ACLs

Why/When are they necessary?

The critical shortfall for permissions is that there are only three opportunities to assign permissions. If you are not the owner then you must be a member of the owning group. If you are not the owner and not a member of the owning group then your permissions are determined by the “everybody else” category.

Access Control Lists augment the standard UNIX file permissions by allowing permissions for more than one user and more than one group. With ACLs you can create a list of users and a list of groups **in addition** to the owner and the owning group (i.e. UID and GID) for each file and directory. Each user and each group will then be assigned file permissions to allow or deny read, write and execute privileges.

Perhaps an example that demonstrates the major advantages ACLs have to offer is appropriate. Let’s assume that the files in the "/opt/documents" directory are owned by “root” and the owning group is “sys” and this directory has rwxrwx--- permissions. In our scenario there are 10 people who need varying degrees of access to these files.

1.-The last three dashes (---) indicate that those users who are not members of the group (sys) have no privileges. To give these ten users access to these files we must either let them login as the “root” user or make them members of the “sys” group.Neither of these choices sounds very good from a security point of view because “root” is a superuser and should not be used except by the administrator and his assistants and the group called “sys” is generally used for UNIX system files.

2.-This logic would lead us to consider changing the owning group to “employee”, for example, to avoid using the “sys” group. Assuming we did that we could then make our 10 users, who work for the office, members of the “employee” group. This strategy would allow all 10 users full access to all files in this directory (still assuming that all files have the same permissions). End.

3.-Imagine, however, that now there are 2 of these 10 employees who must have write permissions to files in this directory but the other 8 should only be able to read them and nothing more. If we could add two owners OR another group we could improve the protection. We could make our two users “additional owners”. Or, we could create/add another group and make our two employees members. The remaining 8 users would be members of a “supplemental” group where they could still read the files but not update them. Following is the reasoning:

a) Scenario rwxr----- : Obligues 2 "special" users to log as root.

b) Scenario rwxrwxr-- : Gives all other users the read permission.

c) Solution 1 with ACLs:

Owner=root	rwx
User=mary	rwx
User=betty	rwx
Group=employee	r--
Other	---

Mary and Betty are the two employees who need to be able to update the files. All of the other people need to be able to read certain files but not update them (these additional users are members of the “employee” group). Any other user cannot access the files. Notice that we have three users (root, mary, and betty) with their own set of permissions.

d) Solution 2 with ACLs:

Owner=root	rwx
Group=bosses	rwx

Group=employee	r--
Other	---

If we move Mary and Betty into the “bosses” group and the other 8 into the “employee” group we can potentially reduce the amount of ACL maintenance we will be faced with in the future. For example, let’s assume there are 800 files in the directory. Each file has an ACL that must be maintained. If Mary should quit we will have to remove her name from all 800 files and later replace her ACL when someone else takes over her duties. Additionally, Alice may be assigned Betty's work while Betty is on sick leave so we would need then to add her to the ACL owner’s list.

In practice

ACLs currently are supported out of the box in XFS, BTRFS and current Ext4 (via /etc/mke2fs.conf) filesystems (in fact, "acl" package is a dependency os systemd). Anyway, we could be sure of this simply executing *tune2fs -l /dev/sda1 | grep "Default mount options:"* and seeing "user_xattr acl" values on screen. More about that in *man acl*

NOTA: If necessary, you could set the default mount options of a filesystem using the tune2fs -o option (for example: *tune2fs -o acl /dev/sdXY*). Using the default mount options instead of an entry in /etc/fstab is very useful for external drives because their partition will be mounted with -in this case- "acl" option also on other Linux machines without the need to edit /etc/fstab on every machine.

ACLs are a type of "extended attribute" (specifically, a "system" type), so in theory they could be managed by standard *getfattr* and *setfattr* commands but there are other utilities much more targeted and convenient to do this job: *getfacl* and *setfacl*.

NOTA: There are two types of ACLs: "normal" ACLs, which are applied to a file or directory, and "default" (optional) ACLs, which can only be applied to a directory. If files inside a directory where a default ACL has been set do not have a ACL of their own, they inherit the default ACL of their parent directory.

To know which ACL is assigned to a specific file, we can do the following:

getfacl /ruta/fitxer (or *getfattr -d -m system /ruta/fitxer*)

To change the ACL of a specific file or folder, we can use the -m ("modify") parameter of *setfacl*:

setfacl -m u:userName:rw /ruta/fitxerOcarpeta <-- Concedeix permisos rw a "userName"
El permís x el deixa com estava.
setfacl -m g:groupName:r-x /ruta/fitxerOcarpeta <-- Concedeix permisos r-x a "groupName"

setfacl -m u:userName:r--,g:groupName:r-- /ruta/fitxerOcarpeta <--Varis permisos de cop

If a directory is managed, *-R* parameter (Recursive down into files and directories) can be used

What about using the *setfacl* command to change normal User, Group, and Other permissions? No problem! This can be used instead of chmod:

setfacl -m u::rwx,g::rwx,o::rwx /ruta/fitxerOcarpeta

To set a default ACL to a directory (which its contents will inherit unless overwritten otherwise), you could add *-d* parameter (optionally with -R to do it recursively) and specify a directory instead of a file name:

setfacl -R -d -m u:userName:rwx /ruta/carpeta

To remove a specific ACL, replace -m in the commands above with -x. For example:

setfacl -x g:grupName /ruta/fitxerOcarpeta

Alternatively, you can also use the -b option to remove ALL ACLs in one step:

setfacl -b /ruta/fitxerOcarpeta

If you see in *a ls* output a "+" sign after permissions column, it reveals that in this particular file there is something different than other files because of adding extended attributes. This trick could be a simple manner to detect wich files have an ACL assigned.

