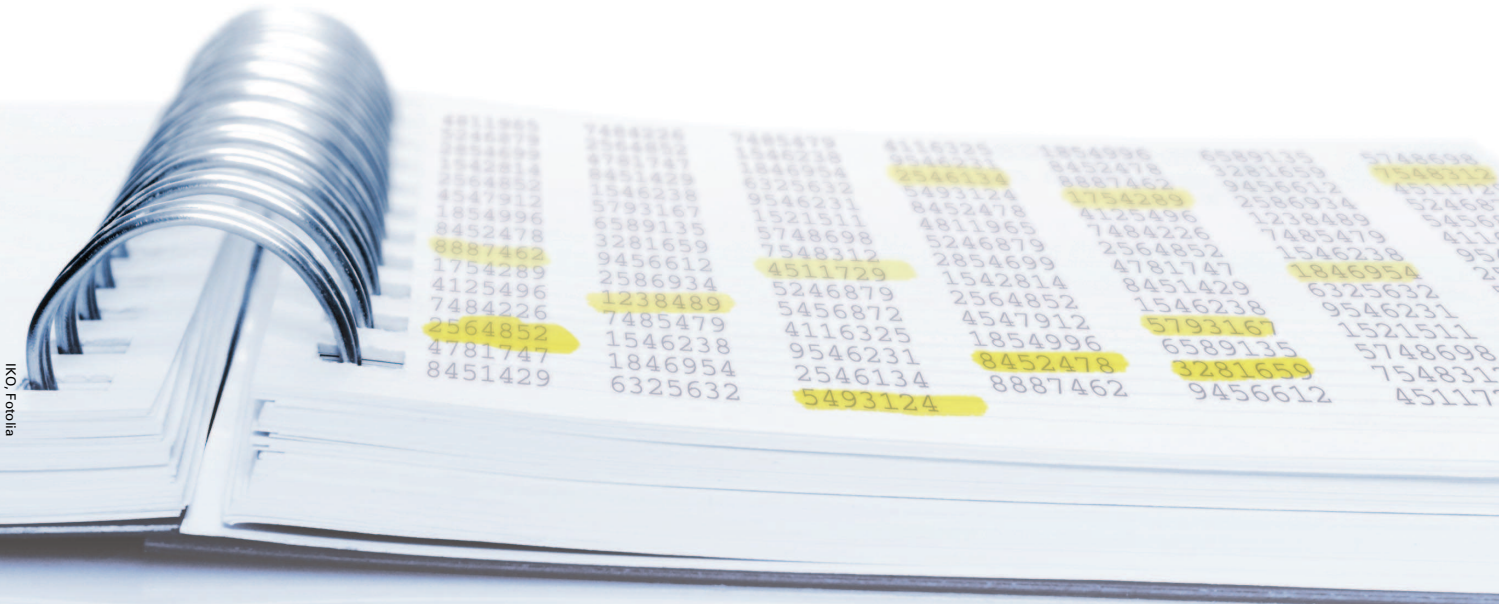


Trabajando con Listas de Control de Acceso

EN LA LISTA



El antiguo sistema de permisos de Linux es a veces insuficiente en entornos de producción complejos. Las Listas de Control de Acceso ofrecen una alternativa flexible. **POR TIM SCHÜRMANN**

Alicia aprecia las ventajas de poseer una agenda electrónica en su PC, pero para mantener su privacidad, ha establecido una serie de permisos estrictos al fichero de la agenda: Ella puede añadir nuevas citas, pero el resto de los miembros de su grupo de trabajo sólo poseen acceso de lectura. El resto de los usuarios que no pertenecen a su grupo de trabajo ni siquiera pueden mirar el contenido.

En principio, esta configuración es la adecuada, pero un día llega Bob de otro departamento para colaborar con Alicia. Para permitir la colaboración tendrá que darle a Bob acceso a su agenda.

