

Directorios en Linux.

Pablo Sanz Mercado.

Cuando utilizamos Linux, no tenemos una especial obligación en la utilización de un cierto directorio u otro para la ubicación de nuestros trabajos. En cualquier directorio, siempre que tengamos permisos para poder guardar ficheros en él, podemos almacenar nuestra información.

Si bien es cierto esto que acabamos de comentar, también lo es que Linux mantiene cierta jerarquía, cierta organización de los diferentes directorios. Respetar esta organización muchas veces es obligatorio, pues por ejemplo ciertos ficheros de configuración el sistema los buscará en determinados directorios y en ningún otro sitio (salvo que realicemos un trabajo extra para cambiar el Sistema Operativo), pero otras muchas veces simplemente se respeta esta distribución por el mero hecho de la buena organización.

Si nosotros guardamos los ficheros de usuarios en subdirectorios de /home, cualquier administrador que nos tenga que ayudar en un momento determinado no tendrá mayor problema al buscar esta información. Si hubiéramos decidido utilizar /casa, probablemente la persona que nos ayude empleará tiempo en darse cuenta de este *cambio* y por lo tanto perderemos efectividad.

De tal forma entonces comprenderemos que hay ciertos directorios en los que encontraremos básicamente siempre cierta información, sobre todo cuando nos referimos a ficheros de configuración del Sistema, archivos importantes del kernel y librerías.

1. /etc

En el directorio /etc nos encontramos los ficheros de configuración del sistema operativo Linux, de tal forma que cualquier cambio de configuración que queramos hacer, si no sabemos dónde encontrar el fichero para guardar este cambio, deberíamos buscar primero en /etc antes que en ningún otro sitio.

En este directorio nos podemos encontrar ficheros de configuración de librerías, de utilización de servicios de red, etc. Además podremos encontrar subdirectorios que incluyen a su vez ficheros de configuración.

Algunos de los ficheros de configuración y directorios más importantes son:

1.1. aliases

Guarda una lista de los alias de correo electrónico que tienen los diferentes usuarios válidos en el sistema, es decir, nos encontraremos ante un fichero organizado en dos columnas, la primera haciendo referencia a un usuario en concreto y la segunda a una dirección de correo electrónico.

Cada vez que el usuario reciba correo electrónico en la máquina, este se redireccionará a la cuenta que tengamos asignada.

Un ejemplo podría ser:

```
root: administrador.ccc@uam.es
```

Fijémonos que después del login encontramos el caracter dos puntos, y a continuación una cuenta de correo electrónico válida.

Decimos *válida* pues si no todo el correo electrónico que se mande a este usuario no se podrá llevar a su destino.

Si en este fichero no está definido un usuario, entonces todo el correo electrónico que le llegue será guardado en su cuenta local en el sistema.

Debemos tener cuidado especial en no crear ningún bucle, es decir, una configuración:

```
pablo: root
root: pablo
```

es decir, el correo electrónico del usuario pablo se envía al usuario root, y el del usuario root se envía al usuario pablo.

Si bien es posible que nos alarmemos ante una configuración de este tipo, hay que tener en cuenta que un fichero *aliases* puede llegar a contener muchas entradas, de tal forma que podríamos tener un descuido de este tipo.

Para que los cambios realizados a este fichero tomen efecto deberemos teclear el comando *newaliases*

1.2. **at.deny y cron.deny**

Estos dos ficheros contendrán una lista de usuarios (ordenados en filas), a los que no permitiremos la utilización de *at* y *cron* respectivamente.

Los ficheros *at.allow* y *cron.allow* tienen el significado contrario, es decir, contienen una lista de los usuarios a los que se les permite utilizar *at* y *cron* respectivamente.

Una forma de configurar estos servicios mediante la utilización de estos ficheros es escribir la lista de los usuarios a los que permitimos utilizar *at* y *cron*, de tal forma que aunque no escribamos nada en *at.deny* y *cron.deny*, sólo los que estén en *at.allow* y *cron.allow* podrán utilizar *at* y *cron* respectivamente.

En este sentido por tanto nada más instalar nuestro sistema tiene sentido que añadamos el login de root a *at.allow* y *cron.allow*, de tal forma que a partir de ese momento sólo root podrá utilizar estas utilidades del sistema, y conforme otros usuarios tengan que utilizarlas se les irá añadiendo.

