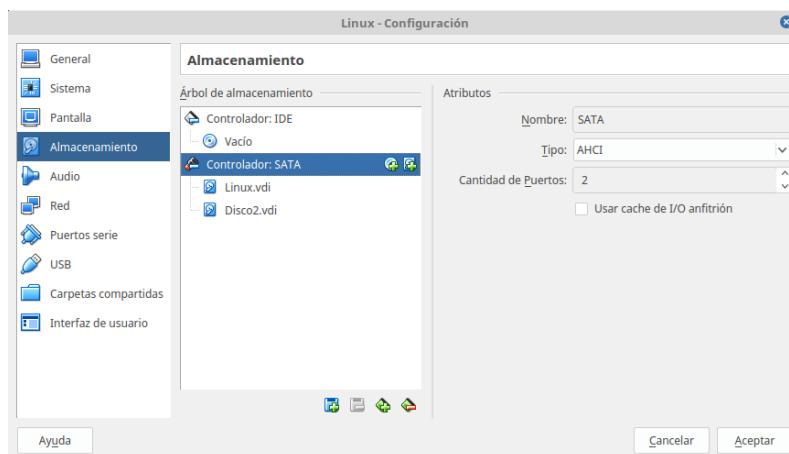


FIGURA 7.12: CONFIGURACIÓN FINALIZADA.

El kernel ve al dispositivo nuevo, en este caso, como «`sdb`», es decir que lo puede ubicar en `/dev/sdb`.

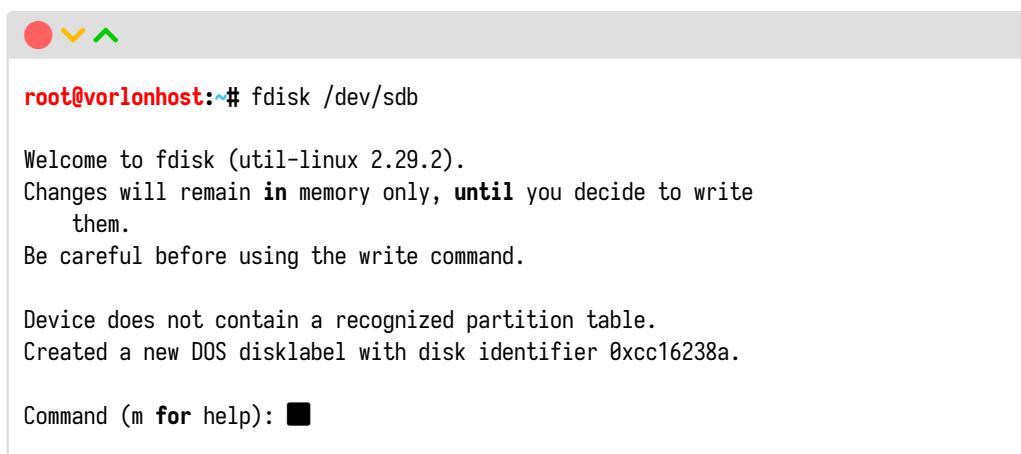
#### UTILIZANDO EL COMANDO `lsblk`

Se puede listar todos los dispositivos de almacenamiento con el comando «`lsblk`».

```
root@vorlonhost:~# lsblk | grep disk
sda                  8:0    0 931,5G  0 disk
sdb                  8:16   0 447,1G  0 disk
root@vorlonhost:~#
```

### 7.5.2 PARTICIONAMIENTO

Una vez conectado físicamente el disco al servidor y luego de haber sido reconocido como dispositivo por parte del Kernel, se procede a su inicialización y particionado con el comando «**CMD fdisk**».



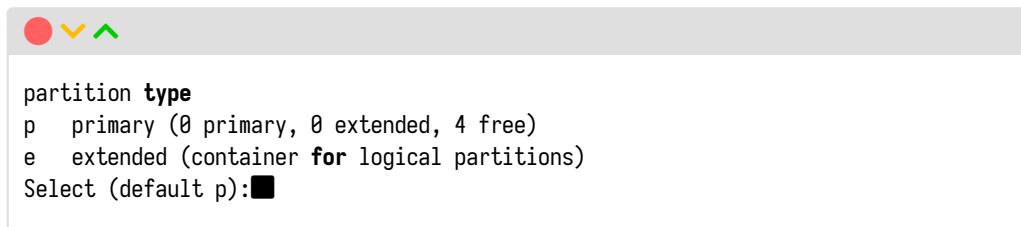
```
root@vorlonhost:~# fdisk /dev/sdb

Welcome to fdisk (util-linux 2.29.2).
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 0xcc16238a.

Command (m for help):
```

Se presiona **n** y luego .



```
partition type
p primary (0 primary, 0 extended, 4 free)
e extended (container for logical partitions)
Select (default p):
```

Se presiona **p** y luego .



```
Select (default p): p
Partition number (1-4, default 1):
```

Se presiona **1** y luego .

```
● ▼ ▲  
Partition number (1-4, default 1): 1  
First sector (2048-16777215, default 2048):■
```

Se presiona  dos veces.

```
● ▼ ▲  
First sector (2048-16777215, default 2048):  
Last sector, +sectors or +size{K,M,G,T,P} (2048-16777215, default 16777215):  
Created a new partition 1 of type 'Linux' and of size 8 GiB.
```

Se verifica:

```
● ▼ ▲  
Command (m for help): p  
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors  
Units: sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disklabel type: dos  
Disk identifier: 0xf58ef347  
  
Command (m for help): ■
```

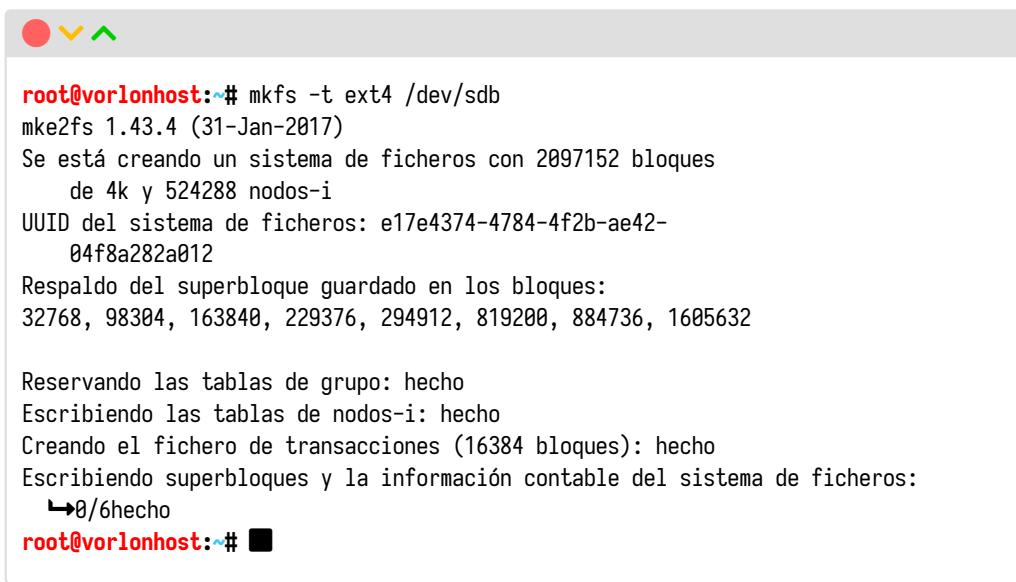
y se sale:

```
● ▼ ▲  
Command (m for help): q  
root@vorlonhost:~# ■
```

### 7.5.3 CREACIÓN DEL FILESYSTEM

Crear un filesystem implica crear e inicializar las estructuras datos que van a almacenar la información que se grabe o lea de este.

Es decir que hay que darle formato con el comando «**CMD mkfs**». A diferencia de otros sistemas operativos, en Linux se dispone de un abanico de distintos tipos de filesystems, algunos de los cuales se han mencionado anteriormente, como ser XFS, JFS, EXT4, ReiserFS, y los más nuevos y con características más avanzadas como BTrFs. Su objetivo es sustituir al actual sistema de archivos EXT3, eliminando el mayor número de sus limitaciones, en especial la del tamaño máximo de los archivos (16TB); además de la adopción de nuevas tecnologías no soportadas por EXT3. Cada uno posee ventajas y desventajas. A los efectos de simplificar la práctica se selecciona uno de los más utilizados hoy en día, EXT4.



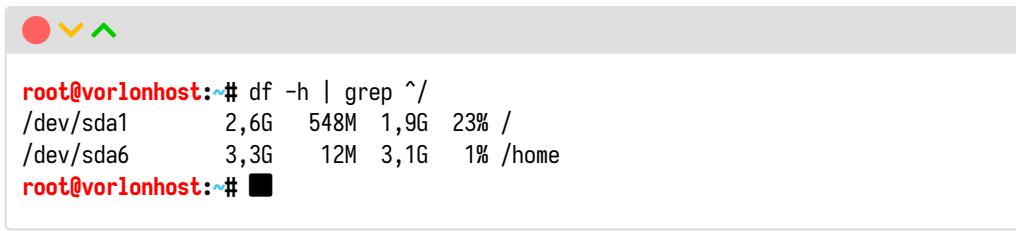
```
root@vorlonhost:~# mkfs -t ext4 /dev/sdb
mke2fs 1.43.4 (31-Jan-2017)
Se está creando un sistema de ficheros con 2097152 bloques
de 4k y 524288 nodos-i
UUID del sistema de ficheros: e17e4374-4784-4f2b-ae42-
04f8a282a012
Respaldo del superbloque guardado en los bloques:
32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Reservando las tablas de grupo: hecho
Escribiendo las tablas de nodos-i: hecho
Creando el fichero de transacciones (16384 bloques): hecho
Escribiendo superbloques y la información contable del sistema de ficheros:
↳0/6hecho
root@vorlonhost:~#
```

### 7.5.4 MIGRACIÓN DEL /home AL NUEVO FILESYSTEM

Primero se debe saber a qué filesystem esta relacionado el directorio `/home`. O más técnicamente hablando, cual es el filesystem «montado» en `/home`.

Se obtiene con el comando **CMD mount -l -t ext4**. En su lugar el Autor sugiere usar **CMD df -h**, que también informa el espacio disponible en cada filesystem. Para una mejor claridad, se filtra la salida para dejar únicamente los dispositivos de bloques.



```
root@vorlonhost:~# df -h | grep '^/'
/dev/sda1      2,6G  548M  1,9G  23% /
/dev/sda6      3,3G   12M  3,1G   1% /home
root@vorlonhost:~#
```

Ahora se sabe que el dispositivo en cuestión es `/dev/sda6`, el cual se puede asociar a otro filesystem más adelante.

Se crea un «punto de montaje» auxiliar. Los puntos de montaje son directorios ordinarios, así que se usará **CMD** `mkdir`.

**CMD** `mkdir /aux`

Ahora se monta el filesystem recién creado en el punto `/aux`.

**CMD** `mount /dev/sdb /aux`

Se verifica

```
root@vorlonhost:~# df -h | grep ^/
/dev/sda1      2,6G   548M  1,9G  23% /
/dev/sda6      3,3G    12M  3,1G   1% /home
/dev/sdb       7,9G   36M  7,4G   1% /aux
root@vorlonhost:~#
```

Se copia el contenido del `/home` a `/aux`.

**CMD** `cp -ar /home/* /aux`

Se verifica someramente la copia

```
root@vorlonhost:~# find /home
/home
/home/lost+found
/home/alumno
/home/alumno/.bashrc
/home/alumno/.profile
/home/alumno/.bash_logout
root@vorlonhost:~#
```

```
root@vorlonhost:~# find /aux
/aux
/aux/alumno
/aux/alumno/.profile
/aux/alumno/.bash_logout
/aux/alumno/.bashrc
root@vorlonhost:~#
```

```
/aux/lost+found  
root@vorlonhost:~# █
```

Y se verifica satisfactoriamente que se copió la información, se desmonta el viejo y se monta el nuevo.

```
● ▼ ▲  
root@vorlonhost:~# umount /aux && umount /home && mount /dev/sdb /home  
root@vorlonhost:~# █
```

Se verifica

```
● ▼ ▲  
root@vorlonhost:~# df -h | grep ^/  
/dev/sda1      2,6G  548M  1,9G  23% /  
/dev/sdb       7,9G   37M  7,4G   1% /home  
root@vorlonhost:~# █
```

Ahora se puede ver que se tiene más espacio en /home.



Se debe recordar que quedó un espacio en el disco que era del antiguo /home (/dev/sdb2). Escapa a los alcances de la materia, pero se debería saber que **ese espacio se puede aprovechar, agrandando la particiones adjuntas.**

## 7.6 ARCHIVO /etc/fstab



El fichero **fstab** («file systems table») se encuentra comúnmente en sistemas \*nix o GNU/Linux (en el directorio **/etc/**) como parte de la configuración del sistema. Lo más destacado de este fichero es la lista de discos y particiones disponibles. En ella se indica como montar cada dispositivo y qué configuración utilizar.

Los comandos **CMD** `mount` y **CMD** `umount` leen este fichero para determinar qué opciones utilizar a la hora de montar el dispositivo especificado. Por ejemplo, «**CMD** `mount /media/musica/`» montaría en el directorio `/media/musica` el dispositivo indicado en `fstab` con las opciones que tiene configuradas, si es que éstas lo permiten.

El usuario root se tiene que encargar de mantener este fichero para el mejor uso de los dispositivos. Esto normalmente se hace mediante un editor de texto aunque existen aplicaciones gráficas para los usuarios más inexpertos.[[wiki:fstab](#)]

### 7.6.1 ESTRUCTURA DEL ARCHIVO `/etc/fstab`

La estructura de las instrucciones es de seis columnas separadas por espacios o tabuladores:

CAMPO	DESCRIPCIÓN
dispositivo	Es el directorio lógico que hace referencia a una partición o recurso. ( <code>/dev/...</code> )
punto_de_montaje	Es la carpeta en que se proyectarán los datos del sistema de archivos. (Cualquier directorio)
sistema de archivos	El tipo de filesystem con el cual fue formateado el dispositivo. (ext2/3/4, xfs, jfs, zfs, etc.)
opciones	Es el lugar donde se especifican los parámetros que <b>CMD</b> <code>mount</code> utilizará para montar el dispositivo, deben estar separados por comas. (noauto, user, etc.)
dump-freq	Es el parámetro que utiliza <b>CMD</b> <code>dump</code> para hacer backup del sistema de archivos, si es cero no se toma en cuenta ese dispositivo para el backup masivo.
pass-num	Indica el orden en que el comando <b>CMD</b> <code>fsck</code> revisará la partición en busca de errores durante el inicio, si es cero el dispositivo no se revisa.

CUADRO 7.2: ESTRUCTURA DEL ARCHIVO `/etc/fstab`

### 7.6.2 EJEMPLO DE ARCHIVO `/etc/fstab`

A continuación se muestra un archivo `/etc/fstab` de ejemplo.

```

/etc/fstab
1 # /etc/fstab: static file system information.
2 #
3 # Use 'blkid' to print the universally unique identifier for a
4 # device; this may be used with UUID= as a more robust way to name devices
5 # that works even if disks are added and removed. See fstab(5).
6 #
7 # <file system> <mount point> <type> <options> <dump> <pass>
8
9 PARTUUID=06f16541-500c-4ef5-a4b8-b0ca657516d4
10   /boot/efi vfat umask=0077 0 0

```

```

11 UUID=c80f4d10-9a42-40c2-992e-c07fb6b63255
12   /home ext4 noatime,errors=remount-ro 0 0
13 UUID=ed745d39-0d4e-4ea1-8267-a85c4fd1c4cf
14   /opt ext4 noatime,errors=remount-ro 0 0
15 UUID=a4a95ac1-5fb5-42eb-8d66-2a216e8d86a3
16   / ext4 noatime,errors=remount-ro 0 0
17 UUID=f5c6e881-2386-4e82-aa81-27805fe2a684
18   /usr ext4 noatime,errors=remount-ro 0 0
19 UUID=d417425e-cabf-491a-b2b5-acd4cc07c0d5
20   /var ext4 noatime,errors=remount-ro 0 0
21 /dev/mapper/vg_datos-lv_swap none      swap    sw      0      0
22 /dev/mapper/vg_datos-lv_u01 /u01 ext4 defaults 0 0
23 /dev/mapper/vg_datos-lv_u02 /u02 ext4 defaults 0 0
24 /dev/mapper/vg_datos-lv_u03 /u03 ext4 defaults 0 0
25 /dev/mapper/vg_datos-lv_steam /steam xfs defaults 0 0
26 /dev/mapper/vg_datos-lv_virtualbox /virtualbox xfs defaults 0 0

```



Al modificar el archivo `/etc/fstab`, se debe tener especial cuidado, ya que cualquier error, **puede traer aparejado que no bootee el sistema operativo.**

## 7.7 LINKS

Uno de los tipos de archivos de Linux son los «links», los cuales se dividen en «hard links» y los «softlinks» (o «symlinks»). A los fines prácticos funcionan de «similar»<sup>7</sup> manera a los llamados «Accesos Directos» de Windows.

### 7.7.1 HARD LINKS

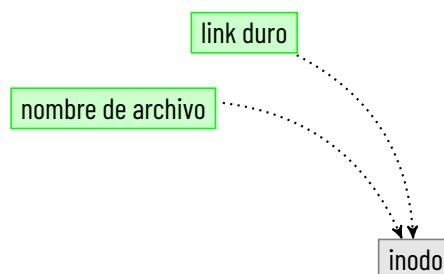


FIGURA 7.13: DIAGRAMA DE UN LINK DURO.

Es una referencia o puntero a un archivo (al dato físico) en un sistema de archivos. Los links duros o «hard links», asocian dos o más ficheros compartiendo el mismo ínodo.

<sup>7</sup>No es tan similar, pero el Autor se permitió una licencia.

## Inodo



Es una estructura de datos (es decir una forma de organizarlos) propia de los sistemas de archivos tradicionalmente empleados en los sistemas operativos tipo \*nix como es el caso de Linux. Un inodo contiene las características de un archivo regular, directorio, o cualquier otro objeto que pueda contener el sistema de ficheros.

Esto hace que cada link duro sea una copia exacta del resto de ficheros asociados, tanto de datos como de permisos, propietario. El nombre asociado a un archivo es simplemente una etiqueta almacenada en una estructura de directorio que referencia el sistema operativo al sistema de archivos, por lo cual, más de un nombre puede ser asociado al mismo archivo.

Cuando se accede a través de diferentes nombres, cualquier cambio hecho afectará el mismo archivo. Los links duros sólo pueden referenciar datos que existen en el mismo sistema de archivos. En la mayoría de los sistemas de archivos, todos los archivos son enlaces duros.

Aunque se llamen de distinta forma, tanto los links duros como el archivo original ofrecen la misma funcionalidad. Al modificar los datos apuntados por cualquiera de ellos, se cambian los datos reales almacenados en disco, quedando modificados para todos por igual. Al crearlos, debido a su naturaleza, los links duros sólo pueden apuntar a datos que estén en el mismo sistema de archivos que el archivo (link) a partir del cual se crean, al ser en realidad una copia de la misma referencia a datos físicos pero apuntada por otra etiqueta. Ver figura 7.13 en la anterior página.

Cada link duro aplica unos permisos de acceso a los datos referenciados.



Al ser indistinguibles de los archivos, se pueden utilizar links duros para ofrecer acceso a datos desde entornos aislados como «chroot» sin necesidad de duplicar los datos en disco.



Existe un procedimiento de recuperación de archivos borrados, que permite la reconstrucción de un enlace a datos que ya no estén asociados con un nombre. Sin embargo este proceso no está disponible en todos los sistemas y a menudo no es seguro que funcione.

## CREANDO LINKS DUROS

El formato del comando es:

```
$ ln <archivo_existente> <link>
```

## Limitaciones del link duro



Recordar que no se pueden realizar links sobre directorios y que el archivo referenciado no puede pasar los límites del filesystem.

Ejemplo:



```
ttyS9@vorlonhost:~$ ls -l *chaja*.jpg
2384743 -rw-r--r-- 2 ttys9 ttys9 175058 jul 15 16:37 chaja.jpg
ttyS9@vorlonhost:~$ ln chaja.jpg link_chaja.jpg
ttyS9@vorlonhost:~$ ls -i *chaja*.jpg
2384743 chaja.jpg 2384743 link_chaja.jpg
ttyS9@vorlonhost:~$ █
```

Se denota que ambos archivos tienen el mismo número de ínodo: #2384743.

## 7.7.2 SOFT LINKS

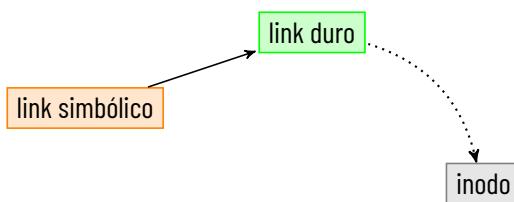


FIGURA 7.14: DIAGRAMA DE UN LINK SIMBÓLICO

Indica un acceso a un directorio o fichero que se encuentra en un lugar distinto dentro de la estructura de directorios. Una modificación realizada utilizando este enlace se reflejará en el original; pero, por el contrario, si se elimina el enlace, no se eliminará el archivo original.

Otra opción menos usual es utilizar un enlace duro «hard link», en el que el acceso es indistinguible del real, y el borrado del enlace provoca el borrado del archivo o directorio si era el último enlace duro al fichero. Ver figura 7.14.

Una ventaja del enlace simbólico frente a los enlaces duros es que es posible realizar enlaces simbólicos que apunten a objetos en sistemas de archivos que se hallan en otros dispositivos

o particiones dentro del mismo dispositivo. Además, cualquier usuario puede crear un enlace simbólico a un directorio, acción que está restringida a root en sistemas \*nix, aunque en los sistemas modernos esta posibilidad no existe.

En resumen, los enlaces simbólicos indican un acceso a un directorio o un fichero que se encuentra en un lugar distinto o no, dentro de la estructura de directorios.

#### CREANDO SOFT LINKS

El formato del comando es:

```
$ ln -s <archivo_existente> <link>
```

Ejemplo:

Se crean dos archivos para comenzar

```
ttyS9@vorlonhost:~$ mkdir -p /tmp/uno/dos
ttyS9@vorlonhost:~$ echo "test_a" >/tmp/uno/dos/a
ttyS9@vorlonhost:~$ echo "test_b" >/tmp/uno/dos/b
ttyS9@vorlonhost:~$ cd /tmp/uno/dos
ttyS9@vorlonhost:~$ ls -l
-rw-r--r-- 1 ttys9 group 7 Jan 01 10:01 a
-rw-r--r-- 1 ttys9 group 7 Jan 01 10:01 b
ttyS9@vorlonhost:~$ █
```

Se crea el link simbólico y se verifica

```
ttyS9@vorlonhost:~$ cd /tmp
ttyS9@vorlonhost:~$ ln -s /tmp/uno/dos tres
ttyS9@vorlonhost:~$ ls -l tres
lrwxrwxrwx 1 ttys9 group 12 Jul 22 10:02 /tmp/tres -> /tmp/uno/dos
ttyS9@vorlonhost:~$ ls -l tres/
-rw-r--r-- 1 ttys9 group 7 Jan 01 10:01 a
-rw-r--r-- 1 ttys9 group 7 Jan 01 10:01 b
ttyS9@vorlonhost:~$ █
```

```
ttyS9@vorlonhost:~$ cd three
ttyS9@vorlonhost:~$ ls -l
-rw-r--r-- 1 user group 7 Jan 01 10:01 a
ttyS9@vorlonhost:~$ █
```

```
-rw-r--r-- 1 user group 7 Jan 01 10:01 b
ttyS9@vorlonhost:~$ cat a
test_a
ttyS9@vorlonhost:~$ cat /tmp/one/two/a
test_a
ttyS9@vorlonhost:~$ echo "test_c" >/tmp/one/two/a
ttyS9@vorlonhost:~$ cat /tmp/one/two/a
test_c
ttyS9@vorlonhost:~$ cat a
test_c
ttyS9@vorlonhost:~$ █
```

**8****Inicio****8.1 systemd & init****8.1.1 ¿QUE ES init?**

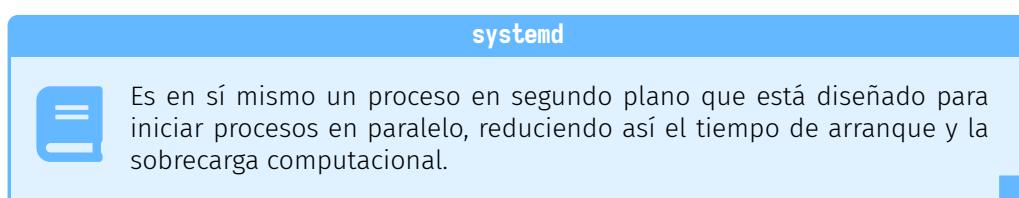
En Linux, init es una abreviatura de «Initialization». El init es un proceso de demonio que se inicia tan pronto como se inicia la computadora y continúa ejecutándose hasta que se apaga. De hecho, init es el primer proceso que se inicia cuando se bootea una computadora, convirtiéndola en el padre de todos los demás procesos en ejecución directa o indirectamente y, por lo tanto, generalmente se le asigna «pid = 1».

Si de alguna manera init no puede iniciarse, no se iniciará ningún proceso y el sistema alcanzará una etapa llamada «Kernel Panic». init se conoce en algunos círculos como como «System V init». «System V» es el primer sistema operativo UNIX comercial y el uso de init en la mayor parte de la distribución Linux de hoy es idéntico al sistema operativo OS V con algunas excepciones como «Slackware» con estilo BSD y «Gentoo» con init personalizado.

**8.1.2 ¿QUE ES systemd?**

systemd es un «daemon» de gestión del sistema, nombrado con la convención UNIX para agregar «d» al final del daemon, para que puedan ser fácilmente reconocidos. Similar a init, systemd es el padre de todos los demás procesos directa o indirectamente y es el primer proceso que comienza en el arranque, por lo tanto, generalmente se le asigna un «pid = 1».

También no hay que olvidar que systemd también puede referirse a todos los paquetes, utilidades y bibliotecas alrededor del demonio.



Por último, systemd fue diseñado para superar las deficiencias de init. Tiene muchas otras características en comparación con init.

Por ejemplo, tiene las ventajas de iniciar los servicios en forma paralela, poder acceder a los «logs» de inicio mediante «journald», y tener los logs en formato binario, lo que le da una capa

adicional de seguridad evitando la manipulación de los mismos.

TIPO DE ACTIVACIÓN	DESCRIPCIÓN
Basada en sockets	En el momento del arranque, systemd crea sockets de escucha para todos los servicios del sistema que admiten este tipo de activación, y los pasa a estos servicios tan pronto como se inician. Esto no solo permite que systemd inicie servicios en paralelo, sino que también permite reiniciar un servicio sin perder ningún mensaje enviado mientras no está disponible: el socket correspondiente permanece accesible y todos los mensajes están en cola.
Basada en bus	Los servicios del sistema que usan D-Bus <sup>1</sup> para la comunicación entre procesos pueden iniciarse a pedido la primera vez que una aplicación cliente intenta comunicarse con ellos. Systemd utiliza archivos de servicio de D-Bus para la activación basada en bus.
Basada en dispositivo	Los servicios del sistema que admiten la activación basada en el dispositivo se pueden iniciar a pedido cuando un tipo particular de hardware se conecta o está disponible. Systemd utiliza unidades de dispositivos para la activación basada en dispositivos.
Basada en rutas	los servicios del sistema que admiten la activación basada en rutas se pueden iniciar a pedido cuando un archivo o directorio en particular cambia su estado. Systemd utiliza unidades de ruta para la activación basada en la ruta.
Gestión de puntos de montaje y montaje automático	Systemd supervisa y gestiona los puntos de montaje y montaje automático. Systemd utiliza unidades de montaje para puntos de montaje y unidades de montaje automático para puntos de montaje automático.
Paralelización agresiva	Debido al uso de la activación basada en sockets, systemd puede iniciar los servicios del sistema en paralelo tan pronto como todos los sockets de escucha estén en su lugar. En combinación con los servicios del sistema que admiten la activación a pedido, la activación paralela reduce significativamente el tiempo requerido para iniciar el sistema.
Lógica de activación de la unidad transaccional	Antes de activar o desactivar una unidad, systemd calcula sus dependencias, crea una transacción temporal y verifica que esta transacción sea coherente. Si una transacción es inconsistente, systemd intenta automáticamente corregirla y eliminar trabajos no esenciales antes de informar un error.

<sup>1</sup>D-Bus es un sistema de bus de mensajes, una forma para que las aplicaciones se comuniquen entre sí. Similar a IPC y RPC.

Cuadro 8.1 – CONTINUACIÓN DE LA TABLA ANTERIOR

TIPO DE ACTIVACIÓN	DESCRIPCIÓN
Compatibilidad con SysV init	Systemd admite scripts de inicio SysV como se describe en la especificación básica de Linux, que facilita la ruta de actualización a las unidades de servicio systemd.

CUADRO 8.1: PRINCIPALES CARACTERÍSTICAS EN SYSTEMD.[rh:systemd]

### 8.1.3 PORQUE REEMPLAZÓ A init?

init comienza en serie, es decir, una tarea comienza solo después de que la última tarea se inició correctamente y se cargó en la memoria. Esto resultaba en un tiempo de arranque demorado y prolongado. Adicionalmente, los procesos eran arrancados por scripts de shell, lo que provocaba una sobrecarga adicional de CPU. Sin embargo, systemd no fue diseñado para la velocidad, sino para hacer las cosas con cuidado, lo que a su vez evita todo el retraso necesario de la CPU.

Asimismo permite un grado mayor de automatización:

Hay mas de un tipo[**definition:systemd**], y estos son:

Tipo de Unit	Extensión	Descripción
Service unit	service	Un servicio del sistema.
Target unit	target	Un grupo de unidades de systemd.
Automount unit	automount	Un punto de automontaje.
Device unit	device	Un archivo de dispositivo reconocido por el núcleo.
Mount unit	mount	Un punto de montaje.
Path unit	path	Un archivo o directorio en un filesystem.
Scope unit	scope	Un proceso creado externamente.
Slice unit	slice	Un grupo de unidades organizadas jerárquicamente que administran procesos del sistema.
Snapshot unit	snapshot	Un punto de guardado del administrador de systemd.
Socket unit	socket	Un socket IPC.
Swap unit	swap	Un archivo de o un filesystem de swap.
Timer unit	timer	Un timer de sysyemd.