En este escenario está claro que el sistema de permisos clásico de Linux no es el adecuado. Para permitir que Bob pueda leer el fichero con los datos de su agenda, Alicia tiene que pedirle a su administrador que meta a su nuevo colega en su grupo de trabajo, pero con ello se le permitiría a Bob ver el resto de los documentos que producen el equipo de trabajo de Alicia.

Otra solución consistiría en establecer temporalmente un nuevo grupo con las cuentas de Bob y Alicia como miembros del mismo. En este simple escenario, un grupo temporal podría ser una solución

aceptable, pero en un entorno empresarial real, la gestión de los grupos llega a ser algo más complicada, y el hábito de crear grupos temporales puede llegar a producir demasiados grupos sin una buena forma de seguirles la pista.

Las Listas de Control de Acceso, o ACLs para abreviar, prometen una solución. Las ACLs añaden un control de acceso flexible al sistema de permisos clásico de UNIX, permitiendo a los usuarios añadir permisos a cualquier grupo o usuario. Alicia no tendrá que molestar a su administrador; simplemente tendrá que poner a Bob en la lista de los usuarios autorizados e incluso puede establecer los permisos por defecto para todos los ficheros nuevos.

Las Listas de Control de Acceso llevan tiempo existiendo y gradualmente se están incorporando en los entornos empresariales; sin embargo, la estructura de seguridad de las ACLs aún no le es familiar a muchos usuarios de Linux. En este artículo vamos a mostrar cómo comenzar con las ACLs en Linux.

Rotación de Discos

Si planea utilizar ACLs, el sistema de ficheros tiene que soportar atributos extendidos.

De los sistemas de ficheros actuales, Ext2, Ext3, Ext4, ReiserFS, JFS y XFS soportan ACLs. JFS y XFS soportan atributos extendidos por defecto; para el resto hay que establecer la opción de montaje *acl* para habilitar el soporte de ACLs.

```
mount -o remount, acl, z
defaults mount_point
```

La mayoría de las distribuciones actuales establecen estos parámetros por defecto en */etc/fstab*:

```
/dev/hda1 / ext3z
acl, user_xattr 1 1
```

Para los discos internos no hay que modificar nada y las ACLs también funcionan con NFS así como con NFSv3, suponiendo que el servidor posea un sistema de ficheros y un sistema operativo que soporte ACLs.

Problemas del Kernel

Además del sistema de ficheros, el kernel también soporta ACLs, después de todo, es el kernel el que finalmente autoriza o niega el acceso al fichero. Todos los kernels actuales de la serie 2.6 soportan las ACLs,

existiendo parches para el kernel 2.4.x. Las principales distribuciones normalmente habilitan los atributos extendidos para todos los sistemas de ficheros mencionados anteriormente, permitiendo a los usuarios comenzar a asignar los permisos desde cero. Para asegurarse, sólo hay que introducir el siguiente comando:

```
grep "XATTR|POSIX_ACL" \
/boot/config-$(uname -r)
```

Debería mostrar dos entradas con `=y` si las ACLs están soportadas. Ext2, por ejemplo, mostraría:

```
CONFIG_EXT2_FS_XATTR=y
CONFIG_EXT2_FS_POSIX_ACL=y
```

De otro modo, no habría más alternativa que instalar un kernel nuevo.

Una Banda de Dos

Además del soporte del sistema de ficheros y del kernel de Linux, hace falta un paquete con aplicaciones que muestren las ACLs de cada fichero y las modifiquen según las necesidades. La mayoría de las distribuciones incluyen paquetes denominados *acl* que se encargan de estas funciones. Dos de los programas que incorporan son particularmente útiles:

- *getfacl* muestra las ACLs de un fichero
- *setfacl* establece o cambia los permisos de un fichero

Ambas herramientas se basan en las librerías *libattr* y *libacl*, que la mayoría de las distribuciones instalan por defecto.