1.3. **cron.hourly, cron.daily, cron.weekly y cron.monthly**

Estos directorios contienen una serie de ejecutables, de tal forma que estos se ejecutarán cada hora, cada día, cada semana o cada mes respectivamente.

Los ejecutables que contengan estos directorios bien pueden ser shell scripts (la mayoría), pero también pueden ser binarios, y por supuesto podremos localizar enlaces simbólicos que apunten a ejecutables situados en otra zona del árbol de directorios del sistema.

La forma de mantener tareas programadas en este sentido tiene sus detractores y también sus amantes. En contra podríamos decir que a priori (aunque lo podemos descubrir en los ficheros de configuración correspondiente), no sabemos exactamente en qué minuto de cada hora se ejecutarán los ejecutables contenidos en `cron.hourly`, a qué hora los contenidos en `cron.daily`, etc. En este sentido la programación mediante `cron` directamente es mucho más directa. Además tenemos muy restringida la programación, es decir, si queremos afinar un poco la programación (por ejemplo que se ejecute todos los domingos a las doce y un minuto de la noche), nos será imposible utilizar esta forma de programar tareas.

A favor tenemos la sencillez, pues simplemente realizando un enlace simbólico al ejecutable que acabamos de instalar en nuestro sistema y que queremos que se ejecute un número determinado de veces (pero nos da un poco igual en qué momento en concreto) tendremos solucionado nuestro requerimiento.

Por ejemplo con esta forma de programar tareas podremos ejecutar tareas de búsqueda de troyanos o de control de logs, que se deben hacer todos los días pero que no es muchas veces imprescindible que sea a una hora en concreto.

1.4. `ssh.login` y `ssh.cshrc`

Son dos ficheros que leerá la shell `ssh` cuando sea llamada, de tal forma que ejecutará lo que se le especifique en estos ficheros, cargando por ejemplo valores en diferentes variables.

El fin de estos ficheros por tanto es que los usuarios tengan un entorno homogéneo y es labor del administrador modificarlos consecuentemente con el fin de que los usuarios tengan las mínimas preocupaciones a la hora de crearse un perfil correcto de ejecución que funcione perfectamente en la máquina deseada.

Los administradores por tanto tendrán que cargar las variables correspondientes que vayan a ser necesarias. Por ejemplo tras la instalación de un programa de cálculo, es posible que para funcione correctamente los usuarios tengan que cargar alguna variable de alguna forma en concreto. Cargando estos valores en este archivo, ningún usuario tendrá que hacer absolutamente nada por su cuenta.

1.5. `exports`

Es el fichero donde tendremos que escribir los recursos que queremos exportar vía NFS, así como las propiedades de exportación.

Este fichero permanecerá vacío siempre y cuando no habilitemos un servidor NFS en nuestra máquina, y deberemos tener extremo cuidado en su configuración si habilitamos el servidor.

Un ejemplo bien podría ser:

```
/home maquina.dominio.es(rw)
```

fijándonos especialmente en los espacios que incluimos. Entre el nombre del recurso a exportar, y el nombre de la máquina (o conjunto de máquinas), que podrán acceder

al recurso, debe haber un espacio, no así entre el nombre de las máquinas y las propiedades de exportación (incluidas entre paréntesis).

1.6. fstab

En este fichero incluiremos los sistemas de archivos que vamos a montar en nuestro equipo.

Estos no tienen que ser montados automáticamente en el arranque ni mucho menos, sino que podemos tener sus líneas correspondientes para hacer más sencillo su montaje cuando sea necesario.

Este fichero se compone de diferentes líneas (una para cada sistema de archivos que queramos montar), y cada una de ellas está organizada en seis columnas.

En la primera columna escribiremos el dispositivo objetivo del montaje, como por ejemplo `/dev/hda1` si queremos especificar el montaje de la primera partición del primer disco IDE del sistema.

La segunda columna hace referencia al directorio donde se montará esta partición, por ejemplo `/home` si queremos que la información del dispositivo sea accesible desde este directorio.

En la siguiente columna especificaremos el sistema de archivos que vamos a utilizar para montar este sistema de archivos. Que podamos montar un sistema de archivos o no depende de si está habilitado en el kernel, es decir, si este *sabe* trabajar con este sistema de archivos. Los más habituales son `ext3` y `iso9660` (sistemas de archivos de linux y de los CDs/DVDs respectivamente), si bien podríamos especificar `vfat` o `ntfs` para particiones Windows, etc.