CUADRO 8.2: TIPOS DE UNIDAD EN SYSTEMD.

## 8.2 SERVICIOS

En la sección 8.1 en página 153, se vieron los conceptos de systemV y systemD. En este capítulo se hará algo de práctica para reforzar esos conceptos.

Toda la operatoria aquí indicada se hace con `systemctl`, cuya sintaxis **reducida** es:

SIN `systemctl <enable|disable|start|stop|status> <nombre archivo unit>`

### 8.3 PASOS PARA DAR DE ALTA UN SERVICIO

Para dar de alta un servicio en systemd, se debe utilizar un archivo que contiene distintas definiciones necesarias para el alta del servicio.

Los pasos para realizar esta tarea son:

- 1 Tener listo el programa que será el servicio a ser administrado. Puede ser un archivo binario o un script.
- 2 Copiar el programa a una ubicación coherente y darle los permisos de ejecución.
- 3 Crear el archivo «unit» para definir un servicio systemd.
- 4 Copiar el archivo unit a /etc/systemd/system y darle permisos.
- 5 Iniciar y Habilitar el Servicio.
- 6 Verificar el estado del servicio.

### 8.4 EJEMPLO

#### 8.4.1 CREACIÓN DE PROGRAMA DE SERVICIO

A los efectos de esta práctica, se creará un script muy sencillo.

Script de servicio miscript.sh

```

1 #!/bin/bash
2
3 echo "Servicio de prueba iniciado el `date '+%d/%m/%Y %H:%M:%S'`" | systemd-cat -p
   ↳ info
4
5 while true; do
6   echo "Servicio de prueba en bucle..."
7   sleep 10
8 done
9
10 exit 0

```

Luego se le otorgan los permisos y se lo copia a un ubicación adecuada.



```

ttyS9@vorlonhost:~$ chmod u+x miscript.sh
ttyS9@vorlonhost:~$ sudo mv miscript.sh /usr/sbin
ttyS9@vorlonhost:~$ █

```

#### 8.4.2 ARCHIVO «UNIT»

Archivo miservicio.service

```

1 [Unit]
2 Description=Ejemplo de servicio en systemd.
3
4 [Service]
5 Type=simple
6 ExecStart=/bin/bash /usr/sbin/miscript.sh
7
8 [Install]
9 WantedBy=multi-user.target
10 tab

```

Se debe copiar al archivo de unit a `/etc/systemd/system`, y dar el permisos respectivo.

```

● ▼ ^
ttyS9@vorlonhost:~$ sudo cp miservicio.service /etc/systemd/system/miservicio.service
ttyS9@vorlonhost:~$ sudo chmod 644 /etc/systemd/system/miservicio.service
ttyS9@vorlonhost:~$ █

```

La extensión `.service`, guarda relación con el tipo de archivo de unit que se genera.

#### 8.4.3 CARGAR EL SERVICIO

Se carga con el argumento `enable` de `systemctl`.

```

● ▼ ^
ttyS9@vorlonhost:~$ sudo systemctl enable miservicio
Created symlink /etc/systemd/system/multi-user.target.wants/miservicio.service →
➥ /etc/systemd/system/miservicio.service.
ttyS9@vorlonhost:~$ █

```

Se verifica

```

● ▼ ^
ttyS9@vorlonhost:~$ sudo systemctl status miservicio
● miservicio.service - Ejemplo de servicio en systemd.
   Loaded: loaded (/etc/systemd/system/miservicio.service; enabled; vendor preset:
     ↳ enabled)

```

```
Active: inactive (dead)
ttyS9@vorlonhost:~$ █
```

#### 8.4.4 INICIAR EL SERVICIO

**CMD** `sudo systemctl start miservicio`

Se verifica

```
ttyS9@vorlonhost:~$ sudo systemctl status miservicio
● miservicio.service - Ejemplo de servicio en systemd.
   Loaded: loaded (/etc/systemd/system/miservicio.service; enabled; vendor preset:
             └─enabled)
   Active: active (running) since Mon 2020-03-02 21:12:50 -03; 4s ago
     Main PID: 13036 (bash)
        Tasks: 2 (limit: 4915)
      CGroup: /system.slice/miservicio.service
              └─13036 /bin/bash /usr/sbin/miscript.sh
                ├─13044 sleep 10
ttyS9@vorlonhost:~$ █
```

## 8.5 GRUB

### 8.5.1 INTRODUCCIÓN

Uno de los temas con más escozor sobre Linux, es la palabra «bootloader». La razón principal es que la mayoría de los nuevos usuarios de Linux solo han usado sistemas operativos Windows. En el mundo de Windows, los usuarios no tienen que lidiar con los gestores de arranque, el problema del arranque del sistema es transparente. A lo sumo, se usa la «Consola de recuperación de Windows» para solucionar problemas. Por lo tanto, se han ahorrado la necesidad de aprender sobre la pieza de software más importante en una computadora: el pequeño programa que hace que todo funcione.

GNU GRUB «GRand Unified Bootloader» es un gestor de arranque (también se puede escribir bootloader) capaz de cargar una variedad de sistemas operativos libres y propietarios. GRUB funcionará bien con Linux, DOS, Windows o BSD.

### 8.5.2 EL VIEJO Y QUERIDO «LILO»

Previo a Grub, el bootloader por excelencia en el mundo Linux, era LiLo «Linux Loader». Hoy es GRUB la mejor opción, por varias razones:

- ▶ LILO solo admite hasta 16 entradas de arranque diferentes; GRUB admite una cantidad ilimitada de entradas de arranque.
- ▶ LILO no puede arrancar desde la red; GRUB puede.
- ▶ LILO debe escribirse en el MBR cada vez que se cambie el archivo de configuración; GRUB no lo necesita.
- ▶ LILO no tiene una interfaz de comando interactiva, sobre todo al inicio que es donde más se lo puede llegar a necesitar.

### 8.5.3 GESTOR DE ARRANQUE

Todo disco rígido tiene un sector cero llamado Master Boot Record (MBR) que es el sector de arranque del disco rígido. En él se aloja el programa encargado de pasar el control, en secuencia de arranque, al sector cero de la partición que contiene el sistema operativo seleccionado. En este sector se aloja, a su vez, un programa encargado de arrancar el sistema operativo instalado en dicha partición.

#### Master Boot Record



Es el sector más importante, tiene solo 512 bytes de longitud y contiene un pequeño fragmento de código (446 bytes) llamado cargador de arranque primario y la famosa tabla de particiones (64 bytes) que describe las particiones primarias y extendidas. Vale decir que si este sector se daña, el disco no sirve más. La información que el había deberá ser recuperada con software de análisis forense.

### 8.5.4 SECUENCIA DE ARRANQUE DEL SISTEMA (RESUMEN)

Primero se debe repasar como es la secuencia de arranque (booteo) de una computadora:

- 1 Se enciende la PC. La BIOS<sup>2</sup> realiza un chequeo de los componentes hardware y utiliza la configuración establecida para comprobar determinados aspectos del equipo (hora del sistema, secuencia de arranque, etc.). A este examen se lo denomina POST «Power On Self Test».
- 2 La BIOS carga en memoria el programa que se encuentra almacenado en el primer sector del primer dispositivo en la secuencia de arranque. Se pasa el control de la máquina a dicho programa, llamado gestor de arranque, que contiene las instrucciones, en código máquina, que arrancan el equipo. Este sector se llama MBR «Master Boot Record».
- 3 Si el gestor de arranque es multiarranque, mostrará un menú donde el usuario debe seleccionar el sistema operativo a arrancar. Una vez elegida la opción, el gestor transfiere el control al primer sector de la partición del disco rígido, donde está el programa cargador de dicho SO.

<sup>2</sup>Basic Input Output System. Es el firmware que se guarda en una EEPROM y el primer código en ser ejecutado, ni bien se enciende una computadora.

- 4 El programa cargador del sistema operativo carga el kernel.
- 5 El kernel comienza reconociendo el hardware e inicializándolo, luego montando los filesystem especificados en la configuración correspondiente.
- 6 El kernel inicia los servicios que brinda el sistema operativo.
- 7 El usuario entra en el sistema introduciendo un nombre de usuario y contraseña.

### 8.5.5 EJECUCIÓN DE GNU GRUB

GRUB reemplaza el MBR predeterminado con su propio código. En concreto, la ejecución de GNU GRUB está dividida en dos etapas (tres contando la etapa 1.5):

- 1 La «Etapa 1» se encuentra en el MBR y apunta principalmente a la etapa 2, ya que el MBR es demasiado pequeño para contener todos los datos necesarios.
- 2 La «Etapa 1.5» también existe y podría usarse si la información de arranque es lo suficientemente pequeña como para caber en el área inmediatamente después de MBR.
- 3 La «Etapa 2» apunta a su archivo de configuración, que contiene todas las complejas interfaces de usuario y opciones con las que normalmente se está familiarizado cuando se habla de GRUB. La etapa 2 se puede ubicar en cualquier parte del disco. Si la Etapa 2 no puede encontrar su tabla de configuración, GRUB detendrá la secuencia de arranque y le presentará al usuario una línea de comando para la configuración manual.

La arquitectura Stage permite que GRUB sea grande (20-30K) y, por lo tanto, bastante complejo y altamente configurable, en comparación con la mayoría de los gestores de arranque, que son escasos y fáciles de ajustar dentro de las limitaciones de la tabla de particiones.

### 8.5.6 NOTACIÓN DE GRUB

Esta sección tiene como objeto que el usuario se familiarice con la notación de GRUB. A diferencia de lo que ya se vió de los dispositivos «`/dev/sdn`», GRUB los referencia con otra notación, por ejemplo:

**(hd0,1)**

Debe recordarse que:

- ▶ Los paréntesis son obligatorios.
- ▶ Se comienza a contar desde cero.
- ▶ «hd» significa disco duro; alternativamente, «fd» significa disquete, «cd» significa CD-ROM.
- ▶ El primer número (entero) se refiere al número del disco duro físico; en este caso, la primera unidad, ya que se cuentan desde cero hacia arriba. Por ejemplo, «hd2» se refiere al tercer disco duro físico.
- ▶ El segundo número se refiere al número de partición del disco duro seleccionado; de nuevo, las particiones se cuentan desde cero hacia arriba. En este caso, «1» representa la segunda partición.

Es evidente que GRUB no discrimina entre unidades IDE o SCSI o particiones primarias o lógicas. La tarea de decidir qué disco duro o partición puede arrancar se deja al BIOS y a la Etapa 1. Como puede ver, la notación es muy simple.

Las particiones primarias están marcadas de 0 a 3 ( $hdn,0$ ), ( $hdn,1$ ), ( $hdn,2$ ), ( $hdn,3$ ). Las particiones lógicas en la partición extendida se cuentan desde 4 en adelante, independientemente del número real de particiones primarias en el disco duro, por ejemplo ( $hd1,7$ ).

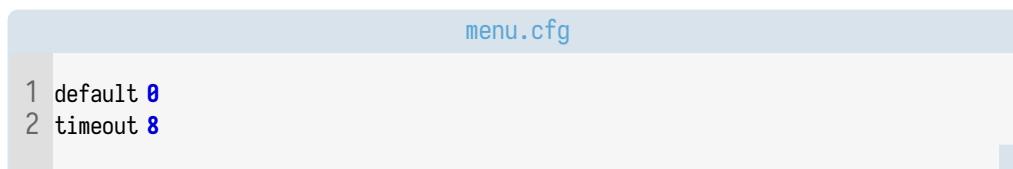
### 8.5.7 CONFIGURACIÓN DE GNU GRUB

Las entradas (también llamadas «stanzas») por sí solas no son suficientes para iniciar un sistema operativo. GRUB también necesita saber qué imágenes del sistema operativo cargar. Estos se asignan como parámetros a cada uno de los dispositivos involucrados, incluidos las directivas especiales. Por ejemplo, el «Modo seguro de Windows» es una directiva especial.

Las configuraciones se dividen en dos grandes grupos. Las de uso particular y las de uso general. Los comentarios se denotan con el símbolo «#».

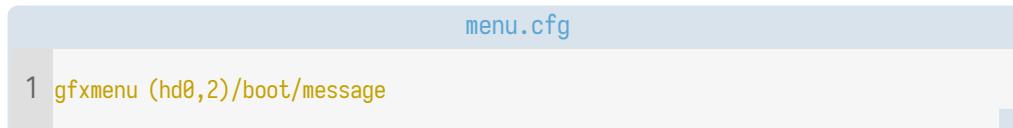
¿Como se determinan cuales son particulares y cuales son generales? Las particulares estan en una sección que comienza con la cláusula «title», las que no, son generales.

Las primeras líneas (sacando los comentarios) son:



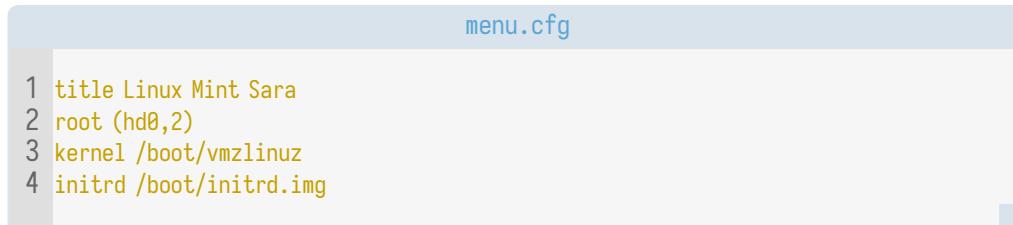
```
menu.cfg
1 default 0
2 timeout 8
```

La primera línea (valor predeterminado «0») significa que se iniciará el primer sistema operativo de la lista. La segunda línea (tiempo de espera «8») indica cuánto tiempo (en segundos) tiene el usuario para elegir antes de cargar la entrada predeterminada.



```
menu.cfg
1 gfxmenu (hd0,2)/boot/message
```

El menú GRUB también puede ser gráfico. Los archivos necesarios para presentar al usuario con un fondo colorido y posiblemente algunos extras se encuentran en el primer disco físico, tercera partición «(hd0,2)». Esta es una partición primaria, como se ha visto anteriormente.



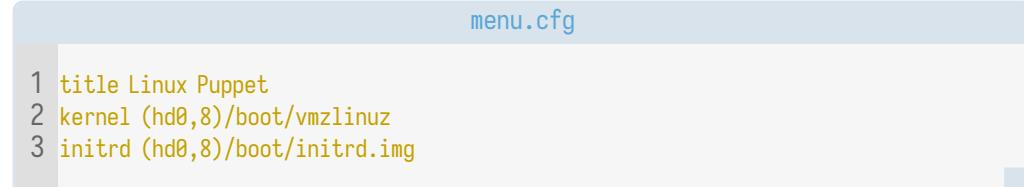
```
menu.cfg
1 title Linux Mint Sara
2 root (hd0,2)
3 kernel /boot/vmlinuz
4 initrd /boot/initrd.img
```

Esta es la primera entrada del sistema operativo en el menú, y las directivas en su interior son particulares.

VARIABLE	DESCRIPCIÓN
<code>title</code>	Es simplemente una cadena de texto destinada a ayudar al usuario a leer el menú en terminología «humana».
<code>root (hd0,2)</code>	Le dice a GRUB dónde se encuentran sus archivos de configuración. En este caso, se pueden encontrar en <code>(hd0,2)/boot/grub</code> .
<code>kernel /boot/vmlinuz</code>	Inicia la imagen real del kernel. Puede haber muchas imágenes disponibles. El hecho de que no haya un dispositivo especificado antes de <code>/boot/vmlinuz</code> indica que la imagen está ubicada en la misma partición que el GRUB. Este suele ser el caso predeterminado para la elección principal del sistema operativo.
<code>initrd /boot/initrd.img</code>	Es el sistema de archivos temporal comprimido, que se descomprime y monta en memoria para realizar los preparativos del sistema (adapta la imagen genérica del núcleo a un hardware específico), antes de montar el verdadero filesystem <code>root</code> .

CUADRO 8.3: CONFIGURACIÓN DE GRUB PARA LINUX CON ROOT APARTE

Los modificadores adicionales escritos después de «`kernel /boot/vmlinuz`» indican dónde se encuentra la raíz real, qué modo gráfico se utiliza y dónde reside la partición de intercambio. No se tratarán en detalles configuraciones de GRUB más avanzadas.



```
menu.cfg
1 title Linux Puppet
2 kernel (hd0,8)/boot/vmlinuz
3 initrd (hd0,8)/boot/initrd.img
```

VARIABLE	DESCRIPCIÓN
<code>title</code>	De nuevo, el título indica un nombre.
<code>kernel (hd0,8)/boot/vmlinuz</code>	Apunta a la novena partición en el primer disco duro ( <code>hd0,8</code> ). Accidentalmente, el indicador raíz ( <code>/dev/sda9</code> ) indica que la partición raíz es la misma que contiene la imagen del núcleo. Normalmente, este es el caso, y por simplicidad, querrá esta opción durante sus instalaciones.
<code>initrd (hd0,8)/boot/initrd.img</code>	Similar a lo visto anteriormente.

CUADRO 8.4: CONFIGURACIÓN DE GRUB PARA LINUX SIN ROOT APARTE,

## Sobre núcleos y particiones



En computadoras más antiguas con BIOS que no admiten acceso a más de los primeros 1024 cilindros, se puede configurar una partición de arranque que contenga la imagen del núcleo, mientras que la raíz misma se encuentra en otro lugar.

Se puede notar que la entrada de Linux Puppet es bastante detallada. Esta característica se llama «Cumplimiento de arranque múltiple»; Linux Mint reconoce Linux Puppet y puede llamar sin problema sus filesystems (incluidos los parámetros especiales) y montar las particiones. Sin embargo, la mayoría de los sistemas operativos son solo parcialmente compatibles con arranque múltiple.

```
menu.cfg
1 title Windows
2 rootnoverify (hd0,0)
3 chainloader (hd0,0)+1
```

VARIABLE	DESCRIPCIÓN
rootnoverify (hd0,0)	Significa que no puede entender al sistema operativo Windows, es decir, no es compatible con el arranque múltiple. Por lo tanto, el sistema operativo se llama sin ningún conocimiento previo del núcleo. GRUB supone que las imágenes de arranque relevantes se encontrarán en la partición de destino y serán montadas por el otro gestor de arranque del sistema operativo. Como se puede ver, Windows se instaló en la primera partición del primer disco duro. Esta es la opción más conveniente.
chainloader (hd0,0)+1	Se usa para sistemas operativos que no se pueden iniciar directamente. No es sorprendente que los sistemas operativos Windows caigan en esta tipificación. Se inician por el método de carga en cadena. Como su nombre lo indica, GRUB pasa el control de la secuencia de arranque a otro gestor de arranque, ubicado en el dispositivo al que apunta la entrada del menú. Este puede ser un sistema operativo Windows, pero también cualquier otro, incluido Linux.

CUADRO 8.5: CONFIGURACIÓN DE GRUB PARA WINDOWS™.

```
menu.cfg
1 title Windows 95/98/NT/2000
2 root (hd0,0)
```

```

3 makeactive
4 chainloader +1
5
6 title Linux
7 root (hd0,1)
8 kernel /vmlinuz root=/dev/hda3 ro

```

Analizando la entrada «Windows»:

VARIABLE	DESCRIPCIÓN
title	Ya visto previamente.
root	Especifica la partición donde se espera encontrar el kernel de Windows y lo monta.
rootnoverify	Deja el trabajo de inicio al gestor de arranque de Windows.
makeactive	Establece la partición activa en el disco root en el dispositivo raíz de GRUB.
chainloader	Se ejecuta sin la partición de destino especificada (ya que la partición de destino es la misma)

CUADRO 8.6: CONFIGURACIÓN DE GRUB PARA UNA ENTRADA WINDOWS.

Analizando la configuración «Linux»:

Se nombra un Linux, se llama a su partición y se arranca el núcleo. Es de destacar, que la imagen del núcleo y la partición raíz (/) del sistema operativo **no** se encuentran en la misma partición. Esto habitualmente sucede en las computadoras más antiguas, o aquellas con una partición de arranque específica.

### 8.5.8 INSTALACIÓN

GRUB se puede instalar en una variedad de dispositivos. Generalmente se instala en el disco duro. Antes de instalar cualquier cosa, se necesita saber dónde están los archivos. Si está instalado, el menú GRUB se encuentra en la partición raíz en /boot/grub/menu.lst

Cabe recordar que siempre debe hacerse una copia de seguridad de este archivo antes de hacer cualquier cambio, por menor que sea.

Los archivos GRUB se pueden encontrar en la imagen del sistema operativo, en: /usr/lib/grub/i386-pc.

#### INSTALACIÓN NATIVA DE GRUB

La instalación nativa, significa colocar la Etapa 1 de GRUB **Stage 1** en el primer sector del disco duro (MBR o tabla de particiones). Esto significa que se podrá arrancar sin un dispositivo secundario, como un disquete (que se ha convertido en una rareza hoy en día).

### Ojo con otras instalaciones



Si se instala más adelante un SO como Windows o si se intenta reparar el MBR por alguna razón (ejecutando `fdisk /MBR` desde el indicador de DOS), se borrará el GRUB y todos los sistemas enumerados en el `menu.lst` no podrán bootearse.

Para instalar GRUB en MBR, se deberá iniciar desde un medio externo (CD o PenDrive de algún Live Linux). Se puede llegar a la línea de comando de GRUB durante el arranque. Cuando se carga el menú GRUB, presionar **C** en el teclado.

Alternativamente, para acceder a GRUB desde la terminal, una vez cargado el sistema operativo, se utiliza el siguiente comando:



```
ttyS9@vorlonhost:~$ sudo grub
grub>■
```

Luego, una vez que se llegue al indicador GRUB, se deben realizar los siguientes pasos:

Se debe encontrar el dispositivo root de GRUB con:



```
grub>find /boot/grub/stage1
```

GRUB buscará todas las «Etapas 1» disponibles y las mostrará. Si se tiene más de una imagen del sistema operativo presente (por ejemplo, «SUSE», «Kubuntu», «Mandriva»), se tendrá más de una «Etapa 1» disponible.

Por ejemplo, si se supone que la computadora tiene los siguientes sistemas operativos instalados en diferentes particiones:



```
grub>find /boot/grub/stage1
Debian on (hd0,1)
Kubuntu on (hd0,2)
Mandriva on (hd0,4)
grub>■
```

Todos estos serán informados como posibles raíces para el dispositivo GRUB (ya que cada sistema operativo tiene sus propios archivos). Si desea utilizar «Debian GRUB», se configurará el dispositivo raíz de GRUB en (hd0,1):



A screenshot of a terminal window showing the GRUB boot loader interface. The prompt is 'grub>'. The user has typed 'root (hd0,1)' followed by the Enter key. The screen also shows a red circle, a yellow triangle pointing down, and a green triangle pointing up.

```
grub>root (hd0,1)
grub>
```

Si por otro lado, se desea «Mandriva»:



A screenshot of a terminal window showing the GRUB boot loader interface. The prompt is 'grub>'. The user has typed 'root (hd0,4)' followed by the Enter key. The screen also shows a red circle, a yellow triangle pointing down, and a green triangle pointing up.

```
grub>root (hd0,4)
grub>
```

Para realizar la instalación en el MBR:



A screenshot of a terminal window showing the GRUB boot loader interface. The prompt is 'grub>'. The user has typed 'setup (hd0)' followed by the Enter key. The screen also shows a red circle, a yellow triangle pointing down, and a green triangle pointing up.

```
grub>setup (hd0)
grub>
```

y luego



A screenshot of a terminal window showing the GRUB boot loader interface. The prompt is 'grub>'. The user has typed 'quit' followed by the Enter key. The screen also shows a red circle, a yellow triangle pointing down, and a green triangle pointing up. The text 'root@vorlonhost:~# ' is displayed in red at the bottom, indicating the system has booted.

```
grub>quit
root@vorlonhost:~#
```

Después de haber instalado GRUB, los sistemas operativos deberían arrancar. Una vez booteado, si el Lector se anima, puede practicar con GRUB, cambiando manualmente la configuración: agregando y eliminando entradas, cargando en cadena otros cargadores de arranque.

### 8.5.9 INSTALANDO GRUB CON grub-install

Este método se considera menos seguro, ya que adivina el mapeo. Aún así, para los que se inician, este podría ser el método preferido. Solo se necesita invocar un solo comando, es decir,

dónde instalar el gestor de arranque.

- ▶ **CMD** `grub-install /dev/hda`
- ▶ **CMD** `grub-install /dev/hd0`
- ▶ **CMD** `grub-install '(hd0)'`
- ▶ **CMD** `grub-install hd0`

## 8.6 GRUB II (EL REGRESO)

GRUB 2 tiene muchos cambios nuevos; mejor portabilidad y modularidad, admite caracteres no ASCII, carga dinámica de módulos, administración de memoria real y otros tantos. Todos estos son prácticamente irrelevantes para la mayoría de los usuarios. Lo que se necesita saber son los cambios en los archivos de configuración, y la forma en que opera GRUB 2.

### 8.6.1 NUEVO DISEÑO

Los archivos de GRUB I se encuentran en `/boot/grub/` incluido el archivo `menu.lst` que se leyó durante el arranque y cuyo contenido se mostró al usuario en forma del menú GRUB. Ahora, GRUB 2 coloca sus archivos en tres ubicaciones principales, tal cual figura en la tabla 8.7.

ARCHIVO	DESCRIPCIÓN
<code>/boot/grub/grub.cfg</code>	Este es el archivo de configuración principal que reemplaza <code>menu.lst</code> . A diferencia de <code>menu.lst</code> , este archivo no se puede editar a mano. Para una mejor seguridad, se recomienda su tratamiento R/O «Read Only»
<code>/etc/grub.d/</code>	Este nuevo directorio contiene los scripts (nuevos) de GRUB. Estos scripts son «bloques de construcción» a partir de los cuales se construye el archivo <code>grub.cfg</code> . Cuando se ejecuta el comando GRUB relevante, los scripts se leen en una secuencia determinada y se crea <code>grub.cfg</code> .
<code>/etc/default/grub</code>	Este archivo contiene la configuración del menú de GRUB que los scripts de GRUB leen y escriben en <code>grub.cfg</code> . Es la parte de personalización de GRUB, similar al antiguo <code>menu.lst</code> , excepto las entradas de arranque reales (las stanzas de GRUB I).

CUADRO 8.7: NUEVO DISEÑO DE GRUB2

Esto significa que si se desea cambiar el menú de GRUB, se tendrá que editar los scripts existentes o crear nuevos, luego actualizar el menú. Esto es más similar a LILO que al viejo GRUB, que permite editar el menú sobre la marcha.

## 8.6.2 ARCHIVO GRUB.CFG DE MUESTRA

8

Se ubica en /boot/grub:

```

root@vorlonhost:~# ls -l /boot/grub
total 2388
drwxr-xr-x 2 root root    4096 dic 28 09:45 fonts
-r--r--r-- 1 root root   12417 dic 28 10:31 grub.cfg
-rw-r--r-- 1 root root    1024 dic 29 08:33 grubenv
drwxr-xr-x 2 root root    4096 dic  6 12:41 locale
drwxr-xr-x 3 root root    4096 dic 28 09:45 themes
-rw-r--r-- 1 root root 2397557 dic  6 12:41 unicode.pf2
drwxr-xr-x 2 root root   12288 dic  6 12:41 x86_64-efi
root@vorlonhost:~# 

```

grub.cfg tiene una apariencia de shell script. Justamente es porque es un shell script.

```

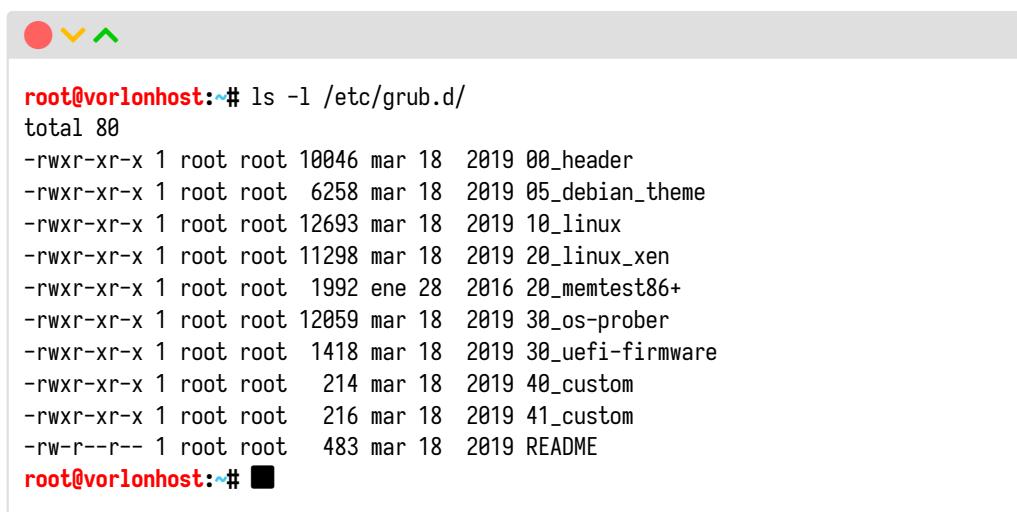
menu.cfg

1 #
2 # DO NOT EDIT THIS FILE
3 #
4 # It is automatically generated by grub-mkconfig using templates
5 # from /etc/grub.d and settings from /etc/default/grub
6 #
7
8 ### BEGIN /etc/grub.d/00_header ####
9 if [ -s $prefix/grubenv ]; then
10 set have_grubenv=true
11 load_env
12 fi
13 if [ "${next_entry}" ] ; then
14 set default="${next_entry}"
15 set next_entry=
16 save_env next_entry
17 set boot_once=true
18 else
19 set default="0"
20 fi

```

## 8.6.3 DIRECTORIO /etc/grub.d/

A continuación se analiza el contenido del directorio /etc/grub.d de la computadora, que en este caso, tiene instalado Linux Mint (un derivado de Ubuntu, que es un derivado de Debian).



```
root@vorlonhost:~# ls -l /etc/grub.d/
total 80
-rwxr-xr-x 1 root root 10046 mar 18 2019 00_header
-rwxr-xr-x 1 root root 6258 mar 18 2019 05_debian_theme
-rwxr-xr-x 1 root root 12693 mar 18 2019 10_linux
-rwxr-xr-x 1 root root 11298 mar 18 2019 20_linux_xen
-rwxr-xr-x 1 root root 1992 ene 28 2016 20_memtest86+
-rwxr-xr-x 1 root root 12059 mar 18 2019 30_os-prober
-rwxr-xr-x 1 root root 1418 mar 18 2019 30_uefi-firmware
-rwxr-xr-x 1 root root 214 mar 18 2019 40_custom
-rwxr-xr-x 1 root root 216 mar 18 2019 41_custom
-rw-r--r-- 1 root root 483 mar 18 2019 README
root@vorlonhost:~#
```

Algunos scripts son:

ARCHIVO	DESCRIPCIÓN
00_header	Es el script que carga la configuración de GRUB desde /etc/default/grub, incluido el tiempo de espera, la entrada de arranque predeterminada y otros.
05_debian_theme	Define el fondo, los colores y los temas. El nombre de este script definitivamente cambiará cuando otras distribuciones adopten GRUB 2.
10_linux	carga Las entradas del menú para la distribución instalada.
20_memtest86+	Carga la utilidad memtest.
30_os-prober	Es el script que escaneará los discos duros en busca de otros sistemas operativos y los agregará al menú de arranque.
40_custom	Es una plantilla que se puede usar para crear entradas adicionales para agregar al menú de inicio.

