



Laboration – Speciella behörigheter och länkar

OBS det kan förekomma små skillnader vad gäller sökvägar samt befintliga kommandon mellan olika distributioner. T.ex. mellan CentOS och Ubuntu,

Material: För att genomföra laborationen behöver man ha tillgång till en dator (vm) med Linux installerat. Windows Subsystem for Linux går att använda också. För vissa övningar kan lokal administratörsbehörighet behövas. I labben utgår det från att man är inloggad med en användare som heter **sysadmin**

Mål: I denna laboration kommer du att utföra följande uppgifter:

- Visa filer med särskilda behörigheter
- Skapa hårda och mjuka länkar

Speciella behörigheter

I denna uppgift kommer du att hitta och förstå syftet med särskilda behörigheter utöver läsa, skriva och exekvera.

Genom att använda **-d**-alternativet för **ls**-kommandot listas kataloginformation; Tillsammans med **-l**-alternativet visas äganderätt och behörigheter för katalogfilerna. Lista detaljerna för **/tmp**- och **/var/tmp**-katalogerna:

```
ls -ld /tmp
ls -ld /var/tmp
```

Utdata visar att behörigheterna på dessa kataloger är desamma:

```
sysadmin@localhost:~$ ls -ld /tmp
drwxrwxrwt 1 root root 4096 Feb 24 20:19 /tmp
sysadmin@localhost:~$ ls -ld /var/tmp
drwxrwxrwt 2 root root 4096 Jan 18 2021 /var/tmp
sysadmin@localhost:~$
```

/tmp- och **/var/tmp**-katalogerna är läs-, skriv- och körbara för alla. Förutom användarnas hemkataloger är dessa två "tillfälliga" kataloger de platser i filsystemet där vanliga användare kan skapa nya filer eller kataloger.

Detta utgör ett problem: om alla användare kan skapa nya filer kan de också radera befintliga filer. Detta beror på att skrivbehörigheten i en katalog ger användare möjlighet att lägga till och ta bort filer i en katalog.

t:et i kör-kolumnen för de andra behörigheterna indikerar att denna katalog har **sticky bit**-behörighetsuppsättningen. Denna särskilda behörighet innebär att även om alla kan lägga till filer i dessa kataloger, är det bara användaren som skapar en fil som kan ta bort den.



Root-användaren påverkas inte av denna behörighet eftersom det kontot kan radera alla filer i katalogen, oavsett ägarskap.

Visa behörigheterna för `/etc/shadow`-filen:

```
ls -l /etc/shadow
```

Resultatet visar att root-användaren har läs- och skrivbehörigheter, medlemmar i shadow gruppen har läsbehörighet, och andra har ingen behörighet på denna fil:

```
sysadmin@localhost:~$ ls -l /etc/shadow
-rw-r----- 1 root shadow 968 Feb  8 2021 /etc/shadow
sysadmin@localhost:~$
```

Liksom de andra filerna i katalogen `/etc` innehåller filen `/etc/shadow` värdspecifik konfigurationsinformation. Specifikt innehåller filen `/etc/shadow` de krypterade lösenorden till alla lokala användarkonton och information om *lösenordsåldrande* (hur länge ett lösenord är giltigt). Eftersom detta är mycket känslig information är åtkomsten till denna fil begränsad till root-användaren och kommandon som exekveras som root, samt medlemmar i shadow-gruppen.

När en användare uppdaterar sitt lösenord med `passwd`-kommandot utförs `passwd`-kommandot med en särskild behörighet kallad *setuid*. Setuid-behörigheten gör att en fil körs som den användare som äger filen, istället för den användare som faktiskt kör kommandot.

Om du kommer från en bakgrund från Microsoft Windows kan du tänka på setuid som funktionen *Kör som administratör* som finns i Windows.

Visa behörigheterna för filen `/usr/bin/passwd`:

```
ls -l /usr/bin/passwd
```

Listningen i denna fil visar:

```
sysadmin@localhost:~$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 59640 Mar 22 2019 /usr/bin/passwd
sysadmin@localhost:~$
```

Lägg märke till `s` i användarens *kolumn för exekveringsbehörighet*. Detta indikerar att denna fil har setuidens behörighetsuppsättning, så den exekveras som användaren som äger den (root) istället för användaren som kör kommandot.