En la cuarta columna indicaremos las opciones que se incluirán a la hora de montar este dispositivo. Muchas veces nos encontraremos *defaults*, ya que así se utilizarán unas opciones que existen por defecto sin necesidad de escribirlas todas, pero podremos especificar todas las opciones que queramos, por ejemplo `rw` si queremos que sea un sistema de lectura y escritura, `user` si queremos que cualquier usuario lo pueda montar, `noauto` si no queremos que se monte automáticamente (con la opción `-a` de `mount`), etc.

La penúltima columna hace referencia a la utilización del comando `dump`. Este comando sirve para realizar copias de seguridad y puede ser utilizado de forma general para realizar un backup global del sistema. Si en la quinta columna de un sistema de archivos determinado aparece el número 1, `dump` podrá realizar el backup global también sobre este sistema de archivos, pero si aparece un 0 no podrá hacerse el backup de esta manera.

La sexta y última columna es utilizada por el comando `fsck` para realizar un chequeo de este sistema de archivos siempre que no hayamos escrito el número 0. En los casos en los que hayamos escrito el número 1 (siempre en el sistema de archivos `/`) o 2 (en cualquier otro sistema de archivos), se comprobará mediante `fsck` cuando sea montado.

1.7. group

En este fichero tendremos una lista con todos los grupos creados en nuestro sistema.

Se compone de una línea por cada grupo, en cada una de ellas podremos encontrar el nombre del grupo, la contraseña del mismo, el número identificador del grupo (gid) y la lista de los usuarios que pertenecen a este grupo. Todos estos campos se separan con el caracter dos puntos.

Es poco habitual que un grupo tenga contraseña, si bien podemos definirla sin mayor problema.

Este fichero debe tener permisos de lectura para todos los usuarios del sistema, ya que si un usuario, al hacer login en el sistema, no puede acceder a esta información, no podrá acceder al sistema.

1.8. gshadow

Es el equivalente a group en sistemas donde tengamos habilitadas las contraseñas shadow. Este sistema aumenta la seguridad del operativo, ya que las contraseñas se separan de los ficheros que tienen que ser accesibles a todos los usuarios del sistema y se alojan en ficheros accesibles sólo a root.

Es posible realizar esta separación ya que el proceso de autenticación lo lleva a cabo el superusuario, por lo tanto sólo este usuario necesita tener acceso al fichero, mientras que el resto de los usuarios simplemente necesita saber la información de su login, información que encontrarán sin mayor problema en los ficheros correspondientes y con permisos para que se pueda llevar a cabo esta lectura.

Hasta la aparición de shadow, en la mayoría de los sistemas encontrábamos las contraseñas, encriptadas, en ficheros accesibles, de tal forma que cualquier usuario podía hacerse con estos ficheros e intentar por tanto el crackeo de las mismas.

1.9. host.conf

En este fichero estableceremos el orden en el que se consultará la resolución de nombres.

Cuando intentamos realizar una conexión a un equipo, necesitamos saber su IP. Si lo que tenemos es su nombre, debe ser resuelto. Este proceso se puede hacer mediante ficheros de configuración en el propio sistema, mediante la consulta en un servidor DNS u otras opciones menos utilizadas.

En este fichero estableceremos por tanto el orden en que realizaremos esta consulta, siendo normalmente la primera opción la consulta local en los ficheros del sistema habilitados, y como segunda opción la consulta en servidores de nombres, si bien puede ser cambiado este orden o incluso modificado para incluir otras opciones (por ejemplo utilizando servicio NIS).

Un ejemplo típico de este fichero es:

```
order hosts,bind
```

donde hosts indica *ficheros locales* y bind *consulta a un servidor de nombres*.

1.10. hosts

Fichero que contiene relaciones IP/nombre para que el sistema operativo pueda resolver direcciones sin necesidad de hacer consultas a los servidores de nombre.

Este fichero está estructurado por líneas, de tal forma que en cada una de ellas nos encontraremos una dirección IP, seguida de su nombre correspondiente y a continuación uno o varios alias para esta dirección IP. De esta forma podremos realizar una conexión a una IP en concreto utilizando su nombre equivalente o cualquier alias que hayamos elegido, siempre y cuando este fichero se consulte en primera instancia.

Si tenemos un error en la configuración de este fichero tenemos que tener en cuenta que las conexiones que realicemos pueden no llevarse a cabo o realizarse de forma errónea.