CUADRO 8.8: SCRIPTS EN /etc/grub.d/

¿Ha notado el lector la numeración en los nombres de los scripts? La numeración define la precedencia. Esto significa que 10\_linux se ejecutará antes de 20\_memtest86+ y, por lo tanto, se colocará antes en el orden del menú de arranque.

Los scripts en sí no son muy interesantes. Al igual que el archivo grub.cfg, no están destinados a ser editados, salvo el script 40\_custom. Se debe tener mucho cuidado al trabajar con estos scripts.

### 8.6.4 ¿COMO TRABAJA GRUB 2?

GRUB 2 en pocas palabras funciona de esta manera: /etc/default/grub contiene personalización; Los scripts en /etc/grub.d/ contienen información de menú de GRUB y scripts de arranque del sistema operativo. Cuando se ejecuta el comando **CMD update-grub**, lee el contenido del archivo grub conjuntamente con los scripts de grub.d y crea el archivo grub.cfg.

#### Notación



GRUB 2 usa la notación de **partición** que comienza con 1 y no con 0 como GRUB 1. En otras palabras, **los dispositivos todavía están numerados desde 0, pero las particiones comienzan con 1**. Por ejemplo, esto significa que sda1 es ahora (hd0,1) y no (hd0,0) como antes.

Para cambiar el archivo grub.cfg, se debe editar el archivo grub o los scripts en grub.d. Los scripts están destinados a ser ejecutados. Esto significa que tienen el bit de ejecución activado. Si se desactiva el bit de ejecución, no se ejecutarán.

### 8.6.5 AGREGAR UN NUEVO SCRIPT GRUB

Para agregar una nueva opción de arranque, se deberá seguir una sintaxis básica:

Se debe crear un nuevo archivo que tenga un prefijo «XX\_» en el nombre, donde «XX» es una secuencia de números. Si se desea que la nueva entrada se coloque sobre otras, se deben usar números menores, si se desea que se coloque debajo de otras, se deben usar números mayores.

Por ejemplo, «11\_algo» se colocará después de las entradas predeterminadas por el sistema operativo, mientras que «08\_algo» se colocará antes de las entradas «10\_linux».

El siguiente paso es escribir el contenido real. Aquí hay una muestra:

```
menu.cfg

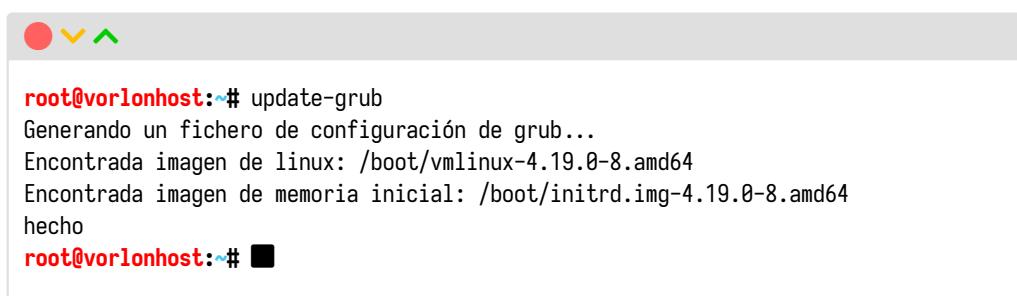
1 #!/bin/sh -e
2
3 echo "Cualquier texto"
4
5 cat << EOF
6 menuentry "Algo" {
7     set root=(hdX,Y)
8     linux /boot/vmlinuz
9     initrd /boot/initrd.img
10 }
11 EOF
```

Como se puede notar, las directivas entre las palabras «EOF», son las mismas que se utilizan en GRUB 1.

Una cosa más que se debe explicar es la información contenida en la sección **CMD cat <<EOF**.

Como se acaba de ver, el comando **CMD cat** define el inicio del código en el script que se agregará al menú de GRUB literalmente y NO será interpretado por el shell. En otras palabras, cualquier cosa que vaya entre **CMD cat<EOF** y EOF son comandos GRUB<sup>3</sup>.

El nuevo script está listo, pero el menú GRUB (grub.cfg) aún no se actualizó. Se necesita ejecutar el comando **CMD update-grub** para que se actualice.



```
root@vorlonhost:~# update-grub
Generando un fichero de configuración de grub...
Encontrada imagen de linux: /boot/vmlinuz-4.19.0-8.amd64
Encontrada imagen de memoria inicial: /boot/initrd.img-4.19.0-8.amd64
hecho
root@vorlonhost:~#
```

### 8.6.6 EDICIÓN DE /etc/default/grub

Este archivo contiene algunos parámetros que permiten la personalización de la entrada pre-determinada seleccionada, el tiempo de espera predeterminado y opciones adicionales.



```
/etc/default/grub

1 # If you change this file, run 'update-grub' afterwards to update
2 # /boot/grub/grub.cfg.
3 # For full documentation of the options in this file, see:
4 #   info -f grub -n 'Simple configuration'
5
6 GRUB_DEFAULT=0
7 GRUB_TIMEOUT_STYLE=hidden
8 GRUB_TIMEOUT=10
9 GRUB_DISTRIBUTOR=`lsb_release -i -s 2>/dev/null || echo Debian`
10 GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
11 GRUB_CMDLINE_LINUX=""
```

Si bien hay muchas otras opciones, algunas se encuentran en la tabla 8.10 de la siguiente página. Es importante que se recuerde que existen y dónde se pueden encontrar. No se deben hacer cambios a ciegas. Se debe consultar la documentación oficial y siempre se debe hacer una copia de seguridad del archivo antes de manipularlo.

<sup>3</sup>A este tipo de texto, se lo conoce como «Inline text».

Archivo	Descripción
GRUB_DEFAULT=0	Especifica la entrada predeterminada. Cuenta desde 0, como cualquier menú. Se puede cambiar al valor que se desee. Si se establece la entrada en GRUB_DEFAULT=saved, arrancará la última opción seleccionada desde el arranque anterior.
GRUB_TIMEOUT="10"	Especifica el tiempo de espera predeterminado. Se puede cambiar al valor que se desee. No se recomiendan valores muy pequeños. Establecer a -1 hará que GRUB espere indefinidamente hasta que seleccione manualmente una entrada y presione Enter.

CUADRO 8.10: PARÁMETROS EN /etc/default/grub

**9****Redes****9.1 UNA (NO TAN CORTA) INTRODUCCIÓN**

Antes de continuar es importante que se establezcan algunos conceptos.

**9.1.1 ¿QUE ES UNA RED?**

Si bien el término «redes» es ambiguo, porque hace alusión a cualquier tipo de red, desde eléctrica, hasta la del pescador, el Autor restringe el significado a redes de computadoras.

**Redes**

Se puede definir a las redes como la conexión existente entre dos o más dispositivos con el objeto de enviar datos de un lado a otro mediante señalización electrónica.

En la vida real, las redes no son simplemente dos latas unidas por un hilo, son millones y millones de latas (técnicamente llamadas «hosts») conectadas entre sí con muchos hilos.

**Host**

Un host es un sistema informático conectado a una red.

**9.1.2 CLIENTES Y SERVIDORES**

En las redes siempre hay dos computadoras conectadas. El servidor y el cliente. El servidor escucha y espera a que los clientes se conecten a él y el cliente se conecta al servidor. Para usar el ejemplo anterior, el servidor es la computadora que contiene el sitio web y el cliente es el navegador web. Entonces, cuando se escribe <http://www.google.com>, el navegador web se conecta al servidor en <http://www.google.com> y comienzan a comunicarse.

**cliente-servidor**

Un protocolo cliente-servidor es aquel en el que un host siempre activo (el servidor) escucha las conexiones de otros hosts (clientes). Cuando se establece una conexión, los datos pueden transferirse entre el cliente y el servidor.

**9.1.3 PROTOCOLOS**

Un protocolo, es «la forma de hacer las cosas». En el ámbito informático, un protocolo es el lenguaje que usan las computadoras para comunicarse entre sí. Si dos computadoras quieren hablar entre sí, necesitan saber el mismo idioma. Si no saben el mismo idioma, no pueden hablar entre ellas. Por ejemplo: alguien que habla chino y no sabe una palabra de español, no le será tan fácil comunicarse con una persona que habla hispana. Será casi imposible. Por ejemplo, el protocolo utilizado en los navegadores web y servidores web es el protocolo http.

**Protocolo**

Un protocolo es un conjunto establecido de reglas que dictan el método de comunicación entre 2 hosts. Los ejemplos incluyen http, ftp, tcp, ppp.

**9.1.4 LA RED DE REDES: INTERNET**

Internet no es un solo término, es una interconexión masiva de hosts como esta computadora y otra en todo el mundo. Entendemos que nuestros ISP<sup>1</sup> brindan acceso a Internet a nuestro host, pero puede que no necesariamente brinde servicio al servidor con el que se está comunicando. En este caso, ¿cómo llega la información desde este host al otro host y viceversa?

La respuesta está en la interconexión de muchos ISP. Los ISP's se pueden clasificar en ISP's de nivel 1, 2 y 3. Los ISP de Nivel 1 son los principales proveedores de servicios de Internet que generalmente venden acceso a ISP de Nivel 2 más pequeños. Los ISP de Nivel 2 pueden prestar servicio a países o ciudades enteros, pero no al resto del mundo. Los ISP de Nivel 3 también son clientes de los ISP de Nivel 2, y generalmente atienden a usuarios finales como el Lector o el Autor. Los ISP se conectan entre sí para permitir que los datos de su host lleguen al otro host. A partir de esto, entendemos que los datos pasan a través de múltiples ISP para ser entregados de una ubicación a otra.

**9.1.5 MÉTODOS DE CONMUTACIÓN «SWITCHING»**

Para transmitir datos de un extremo a otro, se debe adoptar un estándar común para la transmisión de tramas de datos. Aquí consideraremos la commutación de paquetes y la commutación de circuitos.

<sup>1</sup>Son las siglas de «Internet Service Provider» Proveedor de Servicio de Internet.

## CONMUTACIÓN DE CIRCUITOS

Las redes conmutadas son más viejas de lo que se puede llegar a suponer, a título de ejemplo, las viejas redes telefónicas eran redes conmutadas de circuitos.

Para establecer una conexión de datos del punto «A» al punto «Z», una persona debe trazar una ruta directa a través de varias rutas de conexión al destino. Una vez que se ha determinado una ruta, la persona necesita reservar recursos en esa línea para establecer su conexión, después de lo cual puede comenzar a transmitir datos. Si bien se han asignado recursos para esa conexión, nadie más puede usar esa línea hasta que el primer usuario haya desconectado su host, es decir que el enlace es dedicado, lo cual acarrea un costo hoy por hoy, bastante oneroso. Esto plantea algunas preguntas sobre cómo las personas pueden compartir una conexión de conmutación de circuitos, dos de los cuales se describen a continuación.

Una de las desventajas de una red de Circuito Conmutado, es que depende de que un usuario reserve y asigne recursos para sí mismo para usar la red. Esto aumenta el tiempo de conexión y puede generar una sobrecarga significativa al establecer conexiones. En caso de que el usuario decida bajar y tomar un café sin desconectarse, no habrá liberado los recursos reservados para que otros los usen.

## CONMUTACIÓN DE PAQUETES

Esta forma de conmutación funciona sobre el concepto de envío de datos como fragmentos discretos de datos conocidos como «paquetes». Cuando una persona tiene datos para enviar, se envía un flujo de paquetes a la red y los routers envían los paquetes a su destino. No existe una noción de ancho de banda reservado y, como resultado, si se envían demasiados paquetes a la red, la red puede congestionarse y ocurrirá la pérdida de paquetes. Esto lleva al problema del control de la congestión.

La idea principal detrás de las redes de conmutación de paquetes, es que nadie está utilizando todo el ancho de banda de una conexión al mismo tiempo, y por lo tanto, elimina el desperdicio de ancho de banda que generalmente acompaña a las redes de conmutación de circuitos. Esto da como resultado una mayor eficiencia y es la razón principal por la cual las redes de paquetes conmutados forman la columna vertebral de las redes informáticas actuales.

### 9.1.6 EL MODELO OSI

A los efectos de organizar y clasificar los distintos protocolos utilizados en las redes informáticas, se crearon varios modelos. Aquí se va presentar al modelo OSI.

## APLICACIÓN

La capa de aplicación proporciona servicios de red a las aplicaciones del usuario. En esencia, esto no se refiere a la aplicación en sí misma (es decir, cliente de correo electrónico, navegador web) sino a los protocolos reales que se utilizan, tales como: HTTP, POP3, IMAP, SMTP, DHCP, DNS y muchos otros en esa misma categoría. Estos protocolos de nivel superior son los que utiliza la aplicación para presentar la información.

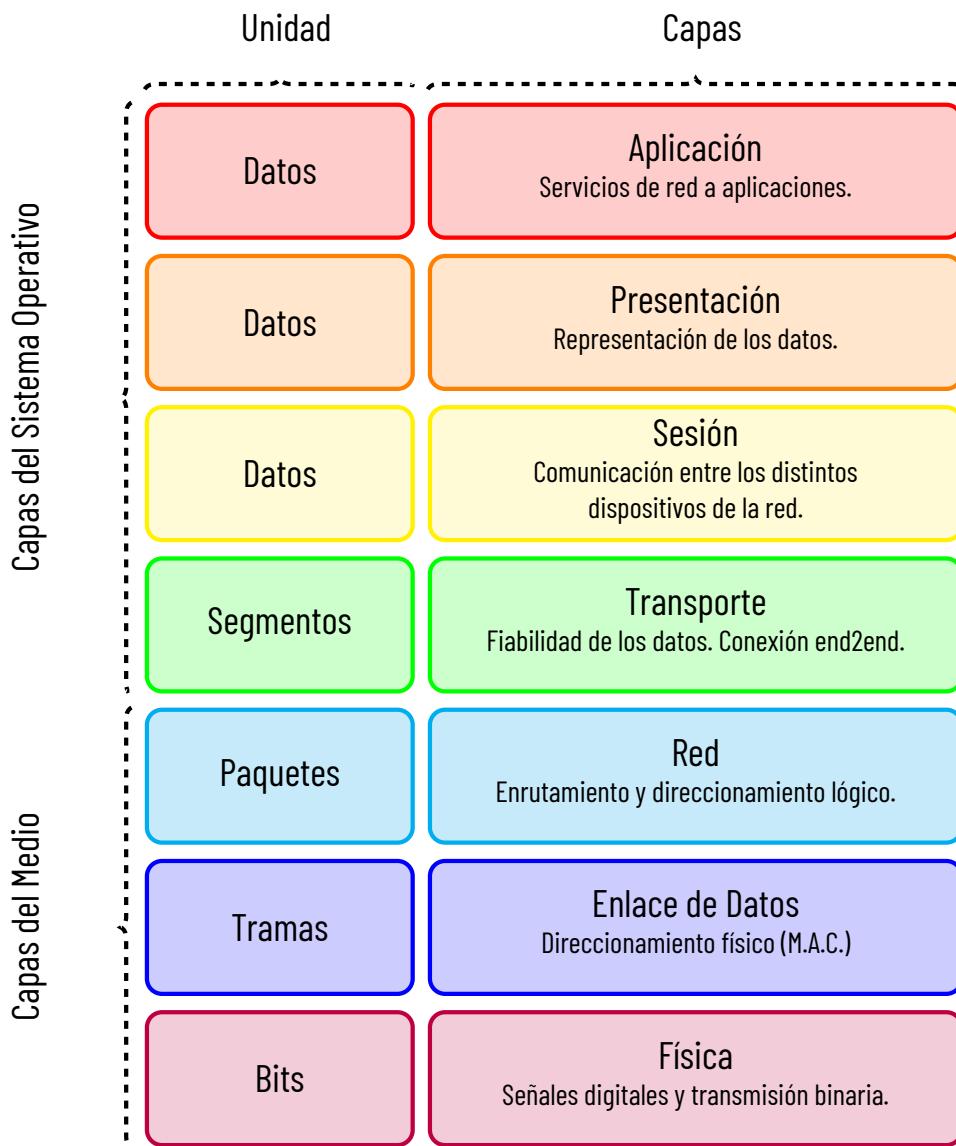


FIGURA 9.1: MODELO OSI

## PRESENTACIÓN

La capa de presentación existe para asegurarse de que la capa de aplicación del remitente envía información que puede leer la capa de aplicación del receptor. En esta capa se incluye la conversión de datos, la compresión de datos y el cifrado de datos. En los escenarios típicos del mundo real, esta capa no siempre se usa.

## SESIÓN

La capa de sesión es responsable del establecimiento, gestión y finalización de las sesiones de comunicación entre las aplicaciones de software de dos hosts. Esto puede parecer confuso ya que se podría creer que este es el trabajo de TCP para establecer, mantener y terminar la comunicación entre hosts. La capa de sesión se ocupa más de cómo dos aplicaciones de software establecen, mantienen y finalizan la comunicación entre ellas completamente separadas de la intercomunicación real de los paquetes de datos. Las capacidades de la capa de sesión casi siempre se encuentran en las API. Ejemplos de «protocolos» de capa de sesión son: NetBIOS, Socket TCP/IP, RPC y Sockets \*nix.

## TRANSPORTE

En el sentido más básico, la capa de transporte controla la segmentación y el reensamblaje del mensaje. Esta capa también es responsable del control de errores y el control de flujo. Esta capa es conocida principalmente por dos protocolos que se encontrarán en escenarios del mundo real: TCP y UDP. TCP y UDP, son responsables de tomar muchas conexiones que entran en un servidor y reenviar los datos correctamente en función de los números de puerto. Por lo tanto, TCP y UDP son responsables de la numeración de puertos para las aplicaciones. Esto se explica mejor si se tiene un servidor que ejecuta estos servicios: Servidor web (80), DNS (53), SMTP (25), IMAP (143) y POP (110):

- **TCP** → «Protocolo de control de transmisión» es el famoso TCP en TCP/IP. El Protocolo de control de transmisión, es un protocolo orientado a la sesión/conexión. TCP, a diferencia de UDP, ofrece recuperación de errores a través de un proceso llamado ventanas, usando señales «SYN» (Sincronizar) y «ACK» (Reconocimiento). Esencialmente, una computadora envía un SYN a un servidor y el servidor responde con un ACK y en el servidor también incluye su propio indicador SYN para el ACK que seguirá desde el host y también incluye el tamaño de la ventana (la cantidad de paquetes no confirmados que se pueden enviar a la vez antes de que se envíe un ACK). Las ventanas permiten que llegue una cierta cantidad de paquetes sin recibir un ACK. Una vez que esta ventana está llena, en teoría, el servidor responde con un ACK y si ese número de ACK está en secuencia con el siguiente SYN que se enviará, el host remitente sabrá que recibió todos los paquetes. Si no obtiene una respuesta, envía todos los paquetes nuevamente y si el ACK del servidor es un valor numérico menor de lo que se envió, el host reenvía los paquetes que estaban después de eso y espera el ACK correctamente numerado.
  
- **UDP** → «Protocolo Uniforme de Datagramas» es un protocolo orientado sin conexión que no proporciona recuperación de errores. Es útil para las comunicaciones que no necesitan ser confiable como: DNS y VOIP, ya que es mas rápido.

## 9

## RED

La capa de red es responsable del transporte de un paquete de una red a otra mediante el direccionamiento lógico. No es necesario si dos computadoras están directamente conectadas entre sí, como en una configuración LAN con el direccionamiento MAC. En esta capa existe la famosa «dirección IP» de TCP/IP «Internet Protocol» (v4 y v6). Esta capa trata con direcciones IP para proporcionar direccionamiento «lógico» de sistemas tanto en entornos LAN<sup>2</sup> como WAN<sup>3</sup>; el enrutamiento también tiene lugar para enrutar datos entre dos subredes diferentes. También es responsable de la fragmentación de los paquetes si se envían por un enlace de capa 2 que tiene una MTU<sup>4</sup> más pequeña. Lo más importante es que IP se utiliza para enrutamiento y direccionamiento lógico de máquinas.



El kernel mantiene una tabla interna denominada tabla de enrutamiento o «routing table», que se mantiene un inventario de las rutas a las distintas redes y la ruta por defecto «routing table». Se puede inspeccionar y manipular con `CMD route`.

## ENLACE DE DATOS

La capa de enlace de datos es responsable del transporte de «tramas» en la misma red. Para lograr esto, se utiliza el direccionamiento físico. Esta capa implementa control de acceso para determinar qué dispositivos pueden transmitir en una red que cuenta con múltiples dispositivos; usando procesos tales como CSMA/CD<sup>5</sup> en redes que utilizan comunicación dúplex completa. Esta capa se refiere al direccionamiento físico (direcciones MAC<sup>6</sup>) que se graban en cada NIC<sup>7</sup> y es única entre todas las tarjetas del mundo<sup>8</sup>, aunque la dirección MAC de ciertos adaptadores de red se puede cambiar. Aquí se hallan los protocolos Ethernet y Token Ring entre otros.



El kernel mantiene una tabla interna llamada «ARP» que relaciona las direcciones MAC con las direcciones IP. Dicha tabla se puede inspeccionar y manipular con `CMD arp`.

## FÍSICA

La capa física controla la transmisión y recepción del flujo de bits a través de un medio físico. Esta capa define medios como UTP/STP, fibra, coaxial, etc. Define los pines de cableado,

<sup>2</sup>Local Area Network.

<sup>3</sup>Wide Area Network

<sup>4</sup>La unidad máxima de transferencia «Maximum Transmission Unit - MTU» es un término que expresa el tamaño en bytes de la unidad de datos más grande que puede enviarse usando un protocolo de comunicaciones.

<sup>5</sup>Acceso múltiple de detección de portador/Detección de colisión

<sup>6</sup>«Media Access Control» Control de acceso al medio.

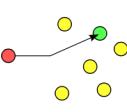
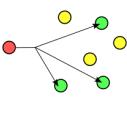
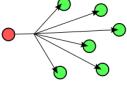
<sup>7</sup>«Network Interface Card» Tarjeta de interfaz de red

<sup>8</sup>Las direcciones MAC son de 48 bits, esto es una cantidad de 281.474.976.710.656 direcciones diferentes.

conductividad eléctrica, amplificación de luz (fibra), distancia de cableado y similares.

### 9.1.7 TIPOS DE COMUNICACIÓN

Dentro de las redes, hay tres tipos de comunicación, los cuales figuran descriptos en el cuadro 9.1.

	Unicast	De host a host.
	Multicast	De un host a muchos hosts.
	Broadcast	De un host a todos los hosts de un segmento.

CUADRO 9.1: TIPOS DE COMUNICACIÓN.

Los protocolos pueden utilizar cualquiera de estos tipos de comunicación.

### 9.1.8 ELEMENTOS TÍPICOS DE UNA RED LAN

En la figura 9.2 de la siguiente página, se pueden apreciar los distintos elementos que componen una red LAN típica.

Archivo	Descripción
PCn	Son las computadoras que forman parte de la LAN.
Switch	Es el dispositivo de capa 2 que se encarga de administrar las conexiones de los equipos directamente conectados a él. Este dispositivo incrementa la cantidad de «dominios de colisión».
Gateway/Router	Es el dispositivo de capa 3 que interconecta redes. La interfaz ethernet que esta conectado a una red LAN, determina un segmento de colisión.

CUADRO 9.2: ELEMENTOS TÍPICOS DE UNA RED LAN.

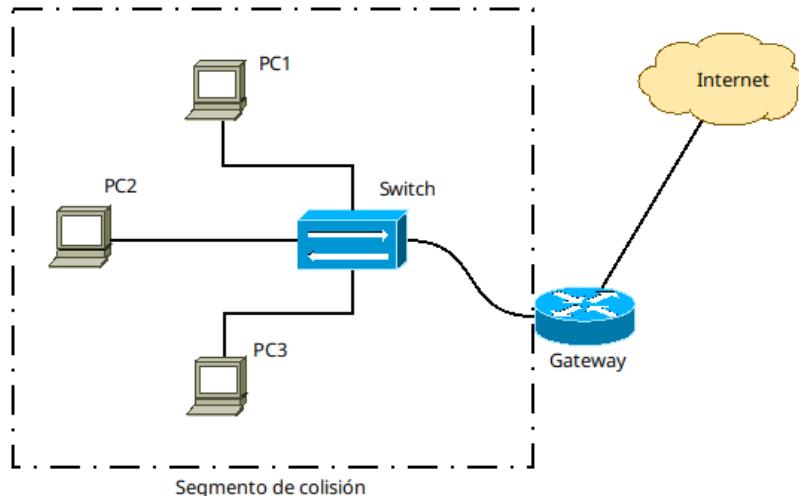


FIGURA 9.2: ELEMENTOS DE UNA RED LAN

## 9.2 EN DEBIAN GNU/LINUX

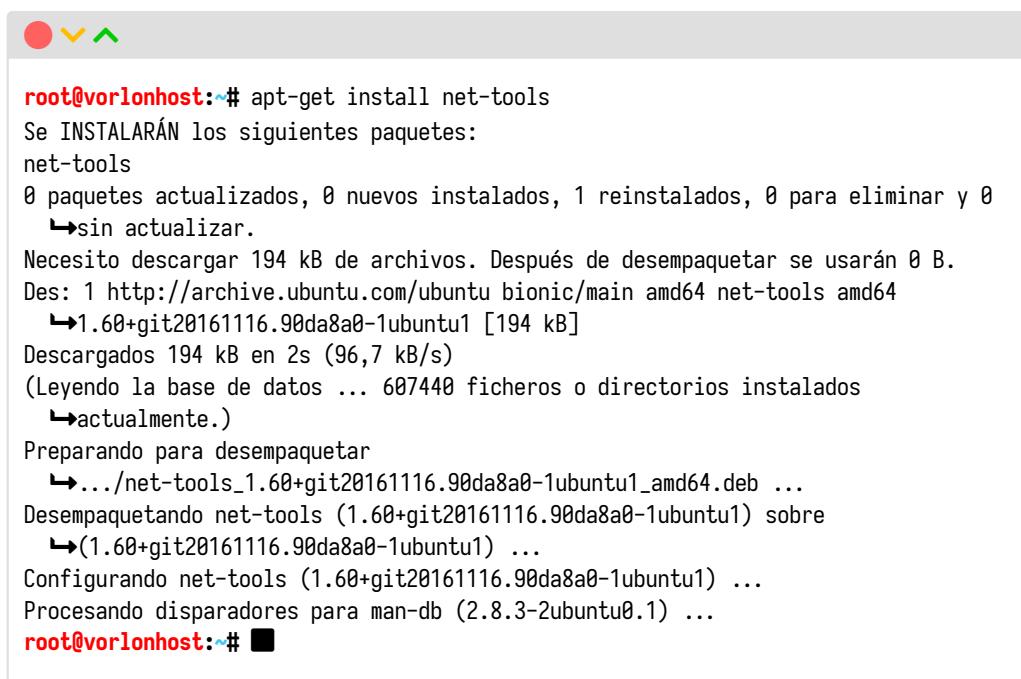
Hasta la versión 8 de Debian, se utilizaba el comando `ifconfig` para la administración de las placas de red. De la versión 9 en adelante se utiliza el comando `ip`, que provee una mayor flexibilidad.

Vemos una equivalencia en la figura 9.3 [net:ipifconfig].

ACCIÓN	<code>ifconfig</code>	<code>ip</code>
Mostrar los dispositivos de red y su configuración.	<code>ifconfig</code>	<code>ip addr show</code> <code>ip link show</code>
Activar interface de red.	<code>ifconfig eth0 up</code>	<code>ip link set eth0 up</code>
Desactivar interface de red	<code>ifconfig eth0 down</code>	<code>ip link set eth0 down</code>
Establecer dirección IP	<code>ifconfig eth0 192.168.1.1</code>	<code>ip address add 192.168.1.1 dev eth0</code>
Eliminar dirección IP	N/D	<code>ip address del 192.168.1.1 dev eth0</code>
Añadir interface virtual o alias	<code>ifconfig eth0:1 10.0.0.1/8</code>	<code>ip addr add 10.0.0.1/8 dev eth0 label eth0:1</code>
Añadir entrada en una tabla ARP <sup>9</sup>	<code>arp -i eth0 -s 192.168.0.1 00:11:22:33:44:55</code>	<code>ip neigh add 192.168.0.1 lladdr 00:11:22:33:44:55 nud permanent dev eth0</code>
Cambiar un dispositivo ARP a off	<code>ifconfig -arp eth0</code>	<code>ip link set dev eth0 arp off</code>

CUADRO 9.3: TABLA DE EQUIVALENCIA ENTRE `ifconfig` E `ip`.