Historia

Para comenzar este estudio sobre las ACLs daremos un rápido repaso al sistema de control de acceso clásico de Linux. Normal-

POSIX y ACLs

Puede que le deje desconcertado el término POSIX (Portable Operating System Interface) ACLs en Internet y en la documentación. Aunque diversos borradores aparecieron a finales del siglo pasado (POSIX 1003.1e, conocido normalmente como POSIX.1e y 1003.2c), por diversas razones, nunca se aprobaron. La mayoría de las implementaciones de las ACLs aún se basan en estos borradores. Para subrayar esta relación, muchos autores utilizan el término POSIX ACLs [1].

mente en Linux, cada objeto del sistema de ficheros distingue entre tres roles diferentes, que se denominan *clases* en la terminología POSIX: el propietario del fichero (*user*), el grupo al que pertenece el fichero (*group*) y el resto de los usuarios del sistema (*other*). El propietario especifica, para cada una de las clases, si es de lectura, de escritura o incluso si el fichero se puede ejecutar. El comando *ls -l* muestra estos permisos como una lista codificada de letras:

```
-rw-r--r-- 1 alicia ateam \
5410 7. Feb 11:21 calendar.cal
```

En este caso, *alicia* es la propietaria y su grupo se denomina *ateam*. *ls* distingue entre lectura (read), escritura (write) y ejecución (execute) por sus primeras letras: *read*, *write*, *execute*. Por ello, para cada clase, se utiliza una terna formada por *rwX* para indicar el permiso que posee. Un guión (-) en cualquier posición indica que la operación está prohibida para esa clase.

Minimalismo

Imagínese una ACL como un trozo de papel en el que se anota una lista con todos los permisos de acceso y los derechos asignados a los mismos. Linux grapa esta lista al fichero y se encarga de que se cumplan sus permisos. En la práctica, Alicia primero comprueba qué permisos están ya asignados para su agenda. Para ello utiliza el comando *getfacl*, que muestra la ACL de un fichero. Como aún no ha incluido a Bob, su lista aparecerá vacía:

```
$ getfacl calendar.cal
# file: calendar.cal
# owner: alicia
# group: ateam
user::rw-
group::r--
other::r--
```

Para una lista que debería aparecer vacía, ésta tiene unas cuantas entradas.

Primero, *getfacl* repite el nombre del fichero, el propietario y el grupo del mismo en las tres primeras líneas. Las líneas siguientes contienen cada una exactamente una entrada de la ACL; a esto se le conoce como una Entrada de Control de Acceso (ACE). Para mantener la compatibilidad con el sistema clásico, la ACL asigna automáticamente los permisos existentes con entradas en la lista, lo que explica las tres

líneas siguientes en el ejemplo anterior. La primera línea muestra los permisos de propietario, la segunda los permisos del grupo y la tercera la del resto de los usuarios.

Las entradas coinciden exactamente con las que muestro el comando *ls -l*. Como estas entradas existen en cualquier ACL, se conocen como ACL mínima. Tan pronto como se añada una entrada, se conocerá como ACL extendida.

Estableciendo las ACEs

Para autorizar a Bob acceder a la agenda, Alicia tiene que añadir otra entrada a esta ACL. La herramienta *setfacl* se encarga de hacerlo:

```
setfacl -m user:bob:rw- \
calendar.cal
```

A primera vista los parámetros parecen enigmáticos: con el parámetro *-m* se crea una nueva entrada en la ACL del fichero *calendar.cal*. Se autoriza el acceso a un único usuario (*user*) llamado *bob*. Bob puede leer y escribir en el fichero, pero no puede ejecutarlo (*rw-*). Ahora *getfacl* muestra la siguiente lista:

```
$ getfacl calendar.cal
# file: calendar.cal
# owner: alicia
# group: ateam
user::rw-
user:bob:rw-
```

Windows y ACLs

Windows, tanto la familia de sistemas operativos NT como XP, también soporta las ACLs, pero sólo con el sistema de ficheros NTFS. Aunque el acceso a NTFS desde Linux normalmente es posible, Linux sólo ha limitado el soporte a su funcionalidad extendida, que desafortunadamente incluye las ACLs.