Un ejemplo podría ser:

```
64.233.183.99 www.google.com buscador
```

de tal forma que podremos acceder a la página web de Google escribiendo en nuestro navegador cualquiera de estas tres opciones:

```
http://64.233.183.99
http://www.google.com
http://buscador
```

1.11. hosts.allow y hosts.deny

Son los ficheros de configuración de la herramienta tcpwrapper.

Esta herramienta nos permite realizar un control de acceso a los servicios de nuestra máquina que ofrezcamos mediante el protocolo tcp. tcpwrapper comprobará la IP del equipo que intenta realizar la conexión y, atendiendo a la configuración de estos ficheros, permitirá el acceso o no.

La configuración por defecto que deberíamos tener en el fichero hosts.deny es:

```
ALL:ALL
```

que impedirá cualquier conexión, configurando entonces el fichero hosts.allow

```
servicio:IP
```

para permitir que el ordenador con ip *IP* acceda al servicio *servicio*.

1.12. inittab

En este fichero se indican qué procesos son ejecutados en el arranque así como durante la operación normal del sistema operativo.

Si bien mucha de la información contenida en este fichero permanece invariable desde la instalación del sistema operativo, muchas veces los administradores modifican la línea donde se indica qué nivel de ejecución se alcanzará tras arrancar la máquina, es decir, qué nivel de ejecución tendremos por defecto

`id:5:initdefault:`

en este ejemplo tendremos que el nivel de ejecución por defecto es el cinco, si bien podríamos cambiarlo a 3 en una máquina de cálculo, para evitar que se arranque el entorno gráfico con el consiguiente consumo de cpu y memoria RAM que conlleva.

La sintaxis de las líneas incluidas en este fichero consiste en cuatro campos separados por el caracter dos puntos. El primer campo es un identificador (de cuatro caracteres máximos, o dos para versiones de sysvinit compiladas con libc5 < 5.2.18). Este identificador tiene que ser único en todo el archivo.

El segundo campo identifica el nivel de ejecución donde se aplicará esta línea. En este campo podremos tener un sólo nivel de ejecución, varios o ninguno, en el caso de que queramos ejecutar algo sólo al alcanzar un nivel de ejecución, cuando alcancemos varios o cuando arranquemos la máquina respectivamente.

El cuarto campo indica el proceso, el comando que se ejecutará.

Finalmente el tercer campo indica la acción de lo que queremos realizar. El número de acciones es finito, siendo su significado:

respawn. El proceso se rearrancará tan pronto se termine. *wait*. El proceso se ejecutará cuando se acceda al nivel de ejecución indicado e init esperará a su ejecución. *once*. El proceso se ejecutará una vez cuando se alcance el nivel de ejecución indicado. *boot*. El proceso se ejecutará en el arranque. El campo de los niveles de ejecución será ignorado por tanto. *bootwait*. El proceso se ejecutará en el arranque, pero en esta ocasión init esperará hasta que se lleve a cabo. El campo de los niveles de ejecución también será ignorado en este caso. *off*. No tiene efecto. *ondemand*. Se ejecutará el proceso cuando se llame al nivel de ejecución ondemand. Estos niveles son el a, b y c, y cuando se llaman no conllevan un cambio en el nivel de ejecución. *initdefault*. Indica el nivel de ejecución al que se accederá cuando la máquina arranque. Si no hay indicación al respecto, el sistema lo preguntará interactivamente en consola. *sysinit*. El proceso se ejecutará en el arranque antes de cualquier proceso con acción boot o bootwait. El campo de los niveles de ejecución es ignorado por tanto. *powerwait*. El proceso se ejecutará cuando init recibe una señal de falta de corriente eléctrica (mandada por un sistema de alimentación ininterrumpida por ejemplo). init esperará hasta que se ejecute el proceso indicado. *powerfail*. Se ejecuta el proceso igual que con powerwait, pero en este caso init no esperará a su final. *powerokwait*. Se ejecutará el proceso cuando init reciba una señal que indique que se ha restablecido el proceso de alimentación eléctrica. init esperará a su ejecución. *powerfailnow*. Si el sistema de alimentación ininterrumpida conectado al sistema es

capaz de informar a `init` que su autonomía está a punto de terminar, se ejecutará el proceso que se indique con esta acción. *ctrlaltdel*. Se ejecutará el proceso cuando se reciba la secuencia `Ctrl+Alt+Spr`. *kbrequest*. Si hemos asignado una combinación de letras a la acción *KeyboardSignal*, cuando se utilice esta combinación se detectará y se ejecutará el proceso asignado a esta acción.