Således kan `passwd`-kommandot uppdatera `/etc/shadow`-filen, eftersom det körs som root-användaren (kom ihåg att root-användaren kan redigera vilken fil som helst, oavsett behörigheter på filen).

Visa behörigheterna på kommandot `/usr/bin/wall`:

```
ls -l /usr/bin/wall
```

Utdata visar nu `s` i exekveringskolumnen för gruppen:

```
sysadmin@localhost:~$ ls -l /usr/bin/wall
-rwxr-sr-x 1 root tty 30800 Sep 16 2020 /usr/bin/wall
```



S:et i kolumnen group *execute* indikerar att denna fil har *setgid-behörighetsuppsättningen*, så den exekveras som gruppen som äger den (**tty**) istället för som användarens grupp som kör kommandot. Därför kan väggkommandot skriva till alla terminaler (tty) när det körs som tty-gruppen.

Notera

Detta liknar mycket *setuid-behörigheten*, men istället för att köra kommandot som användarägare av programmet, körs kommandot som gruppägare av programmet.

Tänk på detta

Hittills har du sett tre typer av specialbehörigheter: *sticky bit* på en katalog, *setuid* på en körbar fil och *setgid* på en körbar fil. Som du har sett finns dessa tre typer i ett typiskt system.

En ytterligare särskild behörighetstyp kan användas; *setgid-behörigheten* kan också tillämpas på en katalog:

```
drwxrwsrwx. 2 root demo 4096 Oct 30 23:20 /tmp/data
```

Om du ser **S** i *exekveringskolumnen* för gruppen som äger katalogen, har katalogen *setgid-behörigheten*. När en katalog har *setgid-behörigheten*, kommer varje ny fil eller katalog som skapas i den katalogen automatiskt att ägas av gruppen som äger katalogen, inte av gruppen för användaren som skapade filen.

Hårda och mjuka länkar

I denna uppgift kommer du att skapa och använda hårda och mjuka länkar.

Byt till din hemkatalog:

```
cd
```

```
sysadmin@localhost:~$ cd
```

Skapa en fil med namnet **source** som innehåller texten **"data"** genom att använda omdirigering:

```
echo "data" > source
```

```
sysadmin@localhost:~$ echo "data" > source
```

Genom att använda **-i**-alternativet med **ls**-kommandot skriver du ut filens indexnummer. Se detaljerna och inodeinformationen i källfilen:

```
ls -li source
```

Den markerade utdatan visar att filen har ett *inode-antal* på **6689431** (detta antal varierar från ett system till ett annat) och ett *länkant* på **1**:

```
sysadmin@localhost:~$ ls -li source
6689431 -rw-rw-r-- 1 sysadmin sysadmin 5 Feb 24 20:32 source
```

Linux-operativsystemet använder inoder för att hålla reda på informationen om en fil. En katalogpost associerar en inode med ett filnamn.

Att skapa en hård länk skapar en annan katalogpost kopplad till en befintlig inode och ökar antalet länkar.



Hårda länkar låter dig använda flera namn för att referera till samma fil. Om något av dessa namn tas bort kan de andra namnen fortfarande användas för att referera till filen.

Faktum är att dessa andra namn till och med kan användas för att skapa ytterligare länkar. Alla dessa namn anses vara ekvivalenta eftersom de alla syftar på en befintlig inode.

Viktigt

Du kan inte skapa hårda länkar till kataloger. Dessutom måste en hård länk till en fil finnas inom samma filsystem (partition) som filen den länkar till.