Si se quiere utilizar el comando `ifconfig` y `route` en Debian9, se debe escribir el siguiente comando:



```

root@vorlonhost:~# apt-get install net-tools
Se INSTALARÁN los siguientes paquetes:
net-tools
0 paquetes actualizados, 0 nuevos instalados, 1 reinstalados, 0 para eliminar y 0
↳sin actualizar.
Necesito descargar 194 kB de archivos. Despues de desempaquetar se usarán 0 B.
Des: 1 http://archive.ubuntu.com/ubuntu bionic/main amd64 net-tools amd64
↳1.60+git20161116.90da8a0-1ubuntu1 [194 kB]
Descargados 194 kB en 2s (96,7 kB/s)
(Leyendo la base de datos ... 607440 ficheros o directorios instalados
↳actualmente.)
Preparando para desempaquetar
↳.../net-tools_1.60+git20161116.90da8a0-1ubuntu1_amd64.deb ...
Desempaquetando net-tools (1.60+git20161116.90da8a0-1ubuntu1) sobre
↳(1.60+git20161116.90da8a0-1ubuntu1) ...
Configurando net-tools (1.60+git20161116.90da8a0-1ubuntu1) ...
Procesando disparadores para man-db (2.8.3-2ubuntu0.1) ...
root@vorlonhost:~# █

```

## 9.3 PREPARACIÓN DEL ENTORNO VIRTUALIZADO

Se debe configurar el software de virtualización y luego la máquina virtual.

### 9.3.1 CONFIGURACIÓN DE VIRTUALBOX™

Se debe configurar una «red secundaria anfitriona».

- 1 Se debe acceder a la configuración del virtualbox. Figura 9.3 de la siguiente página
- 2 Aquí se debe agregar la red del tipo «Solo anfitrion». Figura 9.4 de la siguiente página.
- 3 Se verifica la configuración de la red. Figura 9.5 en página 183.
- 4 Se debe anotar la dirección IP de la red. Figura 9.6 en página 183.
- 5 Si no desea DHCP, se debe destildar la opción correspondiente 9.7 en página 183.

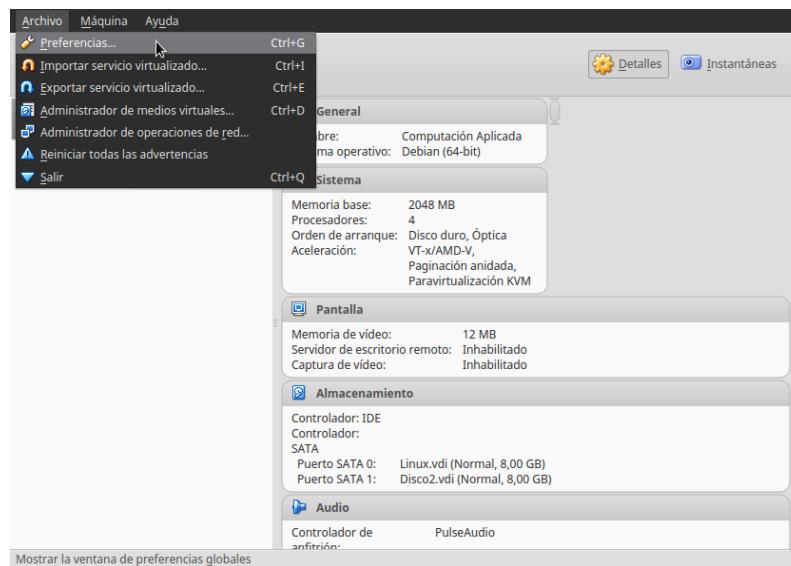


FIGURA 9.3: PASO 1 - CONFIGURACIÓN DEL VIRTUALBOX.

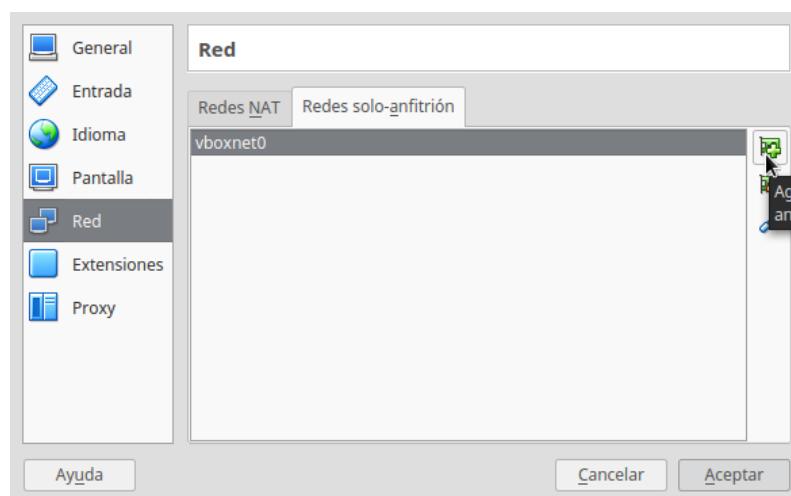


FIGURA 9.4: PASO 2 - AGREGAR RED solo-anfitrión

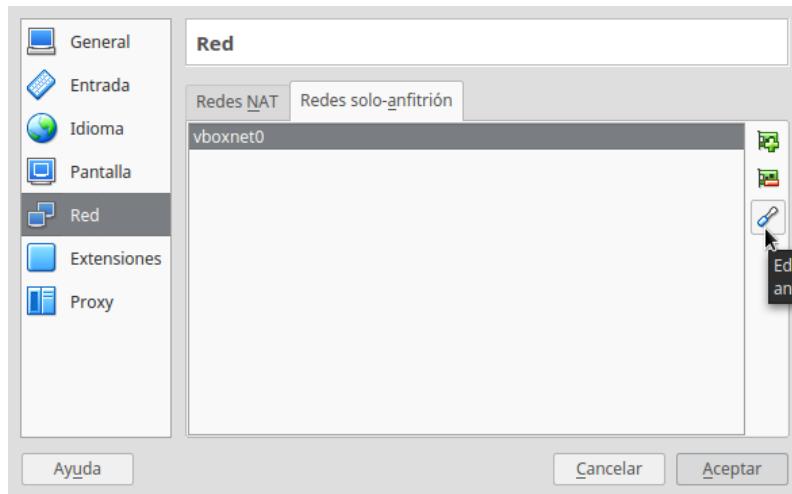


FIGURA 9.5: PASO 3 - VERIFICAR CONFIGURACIÓN DE Dicha RED.



FIGURA 9.6: PASO 4 - TOMAR NOTA DE LA DIRECCIÓN IP.

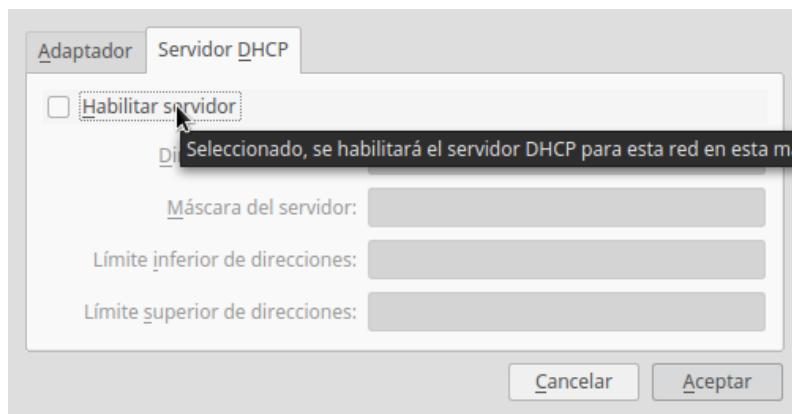


FIGURA 9.7: PASO 5 - VERIFICAR QUE LA OPCIÓN DHCP SE ENCUENTRA DESACTIVADA

### 9.3.2 CONFIGURACIÓN DE LA MÁQUINA VIRTUAL

Una vez declarada la red «solo anfitrión», se debe configurar un nuevo adaptador para nuestra máquina virtual. **Esta configuración puede ser realizada únicamente, si la máquina virtual se encuentra apagada.** Para ello, se accede a la configuración de la VM en la figura 9.8 y se define el adaptador secundario como miembro de la red creada en la sección 9.3.1 en página 181, figura 9.9.

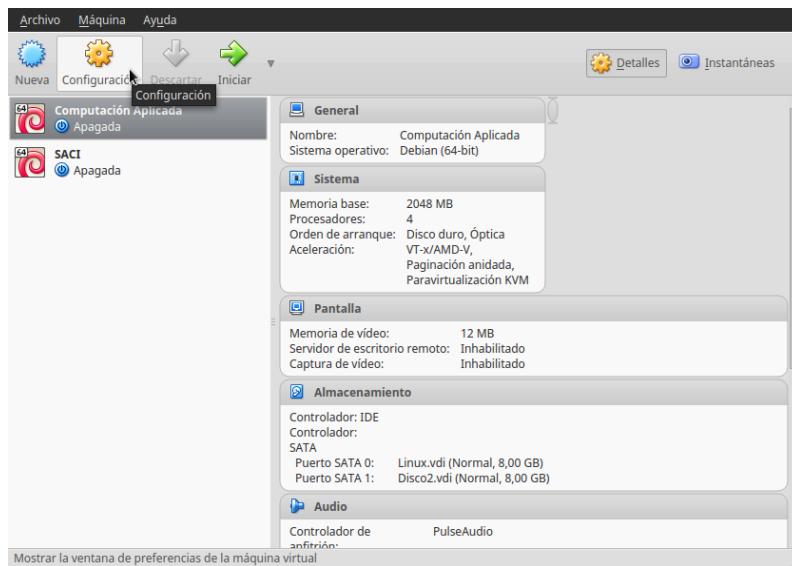


FIGURA 9.8: ACCESO A LA CONFIGURACIÓN DE LA VM

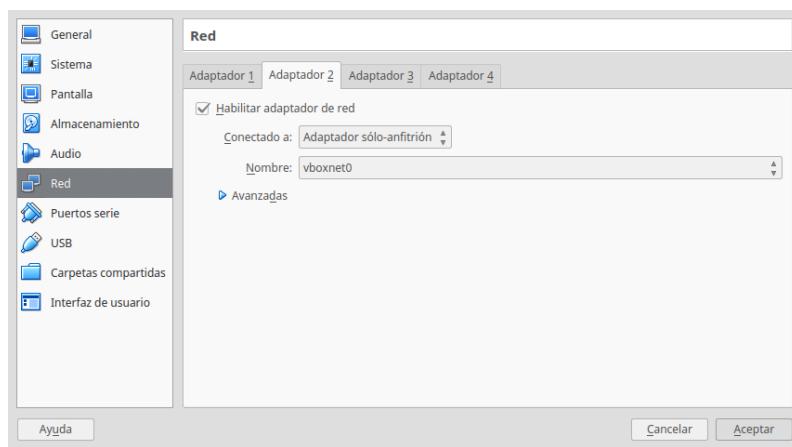


FIGURA 9.9: DEFINIR UN ADAPTADOR SECUNDARIO, ANEXÁNDOLO A LA RED SOLO-ANFITRIÓN

## 9.4 RELEVAR EL HARWARE DE RED DETECTADO

Hay varias maneras para relevar el hardware que el kernel haya detectado.

- ▶ Con `CMD lspci|grep -i net`<sup>10</sup>.

<sup>10</sup>Requiere tener instalado el paquete «pciutils»

- ▶ Revisando con **CMD dmesg**, la salida del kernel que generó cuando booteo.
- ▶ Listando el contenido de `/sys/class/net`. En esta ubicación de hecho también figura el nombre que el sistema operativo le asigna al hardware (`eth0`, `lo`, `enp7s0`, etc.)

## 9.5 CONFIGURAR UNA INTERFAZ ETHERNET

La mayor parte de la configuración de la red se puede hacer desde el archivo de configuración `interfaces` en `/etc/network/interfaces`. Aquí, se le puede dar a la tarjeta de red una dirección IP estática (o usar DHCP), establecer la información de enrutamiento, configurar el enmascaramiento IP y definir las rutas por defecto. Se debe recordar añadir la cláusula «auto» a las interfaces que se deseen activar durante el arranque del sistema.

Por ejemplo:

```

/etc/network/interfaces

1 auto eth0
2 allow-hotplug eth1
3
4 iface eth0 inet dhcp
5
6 iface eth1 inet static
7   address 192.168.1.2/24
8   gateway 192.168.1.1

```

- ▶ **auto** → Las líneas que comienzan con la palabra «auto» se utilizan para identificar las interfaces físicas que se mostrarán cuando `ifup` se ejecute con la opción «-a». (Los scripts de arranque del sistema usan esta opción).
- ▶ **allow** → Las líneas que comienzan con «allow-» se usan para identificar interfaces que varios subsistemas deberían activar automáticamente. Tengase en cuenta que «allow-auto» y «auto» son sinónimos.
- ▶ **iface** → Las líneas que definen las interfaces lógicas comienzan con la palabra «iface» seguida del nombre de la interfaz lógica. El nombre de la interfaz es seguido por el nombre de la familia de direcciones que utiliza la interfaz. Esto será «inet» para las redes TCP/IP. También hay algún soporte para las redes IPX («ipx») y las redes IPv6 («inet6»). Se pueden dar opciones adicionales en líneas posteriores. Las opciones disponibles dependen de la familia y el método. Algunas son:
  - ▶ **address** → Dirección. (Obligatorio)
  - ▶ **netmask** → Máscara de red. (Obligatorio)
  - ▶ **gateway address** → Gateway por defecto.

Véase **CMD man interfaces** para las opciones.

## 9.6 DHCP

DHCP<sup>11</sup> es un protocolo muy conocido que ejecuta la configuración automática de los parámetros de la red, por ejemplo:

- ▶ La dirección IP.
- ▶ La máscara de red.
- ▶ El gateway por defecto.
- ▶ Los servidores DNS.

Para que DHCP funcione, tiene que haber en la red un servidor de DHCP que sea «autoritativo».

DHCP emplea un modelo de servicio sin conexión, utilizando el Protocolo de datagramas de usuario (UDP). Se implementa con dos números de puerto UDP para sus operaciones, que son los mismos que para el protocolo bootstrap (BOOTP). El número de puerto UDP 67 es el puerto de destino de un servidor, y el cliente usa el número de puerto UDP 68.

Las operaciones de DHCP se dividen en cuatro fases:

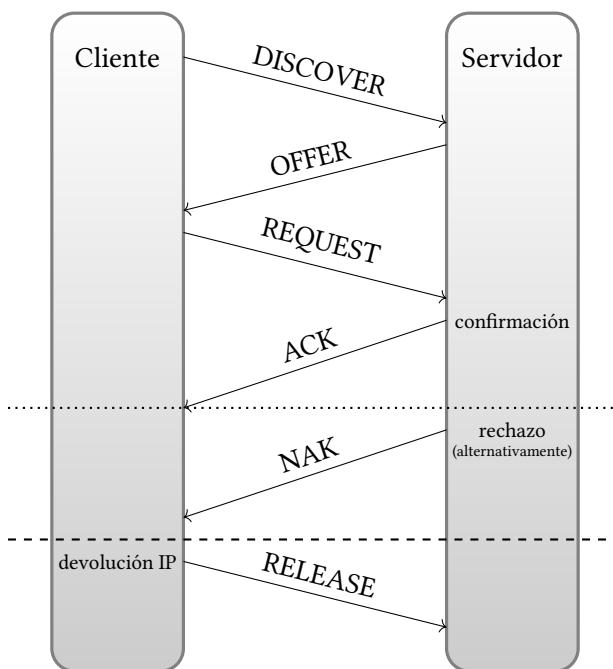


FIGURA 9.10: DIAGRAMA DE TIEMPOS DE UNA NEGOCIACIÓN DHCP

- 1 Descubrimiento del servidor «DISCOVER».
- 2 Oferta de arrendamiento de IP «OFFER».
- 3 Solicitud de arrendamiento «REQUEST» de IP.
- 4 Reconocimiento «ACK» de arrendamiento de IP.

<sup>11</sup>«Dynamic Host Configuration Protocol» Protocolo de Configuración Dinámica del Host.

Estas etapas a menudo se abrevian como «DORA» para descubrimiento, oferta, solicitud y reconocimiento.

La operación DHCP comienza con clientes que transmiten una solicitud. Si el cliente y el servidor están en subredes diferentes, se puede usar un «DHCP Helper» o un «Agente de retransmisión DHCP». Los clientes que solicitan la renovación de un contrato de arrendamiento existente pueden comunicarse directamente a través de unidifusión UDP, que el cliente ya tiene una dirección IP establecida en ese punto. Además, hay un indicador BROADCAST que el cliente puede usar para indicar de qué manera (difusión o unidifusión) puede recibir el DHCPOFFER. Por lo general, el DHCPOFFER se envía a través de unidifusión.

#### 9.6.1 DESCUBRIMIENTO

El cliente DHCP difunde un mensaje en la subred de la red, utilizando la dirección de destino 255.255.255.255 (difusión limitada) o la dirección de difusión de subred específica (difusión dirigida). Un cliente DHCP también puede solicitar su última dirección IP conocida. Si el cliente permanece conectado a la misma red, el servidor puede otorgar la solicitud. De lo contrario, depende si el servidor está configurado como autorizado o no. Un servidor autorizado niega la solicitud, lo que hace que el cliente emita una nueva solicitud. Un servidor no autorizado simplemente ignora la solicitud, lo que lleva un tiempo de espera que dependerá de la configuración del cliente para que el éste expire la solicitud y solicite una nueva dirección IP.

#### 9.6.2 OFRECIMIENTO

Cuando un servidor DHCP recibe un mensaje de un cliente, como por ejemplo una solicitud de arrendamiento de dirección IP, el servidor DHCP reserva una dirección IP para el cliente y hace una oferta de arrendamiento enviando un mensaje al cliente. Este mensaje contiene la identificación del cliente (tradicionalmente una dirección MAC), la dirección IP que ofrece el servidor, la máscara de subred, la duración del arrendamiento y la dirección IP del servidor DHCP que realiza la oferta. El servidor DHCP también puede tomar nota de la dirección MAC a nivel de hardware en la capa de transporte subyacente: de acuerdo con los RFC<sup>12</sup> actuales, la dirección MAC de la capa de transporte se puede usar si no se proporciona una ID de cliente en el paquete DHCP.

El servidor DHCP determina la configuración en función de la dirección de hardware del cliente. Aquí el servidor, 192.168.1.1, especifica la dirección IP del cliente (su dirección IP).

#### 9.6.3 SOLICITUD DE ARRENDAMIENTO

En respuesta a la oferta de DHCP, el cliente responde con un mensaje DHCPREQUEST, transmitido al servidor, solicitando la dirección ofrecida. El cliente puede recibir ofertas de DHCP de varios servidores, pero solo aceptará una. Según la opción de identificación del servidor requerida en la solicitud y la transmisión de mensajes, se informa a los servidores qué oferta ha sido aceptada. Cuando otros servidores DHCP reciben este mensaje, retiran cualquier oferta que hayan hecho al cliente y devuelven la dirección IP ofrecida al conjunto de direcciones disponibles.

---

<sup>12</sup>Requests for Comments

#### 9.6.4 CONFIRMACIÓN

Cuando el servidor DHCP recibe el mensaje DHCPREQUEST del cliente, el proceso de configuración entra en su fase final. La fase de reconocimiento implica enviar un paquete DHCPACK al cliente. Este paquete incluye la duración de la concesión y cualquier otra información de configuración que el cliente haya solicitado. En este punto, se completa el proceso de configuración de IP.

El protocolo espera que el cliente DHCP configure su interfaz de red con los parámetros negociados.

### 9.7 UTILIZAR DHCP PARA CONFIGURAR LA INTERFAZ AUTOMÁTICAMENTE

Si se está utilizando DHCP, entonces sólo se necesita agregar lo siguiente:

#### DHCP en /etc/network/interfaces

```
1 auto eth0
2 allow-hotplug eth0
3 iface eth0 inet dhcp
```

### 9.8 CONFIGURAR LA INTERFAZ MANUALMENTE

Si se está configurando manualmente entonces lo siguiente, definirá la puerta de enlace por defecto (red, difusión o broadcast y gateway o puerta de enlace son opcionales):

#### Interfaz manual en /etc/network/interfaces

```
1 auto eth0
2 iface eth0 inet static
3 address 192.0.2.7
4 netmask 255.255.255.0
5 gateway 192.0.2.254
```

### 9.9 APLICANDO LOS CAMBIOS

Para que los cambios se apliquen, hay dos maneras:

#### 9.9.1 REINICIANDO LOS SERVICIOS DE NETWORKING

Este modalidad sirve cuando se modifica el archivo de configuración /etc/network/interfaces, y se desea que se apliquen esos cambios que **quedan persistentes**.



```
root@vorlonhost:~# /etc/init.d/networking restart
[ ok ] Restarting networking (via systemctl): networking.service.
root@vorlonhost:~# █
```

### 9.9.2 DESACTIVANDO Y ACTIVANDO ADMINISTRATIVAMENTE LA PLACA DE RED

Es cuando se modifica por línea de comandos la configuración de la placa de red, por medio del comando **CMD ifconfig**. **Estos cambios NO quedan persistentes.**



```
root@vorlonhost:~# ifconfig eth0 192.68.47.23 netmask 255.255.255.0
root@vorlonhost:~# ifconfig eth0 down && ifconfig eth0 up
root@vorlonhost:~# █
```

## 9.10 ACTIVAR UNA INTERFAZ SIN DIRECCIÓN IP

Para crear una interfaz de red sin dirección IP, se utiliza la cláusula «manual» y se usan las órdenes «pre-up» y «post-down» para activar o desactivar la interfaz.

/etc/network/interfaces

```
1 iface eth0 inet manual
2 pre-up ifconfig $IFACE up
3 post-down ifconfig $IFACE down
```

Antes de que un ordenador pueda conectarse a un recurso de una red externa (sea, por ejemplo, un servidor web), debe tener los medios para convertir cualquier nombre alfanumérico (p.e. [wiki.debian.org](http://wiki.debian.org)) en direcciones de red numéricas (p.e. 140.211.166.4). La librería C y otras bibliotecas de resolución buscan `/etc/resolv.conf` para tener una lista de servidores de nombres. En el caso más simple, ése es el archivo a editar para poner la lista. Pero debe notarse que otros programas para configuraciones dinámicas es muy probable que sobrescriban estos ajustes:

- ▶ El programa `resolvconf`.
- ▶ El demonio `network-manager`.
- ▶ Clientes `DHCP`.

En la mayoría de las situaciones, el archivo a editar es el de configuración para cada programa.

En las más complejas situaciones, utilizar resolvconf es verdaderamente el modo de hacer las cosas, a pesar de que en configuraciones más simples sea probablemente abrumador.

## 9.11 PRUEBA

Si todo anduvo bien, se puede verificar que nuestra configuración funcionó correctamente si la interfaz secundaria definida puede ser alcanzada desde el sistema operativo «host». Esto se verifica con el comando `ping`.

```
ttyS9@vorlonhost:~$ ping 192.168.56.10
PING 192.168.56.10 (192.168.56.10) 56(84) bytes of data.
64 bytes from 192.168.56.10: icmp_seq=1 ttl=64 time=5.62 ms
64 bytes from 192.168.56.10: icmp_seq=2 ttl=64 time=0.428 ms
64 bytes from 192.168.56.10: icmp_seq=3 ttl=64 time=0.436 ms
ttyS9@vorlonhost:~$
```

## 9.12 CIFRADO

### 9.12.1 INTRODUCCIÓN

Las técnicas de cifrado consisten en manipular la información para asegurar las propiedades enunciadas en el cuadro 9.4.

Propiedad	Descripción
Confidencialidad	No puede acceder a la información nadie que no sea su legítimo destinatario.
Integridad	La información no puede ser alterada (sin ser esto detectado) en el tránsito desde el emisor hacia el destinatario.
No repudio	El creador o emisor de la información no puede dejar de reconocer su autoría.
Autenticación	Tanto el emisor como el receptor pueden confirmar la identidad de la otra parte involucrada en la comunicación.

CUADRO 9.4: PROPIEDADES DE LA TÉCNICAS DE CIFRADO.

En particular, para lo que interesa, se prestará atención a tipos de algoritmos: los de cifrado simétrico (con clave compartida) y los de cifrado asimétrico (con clave pública y clave privada).

<sup>12</sup>El protocolo de resolución de direcciones (ARP, del inglés Address Resolution Protocol) es un protocolo de comunicaciones de la capa de enlace, responsable de encontrar la dirección de hardware (Ethernet MAC) que corresponde a una determinada dirección IP.[wiki:arp]

### 9.12.2 CIFRADO SIMÉTRICO (CON CLAVE COMPARTIDA)

Esta técnica consiste en el uso de una clave que es conocida tanto por el emisor como por el receptor (y, se supone, por nadie más). Tal cual como puede apreciarse en la figura 9.11.

- 1 Dos personas,  $E$  y  $R$  conocen la clave:  $K$ .
- 2 El emisor,  $E$ , desea transmitir el mensaje «Mensaje» a  $R$ .
- 3 Para ello, utilizando determinado algoritmo de cifrado simétrico y la clave  $K$ , genera el mensaje «Mensaje $_K$ », que es transmitido a  $R$ .
- 4 Éste, aplicando la misma clave y el algoritmo inverso, obtiene nuevamente el mensaje original.

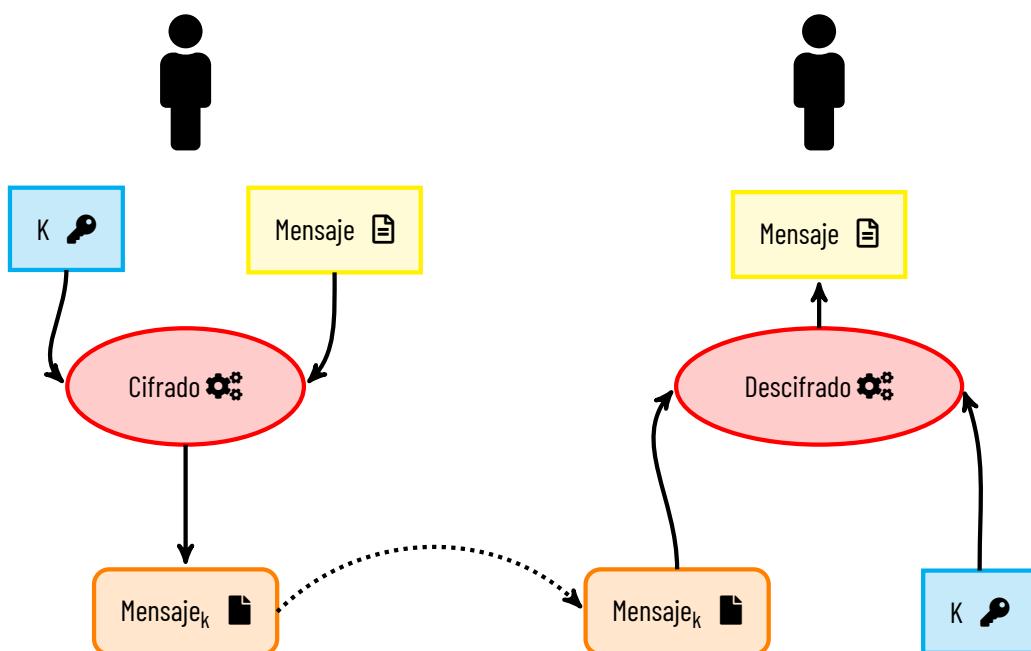


FIGURA 9.11: ESQUEMA DEL CIFRADO SIMÉTRICO

Al respecto del algoritmo utilizado, es deseable que cumpla varias propiedades, entre ellas:

- ▶ Que sea muy difícil descifrar el mensaje sin conocer la clave.
- ▶ Que la publicidad del algoritmo de cifrado/descifrado no tenga influencia en la seguridad del mismo.
- ▶ Que una mínima alteración de la clave, produzca grandes cambios en el mensaje cifrado.

### 9.12.3 CIFRADO ASIMÉTRICO (CON CLAVE PÚBLICA Y PRIVADA)

Las técnicas de cifrado asimétrico se basan en el uso de dos claves: **una pública y otra privada**.

En el ejemplo de la figura 9.12 de la siguiente página:

- 1  $R$  posee dos claves:  $K_R$  (su clave privada, conocida sólo por él) y  $K_{R_p}$  (su clave pública, conocida por cualquiera).
- 2  $E$  desea transmitir el mensaje «Mensaje» a  $R$ , de manera que nadie, excepto este último, pueda conocer su contenido.
- 3 Para ello, utilizando la clave  $K_{R_p}$  y un algoritmo de cifrado asimétrico, genera el mensaje «MensajeKR», el cual es transmitido.
- 4 Luego,  $R$ , utilizando el algoritmo inverso y su clave privada  $K_R$ , reproduce el mensaje original. El algoritmo debe garantizar que nadie que no conozca la clave  $K_R$  pueda obtener el mensaje original.

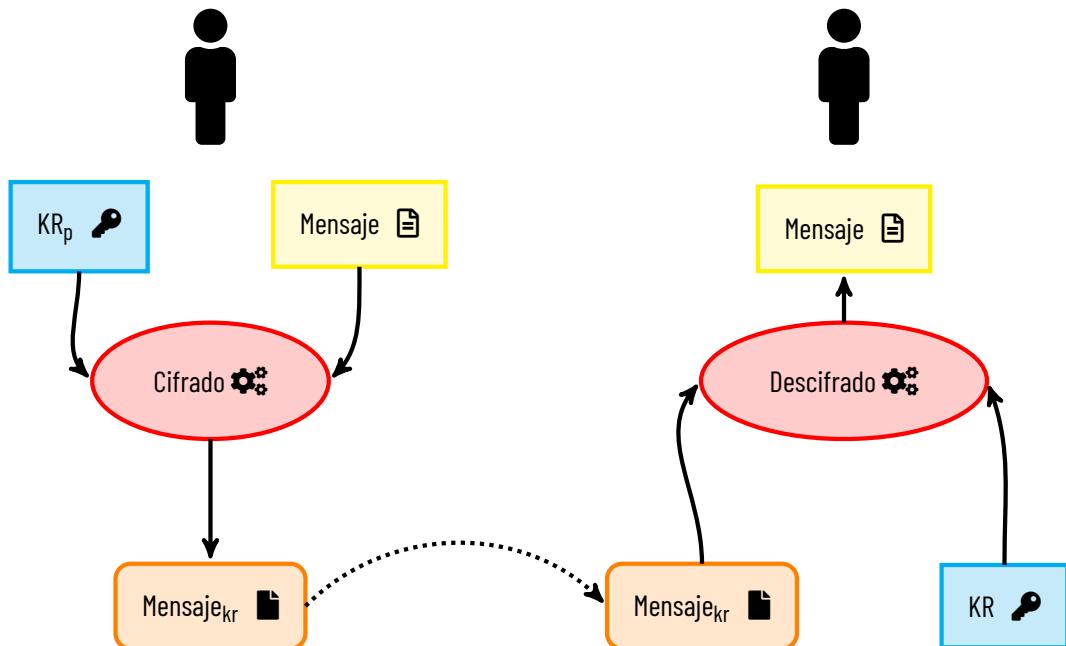


FIGURA 9.12: EJEMPLO DE CIFRADO ASIMÉTRICO.