1.13. `issue`

Es un fichero de texto que se mostrará en pantalla antes del indicador de entrada en el sistema. Podremos modificar su contenido para mostrar mensajes institucionales o con información al usuario.

1.14. `issue.net`

Es equivalente a *issue*, pero su contenido se mostrará en las conexiones telnet que se hagan contra nuestra máquina.

Es conveniente sobre todo en este caso no mostrar información sensible de nuestra máquina en este fichero. Hay que tener en cuenta que su contenido se mostrará antes de que se realice un login correcto en nuestro sistema, de tal forma que si dejamos el contenido por defecto (habitualmente la versión de sistema operativo que se está ejecutando), estamos dando una valiosa información a un hacker que quiera acceder a nuestro equipo, pues a partir de ese momento se centrará en las vulnerabilidades que pueda haber en nuestra versión de operativo en vez de intentar cualquier vulnerabilidad conocida.

1.15. `ld.so.conf`

En es fichero encontraremos la configuración utilizada por el cargador/enlazador `ld.so` (o `ld-linux.so*`), que buscan y cargan las librerías compartidas que necesita un programa que se necesite ejecutar.

En este fichero por tanto escribiremos la ubicación (directorio), con la ubicación de las librerías compartidas que tengamos en nuestro sistema.

Cada vez por tanto que instalemos librerías compartidas en nuestro sistema, que entendamos que necesiten ser cargadas en algún momento, escribiremos este directorio en este archivo y ejecutaremos el comando `ldconfig` para que el cambio tenga efecto.

Actualmente se utiliza una sintaxis un tanto distinta, pero totalmente equivalente, para tener mayor orden en este sentido. En el fichero `ld.so.conf` lo que se hace es incluir la línea:

```
include ld.so.conf.d/*.conf
```

de tal forma que al ejecutar *ldconfig*, al leer este fichero obligaremos a que se lea cualquier fichero, con extensión *.conf*, existente en el directorio `/etc/ld.so.conf.d`

Por lo tanto cada vez que instalemos una librería compartida en nuestro sistema, crearemos un fichero `.conf` en el directorio `/etc/ld.so.conf.d` que contenga el directorio donde se encuentra la (o las) librería compartida.

En este sentido la configuración es mucho más sencilla, pues en vez de tener un archivo único con muchas entradas (muchas veces desordenadas y complicadas de comprobar), tendremos varios archivos con nombres característicos (siempre que acaben en `.conf` dará lo mismo cómo se llamen), que cumplirán la misma función cara al sistema pero que serán mucho más fáciles de ordenar para el administrador.

1.16. `login.defs`

Define la configuración que utilizará `shadow` en la creación de logins.

En este fichero indicaremos las características comunes que tendrán las cuentas que creamos en nuestro sistema, como puede ser la máscara por defecto, la caducidad del login, la creación o no del directorio `home`, etc.

1.17. `logrotate.conf`

Configura la acción de `logrotate`. Este comando puede rotar los logs del sistema, comprimirlos e incluso mandarlos por correo electrónico, según lo configuremos en su fichero de configuración.

Hay que tener en cuenta que si no realizamos ninguna modificación de los ficheros de log, estos al cabo del tiempo se volverán inmanejables debido a su gran tamaño.

En el fichero `logrotate.conf` configuraremos por tanto si estos ficheros tienen que ser rotados, si tienen que ser comprimidos, etc.

Cuando decimos *rotar*, lo que queremos indicar es que el fichero de log cambiará de nombre, por ejemplo se llamará `log.1`, el fichero que se llamara `log.1` se llamará `log.2` y así consecutivamente. El fichero más antiguo se eliminará (por ejemplo), y así tendremos un sistema en el que el fichero de log correspondiente no alcanza un tamaño que le haga inmanejable y donde tendremos un número finito de archivos de log históricos.

Una configuración habitual de este fichero de configuración puede ser:

```
weekly
rotate 4
create
compress
```

donde de forma semanal (*weekly*) se rotarán los logs, guardando cuatro semanas de antigüedad (*rotate 4*). Como cambiamos de nombre los ficheros de log al rotarlos (les añadiremos la extensión `.1`, `.2`, etc), se creará un nuevo fichero de log sin esta extensión para guardar los logs actuales (*create*), y se comprimirán los ficheros de logs (*compress*).

