tutorial acl linux

Una buona spiegazione su cosa sono le acl la possiamo trovare
negli ottimi appunti di informatica libera, a questo link:

http://wwwcdf.pd.infn.it/AppuntiLinux/acl_posix_con_i_sistemi_gnu_linux.htm

Saranno qui trattati solo alcuni esempi pratici, senza pretesa di
completezza.
Le prove sono state effettuate su slackware 14.2 , su diversi
file system, ext4, reiserfs, xfs, jfs .
La prima raccomandazione che apprendiamo fa riferimento al
file /etc/fstab, ove occorre specificare la direttiva " acl " nella
riga che specifica il mount point. In realta' alcune volte questo non
serve, poiche' i file system xfs e jfs hanno applicato i comandi
getfacl e setfacl anche senza specificare " acl " all'atto del mount.
Ecco un estratto del file /etc/fstab del computer slackware 14.2
utilizzato per le prove

```
/dev/sda2 on / type ext4 (rw,acl)
/dev/sdb1 on /mnt/disk2-a type reiserfs (rw,acl)
/dev/sdb2 on /mnt/disk2-b type btrfs (rw,acl)
/dev/sdb3 on /mnt/disk2-c type xfs (rw)
/dev/sdb4 on /mnt/disk2-d type jfs (rw)
```

Il filesystem reiserfs non ne ha voluto sapere di applicare le ACL ,
e non mi sono impegnato a capire il perche'. Di fatto reiserfs non e' piu'
mantenuto. Anzi, credo che solo slackware proponga ancora reiserfs
tra le scelte possibili. Per placara la curiosita', ho iniziato
l'installazione di mint 19, ove volevo appunto utilizzare reiserfs ,
in modo da avere un paragone con slackware 14.2, ma all'atto dell'install
il filesystem reiserfs non era presente come scelta.
Per iniziare l'analisi delle acl, dobbiamo prima sapere quali comandi sono
utilizzabili, e per ogni comando dobbiamo sapere almeno le opzioni princiali
e la loro azione.   Per ottenere questa informazione, lanciamo il comando

```
man -k acl
```

Ci viene restituito un lungo output, ove nella parte finale troviamo cio' che
ci interessa. Concentriamo la nostra attenzione sui comandi

```
getfacl

setfacl
```

Come possiamo intuire , il comando setfacl inposta una determionata ACL , mentre
il comando getfacl interroga un file oppure una direcotry e ci restituisce
le ACL applicate . Non analizzeremo altri comandi riferiti alla ACL .
Le nostre prove iniziano facendo login sul sistema linux utilizzando un account
non amministrativo, per l'esattezza l'utente " claudio ". Successivamente ci
rechiamo in /mnt/disk2-b , formattato con btrfs .
Creiamo un file vuoto

```
claudio@slack-acl:/mnt/disk2-b$ touch file
claudio@slack-acl:/mnt/disk2-b$ ls -l
-rw-r--r-- 1 claudio users 0 Jul  7 17:55 file
```

ora concediamo -utilizzando le ACL- all'utente " ut1" i permessi di lettura,
scrittura ed esecuzione.

```
claudio@slack-acl:/mnt/disk2-b$ setfacl -m user:ut1:rwx file
```

Verifichiamo l'esito della direttiva prima col solito ls -l e poi tramite
il comando getfacl

```
claudio@slack-acl:/mnt/disk2-b$ ls -lh
total 0
-rw-rwxr--+ 1 claudio users 0 Jul  7 17:55 file*

claudio@slack-acl:/mnt/disk2-b$ getfacl file
# file: file
# owner: claudio
# group: users
user::rw-
user:ut1:rwx
group::r--
mask::rwx
other::r--
```

L'output del comando " ls " ci mostra un " + " alla fine della riga,
ad indicare che le regole ACL sono applicate. L'output del comando getfacl
e' piu descrittivo e ci mostra chiaramente che " ut1 " puo' leggere, scrivere
ed eseguire " file " , esattamente come volevamo. Ora concediamo anche a " ut2 "
gli stessi privilegi, e poi verifichiamo tramite il comando getfacl

```
claudio@slack-acl:/mnt/disk2-b$ setfacl -m user:ut2:rwx file

claudio@slack-acl:/mnt/disk2-b$ getfacl -t file
# file: file
USER   claudio   rw-
user   ut1       rwx
user   ut2       rwx
GROUP  users     r--
mask             rwx
other            r--
```