Al menos Samba soporta las ACLs, suponiendo que el sistema subyacente soporte los permisos extendidos. Los ficheros almacenados en un servidor Samba mantienen sus permisos como si estuvieran almacenados en un disco NTFS. Surge un problema: Como las ACLs que utiliza Windows difieren de las utilizadas por Linux, Samba actualmente proporciona sólo una parte de la funcionalidad a la que están acostumbrados los usuarios de Windows.

```
group::r--
mask::rw-
other::r--
```

Cada entrada de la ACL sigue el mismo patrón, comenzando con el tipo, que especifica a quién hay que aplicarle los siguientes permisos, que puede ser a un simple usuario o a todo un grupo. Tras los dos puntos viene una etiqueta que designa el nombre del usuario o del grupo al que pertenece la entrada. En los casos en los que no sea necesario este nombre, simplemente se puede omitir; puede verse un ejemplo de esta situación en las entradas de los permisos estándar. La línea finaliza con la típica terna de permisos.

Investigación de Setacl

setacl tiene los siguientes parámetros: *-m* modifica o crea una entrada nueva, *setfacl -m user:bob:r--* *calendar.cal*

y se borra con *-x*: *setfacl -x user:bob* *calendar.cal*

Con esto sólo elimina la entrada especificada, pero no afecta al grupo al que pertenece Bob. —*set* elimina todas las entradas previas, estableciendo sólo las nuevas: *setfacl --set user:bob:r--* *calendar.cal*

Finalmente, la opción *-b* borra la lista completa. Además, estableciendo el parámetro recursivo *-R* le indica a *setfacl* que aplique la regla al árbol completo. Al mismo tiempo, se pueden establecer varios permisos separados por coma:

```
setfacl -m user:bob:r--,
group:cteam:rw- calendar.cal
```

En vez de los nombres de los usuarios y los grupos, se pueden especificar alternativamente los UIDs y los GIDs; *setfacl* acepta también permisos en formato numérico.

Además pueden aplicarse un par de abreviaturas en *setfacl*. En vez de *user*, es posible usar simplemente *u*. De forma similar, pueden utilizarse las abreviaturas *g*(roup), *m*(ask), *o*(ther) y *d*(efault). Igualmente es posible dejar múltiples secuencias de guiones, siempre que el comando no sea ambiguo:

```
setfacl -m u:bob:r-
calendar.cal
```

¿Quién Tiene Derechos?

A lo largo del proyecto, Alicia tiene que autorizar temporalmente el acceso a su agenda a otros miembros del grupo de Bob. Para ello, simplemente debe añadir una entrada para *bteam*:

```
01 setfacl -m group:bteam:r--
calendar.cal
02 $ getfacl calendar.cal
03 # file: calendar.cal
04 # owner: alice
05 # group: ateam
06 user::rwuser:
07 bob:rw
08 group::r--
09 group:bteam:r--
10 mask::rw-
11 other::r--
```

Cuando se solicita un acceso de lectura para un fichero, Linux simplemente comprueba los permisos uno tras otro.

Primero, el kernel comprueba si el usuario posee una entrada, y si es así, Linux aplica los permisos definidos por la entrada. En este ejemplo, Bob está autorizado a leer y escribir en la agenda. Por el contrario, no existe una entrada personal para Carlos, así que Linux comprueba los permisos de los grupos. Como Carlos es miembro del grupo *bteam*, tendrá acceso de lectura. En el caso de que el kernel sea incapaz de encontrar una entrada personal o de grupo para un individuo, entonces le aplicará los permisos estándar de UNIX.

El Legado

Desafortunadamente, *ls -l* no muestra los permisos extendidos. Por el contrario, tan sólo muestra el signo + para indicar la existencia de permisos extendidos:

```
$ ls -l calendar.cal
-rw-r--r--+ 1 alice ateam
5410 7. Feb 11:21 calendar.cal
```

Como las ACLs aplican los permisos estándar de UNIX, *setfacl* reemplaza el viejo comando *chmod*. Sólo hay que modificar las entradas. Por ejemplo, el comando

```
setfacl -m other::rw-
calendar.cal
```

autoriza el acceso de lectura y escritura para el fichero a todos los usuarios:

```
-rw-r--rw+ 1 alice
ateam 5410 7.
Feb 11:21 calendar.cal
```

Cada ACL extendida contiene una entrada denominada *mask*. La máscara describe los permisos máximos que los usuarios pueden adquirir.