1.18. `modprobe.conf`

Este fichero indica la configuración necesaria para la herramienta `modprobe`. Este comando carga en memoria módulos del kernel, pero incluyendo las dependencias si es que las tiene, a diferencia del comando `insmod`.

Estas dependencias son incluidas en este fichero de configuración, así como en ficheros que podremos incluir en ficheros incluidos en el directorio `/etc/modprobe.d`

Además de dependencias, en este fichero podremos incluir alias, es decir, nombres distintos para los módulos que deseemos, así como características extraordinarias para la carga de módulos.

Esta última utilidad, la de incluir características extra de carga de módulos en memoria, tenemos que configurarlas con mucho cuidado, pues una carga de un módulo de forma incorrecta puede producir errores en el funcionamiento del equipo. Habitualmente es el fabricante del dispositivo al que hace referencia el módulo, o el desarrollador del módulo quien indica qué variables se pueden utilizar en la carga del módulo para modificar su funcionamiento. El realizar nosotros cualquier cambio sin estar seguros de su finalidad última puede acarrear, como hemos indicado, una inestabilidad de la utilidad a la que haga referencia el módulo así como del sistema.

1.19. `motd`

`motd` es el acrónimo de Message Of The Day, es decir, en este fichero podremos escribir la información que nuestros usuarios lean cuando hagan login, y por lo tanto es una buena manera de mantener informados a los usuarios de las noticias que creamos interesantes para ellos.

1.20. `mtab`

Es un fichero donde está incluida la información de los dispositivos montados en nuestro sistema.

Es un fichero que en un principio un administrador no debe modificar en ningún momento, pues es el sistema el que lo modifica cada vez que se monte o desmonte un dispositivo, y cualquier modificación puede llevar consigo un mal funcionamiento del sistema por incongruencia entre lo que se ha montado y lo que el sistema cree que tiene montado (el contenido de este fichero).

Algunas veces es necesario modificar el fichero, y normalmente es cuando hay algún problema grave con alguno de los dispositivos montados. Por ejemplo si accedemos a un dispositivo mediante NFS, y en un momento dado perdemos la conexión (pérdida de conectividad por ejemplo), es posible que ciertas características de nuestro sistema no funcionen correctamente (por ejemplo el comando `df` podrá *colgar* una terminal al ser ejecutado, pues no podrá acceder a la información del recurso importado). Un método (quizás agresivo), que nos puede permitir recuperar cierto control en el sistema (pero que también puede hacerle inestable), es eliminar

la línea de este fichero que esté provocando la inestabilidad. En este momento el sistema creará que no tiene montado este dispositivo y por tanto los errores que estuviéramos teniendo dejarán de existir.

1.21. `passwd`

Es el fichero donde se encuentra las características de los login válidos en nuestro sistema. Este fichero está organizado por filas, cada una de ellas conteniendo la información de un login en concreto.

La organización de las filas viene indicada por campos separados por el caracter dos puntos. El número de campos es siempre siete.

El primer campo siempre es el login del usuario. A continuación tendremos la contraseña encriptada (tendremos el caracter x en el caso de que las contraseñas encriptadas no estén en este fichero sino en el fichero shadow). El tercer campo indica el número identificador del usuario (uid). Seguidamente el cuarto campo indica el número identificador del grupo principal del usuario. A continuación, en el quinto campo, obtendremos una descripción del login (que podremos dejar en blanco si así lo deseamos). El sexto campo es el directorio *home* del usuario. El séptimo y último campo indica la shell por defecto del usuario.

Este fichero es imprescindible en el sistema. Cualquier error en el mismo provocará la imposibilidad de login del usuario afectado, y por lo tanto su ausencia provocará la imposibilidad de ningún usuario pueda acceder al sistema (incluido el superusuario).

Como contiene información vital para los usuarios, estos deben tener permiso de lectura sobre el mismo.

1.22. `profile`

Es el equivalente su cometido a `csh.login` pero para los usuarios de shell `bash`.

1.23. `rc`

Es un script responsable de arrancar/parar servicios cuando un nivel de ejecución cambia.

Este script no debe ser modificado salvo que sepamos exactamente lo que estamos haciendo, pues un mal funcionamiento del mismo haría que los cambios de nivel de ejecución se realizaran erróneamente.