Finally...what's the "mask" entry shown by *getfacl*? This is the "effective rights mask". This entry limits the effective rights granted to all ACL groups and ACL users. Effective masking is used to restrict action(s) for all users and groups, that are defined in ACL. It means, for example, that you can prevent all users from writing to a file by setting the effective mask r-x.The traditional Unix User, Group, and Other entries are not affected. If the mask is more restrictive than the ACL permissions that you grant, then the mask takes precedence. To change it, we must do the following (in this example we change the mask of a file to "r--"...that would prevent any ACL-defined user or group to write on that file regardless of their own permissions):

setfacl -m mask::r-- /ruta/fitxer

EXERCICIS:

1.-a) Sempre com a "root", crea dos usuaris anomenats "alice" i "bob" (i assigna'ls una contrasenya!).

> NOTA: Recorda que per crear un usuari (anomenat "pepe", per exemple) i assignar-li una contrasenya només cal executar (com a root) les comandes: *useradd -m pepe && passwd pepe*

aII) Crea una carpeta anomenada "/opt/documents" que tingui com a propietari l'usuari i grup "root"

> NOTA: Recorda que per fer que una carpeta o fitxer tingui un determinat propietari i grup (per exemple, l'usuari "pepe" i grup "pepe") només cal executar (com a root) la comanda: *chown pepe:pepe ruta/carpeta*

aIII) Fes que la carpeta "/opt/documents" tingui permisos 700

> NOTA: Recorda que per fer que una carpeta o fitxer tingui uns determinats permisos (per exemple, 644) noméscal executar (com a root) la comanda: *chmod 644 /ruta/carpeta*

aIV) Obre un terminal i executa *su -l* per ser l'usuari "root", obre un altre terminal i executa *su -l alice* per ser "alice" i obre un altre terminal i executa *su -l bob* per ser "bob".

b) Fes una ACL que permeti que "alice" pugui llegir, accedir i modificar el contingut de la carpeta /opt/documents. Prova-ho. Si "alice" crea un nou fitxer a dins d'aquesta carpeta, ¿podrà modificar-lo?

c) Fes una ACL que permeti que "bob" només pugui llegir i accedir a la carpeta /opt/documents. Prova-ho.

d) Esborra totes les ACL de /opt/documents per tornar a començar. Comprova llavors que ara ni "alice" ni "bob" poden fer res (llegir, accedir, modificar contingut) en aquesta carpeta. Ara crea un grup d'usuaris anomenat "amics" i fica-hi a dins tant "alice" com "bob". Finalment, fes una altra ACL que permeti que els membres del grup "amics" puguin llegir i accedir a la carpeta /opt/documents. Prova-ho tant amb "alice" com amb "bob".

e) Canvia la màscara ACL de /opt/documents per a què no tingui cap permís. ¿Què poden fer ara "alice" i "bob" en aquesta carpeta? Finalment, torna a establir la màscara a "rwx"

The basic file commands (cp , mv , ls , tar, ...) support ACLs, as do Samba and Nautilus. When archiving a file or file system with tar, however, you must use the *--acls* option to preserve ACLs. Similarly, when using cp to copy files with ACLs you must include the *--preserve=mode* option to ensure that ACLs are copied across too (in addition, the -a option is equivalent to *-d R --preserve= all*

2.-a) Crea (com usuari "normal") un fitxer buit anomenat "formatejar.sh" dins de la teva carpeta personal Assigna-li ara una ACL que permeti al grup "disk" executar aquest fitxer i comprova (amb *getfacl*) que l'hagis assignat correctament

b) Copia ara el fitxer "formatejar.sh" al teu escriptori amb la comanda *cp* sense cap paràmetre especial i comprova si manté la seva ACL o no. ¿I si ara fas el mateix però indicant el paràmetre --*preserve=mode*?

3.-a) Dedueix i explica què fa aquest conjunt de comandes, pas a pas (el pots provar, si vols):

getfacl -R ~/Escriptori > acls.txt
chmod -R ugo-rw ~/Escriptori
cd ~
setfacl --restore=acls.txt

b) Si només es vol "traspassar" una ACL ja definida per escrit a un determinat fitxer (sense fer cap recursió), es pot fer servir en comptes del paràmetre –restore de setfacl el paràmetre -M. Sabent això, explica què fa aquest conjunt de comandes, pas a pas (el pots provar, si vols):

echo "u:usuari:rwx" > acl.txt
touch prova.txt
setfacl -M acl.txt prova.txt

c) Què fa aquesta canonada de comandes? (tingues en compte els diferents paràmetres que apareixen)

getfacl unfitxer.txt | setfacl -b -n -M – prova.txt