För att skapa hårda länkar används `ln`-kommandot med två argument. Det första argumentet är ett befintligt filnamn att länka till, kallat ett *mål*, och det andra argumentet är det nya filnamnet som kommer att länka till målet. Använd `ln`-kommandot för att skapa en hård länk. Se detaljerna och inode-informationen om källkoden och den nya hårdlänksfilen:

```
ln source hardlink
```

```
ls -li source hardlink
```

Observera att `hardlink`-filen och `source` filen delar samma inode:

```
sysadmin@localhost:~$ ln source hardlink
sysadmin@localhost:~$ ls -li source hardlink
6689431 -rw-rw-r-- 2 sysadmin sysadmin 5 Feb 24 20:32 hardlink
6689431 -rw-rw-r-- 2 sysadmin sysadmin 5 Feb 24 20:32 source
```

Använd `ln`-kommandot för att skapa en annan hård länk till `source` filen. Se detaljer och inode-information om källkoden och nya hårdlänkfiler:

```
ln hardlink hardlinktwo
```

```
ls -li hardlink hardlinktwo source
```

Observera att utdata fortsätter att visa att hårda länkar delar samma inode och att länkantalet ökar på alla länkar när en länk läggs till:

```
sysadmin@localhost:~$ ln source hardlinktwo
sysadmin@localhost:~$ ls -li hardlink hardlinktwo source
6689431 -rw-rw-r-- 3 sysadmin sysadmin 5 Feb 24 20:32 hardlink
6689431 -rw-rw-r-- 3 sysadmin sysadmin 5 Feb 24 20:32 hardlinktwo
6689431 -rw-rw-r-- 3 sysadmin sysadmin 5 Feb 24 20:32 source
```

`rm`-kommandot används för att ta bort filer. Ta bort den sista länken som skapades och lista `source` och `hardlink`-filerna:

```
rm hardlinktwo
```

```
ls -li source hardlink
```

Observera att länkantalet minskar när en av de hårdlänkade filerna tas bort:



```
sysadmin@localhost:~$ rm hardlinktwo
sysadmin@localhost:~$ ls -li source hardlink
6689431 -rw-rw-r-- 2 sysadmin sysadmin 5 Feb 24 20:32 hardlink
6689431 -rw-rw-r-- 2 sysadmin sysadmin 5 Feb 24 20:32 source
```

Ta bort `hardlink`-filen och lista källfilens detaljer:

```
rm hardlink
ls -li source
```

```
sysadmin@localhost:~$ rm hardlink
sysadmin@localhost:~$ ls -li source
6689431 -rw-rw-r-- 1 sysadmin sysadmin 5 Feb 24 20:32 source
```

En annan typ av länk som kan skapas kallas symbolisk *länk* eller *mjuk länk*. Symboliska länkar ökar inte antalet länkar för filer de är kopplade till.

Symboliska länkfiler har sin egen inode och filtyp. Istället för att länka och dela en inode, länkar de till filnamnet. Till skillnad från hårda länkar kan mjuka länkar länkas till kataloger och kan korsa enheter och partitioner till sina mål.

`-s`-alternativet för `ln`-kommandot skapar en symbolisk länk istället för en hård länk. Skapa en symbolisk länk till `source` filen och visa detaljerna för båda filerna:

```
ln -s source softlink
ls -li source softlink
```

Den markerade utdatan visar att `softlink` och `source` filen har olika inoder. Observera att länkfiltypen (indikerad av `l`:et före behörigheterna) skapas när en symbolisk länk skapas. Behörigheterna för länken är irrelevanta, eftersom det är behörigheterna för målfilen som avgör åtkomst:

```
sysadmin@localhost:~$ ln -s source softlink
6685169 lrwxrwxrwx 1 sysadmin sysadmin 6 Feb 24 20:45 softlink ->
source
6689431 -rw-rw-r-- 1 sysadmin sysadmin 5 Feb 24 20:32 source
```

Skapa en symbolisk länk till `/proc`-katalogen och visa länken:

```
ln -s /proc crossdir
ls -l crossdir
```

Att dessa kommandon lyckas köras visar att mjuka länkar kan hänvisa till kataloger, och att de kan korsa från ett filsystem till ett annat, vilket är två saker som hårda länkar inte kan göra:

```
sysadmin@localhost:~$ ln -s /proc crossdir
sysadmin@localhost:~$ ls -l crossdir
lrwxrwxrwx 1 sysadmin sysadmin 5 Feb 24 20:59 crossdir -> /proc
```