1.24. `rc0.d`, `rc1.d`, `rc2.d`, `rc3.d`, `rc4.d`, `rc5.d`, `rc6.d`

Cada uno de los directorios que consultará el script `rc` cuando cambiemos al nivel de ejecución al que haga referencia su número.

Es decir, si cambiamos al nivel de ejecución 2, el script `rc` consultará el directorio `rc2.d` y ejecutará (por orden alfabético), todos los scripts que haya en este directorio

(habitualmente enlaces simbólicos que apuntan a scripts ubicados en `/etc/init.d` o en cualquier otra parte accesible del sistema). Esta ejecución será acompañada con el argumento *start* si el nombre del fichero empieza por S, o con *stop* si empieza por K.

Como hemos dicho que la ejecución será en orden alfabético, primero se ejecutarán todos los ficheros que empiecen con K, seguidos de los que empiecen por S.

1.25. `rc.local`

Es un script que se ejecutará tras el resto de los scripts *init*, de arranque del sistema, de tal forma que podremos escribir en él ejecutables (o scripts) que queramos se ejecuten, independientemente del nivel de ejecución alcanzado.

1.26. `rc.sysinit`

Este script se ejecutará en el arranque, y permite por ejemplo cargar las especificaciones de la red, de usb, etc.

Cuando arrancamos el ordenador, el núcleo ejecuta *init*. Este comando buscará el fichero `rc.sysinit` y ejecutará su contenido. A continuación se ejecutará, si existe, el fichero `rc.serial`. Hazto seguido se ejecutará los guiones del nivel de ejecución por defecto (establecido en `inittab`) y finalmente se ejecutará `rc.local`

1.27. `resolv.conf`

Fundamentalmente este fichero contiene la lista de servidores de nombre que consultará el sistema cuando tenga que realizar una resolución de nombres gracias a un servidor externo.

La sintaxis es:

```
nameserver IP
```

donde IP es el número ip del servidor de nombre.

Podremos escribir tantas líneas como queramos, y el orden de consulta será el orden de las líneas, es decir, primero se realizará la consulta al servidor de nombres que aparezca en la primera línea, en el caso de que no se produzca correctamente la consulta se utilizará el siguiente, etc.

Además de especificar la lista de servidores de nombres que consultará el sistema, también podremos especificar nuestro dominio

```
domain universidad.es
```

el orden de búsqueda

```
search uam.es
```

y podremos especificar también opciones de búsqueda.

En cuanto al orden de búsqueda indicar que se utilizará cuanto intentemos una conexión utilizando un nombre de equipo incompleto. Por ejemplo si queremos conectarnos al equipo `www.uam.es`, si escribimos únicamente `www` el ordenador se dará cuenta de que no hay ningún equipo válido con ese nombre y entonces intentará buscar el equipo `www.uam.es` (según hemos indicado con *search uam.es*). Si encuentra entonces un equipo válido entonces realizará la conexión. Podremos escribir tantas líneas *search* como nos sea necesario, utilizándose por el sistema en orden.

Por supuesto si nuestro equipo, antes de realizar una consulta al servidor de nombres, busca la resolución en ficheros locales, si en estos existe un alias que coincida con el nombre corto que hemos utilizado, se utilizará este y no el que se obtenga al usar *search*.

1.28. **securetty**

Es el fichero donde incluiremos las ttys donde root puede hacer login, para evitar que por ejemplo root pueda realizar una conexión mediante telnet o en ciertas consolas.

1.29. **services**

Este fichero incluye una lista de los servicios de red de Internet, es decir, en el tendremos una lista del nombre del servicio, el puerto y protocolo utilizado y en algunas ocasiones un alias:

<code>http</code>	<code>80/tcp</code>	<code>www www-http</code>
<code>http</code>	<code>80/udp</code>	<code>www www-http</code>

Todo programa que necesite realizar una conexión tendrá que leer este fichero para conocer el puerto que debe utilizar. Cualquier error por tanto en la configuración del mismo acarreará un error en la conexión. Si intentamos realizar una conexión y no existe la línea correspondiente, es posible que esta no pueda llevarse a cabo.

1.30. **shadow**

Es el fichero que contiene las contraseñas encriptadas cuando utilizamos shadow.

En este sentido aumentamos la seguridad del sistema, pues ningún usuario (salvo root por supuesto), podrá leer este fichero (si los permisos del mismo son correctos), y por tanto no se podrá realizar un crackeo de contraseñas, que sí se puede llevar a cabo por un usuario no privilegiado del sistema si las contraseñas encriptadas se encuentran en el fichero `passwd`, ya que este debe tener permisos de lectura para los usuarios (aun no siendo privilegiados).