Si la máscara define un permiso más restrictivo que el que posee un usuario en una ACE, la máscara siempre tendrá prioridad. Por ejemplo, Alicia podría haber autorizado a otros miembros del grupo de Bob a acceder a su agenda.

Si ahora quiere revocar temporalmente estos permisos, podría modificarlos para cada miembro del grupo; alternativamente, podría modificar la máscara:

```
setfacl -m mask::r--
calendar.cal
```

No importa qué permisos tengan asignados los usuarios, a partir de ahora, sólo podrán leer la agenda. Por supuesto, esto también se aplica a Bob. Aunque él aún tenga permisos de escritura en la agenda, la máscara tiene prioridad, dejándolo con los mismos permisos que al resto.

Linux se saca de la manga la operación lógica *AND* para calcular los derechos efectivos. Para ser capaz de leer un fichero, el usuario o el grupo necesita el permiso de lectura, permiso de lectura que tiene que existir en la máscara.

Para evitar que el administrador le pierda la pista, *getfacl* muestra los permisos reales efectivos para cada usuario:

Listado 1: Creación de una ACL por Defecto

```
01 $ setfacl -m
    user:bob:rwprojectfolder
02 $ setfacl -d -set
    user:bob:rwprojectfolder
03 $ getfacl projectfolder
04 # file: projectfolder
05 # owner: alice
06 # group: ateam
07 user::rw-
08 user:bob:rwgroup:
09 r-x
10 mask::rw-
11 other::r-x
12 default:user::rw-
13 default:user:bob:rwdefault:
14 mask::rw-
15 default:other::r-x
```

```
$ getfacl calendar.cal
# file: calendar.cal
# owner: alice
# group: ateam
user::rw-
user:bob:rw-      #effective:r-
group::r--
mask::r--
other::r--
```

Aunque Bob posea permiso de escritura, de forma efectiva sólo tiene ahora permiso de lectura para la agenda.

Las máscaras presentan otro inconveniente: *setfacl* modifica de forma autónoma las máscaras cada vez que se modifiquen los permisos del usuario o del grupo. Si no se tiene en cuenta la máscara, como Alicia hizo en este ejemplo, no habrá otra alternativa que comprobar su validez.

Estándar

Mientras Alicia está trabajando en un proyecto crea diversos ficheros de proyectos que Bob tiene que leer. Para cada documento nuevo, ella podría teóricamente modificar los permisos manualmente. Una opción más sencilla consiste en crear los directorios con una ACL por defecto. Los subdirectorios y los ficheros bajo los directorios heredan automáticamente los permisos por defecto. Los directorios tienen tanto una ACL propia como la nueva ACL por defecto del directorio padre. Se puede crear una ACL nueva por defecto ejecutando *setfacl* (véase el Listado 1).

getfacl siempre lista los permisos estándar al final. El formato refleja las entradas clásicas, pero empieza siempre con los permisos por defecto. Si sólo se está interesado en las

Colectivo

Para crear ACLs complejas hay que ejecutar *setfacl* varias veces, una tarea pesada y con la que se pierde tiempo. Afortunadamente, los administradores pueden almacenar las entradas en un fichero de texto con el mismo formato que muestra *getfacl* y ejecutar luego *setfacl* con el parámetro `--set-file="fichero.txt"` para restaurar los permisos. El uso de `-` en vez del nombre del fichero le indica a la herramienta que lea los parámetros desde la entrada estándar. Con esta opción se pueden mover las ACLs desde un fichero a otro:

```
getfacl fichero1 | setfacl -
--set-file=- fichero2
```