Abbiamo invocato il comando getfacl indicando espressamente l'opzione " -t " .
Come possiamo vedere, anche " ut2 " ha i medesimi privilegi di " ut1 " ,
esattamente come volevamo. Ora ci rechiamo nella posizione /mnt/disk2-c
formattata con filesystem xfs. Le nostre prove cominciano a spaziare
in piu' contesti. Anche qui come utente " claudio " creiamo il file
" file " , ed anche qui assegniamo a " ut1 " i permessi rwx

```
claudio@slack-acl:/mnt/disk2-c$ touch file
claudio@slack-acl:/mnt/disk2-c$ setfacl -m user:ut1:rwx file
```

Ora come utente root ed utilizzando le ACL togliamo all'utente " claudio "
i permessi di lettura, scrittura ed esecuzione .

```
root@slack-acl:/mnt/disk2-c# setfacl -m user::--- file
```

Abbiamo utilizzato una sintassi differente rispetto all'esempio precedente.
Controlliamo l'applicazione della ACL

```
root@slack-acl:/mnt/disk2-c# getfacl  -e file
# file: file
# owner: claudio
# group: users
user::---
user:ut1:rwx                    #effective:rwx
group::r--                      #effective:r--
mask::rwx
other::r--
```

Anche qui abbiamo utilizzato una sintassi differente, ora e' apparsa la dicitura
" effetcive " . Per la spiegazione di tale dicitura, fate riferimento al link

di cui sopra :
http://wwwcdf.pd.infn.it/AppuntiLinux/acl_posix_con_i_sistemi_gnu_linux.htm
Ora l'utente proprietario del file non ha alcuna autorita' sul file stesso,
ma puo' cancellarlo, previa conferma

```
claudio@slack-acl:/mnt/disk2-c$ rm file
rm: remove write-protected regular empty file 'file'? y
claudio@slack-acl:/mnt/disk2-c$ ls -l

claudio@slack-acl:/mnt/disk2-c$
```

Ora ci rechiamo in /etc/ , e vediamo come concedere la scrittura al file
fstab agli utenti. Ricordiamo dhe il file /etc/fstab e' editabile
solo da root. Nella root directory " / " viene utilizzato il file system
ext4 , estendiamo cosi' le nostre analisi su di un ulteriore filesystem.

```
root@slack-acl:/etc# ls -lah |  grep fstab
-rw-r--r--  1 root root    748 Jul  5 20:42 fstab
```

Concediamo all'utente " ut2 " i permessi rwx sul file /et/fstab

```
root@slack-acl:/etc# setfacl -m user:ut2:wrx fstab
root@slack-acl:/etc# getfacl -e fstab
# file: fstab
# owner: root
# group: root
user::rw-
user:ut2:rwx                    #effective:rwx
group::r--                      #effective:r--
mask::rwx
other::r--
```

La verifica ci conferma l'applicazione della ACL . Ora concediamo a " ut3 "
i permessi rwx sul file /etc/fstab

```
root@slack-acl:/etc# setfacl -m user:ut3:wrx fstab
root@slack-acl:/etc# getfacl -e fstab
# file: fstab
# owner: root
# group: root
user::rw-
user:ut2:rwx                    #effective:rwx
user:ut3:rwx                    #effective:rwx
group::r--                      #effective:r--
mask::rwx
other::r--
```

Anche ora la verifica conferma l'esecuzione del comando. Annulliamo ora tutte
le ACL riferite al file /etc/fstab , riportandolo alle condizioni originarie

```
root@slack-acl:/etc# setfacl -b fstab
root@slack-acl:/etc# getfacl -e fstab
# file: fstab
# owner: root
# group: root
user::rw-
group::r--
other::r--
```

Riconcediamo a " ut2 " e " ut3 " di poter operare sul flle /etc/fstab

```
root@slack-acl:/etc# setfacl -m user:ut2:wrx fstab
root@slack-acl:/etc# setfacl -m user:ut3:wrx fstab
```

Ora vediamo se il file fstab a cui abbiamo assegnato delle acl, mantiene le sue proprieta' durante una copia

```
root@slack-acl:/mnt# cp /etc/fstab /mnt/disk2-d/
```

il file system di destinazione e' jfs. Vediamo se i permessi ACL sono stati mantenuti

```
root@slack-acl:/mnt/disk2-d# ls -l
-rw-r-xr-- 1 root root 748 Jul 10 14:32 fstab*

root@slack-acl:/mnt/disk2-d# getfacl -t fstab
# file: fstab
USER   root      rw-
GROUP  root      r-x
other            r--
```

I permessi ACL non vengono mantenuti durante la copia. Per ottenere tale risultato dobbiamo invocare il comando cp specificando l'opzione -a

```
root@slack-acl:/mnt/disk2-d# cd /mnt/disk2-d/

root@slack-acl:/mnt/disk2-d# cp -a /etc/fstab .

root@slack-acl:/mnt/disk2-d# getfacl fstab
# file: fstab
# owner: root
# group: root
user::rw-
user:ut2:rwx
user:ut3:rwx
group::r--
mask::rwx
other::r--
```