El que sea posible que el fichero shadow no tenga permisos de lectura para los usuarios, sólo para root, es debido a que el usuario encargado de la autenticación en el sistema es root, y por lo tanto ningún usuario necesita acceder a esta información.

1.31. shells

Cualquier shell que quiera ser utilizada en el sistema debe estar contenida en este fichero, organizado en líneas.

Si realizamos la instalación de una nueva shell en nuestro equipo, esta no funcionará hasta que la incluyamos en este fichero, modificable sólo por root.

Así sólo podrá ser utilizadas, en un principio, las shells autorizadas por el superusuario, evitando cualquier tipo de acceso *anormal* en el sistema.

1.32. sysconfig

El directorio sysconfig alberga una gran cantidad de ficheros de configuración, donde podremos encontrar por ejemplo la configuración de la red.

1.33. sysctl.conf

Este fichero de configuración, desgraciadamente, es un gran desconocido entre los administradores de sistema.

La configuración de este fichero es utilizada por el comando sysctl para configuraciones opciones del kernel existentes en el directorio /proc/sys

Lo que podemos modificar en este directorio /proc/sys lo podemos saber ejecutando el comando

```
sysctl -a
```

cuya ejecución nos dará una larga lista de opciones. Modificar cualquiera de estas opciones cambiará ciertos comportamientos que tiene el kernel, cara al sistema de archivos, red, etc. de tal forma que hay que tener cierto cuidado en realizar cambios sólo y exclusivamente cuando estemos seguros de no causar ningún error en el sistema.

La forma de cambiar estas características es modificando los archivos existentes en /proc/sys, que tienen los valores que hemos obtenidos tras la ejecución de sysctl -a Esta modificación la podemos realizar mediante el comando sysctl:

```
sysctl -w a.configurar="valor"
```

o también modificando el fichero correspondiente mediante el comando echo

```
echo "valor" > archivo.a.configurar
```

Al ser `/proc` un sistema virtual, cualquier modificación se perderá en el siguiente reinicio, por lo tanto si queremos que todas estas modificaciones se realicen siempre que arranque el sistema operativo, las escribiremos en el archivo `sysctl.conf` y por tanto serán modificadas sin nuestra interacción.

Un ejemplo que podríamos poner es desactivar la respuesta al ping, es decir, que nuestra máquina no respondiera al ping de otra máquina remota. Esto lo podemos hacer ejecutando:

```
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

o con el correspondiente comando `sysctl`.

Para evitar tener que teclear esto cada vez que arranquemos el sistema, lo que haremos entonces es incluir en el fichero `sysctl.conf` la siguiente línea:

```
net/ipv4/icmp_echo_ignore_all=1
```

donde podemos observar que la sintaxis cambia un poco: Evitamos la inclusión de `/proc/sys/` en la ruta del archivo a modificar, y lo que hacemos es añadir el signo igual seguido del valor que queremos incluir en este fichero.

1.34. `timezone`

En este fichero tendremos definida la zona horaria donde nos encontramos.

Habitualmente no se suele modificar este archivo, pues no es habitual cambiar un ordenador de localización geográfica, y su configuración viene predefinida por la elección que hayamos hecho a la hora de instalar el sistema operativo. No obstante, si lo necesitamos, siempre podremos modificar su contenido para cambiar nuestra zona horaria.

Una elección incorrecta de la zona horaria de nuestro sistema no sólo nos dará problemas por ejemplo en la recepción/envío de mensajes de correo electrónico (desfasados sus marcas de tiempo debido a esta configuración), sino que afectarán al buen funcionamiento del sistema de colas, por ejemplo, donde integremos el equipo ya que tendrá un desfase horario con respecto al resto de los equipos. También por supuesto podremos tener problemas al utilizar NFS y NIS, y en general con cualquier servicio que necesite una correcta sincronización horaria.

1.35. `xinetd.d`

El directorio `xinetd.d` contiene una serie de archivos que contienen la configuración de diferentes servicios que ofrece el super demonio `xinetd`

Cada uno de los archivos contenidos en este directorio mantiene la configuración de un servicio instalado en nuestro equipo, con el que por ejemplo se podrá permitir o evitar la ejecución de este servicio, controlar el binario que se debe ejecutar, el tipo de seguridad empleado, etc