La complejidad computacional de los algoritmos de cifrado asimétrico hace que sea muy costoso el uso de un esquema de «doble cifrado», por lo cual generalmente se utiliza un esquema mixto, combinando cifrado asimétrico y simétrico.

## 9.13 SECURE SHELL (SSH)

**SSH, es el nombre de un protocolo y del programa cliente que lo implementa.** Sirve para acceder servidores y manejarlos totalmente mediante un intérprete de comandos (siempre y cuando se cuenten con los privilegios correspondientes, o haya un «bug».)

## Ejecución remota de aplicaciones gráficas



Configurado apropiadamente, SSH, permite redirigir el tráfico del subsistema gráfico (X Windows) del servidor ssh, a la terminal cliente. Esto permite que se pueda ejecutar remotamente (y visualizar localmente) cualquier tipo de aplicación gráfica.

Además de la conexión a otros dispositivos, SSH permite copiar datos de forma segura (tanto archivos sueltos como simular sesiones FTP cifradas), gestionar claves RSA para no escribir claves al conectar a los dispositivos y pasar los datos de cualquier otra aplicación por un canal seguro tunelizado mediante SSH.

Hay dos formas de invocación del comando `ssh`:

**SIN** `ssh <host>` → Autentica en el servidor destino con el mismo usuario con el que se está logueado en el origen.

**SIN** `ssh <usuario@host>` → Autentica en el servidor destino con el usuario que se especifica a la izquierda del «@».

### 9.13.1 DESCRIPCIÓN GENERAL DEL FUNCIONAMIENTO DE SSH

Al conectarse un cliente de SSH con el servidor, se realizan los siguientes pasos:

- 1 El cliente abre una conexión TCP al puerto 22 del host servidor
- 2 El cliente y el servidor acuerdan la versión del protocolo a utilizar, de acuerdo a su configuración y capacidades.
- 3 El servidor posee un par de claves pública/privada de RSA<sup>13</sup> <sup>14</sup> (llamadas «claves de host»). El servidor envía al cliente su clave pública.
- 4 El cliente compara la clave pública de host recibida con la que tiene almacenada, para verificar su autenticidad. Si no la conociera previamente, pide confirmación al usuario para aceptarla como válida.
- 5 El cliente genera una clave de sesión aleatoria y selecciona un algoritmo de cifrado simétrico.
- 6 El cliente envía un mensaje contenido la clave de sesión y el algoritmo seleccionado, cifrado con la clave pública de host del servidor usando el algoritmo RSA.
- 7 En adelante, para el resto de la comunicación se utilizará el algoritmo de cifrado simétrico seleccionado y clave compartida de sesión.

<sup>13</sup>En criptografía, RSA (Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública desarrollado en 1977. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente. La seguridad de este algoritmo radica en el problema de la factorización de números enteros. Los mensajes enviados se representan mediante números, y el funcionamiento se basa en el producto, conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto. Actualmente estos primos son del orden de  $10^{200}$ , y se prevé que su tamaño crezca con el aumento de la capacidad de cálculo de los ordenadores. [\[wiki:rsa\]](#)

<sup>14</sup>Este tema lo trata la materia Matemática Discreta.

- 8 Luego se realiza la autenticación del usuario. Aquí pueden usarse distintos mecanismos. Más adelante se analizarán los más importantes.
- 9 Finalmente se inicia la sesión, por lo general, interactiva.

### 9.13.2 MÉTODOS DE AUTENTICACIÓN DE USUARIOS

Existen varios métodos que pueden utilizarse para autenticar usuarios. Aunque son mutuamente excluyentes, tanto el cliente como el servidor pueden soportar varios de ellos. En el momento de la autenticación se aplicarán los métodos que ambos soporten, siguiendo un orden determinado.

- ▶ Autenticación con contraseña.
- ▶ Autenticación con clave pública.

#### AUTENTICACIÓN CON CONTRASEÑA

Es el método más simple de autenticación. El cliente solicita al usuario el ingreso de una contraseña y la misma es enviada al servidor, el cuál la validará utilizando los mecanismos configurados en el sistema; Generalmente, utilizará la autenticación del sistema \*nix para validar la contraseña contra el contenido del archivo /etc/shadow, de la misma manera que lo hace el **CMD** `login`.

Las desventajas de este sistema son:

- ▶ El usuario debe tipear su contraseña cada vez que se conecta al servidor.
- ▶ La contraseña del usuario es enviada hacia el servidor.

Por ejemplo, la sintaxis de **CMD** `ssh` para ingresar como «root» a al servidor cuya dirección IP es 192.168.56.10, sería:

**CMD** `ssh root@192.168.56.10`

Si es la primera vez que se ingresa, dirá que no pueda establecer la identidad del servidor en cuestión y se presentara una «huella digital» única que identifica a ese server (emitida por éste). Para continuar, se escribe «yes» lo cual agrega el certificado digital del servidor remoto a una pequeña «base» de certificados conocidos, la cual esta ubicado en `.ssh/known_hosts`.

Luego pedirá la password del usuario especificado (en este caso, `root`):



```
ttyS9@vorlonhost:~$ ssh root@192.168.56.10
root@192.168.56.10's password: █
```

De haber sido correcta la password ingresada, se mostrará la consola del servidor con el se ha realizado la conexión.



```
Linux debian 4.9.0-3-amd64 #1 SMP Debian 4.9.30-2+deb9u2 (2017-06-26)
x86_64

Last login: Sat Jul 29 10:20:28 2017
root@shadowhost:~#
```

### AUTENTICACIÓN CON CLAVE PÚBLICA

El usuario debe tener un par de claves pública/privada, y la clave pública debe estar almacenada en el servidor. Luego de establecida la conexión, el servidor genera un número aleatorio llamado «desafío» o «challenge» que es cifrado con la clave pública del usuario usando RSA o DSA. El texto cifrado es enviado al cliente, que debe descifrarlo con la clave privada correspondiente y devolverlo al servidor, demostrando de esta manera que el usuario es quien dice ser.

#### Automatización remota de procesos



El método de autenticación por clave pública es muy utilizado para la automatización de procesos.

Las ventajas de este método respecto del anterior son:

- ▶ El usuario no debe tipear (ni recordar) ninguna contraseña.
- ▶ En ningún caso, la contraseña o la clave privada del usuario son enviadas al servidor.

No bien se establece la conexión, y antes de la autenticación del usuario, queda establecido un canal cifrado seguro. En adelante, todo el tráfico de la sesión será prácticamente<sup>15</sup> indescifrable.

Para realizar esta autenticación, primero se tiene que cargar la clave pública del servidor cliente, al servidor remoto. Si bien hay mucho métodos, que dependen principalmente de la seguridad del servidor al cual se quiere conectar, se usará uno que automatiza los procesos involucrados: el comando  `ssh-copy-id`; se utiliza el parámetro `-i` para indicarle el archivo que contiene la clave pública del usuario.



```
ttyS9@vorlonhost:~$ ssh-copy-id -i .ssh/id_rsa.pub root@192.168.56.10
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: .ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
➥out any that are already installed
```

<sup>15</sup>Ningún sistema es 100% seguro.

```
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
    ↪prompted now it is to install the new keys
root@192.168.56.10's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@192.168.56.10'" and check to
    ↪make sure that only the key(s) you wanted were added.
ttyS9@vorlonhost:~$ █
```

Luego se intentará ingresar nuevamente y se verá que no se pide la contraseña.



```
ttyS9@vorlonhost:~$ ssh root@192.168.56.10
Linux debian 4.9.0-3-amd64 #1 SMP Debian 4.9.30-2+deb9u2 (2017-06-26) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the individual
files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jul 29 10:22:36 2017 from 192.168.56.1
root@shadowhost:~# █
```

### 9.13.3 CONFIGURACIÓN DE OPENSSH

Los archivos de configuración de OpenSSH se encuentran en `/etc/ssh` o en `/usr/local/etc/`, dependiendo de cómo se haya instalado.

Los dos archivos principales son:

- ▶ **`sshd_config`** → En este archivo se describe la configuración del **servidor** SSH. Cuadro 9.5 de la siguiente página.
- ▶ **`ssh_config`** → En este archivo se describe la configuración del **cliente** SSH. Cuadro 9.6 de la siguiente página.

---

DIRECTIVA	DESCRIPCIÓN
<code>Port</code>	Especifica el puerto TCP que utilizará el servidor. El valor usual es 22.
<code>Protocol</code>	Versión del protocolo a utilizar. Se usará solamente el valor 2.

Cuadro 9.5 – CONTINUACIÓN...

DIRECTIVA	DESCRIPCIÓN
HostKey	Clave privada de RSA o DSA del host. Normalmente las claves de host son generadas en el momento de la instalación de OpenSSH, y el valor de esta opción es /etc/ssh/ssh_host_rsa_key
PubkeyAuthentication	Permite la autenticación mediante clave pública.
AuthorizedKeysFile	Mediante esta opción se indica al servidor en donde están almacenadas las claves públicas de los usuarios. El valor por defecto es .ssh/authorized_keys, e indica que deben buscarse en el archivo authorized_keys, del directorio .ssh dentro del directorio «home» de cada usuario.
PasswordAuthentication	Permite la autenticación mediante contraseñas.
X11Forwarding	Si el valor de esta opción es yes, se habilita el reenvío de X11 a través de la conexión SSH.

CUADRO 9.5: ALGUNAS DIRECTIVAS DEL ARCHIVO /etc/ssh/sshd\_config

DIRECTIVA	DESCRIPCIÓN
Host	Esta opción actúa como un divisor de sección. Puede repetirse varias veces en el archivo de configuración. Su valor, que puede ser una lista de patrones, determina que las opciones subsiguientes sean aplicadas a las conexiones realizadas a los hosts en cuestión. El valor * significa «todos los hosts».
Port	Es el puerto de TCP que utiliza el servidor (normalmente, 22).
Protocol	Es la versión del protocolo utilizada.
PubkeyAuthentication	Autenticación mediante clave pública (se recomienda yes).
IdentityFile	Archivo que contiene la clave pública (en caso de usar RSA, lo usual es ~/ .ssh/id_rsa).
PasswordAuthentication	Autenticación mediante contraseñas (yes o no).
StrictHostKeyChecking	Define que hará el cliente al conectarse a un host del cual no se dispone de su clave pública. El valor no hace que sea rechazara la clave del servidor (y por lo tanto, se aborte la conexión), el valor yes hace que se acepte automáticamente la clave recibida, y el valor ask hace que se pida confirmación al usuario.
Ciphers	Algoritmos de cifrado simétrico soportados para su uso durante la sesión.
ForwardX11	Reenvío de aplicaciones X11.

CUADRO 9.6: SCRIPTS EN /etc/ssh/ssh\_config

Tabla	Descripción
raw	Filtre los paquetes antes que cualquier otra tabla.
filter	Es la tabla por defecto (si no se pasa la opción -t).
nat	Se utiliza para la traducción de dirección de red (por ejemplo, el redirección de puertos).
mangle	Se utiliza para la alteración de los paquetes de red especializados.
security	Se utiliza para reglas de conexión de red «Mandatory Access Control».

CUADRO 9.7: TABLAS DE iptables.

## 9.14 IP TABLES

**iptables**



Es el firewall integrado en el kernel de Linux y que forma parte del proyecto «netfilter». Netfilter es un framework disponible en el núcleo Linux que permite interceptar y manipular paquetes de red. Dicho framework permite realizar el manejo de paquetes en diferentes estados del procesamiento.

**CMD** `iptables` puede ser configurado directamente, como también por medio de un frontend o una GUI<sup>16</sup>. **CMD** `iptables` es usado por IPv4 e IPV6.

### 9.14.1 TABLAS

**CMD** `iptables` cuenta con cinco tablas (Cuadro 9.7), que son zonas en las que una cadena de reglas se puede aplicar:

#### FILTER

La tabla «filter» es usada para permitir o bloquear tráfico, y contiene 3 cadenas «CHAIN»: INPUT, OUTPUT y FORWARD.

#### NAT

Esta tabla es utilizada para crear reglas de traducción de direcciones, y permitir que un paquete con IP privada, pueda salir a Internet con una IP pública, o viceversa. Esta tabla contiene las cadenas:

- **PREROUTING** → Es donde llegan los paquetes antes de procesarse por la tabla de rutas local.
- **POSTROUTING** → Es donde llegan los paquetes una vez han sido procesados por la tabla de rutas.

<sup>16</sup>«Graphic User Interface» Cualquier aplicación con interfaz gráfica.

Target	Descripción
ACCEPT	significa dejar pasar el paquete.
DROP	significa dejar caer el paquete, es decir, descartar y no enviar ninguna respuesta.
QUEUE	significa pasar el paquete al espacio de usuario <sup>17</sup> . Se utiliza junto con programas o utilidades que son ajenos a iptables y se puede usar, por ejemplo, con contabilidad de red, o para aplicaciones específicas y avanzadas que procesan o filtran paquetes.
RETURN	significa dejar de atravesar esta cadena y reanudar en la siguiente regla de la cadena anterior (llamada).

CUADRO 9.8: TARGETS INTEGRADOS.

### 9.14.2 REGLAS

El filtrado de los paquetes de red se basa en reglas «rules», que se especifican por diversas coincidencias «matches», es decir, condiciones que el paquete debe satisfacer para que la regla se puede aplicar, y un objetivo «target», es decir, una acción a tomar cuando el paquete coincide con la condición.

#### Orden en las reglas



Se debe tener presente que las reglas hay un orden, que el se determina cuando las mismas se dan de alta, con -I o -A, **y estas reglas se ejecutan de arriba hacia abajo**

#### TARGET

Se especifican mediante la opción «-j» o «--jump». Los «targets» pueden ser tanto las cadenas definidas por el usuario, como uno de los targets integrados especiales, o una extensión de target. Los targets integrados figuran listados en la tabla 9.8.

#### EXTENSIONES

Las extensiones de «target» son, por ejemplo, «REJECT» y «LOG». Si el target es un target integrado, el destino del paquete es decidido inmediatamente y el procesamiento del paquete en la tabla actual se detiene. Si el target es una cadena definida por el usuario y el paquete supera con éxito esta segunda cadena, se moverá a la siguiente regla de la cadena inicial.

### 9.14.3 COMANDOS DE IPTABLES

El cuadro 9.9 de la siguiente página contiene los comandos para usar con el comando **cmd** `iptables` [andreasson2014].

CMD	Descripción
-A	Agrega la regla definida al final de la cadena.
-D	Borra la regla, ya sea definida con número de orden, o sino especificándola.
-R	Reemplaza la regla especificada.
-I	Inserta la regla especificando el número de orden puntual.
-L	Listas las reglas.
-P	Define la política por defecto de la cadena.
-F	Purga («flush») la cadena en su totalidad. Borra todas las reglas.

CUADRO 9.9: COMANDOS DE IPTABLES.

CMD	Descripción
-p, --protocol	Especifica el protocolo. Si no encuentra lo especificado, se fija en el archivo /etc/protocols.
-s, --src, --source	Especifica el origen del paquete, basado en su dirección IP.
-d, --dst, --destination	Especifica la dirección destino del paquete.
-i, --in-interface	Interface de red por donde ingresó el paquete.

CUADRO 9.10: MATCHES GENERALES.

#### 9.14.4 MATCHES GENERALES

En el cuadro 9.10, figuran los distintos tipos de «matches» que iptables posee para construir las reglas.

#### 9.14.5 MATCHES TCP/UDP

El cuadro siguiente muestra los matches específicos para TCP/UP.

- ▶ **--sport / --source-port** → Especifica puerto de origen.
- ▶ **--dport / --destination-port** → Especifica puerto destino.

#### 9.14.6 TARGETS/JUMPS

Especifican que hacer, cuando un paquete coincide con las reglas especificadas con las opciones arriba mencionadas.

En el cuadro 9.11 de la siguiente página se listan las acciones que se pueden realizar cuando el match da verdadero.

---

ACCEPT	Acepta el paquete.
DROP	Rechaza el paquete sin avisar en absoluto.
LOG	Escribe en el log, para futuras auditorías.
REJECT	Rechaza el paquete, pero informándolo al destinatario. <sup>18</sup>

---

CUADRO 9.11: TARGETS/JUMPS.

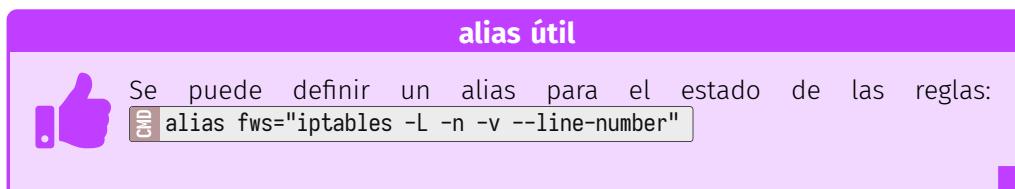
**Salida de log**

```

1 INPUT:DROP:IN=enp0s8 OUT=MAC=ff:ff:ff:ff:ff:ff:0a:00:27:00:00:00:08:00
   ↳ SRC=192.168.56.1 DST=192.168.56.255 LEN=185 TOS=0x00 PREC=0x00 TTL=64 ID=5007
   ↳ DF PROTO=UDP SPT=17500 DPT=17500 LEN=165
2 INPUT:DROP:IN=enp0s8 OUT= MAC=ff:ff:ff:ff:ff:ff:0a:00:27:00:00:00:08:00
   ↳ SRC=192.168.56.1 DST=192.168.56.255 LEN=185 TOS=0x00 PREC=0x00 TTL=64 ID=7811
   ↳ DF PROTO=UDP SPT=17500 DPT=17500 LEN=165

```

FIGURA 9.13: VISTA DEL ARCHIVO SYSLOG.



## 9.15 REGISTRAR LAS INCIDENCIAS DE IPTABLES

Se pueden registrar las incidencias de iptables en el log del sistema. Para esto hay que asegurarse que las últimas dos reglas de la cadena sean LOG y DROP.

```

root@vorlonhost:~#
iptables -A INPUT -j LOG --log-prefix "INPUT:DROP:" --log-level 6
root@vorlonhost:~# iptables -A INPUT -j DROP
root@vorlonhost:~# █

```

En la figura 9.13 se muestra un extracto de un log por medio de **CMD** tail -f /var/log/syslog .

## 9.16 EJEMPLOS

Se eliminan las reglas existentes con **CMD** iptables -F .

Estado de las cadenas luego de la ejecución:

```
root@vorlonhost:~# iptables -L
Chain INPUT (policy ACCEPT)
target    prot opt source          destination
Chain FORWARD (policy ACCEPT)
target    prot opt source          destination
Chain OUTPUT (policy ACCEPT)
target    prot opt source          destination
root@vorlonhost:~# █
```

Se establece la política por defecto:

```
root@vorlonhost:~#
iptables -A INPUT -j LOG --log-prefix "INPUT:DROP:" --log-level 6
root@vorlonhost:~# iptables -A INPUT -j DROP
root@vorlonhost:~# █
```

Estado de las cadenas:

```
root@vorlonhost:~# iptables -L
Chain INPUT (policy ACCEPT)
target    prot opt source          destination
LOG      all  --  anywhere        anywhere        LOG level info
      ↳prefix "INPUT:DROP:"
DROP    all  --  anywhere        anywhere
Chain FORWARD (policy ACCEPT)
target    prot opt source          destination
Chain OUTPUT (policy ACCEPT)
target    prot opt source          destination
root@vorlonhost:~# █
```

El localhost se permite (por ejemplo conexiones locales a mysql<sup>19</sup>)

Permitir trafico a la interfaz local

**CMD** `iptables -I INPUT -i lo -j ACCEPT`

Permitir trafico al puerto 22 (ssh):

**CMD** `iptables -I INPUT -p tcp --dport 22 -j ACCEPT`

Permitir trafico al puerto 80 (www):

**CMD** `iptables -A INPUT -p tcp --dport 80 -j ACCEPT`

Estado de las tablas:

```

root@vorlonhost:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT    all  --  anywhere             anywhere
ACCEPT    tcp  --  anywhere             anywhere             tcp dpt:ssh
LOG       all  --  anywhere             anywhere           LOG level info
      ↳prefix "INPUT:DROP:"
DROP     all  --  anywhere             anywhere
ACCEPT    tcp  --  anywhere             anywhere           tcp dpt:http

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@vorlonhost:~# █

```

¿Que se puede notar? Al usar el parámetro A de iptables, la regla del puerto 80 se agregó al final, es decir pasando el «DROP» que es la regla por defecto.

### Orden en las reglas II



Es un error común, que agreguen reglas luego de la que es por defecto, y como las reglas se «ejecutan» de arriba hacia abajo, la regla que corresponde al puerto 80, **nunca será ejecutada**.

<sup>19</sup>MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.[\[wiki:mysql\]](#)

```

root@vorlonhost:~# fws
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source        destination
1    565  42104 ACCEPT     tcp   --  *       *       0.0.0.0/0    0.0.0.0/0
    ↳      tcp dpt:22
2    189  29565 LOG        all   --  *       *       0.0.0.0/0    0.0.0.0/0
    ↳      LOG flags 0 level 6 prefix      "INPUT:DROP:"
3    189  29565 DROP       all   --  *       *       0.0.0.0/0    0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source        destination

Chain OUTPUT (policy ACCEPT 455 packets, 72080 bytes)
num  pkts bytes target     prot opt in     out      source        destination
root@vorlonhost:~# █

```

### Persistencia de las reglas



Las reglas de `iptables`, se encuentran almacenadas en memoria RAM, es decir que de reiniciarse el sistema operativo, las mismas se pierden. Para evitar esto se deben utilizar los comandos `CMD iptables-save` e `CMD iptables-restore`.

#### 9.16.1 GUARDAR LA CONFIGURACIÓN DE `iptables`

`iptables` se configura en el «kernel», en consecuencia, al reniciar el sistema operativo, toda la configuración se pierde, es decir no queda persistente.

Para dejar persistente la configuración, se utilizan dos comandos:

COMANDO DE BASH	DESCRIPCIÓN
<code>CMD iptables-save</code>	Imprime por stdOut, la información correspondiente a <code>iptables</code> .
<code>CMD iptables-restore</code>	Configura <code>iptables</code> acorde a la información provista por un archivo o stdIn.

CUADRO 9.12: GUARDAR LA CONFIGURACIÓN DE `iptables`

```
root@vorlonhost:~# iptables-save
# Generated by iptables-save v1.4.21 on Fri Nov  2 07:36:36 2018
*filter
:INPUT DROP [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [5:512]
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
COMMIT
# Completed on Fri Nov  2 07:36:36 2018
root@vorlonhost:~# █
```

Para almacenarlo en un archivo, se debe redirigir stdOut.

CMD `iptables-save > myfw.cfg`

Para recuperar la información, se toma desde un archivo por stdIn.

CMD `iptables-restore < myfw.cfg`

Para planificar que estos comandos se realicen, se pueden utilizar los scripts de arranque que se ve en la sección 8.2 o mediante «crontab», que se ve en la sección 10.13.



# 10 Automatización

## 10.1 SCRIPTING

Si ya posee conocimientos de programación, esta sección le resultará bastante sencilla y puede ser omitida.

### 10.1.1 ¿QUE ES UN PROGRAMA?

Para contestar esto, se decidió a utilizar una analogía con algo bien conocido: «La Cocina».

### 10.1.2 LA COCINA

- ▶ Cocina → Computadora.
- ▶ Cocinero → Procesador.
- ▶ Mesada → Memoria.
- ▶ Receta → Programa.
- ▶ Ingredientes → Datos.

Lo importante de esta analogía es: el programa dictamina como se trabaja con los datos, al igual que un cocinero siguiendo las instrucciones de una receta para procesar los ingredientes.

El script no es nada mas y ni nada menos que un programa.

Existen varios lenguajes para escribir programas. Una forma de clasificarlos es en función de su «distancia» al CPU.

- ▶ **Bajo Nivel** → Los lenguajes de bajo nivel permiten un mayor control, con un detalle mas fino sobre las tareas que se realizan, pero esto conlleva que una operación sencilla implica programar muchas instrucciones. Las instrucciones de estos lenguajes están mas cercanos a la electrónica digital que gobierna el CPU. Algunos ejemplos de Bajo Nivel, son: Assembler, Lenguaje (o código de máquina), C<sup>1</sup>, C#
- ▶ **Alto Nivel** → Son aquellos lenguajes que con una sola instrucción realizan muchas operaciones (que no es necesario que los usuarios deban programar). Las instrucciones de estos lenguajes son mas parecidas al lenguaje coloquial inglés. Ejemplos son: PHP, Python, Java, .NET, nodeJS y lenguajes de scripting en general.

---

<sup>1</sup>Algunos consideran que este lenguaje es de medio nivel, ya que retiene características de alto nivel y de bajo nivel.

## 10.2 ¿QUÉ ES UN SCRIPT?

10



Es un archivo de texto que contiene una sucesión de comandos de **shell** que pueden ejecutar diversas tareas de acuerdo al contenido del texto del script.

Por medio de un script se pueden automatizar muchas acciones para alguna necesidad particular o para la administración de sistemas. El script debe escribirse en un orden lógico pues Bash (el shell) ejecutará el código **en el orden en que se escriben las líneas**, de la misma forma que cuando se realiza una tarea cualquiera por una persona, por ejemplo; primero hay que colocar la escalera en posición y luego subirla.

Cuando se ejecuta un shell script, realmente lo que se hace es lanzar un shell (**CMD bash**, **CMD ksh**, **CMD csh**, etc.) que se encarga de interpretar y ejecutar las órdenes contenidas en el archivo del script.

Para ejecutar un script se puede hacer de varias maneras:

- 1 Dándole permisos de ejecución y lanzarlo escribiendo el path donde se encuentra y el nombre. Por ejemplo si se tiene un script que se llama «check\_tcp» y que se encuentra en el directorio /opt/scripts, el comando para ejecutarlo (una vez dados los permisos de ejecución) sería: **CMD /opt/scripts/check\_tcp**
- 2 Dándole permisos de ejecución y situarlo en alguno de los directorios de la variable PATH (por ejemplo en /usr/local/bin). Es similar al anterior, pero para lanzarlo no hay que escribir la ruta de donde se encuentra, ya que es suficiente con solo su nombre.
- 3 Ejecutando explícitamente el «shell» y pasándole por parámetro el nombre del script (no son necesarios permisos de ejecución). En 2 sería: **CMD bash /opt/scripts/check\_tcp**

La primera línea que se suele poner en un script, que no es obligatoria para el caso 3, es la que le indica al sistema cuál es el intérprete que debe utilizarse para ejecutar el script (si no se pone nada, se utilizará el shell por defecto que generalmente suele ser «bash»).

### «Magic Number»



Tanto en \*nix, como en GNU/Linux los tipos de archivo **se identifican por sus dos primeros bytes, y no por la extensión** (como otros sistemas operativos =). A estos dos primeros bytes, en el caso de los scripts «#!», se los denomina «SheBang» o «ShaBang». Nótese en que los lenguajes de scripting, el «#» es comentario, osea que va a ser ignorado por el intérprete del mismo script durante la ejecución.

Variable	Descripción
\$0	Nombre del script.
\${1}....\${n}	Variables que almacenan los <i>n</i> argumentos (opciones) proporcionados al script. Las llaves «{}» se pueden suprimir, si la cantidad de argumentos como máximo, nueve.
\$#	Variable que contiene el total de los argumentos proporcionados.
\$*	Conjunto de los argumentos.
\$?	Valor de ejecución del comando anterior «return code», si es cero es que el comando anterior se ejecutó sin errores, de lo contrario hubo algún error.
\$\$	Identifica el PID del script.
\$!	Identifica el último proceso arrancado en background.

CUADRO 10.1: VARIABLES INTRÍNSECAS DE BASH.

## 10.3 VARIABLES

Es impensable elaborar scripts de bash sin el uso de las variables. Una variable es una estructura de texto (una letra, un número o sucesiones de ellos) que representa alguno de los elementos que varían en valor y/o significado en el entorno del shell, sirviendo como elemento básico de entrada/salida de valores a y desde los comandos en su ejecución consecutiva. Para **referenciar** una variable se utiliza el carácter especial \$ precediendo al nombre de la variable.

### Error de programación



Es un error común de programación usar el símbolo \$, cada vez que se habla de una variable. Al contrario que en otros lenguajes, como PHP, **el símbolo \$, solo se usa cuando se referencia una variable, y no cuando se la define.**

### 10.3.1 VARIABLES INTRÍNSECAS DE BASH

Estas son elaboradas por el propio bash y figuran en el cuadro 10.1.

### 10.3.2 VARIABLES CREADAS POR EL PROGRAMADOR

Las variables pueden ser creadas en cualquier momento, pero siempre antes de su utilización de manera muy simple, se escribe, tal cual figura a continuación.

**SM** `nombre_variable=valor_variable`

# 10

## Buena Práctica



En bash scripting, es considerada buena práctica, nombrar las variables totalmente en mayúsculas.

En cualquier momento posterior a la creación si se coloca \$nombre\_variable dentro del entorno de la «shell» el sistema colocará allí valor\_variable.

**CMD** `SALUDO=Bienvenido`

En cualquier momento posterior si se pone \$SALUDO, bash colocará ahí «Bienvenido».

Una variable también puede ser la salida de un comando si se coloca al principio y final del mismo, un acento invertido «`».

**SIN** `SALIDA=`comando``

Le indicará al sistema que donde se escriba \$SALIDA debe poner la salida de ese comando. Es práctica común utilizar mayúsculas para las variables a fin de identificarlas fácilmente dentro del script.