Ora vediamo come modificare i permessi ACL di un file, prendendo come base di partenza un altro file, ove le ACL sono gia' assegnate. Detto brutalmente, faremo "copia ed incolla" delle ACL da un file ad un altro file. Faremo la prova come utente " users " e non " root " .

```
claudio@slack-acl:/mnt/disk2-d$ touch provacopia

claudio@slack-acl:/mnt/disk2-d$ getfacl --access fstab | setfacl --set-file=-
provacopia

claudio@slack-acl:/mnt/disk2-d$ ls -lh

-rw-rwxr--+ 1 root    root  748 Jul  5 20:42 fstab*
-rw-rwxr--+ 1 claudio users   0 Jul 10 14:47 provacopia*

claudio@slack-acl:/mnt/disk2-d$ getfacl provacopia

# file: provacopia
# owner: claudio
# group: users
user::rw-
user:ut2:rwx
user:ut3:rwx
group::r--
mask::rwx
other::r--
```

Come Volevasi dimostrare, i permessi ACL sono stati assegnati al file " provacopia "

a partire dal file fstab. Il tutto come utente declassato. Continuiamo le nostre prove, ci troviamo sempre in /mnt/disk2-d e siamo sempre utenti non privilegiati.
Creaimo una directory

```
claudio@slack-acl:/mnt/disk2-d$ mkdir babbano
```

Vogliamo fare in modo che l'utente " ut1 " non possa accede alla directory

```
claudio@slack-acl:/mnt/disk2-d$ setfacl -m user:ut1:--- babbano

claudio@slack-acl:/mnt/disk2-d$ getfacl -e babbano
# file: babbano
# owner: claudio
# group: users
user::rwx
user:ut1:---                    #effective:---
group::r-x                      #effective:r-x
mask::r-x
other::r-x
```

Facciamo login come ut1 e proviamo l'ingresso nella directory " babbano "

```
ut1@slack-acl:/mnt/disk2-d$ cd babbano/
-bash: cd: babbano/: Permission denied
```

Esattamente quello che volevamo. Vediamo se riusciamo ad impedire l'ingresso anche a root

```
claudio@slack-acl:/mnt/disk2-d$ setfacl -m user:root:--- babbano
claudio@slack-acl:/mnt/disk2-d$ getfacl -e babbano
# file: babbano
# owner: claudio
# group: users
user::rwx
user:root:---                   #effective:---
user:ut1:---                    #effective:---
group::r-x                      #effective:r-x
mask::r-x
other::r-x
```

Rientriamo nel sistema coi privilegi di root e controlliamo

```
root@slack-acl:/mnt/disk2-d# cd babbano/
root@slack-acl:/mnt/disk2-d/babbano# pwd
/mnt/disk2-d/babbano
root@slack-acl:/mnt/disk2-d/babbano#
```

Il trucco non funziona. non si puo' impedire accesso a root. di fatto root e' il "cavaliere nero" nei sistemi unix/linux . nulla gli e' impedito. Nei sistemi windows l'utente Administrator non ha lo stesso potere dell'utente root su linux. Ora impostiamo una ACL di default per una dorectory. Cosi' facendo, i file generati all'interno di detta directory erediteranno determinate ACL .

vogliamo ora creare una directory in /mnt/sdc2-c , quindi con filesystem xfs, e fare in modo che qualunque utente generi un file in questa directory, detto file assuma i permessi rwx. Per il gruppo " users " ovviamente.

```
root@slack-acl:/mnt/disk2-c# mkdir directory
root@slack-acl:/mnt/disk2-c# setfacl -m g:users:rwx directory/
root@slack-acl:/mnt/disk2-c# setfacl -d -m g:users:rwx directory/

root@slack-acl:/mnt/disk2-c# getfacl -t directory/
```

```
# file: directory/
USER   root      rwx  rwx
GROUP  root      r-x  r-x
group  users     rwx  rwx
mask             rwx  rwx
other            r-x  r-x
```

ora facciamo la verifica. Generiamo un file all'interno di detta directory da parte di due utenti, uno appartenente al gruppo users, l'altro non appartenente al gruppo users

```
claudio@slack-acl:/mnt/disk2-c/directory$ touch claudio
claudio@slack-acl:/mnt/disk2-c/directory$ getfacl -t claudio
# file: claudio
USER   claudio   rw-
GROUP  users     r-X
group  users     rwX
mask             rw-
other            r--
```