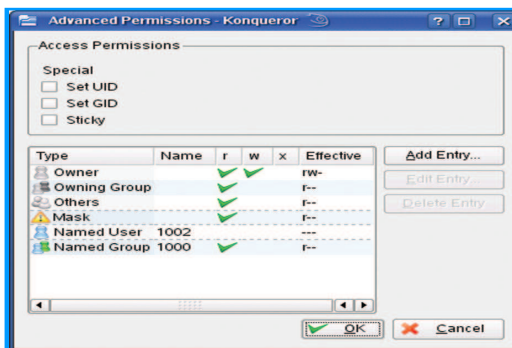


Figura 1: En Konqueror de KDE, el botón de permisos extendidos en el cuadro de diálogo Propiedades de Ficheros muestra la siguiente ventana, donde se pueden modificar convenientemente las entradas de las ACLs.

entradas por defecto, se puede especificar *getfacl -d*; por otro lado, con *getfacl -a* se evita que se muestren estas entradas por defecto.

Como el acceso a los ficheros se gestiona desde el propio kernel, los programas clásicos no presentan ningún problema con los permisos extendidos, lo que no es cierto para las aplicaciones que manipulan los permisos, como Konqueror o Nautilus. La versión 3.5 de Konqueror o posteriores pueden gestionar las ACLs (tal como se puede observar en la Figura 1), así como con la versión 2.16 o posteriores de Nautilus.

Los comandos estándar de UNIX como *cp* o *mv* han sido modificados para que puedan funcionar con las ACLs. Tanto el comando para copiar como el comando para mover ficheros suponen que el sistema de ficheros de destino soporta las ACLs. Si no las soportara, sólo se mantendrían los permisos clásicos, perdiéndose los extendidos.

Los programas que no se hayan modificado para soportar las ACLs simplemente modificarán los permisos estándar. Puede verse un ejemplo al realizar una copia de seguridad. El comando *tar* clásico no mantiene las ACLs. En este caso, habrá que elegir una alternativa, como *star*, que se incluye en muchas distribuciones. El comando:

```
star H=pax -acl -c -f backup.pax project_folder/
```

crea `(-c)` el archivo `(-f)` *backup.pax* y almacena el contenido de la carpeta *project_folder* en él. El programa utiliza el formato de ficheros PAX. El comando:

```
star -acl -xp -acl -f backup.pax
```

descomprime el paquete creado con el comando previo.

Si se prefieren evitar los formatos exóticos, recomiendo el siguiente truco: se le puede indicar a *setfacl* que tome los parámetros desde un fichero de texto. El formato coincide precisamente con la salida de *getfacl*, así que puede utilizarse para almacenar todas las ACLs en un fichero de texto y luego ejecutar el comando *setfacl* para restaurarlos. Para comenzar, se almacenan las ACLs de salida de *getfacl* en un fichero de texto:

```
getfacl -R --skip-
base directory/ >
/backup.acl
```

Con este comando se le indica al comando *getfacl* que entre en el directorio y escriba las ACLs de todos los objetos que encuentre en el fichero *backup.acl*. El parámetro `-R` hace que el proceso sea recursivo. Luego puede almacenarse el fichero ACL dentro del archivo de la copia de seguridad. Para restaurar las ACLs hay que ejecutar:

```
setfacl --restore=backup.acl
```

Conclusiones

Las Listas de Control de Acceso ofrecen una gestión de permisos extremadamente flexible. Al mismo tiempo, las ACLs descargan de trabajo a los administradores pasándole la responsabilidad de la gestión de los permisos de acceso a los propietarios de los ficheros. Gracias a las ACLs por defecto y a las máscaras, los administradores aún mantienen el control.

El intercambio de datos presenta un problema. Como cada sistema operativo posee una variante diferente de las ACLs, las diversas versiones son incompatibles o sólo parcialmente convertibles, así que los permisos a menudo se pierden en el proceso de conversión. En entornos heterogéneos los administradores tendrán que tener cuidado. ■

RECURSOS

- [1] Borradores POSIX ACL: <http://wt.xpilot.org/publications/posix.1e>
- [2] Konqueror: <http://www.konqueror.org/>
- [3] Nautilus: <http://www.gnome.org/projects/>