Cuando se ejecutan Scripts que pueden ser "hijos" de otro script en ocasiones es necesario exportar las variables, esto se hace escribiendo:

**SIN** `export <nombre_variable>`

## 10.4 CARÁCTERES ESPECIALES

### El carácter «-»



Debe evitarse en lo posible en scripting a «-» ya que para la mayoría de los programas, se usa para indicarle al propio programa que lo que sigue es una de sus opciones. De manera tal por ejemplo, si se crea un archivo con nombre -archivo (en caso que pueda) después será difícil borrarlo ya que **CMD** `rm` (el comando que borra) tratará el archivo como una de sus opciones (al «ver» el script) y dará de error algo como: «Opción -archivo no se reconoce».

Existe un grupo de caracteres especiales (también llamados «meta caracteres») que tienen significado propio para bash. Algunos son los mencionados en el cuadro 10.2 de la siguiente página.

### El «escape»



Para poder utilizar los caracteres reservados de todas maneras, se los utiliza mediante «el escape». Este procedimiento consiste en antecederlos con la antibarra «\». Por ejemplo, «\\$».

CARACTER	DESCRIPCIÓN
\	Le indica a bash que ignore el carácter especial que viene después. A este procedimiento se le denomina «escapar» un carácter.
""	Cuando se encierra entre comillas dobles un texto o una variables si esta es una frase (cadena de palabras) bash lo interpretará como una cadena única.
\$	Identifica que lo que le sigue es una variable.
' '	Las comillas simples se usan para desactivar todos los caracteres especiales encerrados dentro de ellas, así que si escribe '\$VARIABLE' bash interpreta literalmente lo escrito y no como el nombre de variable.
#	Cuando se coloca este carácter dentro de una linea del script, bash ignora el resto de la linea. Muy útil para hacer comentarios y anotaciones o para inhabilitar una linea de comandos al hacer pruebas o debugging.
;	Este carácter se usa para separar la ejecución de distintos comandos en una misma linea de comandos.
``	Se utiliza como se explicó en el punto anterior, para convertir la salida (stdOut) de un comando en el contenido de una variable. El comando en cuestión se ejecuta en una «sub shell».

CUADRO 10.2: CARACTERES ESPECIALES.

### El carácter «espacio»



El espacio ( ) es otro carácter especial y se interpreta por bash como el separador del nombre del programa y las opciones dentro de la linea de comandos, por esta razón es importante encerrar entre comillas dobles el texto o las propias variables cuando son una frase de varias palabras.

## 10.5 PALABRAS RESERVADAS

# 10

Hay un grupo de palabras que tienen significado especial para bash, y que siempre que se pueda, deben evitarse cuando se escriben líneas de comandos; para no crear «confusiones», algunas son: `CMD exit`, `CMD break`, `CMD continue`, `CMD true`, `CMD false`, `CMD return`, cuyo significado es el indicado en el cuadro 10.3.

PALABRA	DESCRIPCIÓN
<code>CMD exit</code>	Se sale del script.
<code>CMD break</code>	Se manda explícitamente a salir de un ciclo.
<code>CMD continue</code>	Se manda explícitamente a retornar en un ciclo.
<code>CMD return</code>	Como <code>CMD exit</code> pero solo se sale del comando u operación sin cerrar el script.
<code>CMD true</code>	Indica que una condición es verdadera.
<code>CMD false</code>	Indica que una condición es falsa.

CUADRO 10.3: PALABRAS RESERVADAS

## 10.6 EL EJECUTABLE [

A diferencia de otros sistemas operativos, en los filesystems de \*nix y Linux, puede haber ejecutables con nombres raros, por ejemplo `CMD [ ]`. `CMD [ ]` es un comando de evaluación que se usa mucho en scripting, principalmente dentro de las sentencias de evaluación condicional como `CMD if` o `CMD while`, que se ven en la sección 10.8 en página 216. En el cuadro 10.4 los argumentos que utiliza.

ARGUMENTO	DESCRIPCIÓN
<code>-d</code>	Archivo existe y es un directorio.
<code>-c</code>	Archivo existe y es de caracteres.
<code>-e</code>	Archivo existe.
<code>-h</code>	Archivo existe y es un vínculo simbólico.
<code>-s</code>	Archivo existe y no está vacío.
<code>-f</code>	Archivo existe y es archivo regular.
<code>-r</code>	Tienes permiso de lectura del archivo.
<code>-w</code>	Tienes permiso de escritura en el archivo.
<code>-x</code>	Tienes permiso de ejecución del archivo.
<code>-0</code>	Eres propietario del archivo.
<code>-G</code>	Perteneces al grupo que tiene acceso al archivo.
<code>-n</code>	Variable existe y no es nula.
<code>Arch1 -nt Arch2</code>	Archivo1 es más nuevo que Archivo2.
<code>Arch1 -ot Arch2</code>	Archivo1 es más viejo que Archivo2.

CUADRO 10.4: EVALUACIÓN PARA ARCHIVOS

### 10.6.1 EVALUACIÓN PARA ARCHIVOS

Este ejecutable, al igual que muchos otros, recibe argumentos, por ejemplo:

```
ttyS9@vorlonhost:~$ touch /tmp/archivo ;; [ -f /tmp/archivo ] && echo "El archivo existe."
El archivo existe.
ttyS9@vorlonhost:~$ █
```



Recordar que tanto en \*nix o Linux, **todo es un archivo**

### 10.6.2 EVALUACIÓN PARA CADENAS DE CARACTERES

Asimismo también posee argumentos (cuadro 10.5) para evaluar cadenas de caracteres.

ARGUMENTO	DESCRIPCIÓN
-z	La cadena está vacía.
-n	La cadena no está vacía.
cadena == cadena2	Si las cadenas son iguales.
cadena1 != cadena2	Si las cadenas son diferentes.
cadena1 <> cadena2	Si la cadena 1 va antes en el orden lexicográfico.
cadena1 > cadena2	Si la cadena 1 va después en el orden lexicográfico.

CUADRO 10.5: EVALUACIÓN PARA CADENAS DE CARACTERES

Por ejemplo, si se accede como el usuario «pablo»:

```
ttyS9@vorlonhost:~$ [ "$USER" = "tty" ] && echo "A vos te conozco."
A vos te conozco.
ttyS9@vorlonhost:~$ █
```

## 10.6.3 ENTRADA/SALIDA

## 10

En algunas ocasiones será necesario leer ciertas variables desde el teclado o imprimirlas a la pantalla, para imprimir a la pantalla se pueden invocar dos programas en la línea de comandos:

**SIN** `echo [cadena,variable,expresión]` →Envia mensajes a la pantalla.

**SIN** `printf [<cadena formateada> <valores>]`, cuya funcionalidad es similar al `printf` del lenguaje C.

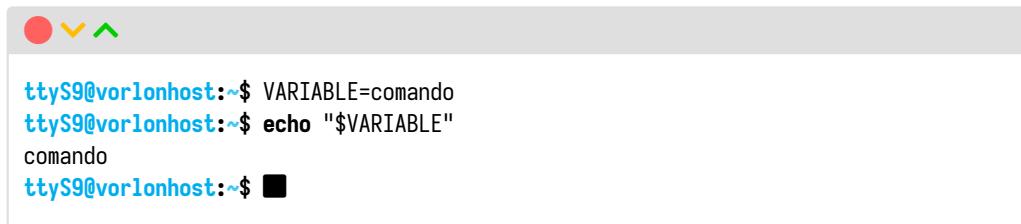
Y para leer desde el teclado se usa:

**SIN** `read [variable]`

Si se ejecuta un `CMD read` sin asignar variable, el dato de almacena en `$REPLY` una variable del shell. Tanto el comando `CMD echo` como `CMD read` tienen sus propias opciones.

## 10.6.4 EJEMPLOS

Si se crea en una línea del script, una variable como un comando y se quiere imprimir dicha variable en la pantalla se puede hacer algo así:



```
ttyS9@vorlonhost:~$ VARIABLE=comando
ttyS9@vorlonhost:~$ echo "$VARIABLE"
comando
ttyS9@vorlonhost:~$ █
```

La palabra `$VARIABLE` está puesta entre comillas dobles para que se imprima todo el texto ignorando los espacios entre palabras.

Si se escribe en una línea del script



```
ttyS9@vorlonhost:~$ read PREGUNTA
Hola, ¿Cómo estás? █
```

se habrá creado una variable de nombre `PREGUNTA`. Si luego se escribe



```
ttyS9@vorlonhost:~$ echo "$PREGUNTA"
Hola, ¿Como estás?
ttyS9@vorlonhost:~$ █
```

Se imprimirá a la pantalla lo que se escribió en el teclado al presionar la tecla .

## 10.7 LOS PRIMEROS SCRIPTS

Con los elementos tratados hasta aquí ya se pueden escribir los primeros scripts.

### 10.7.1 SCRIPT 1: HOLA MUNDO

Hola mundo

```
1 #!/bin/bash
2
3 echo Hola mundo
```

Cuando se corre este script se imprimirá a la pantalla «Hola mundo»

### 10.7.2 SCRIPT 2: LO MISMO USANDO UNA VARIABLE

Hola mundo con variable

```
1 #!/bin/bash
2
3 VARIABLE=Hola mundo
4 echo "$VARIABLE"
```

Nótese la variable entre comillas dobles para que imprima todo el texto.

### 10.7.3 SCRIPT 3: CUANDO SE USAN MÁS DE UNA VARIABLE

Hola mundo con más de una variable

```
1 #!/bin/bash
2
3 VARIABLE=Hola
4 SALUDO=mundo
5 echo "$VARIABLE" "$SALUDO"
```

En los tres casos se imprimirá a la pantalla «Hola mundo»

## 10

## 10.7.4 SCRIPT 4: SI SE USAN CARACTERES ESPECIALES LA COSA PUEDE CAMBIAR

## Caracteres especiales

```

1 #!/bin/bash
2
3 VAR=auto
4 echo "Me compré un $VAR" #Imprimirá Me compré un auto
5 echo 'Me compré un $VAR' #Imprimirá Me compré un $VAR

```

## Comillas simples



Nótese como las comillas simples y el carácter hacen que bash ignore la función del carácter especial \$. Siempre las comillas simples harán que se ignore todos los meta caracteres encerrados entre ellas.

## 10.8 CONDICIONALES

Los condicionales son claves para «explicarle» a bash como debe proceder en una tarea cualquiera, esto se hace casi como si se estuviera explicando una tarea a ejecutar a otra persona.

## 10.8.1 if-then-fi

El condicional por excelencia tiene cinco palabras claves que son **CMD if**, **CMD elif**, **CMD else**, **CMD then** y **CMD fi**. Donde las palabras tienen un significado comunicativo (en Inglés) casi literal, tal y cual se tratará con otra persona y que bash por defecto las entienda con ese significado.

- ▶ **CMD if** →«si» condicional (de si esto o lo otro).
- ▶ **CMD elif** →También si (contracción de «else if»).
- ▶ **CMD else** →De cualquier otra manera.
- ▶ **CMD then** →Entonces.
- ▶ **CMD fi** →if invertido, indica que se acabó la condicional abierta con if.

Solo son imprescindibles en la estructura del Script **CMD if**, **CMD then** y **CMD fi**.

Obsérvese que la acción a ejecutar (equivalente al comando) se hace, si la condición se evalúa como verdadera, de lo contrario se ignora y se pasa a la próxima, si ninguna es verdadera se ejecuta finalmente la acción después del **CMD else**. La sintaxis de bash se debe tener en cuenta a la hora de escribir el script o de lo contrario bash no entenderá lo que se le quiso decir, Ejemplos de scripts reales

## SCRIPT 5

## Comparación de cadenas

```

1 #!/bin/bash
2
3 VAR1=Pablo
4 VAR2=Pedro
5
6 if [ "$VAR1" = "$VAR2" ]; then
7 echo Son iguales
8 else
9 echo Son diferentes
10 fi

```

Los corchetes son parte de la sintaxis de bash y en realidad es el ejecutable **CMD** [ ] (visto en la sección 10.6 en página 212) que es el que ejecuta la acción de comparación. Obsérvese siempre los espacios vacíos entre los elementos que conforman la línea de comandos (excepto entre el último corchete y el «;» recuerde que ese espacio vacío por defecto bash lo interpreta como final de un elemento y comienzo de otro. Si corre este script siempre se imprimirá a pantalla Son diferentes, ya que la condición es falsa. Pero si cambia el valor de **CMD** **VAR2=Pablo** entonces se imprime Son iguales.

## 10.8.2 case-in-esac

Cuando una variable puede adquirir varios valores o significados diferentes, ya se ha visto como puede usarse la palabra **CMD elif** para hacer diferentes ejecuciones de comandos dentro de una misma condicional **SIN if-then-fi** de acuerdo al valor de la variable. Una forma de realizar la misma acción sin escribir tantas líneas de condicionales **CMD elif** y con ello disminuir el tamaño del script, es la utilización de la sentencia **SIN case-in-esac**. Esta sentencia permite vincular patrones de texto con conjuntos de comandos; cuando la variable de la sentencia coincide con alguno de los patrones, se ejecuta el conjunto de comandos asociados.

## case-in-esac

```

1 #!/bin/bash
2
3 echo "Diga si o no:"
4 read VAR
5
6 case "$VAR" in
7 si) echo "Escribiste -si-" ;;
8 no) echo "Escribiste -no-" ;;
9 *) echo "Lo que escribió no se acepta" ;;
10 esac

```

Este Script es el mismo que el Script 7 pero utilizando la sentencia **SIN case-in-esac**. Obsérvese que el carácter «\*» utilizado en la última opción significa «patrón no contemplado» en este caso.

## 10.9 CICLOS O BUCLES

# 10

### 10.9.1 while-do-done

La sentencia **SIN while-do done** se utiliza para ejecutar un grupo de comandos en forma repetida mientras una condición sea verdadera. Su sintaxis es:

```
SIN while <expresión lógica>; do lista de comandos 1; done
```

Mientras la condición de control «lista de comandos1» sea verdadera, se ejecutarán los comandos comprendidos entre **CMD do** y **CMD done** en forma repetida, si la condición da falsa (o encuentra una interrupción explícita dentro del código) el programa sale del bucle (se detiene) y continúa la ejecución por debajo del **CMD while**. Un ejemplo de la utilidad de este bucle es la posibilidad de poder escoger varias opciones de un menú, sin tener que correr el script para cada opción. Es decir se escoge y evalúa una opción y el programa no se cierra, vuelve al menú principal y se puede escoger otra opción, tantas veces como sea necesario. A continuación un ejemplo de cómo elaborar un menú de opciones.

**while-do-done**

```

1 #!/bin/bash
2
3 while [ "$OPCION" != 5 ]; do
4 echo " Listar archivos"
5 echo " Ver directorio de trabajo"
6 echo " Crear directorio"
7 echo " Crear usuario"
8 echo " Salir"
9 read -p "Ingrese una opción: " OPCION
10 case $OPCION in
11 1) ls;;
12 2) pwd;;
13 3) read -p "Nombre del directorio: " DIRECTORIO
14 mkdir $DIRECTORIO;;
15 4) if id | grep uid=0; then
16 read -p "Nombre del usuario: " NOMBREUSUARIO
17 useradd $NOMBREUSUARIO
18 else
19 echo "Se necesitan permisos de root"
20 fi;;
21 5) ;;
22 *) echo "Opción ingresada invalida, intente de nuevo";;
23 esac
24 done
25
26 exit 0

```

- 1 En la primera línea se condiciona el bucle a que la opción escogida sea diferente de 5.
- 2 Luego se hace una lista de echos de las opciones desde 1 hasta 5 con su descripción para que sean imprimidas a la pantalla y así poder escoger alguna.
- 3 Le sigue el comando **CMD read** para que lea del teclado la opción escogida (variable OPCION), a **CMD read** se le ha agregado -p que hace que imprima un mensaje, en este caso imprime

«Ingrese una opción».

- 4 Para ahorrar líneas del script se elabora un **CMD case** con los comandos que deben ejecutarse en cada caso de **CMD ls** para listar los archivos , **CMD pwd** para ver directorio de trabajo, otro **CMD read** para escribir el nombre del directorio que quiere crear y hacer la variable DIRECTORIO seguido por **CMD mkdir** que crea el directorio, luego se crea una condicional **SIN if-fi** para chequear si el usuario tiene permisos de root, necesario para la opción de crear un usuario rechazándolo de lo contrario, después viene la opción vacía que ejecuta el comando **CMD exit 0**, finalmente se incluye «cualquier otra cosa» con el carácter «\*».

Este script resulta interesante porque se usan las dos formas de compactar el script vistas hasta ahora, la sentencia **SIN case-in-esac** y **SIN while-do-done**. Además empiezan a aparecer incluidos en los comandos algunos de los programas muy usados de Linux al escribir scripts.

### 10.9.2 until-do-done

La sentencia **SIN until-do done** es lo contrario de **SIN while-do done** es decir el bucle se cierra o para, cuando la condición sea falsa. Si parece que ambas son muy parecidas se está en lo cierto. En ambos casos se pueden elaborar bucles o ciclos infinitos si la condición de control es siempre verdadera o falsa según el caso. Bucle infinito son aquellos donde la ejecución continua dentro del bucle indefinidamente. Como hacer un bucle infinito mediante **CMD while** :

```
SIN while true; do comando 1; comando n; done
```

La condición siempre es verdadera y se ejecutará el bucle indefinidamente.

Si, por otro lado, se realiza mediante **CMD until**, entonces sería así:

```
SIN until false; do comando 1; comando n; done
```

Existe la posibilidad de salir de un bucle, independientemente del estado de la condición, el comando **CMD break** produce el abandono del bucle inmediatamente. Ejemplo sobre la creación de un menú utilizando un bucle infinito y el comando **CMD break**

#### until-do-done

```

1 while true; do
2   echo " Listar archivos"
3   echo " Ver directorio de trabajo"
4   echo " Crear directorio"
5   echo " Crear usuario"
6   echo " Salir"
7   read -p "Ingrese una opción: " OPCION
8   case $OPCION in
9     1) ls;;
10    2) pwd;;
11    3) read -p "Nombre del directorio: " DIRECTORIO
12      mkdir $DIRECTORIO;;
13
14    4) if id | grep uid=0; then
15      read -p "Nombre del usuario: " NOMBREUSUARIO
16      useradd $NOMBREUSUARIO

```

```

17     else
18         echo "Se necesitan permisos de root"
19     fi;;
20
21 5) echo "Abandonando el programa..."
22     break;;
23
24 *) echo "Opción ingresada invalida, intente de nuevo";;
25 esac
26 done
27
28 exit 0

```

### 10.9.3 for-in-done

Es otro tipo de ciclo o bucle disponible, la diferencia con los anteriores es que no se basa en una condición, sino que ejecuta el bucle una cantidad determinada de veces. Su sintaxis es la siguiente:

**SIN** for variable in arg1.....argn; do comando1 comand0n; done

Ejemplos

#### for-in-done

```

1 for LETRA in a b c d e f; do
2     echo $LETRA
3 done

```

En este script el comando **CMD echo** se ejecutará tantas veces como argumentos se hayan puesto después del «in», por lo tanto imprimirá seis líneas cada una con una letra de la «a» a la «f».

#### for-in-done

```

1 for ARCHIVO in *; do
2     if [ -d $ARCHIVO ]; then
3         cd $ARCHIVO
4         rm *.tmp
5         cd ..
6     fi
7 done

```

Este es un script que entra en todos los subdirectorios del directorio actual de trabajo y borrará todos los archivos .tmp (temporales). En este caso el carácter \* se usa en la primera línea con el significado «tantas veces como sea necesario» y en la penúltima línea «como cualquier cosa».

## 10.10 GLOBALES Y EXPANSIONES

### 10.10.1 GLOBALES

Estos son aliados cuando se quiere ahorrar teclas y funcionan como «generalizadores», los globales más comunes son descriptos en el cuadro 10.6.

ARCHIVO	DESCRIPCIÓN
*	Le dice a bash que es el directorio home del usuario.
*	Significa «todo lo que puedes incluir ahí» de forma tal que si se escribe el comando <code>CMD ls /*.wav</code> listará todos los archivos .wav que están en el directorio home del usuario. Ahora si se escribe <code>CMD ls /m*</code> se listará todos los archivos de home que empiecen con «m».
.	Un punto en el entorno de la shell significa «el directorio donde se está trabajando»

CUADRO 10.6: GLOBALES

Ejemplo:

**Globales**

```

1 #!/bin/bash
2
3 DIR=.
4 mkdir "$DIR"
5 echo "$?"

```

Si se escribe este script y se lo ejecuta dará un error. Por supuesto, se le está mandando a crear el directorio donde se está. Habráse notado que es muy común a la hora de compilar programas desde el fuente, utilizar `CMD ./configure`, con esto se le está diciendo a bash «corre el archivo configure que está en este mismo directorio».

### 10.10.2 EXPANSIONES

Las expansiones son más configurables y trabajan con argumentos mucho más definidos, están claramente hechas para hacer más inteligente al shell. Cuando se especifica una lista de valores o argumentos separados por comas entre llaves, bash la expande convirtiéndola en la cadena expandida con cada uno de los argumentos, por ejemplo:

● ▼ ^

```

ttyS9@vorlonhost:~$ echo este/directorio/{algo,muy,demasiado}/largo
este/directorio/algo/largo este/directorio/muy/largo
→este/directorio/demasiado/largo
ttyS9@vorlonhost:~$ █

```

## 10

Hay que tener en cuenta que:

La expansión funciona sobre una sola palabra sin espacios, si se escribe:

```
● ▼ ^  
ttyS9@vorlonhost:~$ echo esto {es,parece} difícil  
esto es parece difícil  
ttyS9@vorlonhost:~$ █
```

**La expansión no se realiza entre comillas simples ni dobles** por lo que no sirve para corregir el ejemplo anterior:

```
● ▼ ^  
ttyS9@vorlonhost:~$ echo "esto {es,parece} difícil"  
esto {es,parece} difícil  
ttyS9@vorlonhost:~$ █
```

Lo que debe hacerse es ignorar o escapar los espacios y escribir

```
● ▼ ^  
ttyS9@vorlonhost:~$ echo esto {es,parece} confuso  
esto es difícil esto parece confuso  
ttyS9@vorlonhost:~$ █
```

Pueden ponerse múltiples expansiones en una sola linea y se obtendrán todas las combinaciones posibles.

```
● ▼ ^  
ttyS9@vorlonhost:~$ echo {una,otra} combinación { bastante,muy} difícil.  
una combinación bastante difícil. otra combinación bastante  
difícil. una combinación muy difícil. otra combinación muy difícil  
ttyS9@vorlonhost:~$ █
```

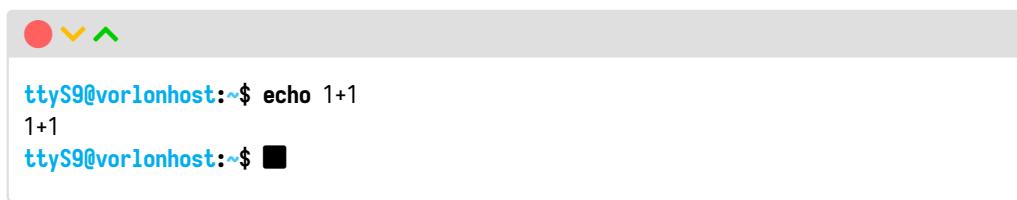
## 10.11 ARITMÉTICA DE BASH

Se pueden ejecutar en bash las principales acciones aritméticas entre las variables utilizando los signos, eso si, siempre en el conjunto de números enteros ( $\mathbb{Z}$ ):

- ▶ + suma
- ▶ - resta
- ▶ \* multiplicación
- ▶ / división

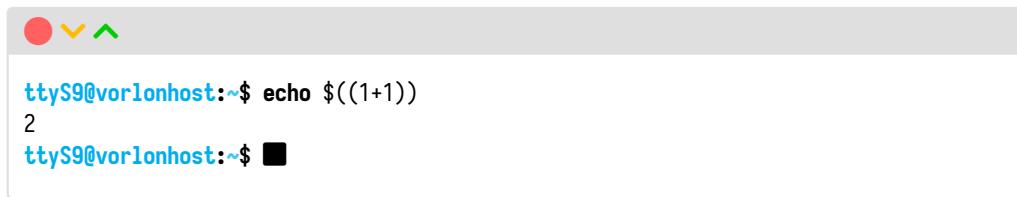
Las operaciones tienen su sintaxis que debe ser respetada para que bash lo haga adecuadamente.

Pruebe esto en el shell o la linea de comandos (consola).



```
ttyS9@vorlonhost:~$ echo 1+1
1+1
ttyS9@vorlonhost:~$ █
```

Esto es porque bash lo interpreta como caracteres simples, para que se realice la operación de suma hay que escribir:



```
ttyS9@vorlonhost:~$ echo $((1+1))
2
ttyS9@vorlonhost:~$ █
```

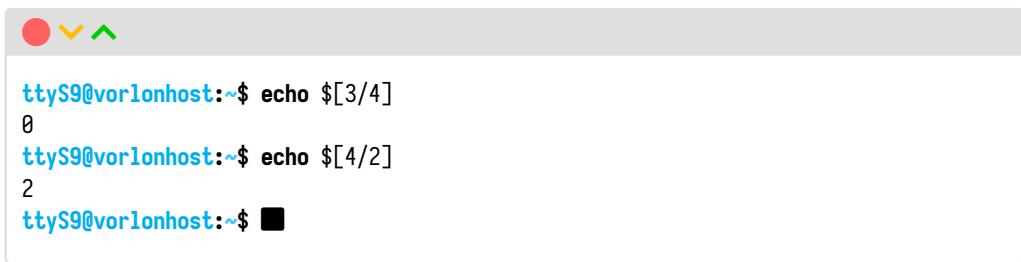
ó



```
ttyS9@vorlonhost:~$ echo ${[1+1]}
2
ttyS9@vorlonhost:~$ █
```

Hay que recordar que bash maneja solo números enteros ( $\mathbb{Z}$ ) por lo tanto si se escribe:

## 10

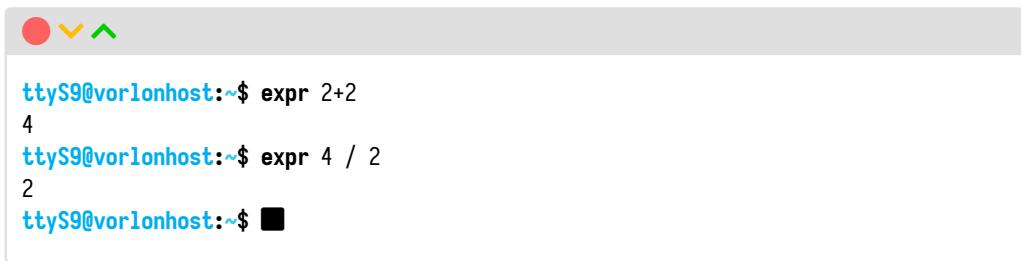


```
ttyS9@vorlonhost:~$ echo ${3/4}
0
ttyS9@vorlonhost:~$ echo ${4/2}
2
ttyS9@vorlonhost:~$ █
```

También se podrá utilizar a **CMD expr** para las operaciones de la forma siguiente:

**SIN** `expr <argumento1> signo <argumento2>`

pruébese en la consola



```
ttyS9@vorlonhost:~$ expr 2+2
4
ttyS9@vorlonhost:~$ expr 4 / 2
2
ttyS9@vorlonhost:~$ █
```

Cuando se use es signo \* para la multiplicación debe anteponerle una barra invertida para que bash no lo interprete como un «global», entonces sería:



```
ttyS9@vorlonhost:~$ expr 10 * 10
100
ttyS9@vorlonhost:~$ █
```

El comando **CMD expr** da sus resultados directamente a la salida estándar pero tampoco maneja números fraccionarios. Hay que observar siempre un espacio entre los argumentos.

Para operar con números fraccionarios se debe utilizar «|» con la expresión, y el programa **CMD bc** de la forma siguiente:

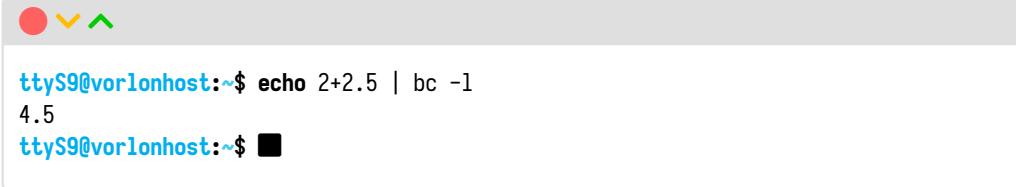
**SIN** `echo <operación> | bc -l`

por ejemplo:



```
ttyS9@vorlonhost:~$ echo 3/4 | bc -l
0.75
ttyS9@vorlonhost:~$ █
```

ó



```
ttyS9@vorlonhost:~$ echo 2+2.5 | bc -l
4.5
ttyS9@vorlonhost:~$ █
```

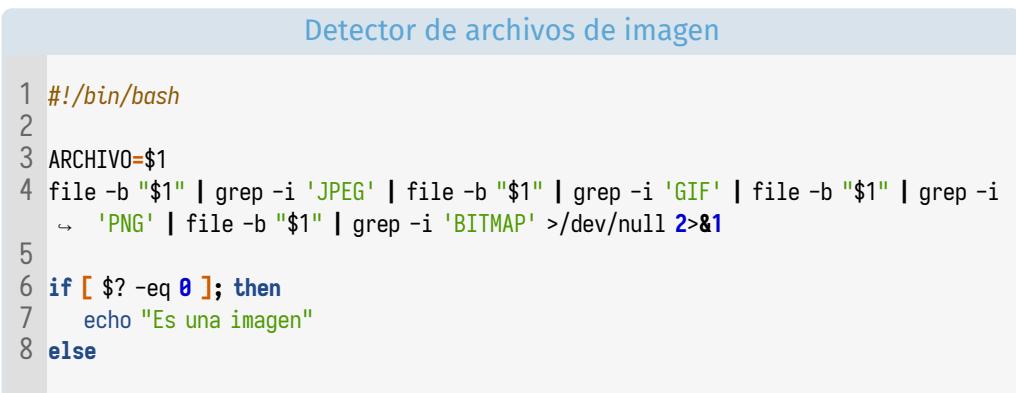
En algunas distribuciones el programa **bc** no se instala por defecto. Hay otras expresiones que bash maneja aritméticas y la compara con los siguientes argumentos (Cuadro 10.7).