l'utente " esterno " puo' accedere alla directory, ma non puo' creare nessun file al suo interno, poiche' non appartenente al gruppo " users " . Concediamo ora i privilegi necessari

```
root@slack-acl:/mnt/disk2-c# setfacl -m g:esterno:rwx directory/
root@slack-acl:/mnt/disk2-c# setfacl -d -m g:esterno:rwx directory/
root@slack-acl:/mnt/disk2-c# getfacl -t directory/
# file: directory/
USER   root      rwx  rwx
GROUP  root      r-x  r-x
group  users     rwx  rwx
group  esterno        rwx
mask             rwx  rwx
other            r-x  r-x
```

ed ora come utente " esterno " ripetiamo il tutto

```
esterno@slack-acl:/mnt/disk2-c$ pwd
/mnt/disk2-c
esterno@slack-acl:/mnt/disk2-c$ cd directory/
esterno@slack-acl:/mnt/disk2-c/directory$ touch esterno
esterno@slack-acl:/mnt/disk2-c/directory$ getfacl -t esterno
# file: esterno
USER   esterno   rw-
user   esterno   rwX
GROUP  esterno   r-X
group  users     rwX
group  esterno   rwX
mask             rw-
other            r--
```

torniamo all'identita' di root e posizioniamoci nella directory " directory "

```
root@slack-acl:/mnt/disk2-c# cd directory/
root@slack-acl:/mnt/disk2-c/directory# pwd
/mnt/disk2-c/directory
```

creaimo un file e leggiamo le acl di questo file

```
root@slack-acl:/mnt/disk2-c/directory# touch root
root@slack-acl:/mnt/disk2-c/directory# getfacl -t root
# file: root
USER   root      rw-
```

```
user   esterno   rwX
GROUP  root      r-X
group  users     rwX
group  esterno   rwX
mask             rw-
other            r--
```

il file e' stato creto, le ACL di default sono state applicate, ed anche gli utenti " claudio " ed " esterno " possono editare il file. Ora qualche esempio sulle direttive " mask:: " e " other:: " . Ci posizionamo in / , creiamo una directory " zzzz " e diamo un chmod 777 alla direcotry. Siamo quindi su filesystem reiserfs.

```
root@slack-acl:/# cd /
root@slack-acl:/# mkdir zzzz
root@slack-acl:/# chmod 777 zzz
root@slack-acl:/# chmod 777 zzzz
root@slack-acl:/# cd zzzz/
root@slack-acl:/zzzz#
```

ora creiamo una direcoty " prova " e applichiamo una ACL " other "

```
root@slack-acl:/zzzz# mkdir prova
root@slack-acl:/zzzz# setfacl -m other::--- prova
root@slack-acl:/zzzz# ls -l
total 4
drwxr-x--- 2 root root 4096 Jul 15 07:21 prova/
root@slack-acl:/zzzz# getfacl -e prova/
# file: prova/
# owner: root
# group: root
user::rwx
group::r-x
other::---
```

abbiamo impedito l'ingresso, la scrittura e l'esecuzione a tutti gli user che non rientrino espressamente nel gruppo " root " ed a root stesso ovviamente

```
claudio@slack-acl:~$ cd /zzzz/
claudio@slack-acl:/zzzz$ cd prova/
-bash: cd: prova/: Permission denied
claudio@slack-acl:/zzzz$
```

ora concediamo tutti i permessi possibili a " other " , ma tramite il comando chmod

```
root@slack-acl:/zzzz# chmod o+wrx prova/
root@slack-acl:/zzzz# ls -ld prova/
drwxr-xrwx 2 root root 4096 Jul 15 07:21 prova//
root@slack-acl:/zzzz#
```

verifichiamo

```
claudio@slack-acl:/zzzz$ cd prova/
claudio@slack-acl:/zzzz/prova$ pwd
/zzzz/prova
claudio@slack-acl:/zzzz/prova$
```

in questo caso specifico i due comandi " setfacl " e " chmod " si equivalgono. Per quanto riguarda la direttiva " mask:: " , diventa troppo lungo l'esame. Rimando quindi ai link sottostanti, inoltre anche la pagina man 5 acl puo' fugare molti dubbi.

https://unix.stackexchange.com/questions/147499/what-relationships-tie-acl-mask-and-standard-group-permission-on-a-file

http://www-uxsup.csx.cam.ac.uk/pub/doc/suse/sles9/adminguide-sles9/ch27s03.html

https://unix.stackexchange.com/questions/475698/what-is-the-exact-purpose-of-mask-in-file-system-acl

https://tylersguides.com/guides/linux-acl-permissions-tutorial/#:~:text=ACL%20masks%20are%20a%20way%20to%20enforce%20a,the%20example%2C%20the%20mask%20is%20set%20to%20r-x.