ARGUMENTO	DESCRIPCIÓN
-lt	Menor que.
-le	Menor o igual que.
-eq	Igual que.
-ge	Mayor o igual que.
-gt	Mayor que.
-ne	Distinto que.

CUADRO 10.7: COMPARADORES ARITMÉTICOS.

## 10.12 CASOS DE EJEMPLO DE SCRIPTING

### 10.12.1 DETECTOR DE ARCHIVOS DE IMAGEN



Detector de archivos de imagen

```
1 #!/bin/bash
2
3 ARCHIVO=$1
4 file -b "$1" | grep -i 'JPEG' | file -b "$1" | grep -i 'GIF' | file -b "$1" | grep -i
   ~ 'PNG' | file -b "$1" | grep -i 'BITMAP' >/dev/null 2>&1
5
6 if [ $? -eq 0 ]; then
7   echo "Es una imagen"
8 else
```

## 10

```

9 echo "No es una imagen"
10 fi

```

En este script se ha supuesto que un archivo cualquiera se convierte en la variable \$1 y se desea averiguar si el archivo es una imagen en alguno de los formatos más comunes, primero se acude a **CMD file** para que «lea» el texto que contiene el archivo y se lo une con pipe a **CMD grep** que buscará patrones de texto de lo que le entrega **CMD file**. Como se necesita averiguar si alguno de los patrones «JPEG», «GIF», «PNG» o «BITMAP» aparece dentro del archivo se utilizan varias instancias de **CMD file** y **CMD grep** separadas con OR (||), de esta forma se le esta diciendo en el comando «busca si aparece JPEG o GIF o PNG o BITMAP, si lo encuentras entonces imprime» «Es una imagen de cualquier otra forma imprime No es una imagen».

## 10.12.2 PARSEADOR DE ARCHIVO CSV

## Archivo actas.txt

```

1 #
2 # Acta de las materias
3 # Universidad de GNU/Linux
4 #
5 # Estructura de la linea: materia;alumno;nota
6 #
7 linux;jose;9
8 discreta;pedro;10

```

## Script listanotas.sh

```

1#!/bin/bash
2
3# FUNCIONES
4source misfunciones.sh
5
6#
7# MAIN
8#
9ARCHIVO_ACTAS="$1" # Primer parametro es el archivo a ser parseado
10
11# Valido que hayan invocado el script con el argumento.
12if [ ! $# -eq 1 ]; then
13log "Falta el argumento."
14exit 2
15fi
16
17# Si ingresan el parametros, verifico que exista.
18if [ ! -f "$ARCHIVO_ACTAS" ]; then
19log "El archivo $ARCHIVO_ACTAS no se encuentra."
20exit 1
21fi
22
23# Limpio los comentarios del archivo y genero un archivo temporal

```

```

24 grep -v ^# $ARCHIVO_ACTAS > /tmp/$$.txt
25
26 # Comienzo el parseo
27 while read LINEA; do
28
29 # LINEA contiene linea actual
30 MATERIA=`echo $LINEA|cut -d\; -f1`" # Primer campo
31 ALUMNO=`echo $LINEA|cut -d\; -f2`" # Segundo campo
32 NOTA=`echo $LINEA|cut -d\; -f3`" # Tercer campo
33
34 echo "MATERIA: $MATERIA"
35 echo "ALUMNO: $ALUMNO"
36 echo "NOTA: $NOTA"
37 done < /tmp/$$.txt
38
39 # Borro el archivo temporal.
40 rm -f /tmp/$$.txt
41
42 # Salgo con RC=0
43 exit 0

```



```

ttyS9@vorlonhost:~$ ./listanotas.sh actas.txt
MATERIA: linux
ALUMNO: jose
NOTA: 9
MATERIA: discreta
ALUMNO: pedro
NOTA: 10
ttyS9@vorlonhost:~$ █

```

### Biblioteca de funciones misfunciones.sh

```

1 #
2 # Archivo de biblioteca de funciones
3 #
4
5 # Funcion log
6 function log {
7     LOGFILE=`basename $0|cut -d. -f1`.log"
8
9     TEXTO="$1"
10    echo "`basename $0`: $1"
11    echo "$1" > $LOGFILE
12 }

```

## 10.12.3 EJEMPLO DE case Y while

10

## Ejemplo de case y while

```

1 #!/bin/bash
2
3 # Inicio ciclo infinito
4 while true; do
5
6     # Muestro menu
7     echo "MENU:"
8     echo
9     echo "1- Listar archivos."
10    echo "2- Crear archivo."
11    echo "3- Borrar archivo."
12    echo "4- Salir."
13
14    # Leo opcion del usuario
15    read -p "Ingrese una opcion: " OPCION
16
17    # Comienza case
18    case "$OPCION" in
19
20        # Listo archivos
21        1) echo "Listando archivos.."
22            ls -l
23            ;; # Interrumpo ejecucion
24
25        # Creo archivos (usando touch)
26        2) read -p "Ingrese archivo a crear: " NOMBRE
27            touch $NOMBRE
28            ;; # Interrumpo ejecucion
29
30        # Borro archivos
31        3) read -p "Ingrese nombre a borrar: " NOMBRE
32            rm -i $NOMBRE
33            ;; # Interrumpo ejecucion
34
35        # Me voy.
36        4) echo "ADIOS"
37            exit 0
38            ;; # Interrumpo ejecucion (aqui solo por prolijidad)
39
40        # Default.
41        *) echo "Opcion no reconocida."      # Default
42            exit 2
43    esac
44 done

```

## 10.13 CRON

### Cron



Es el nombre del programa que permite a usuarios Linux/\*nix ejecutar automáticamente comandos o scripts (es decir un grupos de comandos) a una hora o fecha específica.

Es usado normalmente para comandos de tareas administrativas, como respaldos, pero puede ser usado para ejecutar cualquier cosa. Como se define en las páginas del manual de cron (`man cron`), es un demonio que ejecuta programas agendados. En prácticamente todas las distribuciones de Linux se usa la versión Vixie Cron, por la persona que la desarrolló, que es Paul Vixie, uno de los grandes gurús de \*nix, también creador, entre otros sistemas, de «BIND» que es uno de los servidores DNS más populares del mundo.

## 10.14 INICIAR CROND

Cron es un demonio<sup>2</sup>, lo que significa que solo requiere ser iniciado una vez, generalmente con el mismo arranque del sistema. El servicio de cron se llama `crond`. En la mayoría de las distribuciones el servicio se instala automáticamente y queda iniciado desde el arranque del sistema, se puede comprobar de varias maneras, dependiendo de la distribución:

Verificando que el servicio este en ejecución:

```
● ▼ ▲
root@vorlonhost:~# /etc/init.d/crond status
cron.service - Regular background program processing daemon
  Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2020-01-06 18:32:20 -03; 3 days ago
    Docs: man:cron(8)
   Main PID: 1767 (cron)
      Tasks: 1 (limit: 4915)
     CGroup: /system.slice/cron.service
           1767 /usr/sbin/cron -f
root@vorlonhost:~# █
```

o si se tiene el comando `service` instalado:

---

<sup>2</sup>«Servicio» en Windows.

# 10

```
root@vorlonhost:~# service crond status
crond.service - Regular background program processing daemon
   Loaded: loaded (/lib/systemd/system/crond.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-01-06 18:32:20 -03; 3 days ago
     Docs: man:cron(8)
 Main PID: 1767 (crond)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/crond.service
           1767 /usr/sbin/crond -f
root@vorlonhost:~#
```

se puede también revisar a través del comando ps:

```
root@vorlonhost:~# pgrep cron
1767
root@vorlonhost:~#
```

## 10.15 USANDO CRON

Hay al menos dos maneras distintas de usar cron:

La primera es en el directorio /etc, donde se encontrarán los siguientes directorios:

- ▶ cron.hourly/
- ▶ cron.daily/
- ▶ cron.weekly/
- ▶ cron.monthly/

Si se coloca un archivo tipo script en cualquiera de estos directorios, entonces el script se ejecutará cada hora, cada día, cada semana o cada mes, dependiendo del directorio. Para que el archivo pueda ser ejecutado tiene que ser algo similar a lo siguiente:

### Script de respaldo

```
1 #!/bin/sh
2 #script que genera un respaldo
3
```

```

4 cd /usr/documentos
5 tar czf * respaldo
6 cp respaldo /otra_directorio/.

```

Nótese que la primera línea empieza con `#!`, que indica que se trata de un script, las demás líneas son los comandos que deseamos ejecutar el script. Este script podría nombrarse por ejemplo `respaldo.sh` y también se debe cambiarle los permisos correspondientes para que pueda ser ejecutado, por ejemplo:

```

root@vorlonhost:~# chmod 700 respaldo.sh
root@vorlonhost:~# ls -l respaldo.sh
-rwx----- 1 root root 0 Jul 20 09:30 respaldo.sh
root@vorlonhost:~# █

```

La «x» en el grupo de permisos del propietario (`rwx`) indica que puede ser ejecutado. Tal cual se ha visto en la sección 5.7 en página 105. Si este script lo dejamos en `cron.hourly`, entonces se ejecutará cada hora con un minuto de todos los días.

Como segundo modo de ejecutar o usar cron es a través de manipular directamente el archivo `/etc/crontab`. En la instalación por defecto de varias distribuciones Linux, este archivo se verá a algo como lo siguiente<sup>3</sup>:

```

root@vorlonhost:~# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
^^I
# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
root@vorlonhost:~# █

```

Las primeras cuatro líneas son variables que se describen en el cuadro 10.8 de la siguiente página.

---

<sup>3</sup>Puede haber diferencias entre distribuciones, principalmente por la implementación de cron que sea instalada.

EJEMPLO	DESCRIPCIÓN
SHELL	Es el «shell» bajo el cual se ejecuta el cron. Si no se especifica, se tomará por defecto el indicado en la línea /etc/passwd correspondiente al usuario que este ejecutando cron.
PATH	Contiene o indica la ruta a los directorios en los cuales cron buscará el comando a ejecutar. Este path es distinto al path global del sistema o del usuario.
MAIL TO	Es la dirección de correo (interna o externa) a la cual se le enviará la salida del comando (si es que este tiene alguna salida). Cron enviará un correo a quien se especifique en este variable, es decir, debe ser un usuario válido del sistema o de algún otro sistema. Si no se especifica, entonces cron enviará el correo al usuario propietario del comando que se ejecuta.
HOME	Es el directorio raíz o principal del comando cron, si no se indica entonces, la raíz será la que se indique en el archivo /etc/passwd correspondiente al usuario que ejecuta cron.

CUADRO 10.8: VARIABLES EN /etc/crontab

Los comentarios se indican con «#» al inicio de la línea.

Después de lo anterior vienen las líneas que ejecutan las tareas programadas propiamente. No hay límites de cuantas tareas pueda haber, una por renglón. Los campos descritos en la tabla 10.9, están listados en el mismo orden en el que aparecen (de izquierda a derecha) en el crontab.

CAMPO	DESCRIPCIÓN
Minuto	Controla el minuto de la hora en que el comando será ejecutado, este valor debe de estar entre 0 y 59.
Hora	Controla la hora en que el comando será ejecutado, se especifica en un formato de 24 horas, los valores deben estar entre 0 y 23, 0 es medianoche.
Día del Mes	Día del mes en que se quiere ejecutar el comando. Por ejemplo se indicaría 20, para ejecutar el comando el día 20 del mes.
Mes	Mes en que el comando se ejecutará, puede ser indicado numéricamente (1-12), o por el nombre del mes en inglés, solo las tres primeras letras.
Día de la semana	Día en la semana en que se ejecutará el comando, puede ser numérico (0-7) o por el nombre del día en inglés, solo las tres primeras letras. (0 y 7 = domingo)
Usuario	Usuario que ejecuta el comando.
Comando	Comando, script o programa que se desea ejecutar. Este campo puede contener múltiples palabras y espacios.

CUADRO 10.9: CAMPOS DEL crontab

Un asterisco «\*» como valor en los primeros cinco campos, indicará inicio-fin del campo, es decir todo. Un «\*» en el campo de minuto indicará todos los minutos.

Para entender bien esto de los primeros cinco campos y el asterisco se recurrirá a varios ejemplos que figuran en la tabla 10.10.

EJEMPLO	DESCRIPCIÓN
@reboot	Se ejecuta cuando la computadora se reinicia.
01 * * * *	Se ejecuta al minuto 1 de cada hora de todos los días.
15 8 * * *	A las 8:15 a.m. de cada día.
15 20 * * *	A las 8:15 p.m. de cada día.
00 5 * * 0	A las 5 a.m. todos los domingos.
* 5 * * Sun	Cada minuto de 5:00a.m. a 5:59a.m. todos los domingos.
45 19 1 * *	A las 7:45 p.m. del primero de cada mes.
01 * 20 7 *	Al minuto 1 de cada hora del 20 de julio.
10 1 * 12 1	A la 1:10 a.m. todos los lunes de diciembre.
00 12 16 * Wen	Al mediodía de los días 16 de cada mes y que sea Miércoles.
30 9 20 7 4	A las 9:30 a.m. del dia 20 de julio y que sea jueves.
30 9 20 7 *	A las 9:30 a.m. del dia 20 de julio sin importar el día de la semana.
20 * * * 6	Al minuto 20 de cada hora de los sábados.
20 * * 1 6	Al minuto 20 de cada hora de los sábados de enero.

CUADRO 10.10: EJEMPLOS DE crontab

También es posible especificar listas en los campos. Las listas pueden estar en la forma de 1,2,3,4 o en la forma de 1-4 que sería lo mismo. Cron, de igual manera soporta incrementos en las listas, que se indican de la siguiente manera:

## 10.16 VALOR O LISTA/INCREMENTO

De nuevo, es más fácil entender las listas e incrementos con ejemplos, que figuran en la tabla 10.11 de la siguiente página.

EJEMPLO	DESCRIPCIÓN
59 11 * 1-3 1,2,3,4,5	A las 11:59 a.m. de lunes a viernes, de enero a marzo.
45 * 10-25 * 6-7	Al minuto 45 de todas las horas de los días 10 al 25 de todos los meses y que el día sea sábado o domingo.
10,30,50 * * * 1,3,5	En el minuto 10, 30 y 50 de todas las horas de los días lunes, miércoles y viernes.
*/15 10-14 * * *	Cada quince minutos de las 10:00a.m. a las 2:00p.m.
* 12 1-10/2 2,8 *	Todos los minutos de las 12 del día, en los días 1,3,5,7 y 9 de febrero y agosto. (El incremento en el tercer campo es de 2 y comienza a partir del 1).
0 */5 1-10,15,20-23 * 3	Cada 5 horas de los días 1 al 10, el día 15 y del día 20 al 23 de cada mes y que el día sea miércoles.

Cuadro 10.11 – CONTINUACIÓN...

EJEMPLO	DESCRIPCIÓN
3/3 2/4 2 2 2	Cada 3 minutos empezando por el minuto 3 (3,6,9, etc.) de las horas 2,6,10, etc (cada 4 horas empezando en la hora 2) del día 2 de febrero y que sea martes.

CUADRO 10.11: EJEMPLOS DE crontab CON VALOR O LISTA/INCREMENTO.

Como se puede apreciar en el último ejemplo la tarea cron que estuviera asignada a ese renglón con esos datos, solo se ejecutaría si se cumple con los cinco campos (es decir, como el operador lógico AND). Es decir, para que la tarea se ejecute tiene que ser un martes 2 de febrero a las 02:03.

### Lógica de los campos



Los campos especificados en crontab, siempre son un AND booleano que solo resulta verdadero si los cinco campos son ciertos.

El caso anterior deja claro entonces que: El programa **cron** se invoca cada minuto y ejecuta las tareas que sus campos se cumplan en ese preciso minuto.

Incluyendo el campo del usuario y el comando, los renglones de crontab podrían quedar entonces de la siguiente manera:

```

crontab

1 0 22 * * * root /usr/respaldodiario.sh
2 0 23 * * 5 root /usr/respaldosemanal.sh
3 0 8,20 * * * erik mail -s "sistema funcionando" el@freemutants.org

```

Las dos primeras líneas las ejecuta el usuario root y la primera ejecuta a las 22hs todos los días el script que genera un respaldo diario. La segunda ejecuta a las 23hs todos los viernes un script que genera un respaldo semanal. La tercera línea la ejecuta el usuario erik y se ejecutaría a las 8 y 20hs todos los días y el comando es enviar un correo a la cuenta `el@freemutants.org` con el asunto «sistema funcionando», una manera de que un administrador esté enterado de que un sistema remoto está activo en las horas indicadas, sino recibe un correo en esas horas, algo andaría mal.

Siendo root, es posible entonces, modificar directamente crontab:

**vi /etc/crontab**

## 10.17 EJECUTANDO CRON CON MÚLTIPLES USUARIOS, COMANDO crontab

Linux es un sistema multiusuario y cron es de las aplicaciones que soporta el trabajo con varios usuarios a la vez. Cada usuario puede tener su propio archivo crontab, de hecho, /etc/crontab se asume que es el archivo crontab del usuario root, aunque no hay problema que se incluyan otros usuarios, «y de ahí el sexto campo que indica precisamente quien es el usuario que ejecuta la tarea y es obligatorio en /etc/crontab».

Pero cuando los usuarios normales (e incluso root) desean generar su propio archivo de crontab, entonces se utilizará **CMD crontab**.

En el directorio /var/spool/cron (puede variar según la distribución), se genera un archivo cron para cada usuario, este archivo aunque es de texto, **no debe editarse directamente**.

Se tiene entonces, dos situaciones, generar directamente el archivo crontab con el comando:

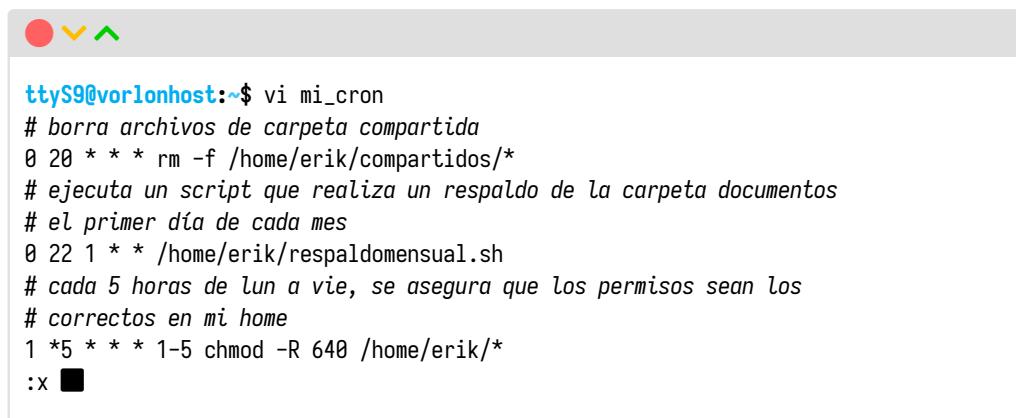
**CMD crontab -e**

Con lo cual se abrirá el editor (generalmente vi) con el archivo llamado crontab vacío y donde el usuario ingresará su tabla de tareas y que se guardará automáticamente como /var/spool/cron/usuario.

El otro caso es que el usuario edite un archivo de texto con las entradas de las tareas y como ejemplo lo nombre mi\_cron, después el comando

**CMD crontab mi\_cron**

se encargará de establecerlo como su archivo cron del usuario en /var/spool/cron/usuario:



```
ttyS9@vorlonhost:~$ vi mi_cron
# borra archivos de carpeta compartida
0 20 * * * rm -f /home/erik/compartidos/*
# ejecuta un script que realiza un respaldo de la carpeta documentos
# el primer día de cada mes
0 22 1 * * /home/erik/respaldomensual.sh
# cada 5 horas de lun a vie, se asegura que los permisos sean los
# correctos en mi home
1 *5 * * * 1-5 chmod -R 640 /home/erik/*
:x
```



```
ttyS9@vorlonhost:~$ ls
mi_cron
ttyS9@vorlonhost:~$
```

# 10

Resumiendo lo anterior y considerando otras opciones de crontab:

- ▶ **CMD** `crontab archivo.cron` → Establecerá el archivo.cron como el crontab del usuario.
- ▶ **CMD** `crontab -e` → Abrirá el editor preestablecido donde se podrá crear o editar el archivo crontab.
- ▶ **CMD** `crontab -l` → Lista el crontab actual del usuario, sus tareas de cron.
- ▶ **CMD** `crontab -r` → Elimina el crontab actual del usuario.

En algunas distribuciones cuando se editan crontabs de usuarios es necesario reiniciar el servicio para que se puedan releer los archivos de crontab en `/var/spool/cron`

**CMD** `service cron restart`

## 10.18 CONTROLANDO EL ACCESO A CRON

Cron permite controlar qué usuarios pueden o no pueden usar los servicios de cron. Esto se logra de una manera muy sencilla a través de los siguientes archivos:

- ▶ `/etc/cron.allow`
- ▶ `/etc/cron.deny`

Para impedir que un usuario utilice cron o mejor dicho el comando crontab, basta con agregar su nombre de usuario al archivo `/etc/cron.deny`, para permitirle su uso entonces sería agregar su nombre de usuario en `/etc/cron.allow`, si por alguna razón se desea negar el uso de cron a todos los usuarios, entonces se puede escribir la palabra ALL al inicio de `cron.deny` y con eso bastaría.

**CMD** `echo ALL >>/etc/cron.deny`

o para agregar un usuario más a `cron.allow`

**CMD** `echo charles >>/etc/cron.allow`

Si no existe el archivo `cron.allow` ni el archivo `cron.deny`, en teoría el uso de cron esta entonces sin restricciones de usuario. Si se añaden nombres de usuarios en `cron.allow`, sin crear un archivo `cron.deny`, tendrá el mismo efecto que haberlo creado con la palabra ALL. Esto quiere decir que una vez creado `cron.allow` con un solo usuario, siempre se tendrán que especificar los demás usuarios que se quiere usen cron, en este archivo. ■

# 11 | Práctica - Parte I

## 11.1 COMANDOS

- 1 Inicia sesión en una máquina con Linux.
- 2 Ingresa estos comandos en el indicador de \*nix e intenta interpretar la salida. No tengas miedo de experimentar (como usuario normal no se puede hacer mucho daño):

- ▶ `CMD echo hello world`
- ▶ `CMD passwd`
- ▶ `CMD date`
- ▶ `CMD hostname`
- ▶ `CMD arch`
- ▶ `CMD uname -a`
- ▶ `CMD dmesg | more` (Se debe presionar `q` para salir)
- ▶ `CMD uptime`
- ▶ `CMD who am i`
- ▶ `CMD who`
- ▶ `CMD id`
- ▶ `CMD last`
- ▶ `CMD finger`
- ▶ `CMD w`
- ▶ `CMD (` Se debe presionar `q` para salir)
- ▶ `CMD echo $SHELL`
- ▶ `CMD echo {con,pre}{sent,fer}{s,ed}`
- ▶ `CMD man "automatic door"`
- ▶ `CMD man ls` (Se debe presionar `q` para salir)
- ▶ `CMD man who` (Se debe presionar `q` para salir)
- ▶ `CMD who can tell me why i got divorced`
- ▶ `CMD lost`
- ▶ `CMD clear`

## 11

- ▶ `CMD cal 2000`
- ▶ `CMD cal 9 1752` (¿Se nota algo raro?)
- ▶ `CMD bc -l` (Se debe presionar `Ctrl`+`d` para salir)
- ▶ `CMD echo 5+4 | bc -l`
- ▶ `CMD yes please` (Se debe presionar `Ctrl`+`c` para salir)
- ▶ `CMD time sleep 5`
- ▶ `CMD history`

## 11.2 NAVEGACIÓN Y ARCHIVOS

- 1 Probar la siguiente secuencia de comandos:

- ▶ `CMD cd`
- ▶ `CMD pwd`
- ▶ `CMD ls -al`
- ▶ `CMD cd .`
- ▶ `CMD pwd` (¿dónde te lleva?)
- ▶ `CMD cd ..`
- ▶ `CMD pwd`
- ▶ `CMD ls -al`
- ▶ `CMD cd ..`
- ▶ `CMD pwd`
- ▶ `CMD ls -al`
- ▶ `CMD cd ..`
- ▶ `CMD pwd` (¿que pasa ahora?)
- ▶ `CMD cd /etc`
- ▶ `CMD ls -al | more`
- ▶ `CMD cat passwd`
- ▶ `CMD cd -`
- ▶ `CMD pwd`

- 2 Continuar explorando el árbol del sistema de archivos usando `CMD cd`, `CMD ls`, `CMD pwd` y `CMD cat`. Buscar en `/bin`, `/usr`, `/bin`, `/sbin`, `/tmp` y `/boot`. ¿Qué se ve?
- 3 Explorar `/dev`. ¿Se pueden identificar qué dispositivos están disponibles? ¿Cuáles están orientados a los caracteres y cuáles están orientados a los bloques? ¿Se puede identificar el dispositivo `tty` (terminal) ¿Quién es el dueño de la `tty` (usar `CMD ls -l`)?
- 4 Explorar `/proc`. Imprimir por pantalla el contenido de los archivos `interrupts`, `devices`, `cpuinfo`, `meminfo` mediante `cat`. ¿Puedes ver por qué decimos que `/proc` es un pseudo-sistema de archivos que permite acceder a las estructuras de datos del núcleo?

- 5 Cambia al directorio de inicio de otro usuario directamente, usando `CMD cd ~nombre de usuario`.
- 6 Cambiar de nuevo al directorio personal.
- 7 Crear subdirectorios llamados `trabajo` y `juego`.
- 8 Eliminar el subdirectorio llamado `trabajo`.
- 9 Copiar el archivo `/etc/passwd` al directorio de inicio.
- 10 Moverlo al subdirectorio `juego`.
- 11 ¿Cuál es la diferencia entre enumerar los archivos de un directorio con `CMD ls -l` y `CMD ls -L`?
- 12 Imaginar que se estaba trabajando en un sistema y alguien eliminó accidentalmente el comando `CMD ls` (`/bin/ls`). ¿Cómo se podría obtener una lista de los archivos en el directorio actual? Intentalo.
- 13 ¿Cómo se crearía y luego eliminaría un archivo llamado `"$SHELL"`? Intentalo.
- 14 ¿Cómo se crearía y luego se eliminaría un archivo que comienza con el símbolo `#`? Intentalo.
- 15 ¿Cómo se crearía y luego se eliminaría un archivo que comienza con el símbolo `-`? Intentalo.
- 16 ¿Cuál es la salida del comando: `CMD echo {con, pre} {enviado, fer} {s, ed}`?
- 17 Ahora, desde el directorio de inicio, copiar `/etc/passwd` y `/etc/group` al directorio de inicio con un comando solamente dado que hipotéticamente solo puede leer `/etc` una sola vez.
- 18 Aún en el directorio de inicio, copiar la estructura completa del directorio en un directorio llamado `trabajo`
- 19 Eliminar el directorio de `trabajo` y su contenido con un comando, sin ningún tipo de confirmación o mensaje de error.
- 20 Cambiar a un directorio que no le pertenece e intentar eliminar todos los archivos (¡ojo con `/proc` o `/dev`, por si acaso!)
- 21 Experimentar con las opciones en el comando `CMD ls`. ¿Qué hacen las opciones `d`, `i`, `R` y `F`?
- 22 Crear un archivo llamado `listado_bin` que contenga el listado del directorio `/bin`.
- 23 Crear un archivo llamado `listado_sbin` que contenga el listado del directorio `/sbin`.
- 24 Crear un archivo llamado `binarios` que contenga ambos listados.
- 25 Ordenar alfabéticamente el listado `binarios` y guardar el resultado en un archivo `binarios2`.
- 26 Verificar que los datos en `binarios2` sean correctos.
- 27 Crear un archivo llamado `datosv` con los siguientes datos personales dentro: Nombre, Apellido y DNI.
- 28 Agregar a `datosv` una línea que indique el directorio actual.
- 29 Agregar a `datosv` un listado en formato lista del directorio `/etc`.
- 30 Observar (por pantalla) el archivo `datosv` resultante a través del filtro `CMD more` y verificar que los datos estén correctos

## 11

## 11.3 PIPE'S Y REDIRECCIONES

- 1 Obtener un listado en orden alfabético inverso de los primeros 3 comandos del directorio /bin. Utilizar: `CMD ls`, `CMD sort` y `CMD head`.
- 2 Obtener el mismo resultado que en el ejercicio anterior, pero utilizando el comando `CMD tail` en vez de `CMD head`.
- 3 Contar la cantidad de archivos en el directorio /bin. Ayuda: Para contar palabras se utilizará el comando `CMD wc -w`.
- 4 Buscar dentro de /usr/doc o /usr/share/doc algún archivo que contenga gz
- 5 Obtener (utilizando una sola línea de comandos y pipes) un archivo donde figuren todos los usuarios y los grupos definidos en el sistema. Ordenados alfabéticamente. Visualizado a través del `CMD less`. Utilizar `CMD cat` para juntar ambos archivos, ordenar las líneas alfabéticamente y mostrar el resultado.
- 6 Obtener la cantidad de usuarios y grupos definidos en el sistema (suma de ambos).
- 7 Explicar que hacen:

▶ `CMD ls | head -3 | tail -1`

▶ `CMD ls -l /etc | tail -n +2 | sort`

▶ `CMD ls -l | grep '^.....w'`

▶ `CMD grep -v "#" /etc/apache2/sites-available/default-ssl`

▶ `CMD grep -r "function" *`

▶ `CMD grep -n "main" setup.py`

## 11.4 PERMISOS

- 1 En un directorio vacío (nuevo), crear 6 archivos (archiv1, archiv2, etc.) utilizando el comando `CMD touch`. Quitarles todos los permisos con el comando `CMD chmod a-rwx archiv*`
- 2 Modificar los permisos usando el operador '=' del '`CMD chmod`', para que queden de la siguiente manera:

```
1 archiv1 -rwx-----
2 archiv2 -rw-----
3 archiv3 -rwxrwxrwx
4 archiv4 -rwxrw-r--
5 archiv5 -rwxr-----
6 archiv6 -r-xrw-r--
```

- 3 Modificar los permisos de los archivos anteriores utilizando los operadores + y - del 'chmod' para que queden de la siguiente manera:

```

1 archiv1 -rwx---r-- (agrega lectura para otros)
2 archiv2 -r----- (quita escritura para propietario)
3 archiv3 -rw-rw-rw- (quita ejecución para todos)
4 archiv4 -rwx-w--- (quita lectura para grupo y otros)
5 archiv5 -rwx----wx (quita lectura al grupo, agrega esc. y ejec para otros)
6 archiv6 -rwxrw---- (agrega escritura al propietario, quita lectura a otros)

```

- 4 Crear seis archivos (num1, num2, etc.) utilizando el comando **CMD touch**.
- 5 Sobreescribir los permisos utilizando el comando **chmod** con argumento numérico (octal) para que queden de la siguiente manera:

```

1 num1 -r---w---x
2 num2 -----
3 num3 -rwxrwxrwx
4 num4 -r-xrw-r--
5 num5 -rwxr-----
6 num6 -rw-r--r--

```

- 6 Con una sola instrucción, quitar permisos de lectura, escritura y ejecución para otros a todos los archivos utilizados en el último ejercicio.
- 7 Crear un directorio y quitarle todos los permisos de ejecución. Explicar qué pasa al intentar entrar al directorio con el comando **CMD cd**. Explicar el significado de los permisos r,w y x para directorios.
- 8 Informarse sobre los grupos a los que pertenece su usuario.
- 9 Utilizando los comandos **CMD chown** y **CMD chgrp**, intentar cambiar el propietario y el grupo del archivo "num3". ¿Cuál es el problema?

## 11.5 USUARIOS



Orden en las prácticas Es recomendable realizar primero la práctica de permisos

- 1 Crea los grupos «oficina1» y «oficina2».
- 2 Crea los usuarios «damian» y «pablo». Estos usuarios deben pertenecer únicamente al grupo «oficina1».
- 3 Crea los usuarios «alba» y «nerea». Estos usuarios deben pertenecer únicamente al grupo oficina2.

## 11

- 4 Como usuario «damian», crea un archivo con nombre «topsecret.txt» en su directorio de trabajo al que únicamente él tenga acceso, tanto de lectura como de escritura.
- 5 Crea otro archivo, también como usuario «damian», con nombre «ventas\_trimestre.txt» al que tengan acceso, tanto para leer como para escribir todos los usuarios que pertenezcan al mismo grupo. Se deben dejar los permisos que haya por defecto para el dueño y para el resto de usuarios. Comprobar como usuario «pablo» que se puede modificar el archivo.
- 6 Como usuario alba, crea un archivo con nombre «empleados.txt» al que pueda acceder cualquier usuario para leer su contenido, y cualquier usuario del mismo grupo para leer o escribir.
- 7 Copia el archivo «empleados.txt» al directorio de trabajo de alumno. Cambiar el propietario y el grupo al que pertenece el archivo, ahora debe ser «alumno».
- 8 Crea el usuario «rosa», perteneciente a «oficina2». Dentro de su directorio de trabajo, crea un directorio de nombre «compartido\_con\_todos».
- 9 Cambia de usuario y entra como «rosa». Crea los archivos «telefono\_contactos», «gastos\_marzo» y «sueldos». Inserta varias entradas en cada uno de los archivos y grábalos todo en el directorio «compartido\_con\_todos».
- 10 Da permiso de lectura a la carpeta «compartido\_con\_todos» y a todos los archivos que contenga para todos los usuarios.
- 11 Restringe el acceso de escritura sobre el archivo «telefono\_contactos» para que sólo lo puedan modificar los usuarios del grupo al que pertenece su propietario.
- 12 Cambia los permisos de «gastos\_marzo» para que sólo pueda modificarlo su propietario y leerlo cualquiera del mismo grupo.
- 13 Cambia los permisos de sueldos para que sólo su dueño tenga acceso a él, tanto para lectura como para escritura.
- 14 Si un usuario tiene permiso de lectura sobre un archivo pero ese archivo se encuentra dentro de un directorio sobre el que no tiene permiso de lectura, ¿Se podrá leer el archivo?, hacer la prueba.

## 11.6 PROCESOS

- 1 Ejecutar en la siguiente secuencia:

```

1 $ bash
2 $ yes >/dev/null &
3 $ yes >/dev/null &
4 $ bash
5 $ ping host_valido >/dev/null &
6 $ ping otro_host_valido >/dev/null &

```

- 2 Utilizar el comando  `pstree` para ver el árbol de procesos generado.
- 3 Cambiar de consola y matar todas las tareas relacionadas con la consola anterior (utilizar algún comando para eliminar grupos de procesos).

- 4 Ejecutar varios procesos `CMD yes >/dev/null &` y varios `CMD ping host >/dev/null &`.
- 5 Eliminar todos los procesos `CMD yes`. Verificar lo realizado y luego eliminar todos los procesos `CMD ping`.
- 6 Leer la página de manual del `CMD ls`. Suspender el proceso.
- 7 Leer la página de manual del `CMD cat`. Suspender el proceso.
- 8 Hacer un `CMD ping` a otra máquina. Suspender el proceso.
- 9 ¿Que otros procesos se han disparado? ¿De qué proceso son hijos?
- 10 Retomar el proceso `CMD man ls` en primer plano y terminarlo normalmente
- 11 Pasar el proceso `CMD ping` de suspendido a ejecución en segundo plano. Eliminar luego el proceso `CMD ping`.
- 12 Ejecutar un procesos `CMD yes >/dev/null &` y un `CMD ping host >/dev/null &`. Cuanto porcentaje de cpu consume cada uno? ¿es constante?
- 13 Matar todos los procesos que han quedado **perdidos** antes de cerrar las sesiones.
- 14 Ejecutar el comando `CMD sleep 5`. ¿Qué hace este comando?
- 15 Ejecutar el mismo comando en segundo plano usando `CMD &`.
- 16 Ejecutar `CMD sleep 15` en primer plano, suspenda con `Ctrl + z` y luego póngalo en segundo plano con `CMD bg`. Escribir `CMD ps`. Volver a poner el trabajo en primer plano con `CMD fg`.
- 17 Ejecutar `CMD sleep 15` en segundo plano con `CMD &`, y luego usar `CMD kill` para finalizar el proceso por su número de trabajo. Repetir, excepto que esta vez finalice el proceso especificando su PID.
- 18 Iniciar una serie de procesos de `CMD sleep 60` en segundo plano y finalícelos todos al mismo tiempo con el comando `CMD pkill`.
- 19 Iniciar un proceso de `CMD sleep 300` ejecutándose en segundo plano. Cierren sesión en el servidor y vuelvan a iniciar sesión. Enumerar todos los procesos que está ejecutando. ¿Qué pasó con el proceso de `CMD sleep`? Ahora repetir, excepto que esta vez se debe comenzar ejecutando `CMD nohup sleep 300`.



## 12 Práctica - Parte II

### 12.1 PROCEDIMIENTO DE RECUPERACIÓN DE PASSWORD

Para lograr dicho cometido, debemos seguir los siguientes pasos:

- 1 Parar la cuenta regresiva del GRUB presionando cualquier tema. Figura 12.1
- 2 Editar la entrada de grub que usualmente booteamos al inicio. Figura 12.2 de la siguiente página
- 3 Agregarle la entrada `init=/bin/bash`. Figura 12.3 de la siguiente página
- 4 Bootear con **F10** o **Ctrl**+**x**. Figura 12.4 en página 247
- 5 Verificar el estado del filesystem `/`, y ver que esta montado Read Only. Figura 12.5 en página 247
- 6 Remontar Read Write y cambiar la password. Figura 12.6 en página 248

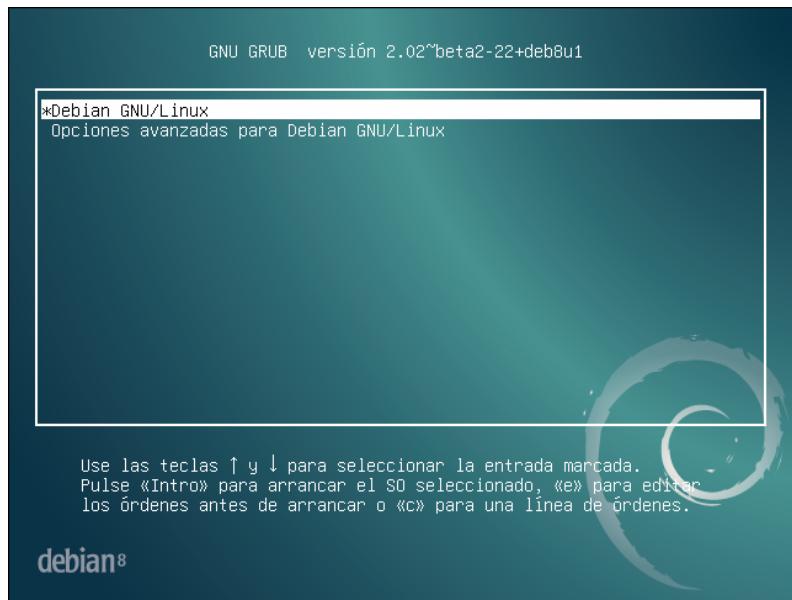


FIGURA 12.1: MENU DEL GRUB

### 12.2 FIREWALL

- 1 Listar las reglas activas actualmente.

## 12

GNU GRUB versión 2.02~beta2-22+deb8u1

```

fi
    insmod part_msdos
    insmod ext2
    if [ $feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root 42290d8e-78ee-4604-a4\|
3c-0099580d5d5f
    else
        search --no-floppy --fs-uuid --set=root 42290d8e-78ee-4604-a43\|
c-0099580d5d5f
    fi
    echo      'Cargando Linux 3.16.0-6-amd64...'
    linux    /boot/vmlinuz-3.16.0-6-amd64 root=UUID=42290d8e-78e\|
e-4604-a43c-0099580d5d5f ro quiet_
    echo      'Cargando imagen de memoria inicial...'
    initrd   /boot/initrd.img-3.16.0-6-amd64

```

↑ ↓

Se soporta una edición de pantalla mínima al estilo de Emacs.  
 «TAB» enumera las posibles palabras a completar. Pulse «Ctrl-x» o «F10» para arrancar, «Ctrl-c» o «F2» para una línea de órdenes o «ESC» para descartar las ediciones y volver al menú de GRUB.

debian®

FIGURA 12.2: A PUNTO DE EDITAR LA ENTRADA CORRESPONDIENTE

GNU GRUB versión 2.02~beta2-22+deb8u1

```

fi
    insmod part_msdos
    insmod ext2
    if [ $feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root 42290d8e-78ee-4604-a4\|
3c-0099580d5d5f
    else
        search --no-floppy --fs-uuid --set=root 42290d8e-78ee-4604-a43\|
c-0099580d5d5f
    fi
    echo      'Cargando Linux 3.16.0-6-amd64...'
    linux    /boot/vmlinuz-3.16.0-6-amd64 root=UUID=42290d8e-78e\|
e-4604-a43c-0099580d5d5f ro quiet init=/bin/bash_
    echo      'Cargando imagen de memoria inicial...'
    initrd   /boot/initrd.img-3.16.0-6-amd64

```

↑ ↓

Se soporta una edición de pantalla mínima al estilo de Emacs.  
 «TAB» enumera las posibles palabras a completar. Pulse «Ctrl-x» o «F10» para arrancar, «Ctrl-c» o «F2» para una línea de órdenes o «ESC» para descartar las ediciones y volver al menú de GRUB.

debian®

FIGURA 12.3: AGREGANDO INIT=/BIN/BASH

- 2 Borrar las reglas.
- 3 Definir la política por defecto para trafico entrante como denegada.
- 4 Crear una regla que permita el trafico entrante del protocolo SSH.
- 5 Dejar persistentes los cambios hechos al firewall.
- 6 Crear una regla que permita el trafico entrante del protocolo HTTP.
- 7 Listar las reglas activas actualmente, con número de línea.
- 8 Insertar una regla entre 1 y 2.

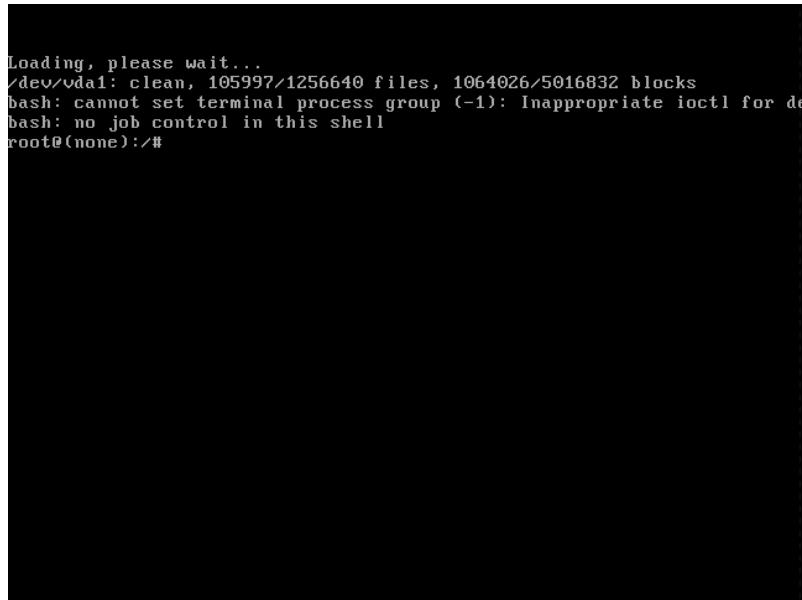


FIGURA 12.4: BOOTEO LA ENTRADA CON F10

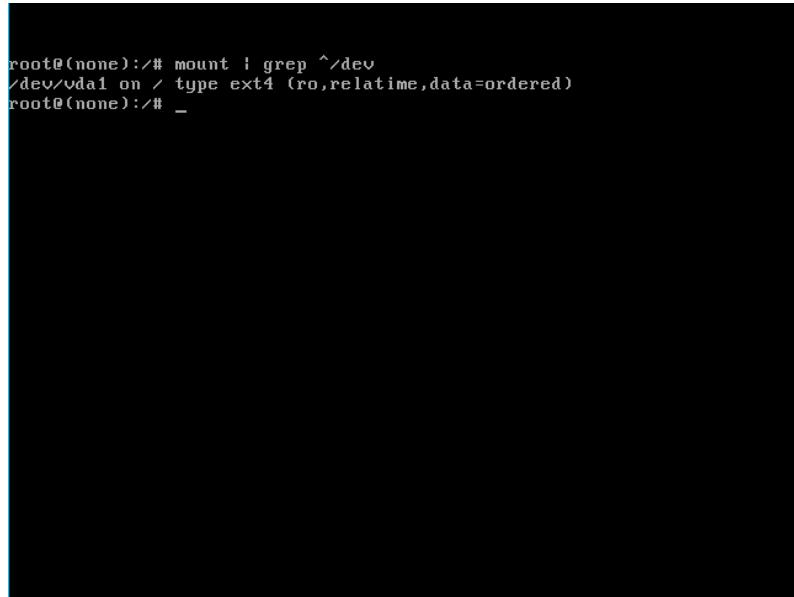


FIGURA 12.5: VERIFICO EL ESTADO DEL FILESYSTEM /

- 9 Dejar persistentes los cambios hechos al firewall.

### 12.3 SCRIPTING

- 1 Escribir un script que muestre por consola los números del 1 al 5.
- 2 Escribir un script que muestre por consola la fecha del sistema cada dos segundos, 10 veces.
- 3 Escribir un script que, al ejecutarlo como root, reinicie el equipo después de un minuto. Si el usuario ejecutor no es root, informar y finalizar el programa.

```

root@none:/# mount | grep '^/dev
/dev/vda1 on / type ext4 (ro,relatime,data=ordered)
root@none:/# mount -o remount,rw /
root@none:/# passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@none:/# -

```

FIGURA 12.6: REMONTAR R/W Y CAMBIAR LA PASSWORD

- 4 Escribir un script que recibe dos parámetros: una cadena de caracteres y el path a un archivo de texto plano. Se deberá buscar dicha cadena dentro del archivo e imprimir por consola «Palabra encontrada» en caso de encontrarla o «Palabra no encontrada» en caso contrario. Controlar que sea correcta la cantidad de parámetros.
- 5 Escribir un script que reciba como parámetro el path a un archivo y determine si existe o no y que tipo de archivo es.
- 6 Escribir un script que lea dos números enteros desde el teclado y determine si son iguales, o cual es mayor que el otro (Hacerlo con al menos una función).
- 7 Escribir un script que reciba dos números enteros como parámetro y determine si son iguales, o cual es mayor que el otro (Hacerlo con al menos una función).
- 8 Escribir un script que muestre un menú y haga lo siguiente:
  - ▶ Mostrar el directorio actual.
  - ▶ Pedir un nombre de directorio y si es correcto, establecerlo como actual.
  - ▶ Listar los elementos del directorio actual que solo sean archivos (Sin usar el comando **CMD ls**).
  - ▶ Pedir un nombre de directorio y crearlo. Si no se puede crear, personalizar el error.
  - ▶ Salir (Saludando al usuario que ejecuto el programa).
  - ▶ Si se presiona otra tecla, mostrar «Opción incorrecta».
- 9 Escribir un script que reciba como parámetro una cadena de caracteres y determine si es palíndromo<sup>1</sup> o no.

<sup>1</sup>Un palíndromo (del griego palin dromein, volver a ir atrás), también llamado palíndromo, palíndroma o palindroma, es una palabra, número o frase que se lee igual adelante que atrás.

## 12.4 INTEGRADORES

- 1 Dado un archivo log que puede contener mensajes de error, warning, critical, o info, se pide que ese archivo sea dividido, por criticidad de error, en cuatro archivos.
- 2 Dado un archivo que contiene las notas de los alumnos de una cursada, se pide obtener el promedio de notas, y la nota más alta. Informar ambas al finalizar la ejecución. El contenido del archivo es de la forma «alumno;nota». Tener en cuenta que puede haber cabeceras informativas antecedidas con el símbolo numeral «#»
- 3 De seguridad informática, se informa que hubo una incidencia no localizada a las dos de la mañana del domingo. Como parte de la investigación se tiene que verificar que no hay un job planificado a esa hora, con una lista de un número indeterminado de servidores.
- 4 Se pide de RRHH que se verifique en forma continua los logs de acceso, y cuando aparezca un legajo determinado, enviar un mail a rrhh@volonempire.gov.ar informando tal situación.
- 5 Escribir un script que: dado un archivo con contenido de la forma «UDP;166», debe realizar lo siguiente:
  - ▶ Leer ese archivo y aplique esas reglas para permitir el tráfico entrante de los protocolos y puertos especificados allí. Debe tomar el nombre de archivo como parámetro, y validar su existencia.
  - ▶ Al inicio, el script debe purgar las reglas existentes.
  - ▶ Definir como política por defecto, denegar el tráfico.
  - ▶ Al finalizar la ejecución, debe dejar las reglas persistentes en el sistema.
- 6 Dado un conjunto de archivos, cuyo nombre es de la forma: AUD\_YYYYMM. Ejemplo AUD\_200902.txt, se pide, escribir en una sola linea de sentencias anidadas, que cumpla lo siguiente:
  - ▶ Crear un directorio de la forma AAAA.txt.
  - ▶ Si el punto anterior finalizó satisfactoriamente, mover los archivos que cumplan con el AAAA del punto anterior al directorio respectivo.
  - ▶ Si la operación anterior finalizó satisfactoriamente, comprimir el directorio correspondiente generando un archivo comprimida de la forma AAAA.tgz.
  - ▶ Si la operación anterior finalizó satisfactoriamente, borrar el directorio correspondiente.



## Bibliografía

ÍNDICE DE REFERENCIAS



## Índice de figuras

1.1	Distribución de Sistema Operativos en Servidores. . . . .	21
1.2	Algunas de las distribuciones más populares. . . . .	23
1.3	Diálogo principal de VirtualBox. . . . .	26
1.4	Configuración de la memoria de la maquina virtual nueva. . . . .	26
1.5	Configuración de la unidad de almacenamiento. . . . .	27
1.6	Configuración formato de unidad de almacenamiento. . . . .	27
1.7	Crecimiento de la unidad de almacenamiento. . . . .	28
1.8	Tamaño inicial de la unidad de almacenamiento. . . . .	28
1.9	Finalización de la configuración de la VM. . . . .	29
1.10	Definición de la unidad de CD/DVDROM. . . . .	29
1.11	Selección del ISO correspondiente. . . . .	30
1.12	Booteo inicial de la VM. . . . .	30
1.13	Se selecciona de idioma de la instalación. . . . .	31
1.14	País de localización. . . . .	31
1.15	Distribución del teclado. . . . .	32
1.16	Autoconfiguración de la Red. . . . .	32
1.17	Se le da nombre al equipo «host». . . . .	33
1.18	Se le configura el dominio de la red. . . . .	33
1.19	Se configura la clave del usuario root. . . . .	34
1.20	Nuevamente se vuelve a introducir la contraseña de root. . . . .	34
1.21	Se crea un nombre de usuario completo no privilegiado. . . . .	35
1.22	Se crea un nombre de usuario no privilegiado. . . . .	35
1.23	Se elige la contraseña para ese nuevo usuario. . . . .	36
1.24	Se vuelve a introducir la contraseña. . . . .	36
1.25	Se selecciona todo el disco. . . . .	37
1.26	Se selecciona el disco. . . . .	37
1.27	Se selecciona partición separada para /home. . . . .	38
1.28	Se finaliza el particionado del disco. . . . .	38
1.29	Se escriben los cambios en el disco. . . . .	39
1.30	Comenzando a instalar paquetes. . . . .	39
1.31	Sigue instalando paquetes... . . . .	40

1.32	Sigue instalando paquetes... . . . .	40
1.33	Sigue instalando paquetes... . . . .	41
1.34	Sigue instalando paquetes... . . . .	41
1.35	Termino de revisar el ISO1 y se responde que no revise mas. . . . .	42
1.36	Se elije NO utilizar una replica de Red. . . . .	42
1.37	Se elije NO participar en la encuesta... . . . .	43
1.38	Se selecciona únicamente Utilidades Estándar del Sistema... . . . .	43
1.39	Se configura instalar GRUB. . . . .	44
1.40	Se elige el dispositivo donde se instala GRUB. . . . .	44
1.41	La instalación finalizó satisfactoriamente. . . . .	45
1.42	Se bootea el VM nuevamente y aparece el menu de GRUB. . . . .	45
1.43	La VM ya lista. . . . .	46
2.1	Elementos de una terminal de comandos . . . . .	55
2.2	RegExp: Solo números. . . . .	64
2.3	RegExp: Validador de Mail. . . . .	64
2.4	RegExp: Validador de hora en formato 24Hs . . . . .	65
2.5	RegExp: Búsqueda de expresión. . . . .	66
3.1	Proceso de instalación de software con apt. . . . .	70
3.2	Esquema de funcionamiento de <code>apt-cache search</code> . . . . .	71
3.3	Esquema de funcionamiento de <code>apt-get update</code> . . . . .	72
3.4	<code>https://debgen.github.io/</code> . . . . .	74
3.5	Estructura de los streams de un comando. . . . .	75
3.6	Una pipeline de programas ejecutándose desde una terminal. . . . .	80
4.1	Ranking de IDEs.[ <b>article:encuesta</b> ] . . . . .	85
4.2	Modos de operación del <code>vi/vim</code> . . . . .	86
4.3	Comando «set showmode». . . . .	88
4.4	Reemplazo en <code>vi</code> . . . . .	92
4.5	<code>vim-adventures</code> . . . . .	94
5.1	Relación entre los archivos <code>shadow</code> , <code>passwd</code> y <code>groups</code> . . . . .	96
5.2	Distribución del esquema de permisos . . . . .	107
6.1	Estados de los procesos en Linux. . . . .	117
7.1	Estructura de un filesystem conceptual vacío. . . . .	132
7.2	Estructura de un filesystem conceptual con «Hola». . . . .	134
7.3	Estructura de un filesystem conceptual con «Buen día». . . . .	134
7.4	Direccionamiento en filesystems de íodos . . . . .	135
7.5	Tres dispositivo montados. . . . .	137

7.6	Diálogo de VirtualBox. Configuración. . . . .	138
7.7	Almacenamiento. . . . .	139
7.8	Confirmación. . . . .	139
7.9	Se elige tipo de unidad. . . . .	140
7.10	Tamaño de la unidad. . . . .	140
7.11	Tamaño inicial. . . . .	141
7.12	Configuración finalizada. . . . .	141
7.13	Diagrama de un link duro. . . . .	148
7.14	Diagrama de un link simbólico . . . . .	150
9.1	Modelo OSI . . . . .	176
9.2	Elementos de una red LAN . . . . .	180
9.3	Paso 1 - Configuración del virtualbox. . . . .	182
9.4	Paso 2 - Agregar red solo-anfitrión . . . . .	182
9.5	Paso 3 - Verificar configuración de dicha red. . . . .	183
9.6	Paso 4 - Tomar nota de la dirección IP. . . . .	183
9.7	Paso 5 - Verificar que la opción DHCP se encuentra desactivada . . . . .	183
9.8	Acceso a la configuración de la VM . . . . .	184
9.9	Definir un adaptador secundario, anexándolo a la red solo-anfitrión . . . . .	184
9.10	Diagrama de tiempos de una negociación DHCP . . . . .	186
9.11	Esquema del cifrado simétrico . . . . .	191
9.12	Ejemplo de cifrado asimétrico. . . . .	192
9.13	Vista del archivo syslog. . . . .	201
12.1	Menu del GRUB . . . . .	245
12.2	A punto de editar la entrada correspondiente . . . . .	246
12.3	Agregando init=/bin/bash . . . . .	246
12.4	Booteo la entrada con F10 . . . . .	247
12.5	Verifico el estado del Filesystem / . . . . .	247
12.6	Remontar R/W y cambiar la password . . . . .	248



## Índice de cuadros

1.1	Diferencias entre GNU/Linux y *nix. . . . .	20
2.1	Clasificación de directorios- . . . . .	52
2.2	Comandos . . . . .	58
2.3	Secciones de las páginas man. . . . .	60
2.4	Concatenación de comandos en BASH. . . . .	61
3.1	Filtros . . . . .	81
3.2	Opciones usuales de grep. . . . .	81
4.1	Comandos básicos de vi. . . . .	87
4.2	Formas de invocar a vi. . . . .	87
4.3	Ejemplos de manejo de vi. . . . .	89
4.4	Movimientos del cursor en vi. . . . .	89
4.5	Control de pantalla en vi. . . . .	90
4.6	Comandos para ingreso de texto en vi. . . . .	90
4.7	Comandos para borrado de texto en vi . . . . .	90
4.8	Copiar y pegar en vi . . . . .	91
4.9	Búsqueda en vi. . . . .	91
4.10	Otros comandos en vi. . . . .	92
4.11	Modo «ex» o última línea. . . . .	92
4.12	Saltar a una línea específica en vi. . . . .	93
4.13	Algunas configuraciones en vi. . . . .	93
5.1	Campos en /etc/passwd . . . . .	97
5.2	Campos en /etc/shadow . . . . .	98
5.3	Campos en /etc/group . . . . .	99
5.4	Argumentos de useradd . . . . .	100
5.5	Argumentos de usermod . . . . .	101
5.6	Archivos de configuración de bash. . . . .	104
5.7	Comandos de administración y control de usuarios. . . . .	105
5.8	Tipos de archivo . . . . .	106

5.9	Esquema de permisos . . . . .	106
5.10	Equivalencias de permisos de archivos entre simbólico y octal. . . . .	108
5.11	Ejemplos de otorgamiento de permisos de archivos. . . . .	109
6.1	Descripción de los estados de los procesos en Linux. . . . .	118
6.2	Códigos adicionales a los estados de los procesos en Linux. . . . .	119
6.3	Columnas de <code>ps</code> . . . . .	120
6.4	Otros comandos relacionados . . . . .	128
7.1	Diferencias entre los distintos filesystems actuales. . . . .	133
7.2	Estructura del archivo <code>/etc/fstab</code> . . . . .	147
8.1	Principales características en systemd.[ <b>rh:systemd</b> ] . . . . .	155
8.2	Tipos de unidad en systemd. . . . .	155
8.3	Configuración de grub para Linux con root aparte . . . . .	162
8.4	Configuración de grub para Linux sin root aparte, . . . . .	162
8.5	Configuración de grub para Windows™. . . . .	163
8.6	Configuración de grub para una entrada Windows. . . . .	164
8.7	Nuevo diseño de GRUB2 . . . . .	167
8.8	Scripts en <code>/etc/grub.d/</code> . . . . .	169
8.10	Parámetros en <code>/etc/default/grub</code> . . . . .	172
9.1	Tipos de comunicación. . . . .	179
9.2	Elementos típicos de una red LAN. . . . .	179
9.3	Tabla de equivalencia entre <code>ifconfig</code> e <code>ip</code> . . . . .	180
9.4	Propiedades de la técnicas de cifrado. . . . .	190
9.5	Algunas directivas del archivo <code>/etc/ssh/sshd_config</code> . . . . .	197
9.6	Scripts en <code>/etc/ssh/ssh_config</code> . . . . .	197
9.7	Tablas de <code>iptables</code> . . . . .	198
9.8	Targets integrados. . . . .	199
9.9	Comandos de <code>iptables</code> . . . . .	200
9.10	Matches generales. . . . .	200
9.11	Targets/Jumps. . . . .	201
9.12	Guardar la configuración de <code>iptables</code> . . . . .	204
10.1	Variables intrínsecas de bash. . . . .	209
10.2	Caracteres especiales. . . . .	211
10.3	Palabras reservadas . . . . .	212
10.4	Evaluación para archivos . . . . .	212
10.5	Evaluación para cadenas de caracteres . . . . .	213
10.6	Globales . . . . .	221

10.7	Comparadores aritméticos.	225
10.8	Variables en <code>/etc/crontab</code>	232
10.9	Campos del <code>crontab</code>	232
10.10	Ejemplos de <code>crontab</code>	233
10.11	Ejemplos de <code>crontab</code> con valor o lista/incremento.	234