FreeBSD – a szomszéd vár (6. rész)

Kiszolgálók – ACL és kiterjesztett attribútum a fájlrendszerben.

FreeBSD – a legtöbb UNIX rendszerhez hasonlóan – a könyvtárak és az állományok eléréséhez megfelelő jogokat követel meg, ezekkel a jogokkal képesek a felhasználók az állományokhoz hozzáférni. Ez a jogrendszer alapvetően három részre osztotta a felhasználók körét: tulajdonos, csoporttag és mindenki más. A UNIX történelem első évtizedében elegendőnek bizonyult, azonban mostanság ezek a jogok néhol erősen szűkösek lettek. S ez súlyos hiányosság, ha kiszolgálóként üzemeltetünk egy ilyen rendszert.

A UNIX jogosultságok rendszere

A régi nagygépes rendszerben minden bit kihasználásra került, mivel az erőforrások szűkösnek bizonyultak, s ennek megfelelően kellett jogosultságokat osztogatni állományokhoz és könyvtárakhoz. A leginkább erőforráskímélő megoldás felé hajlottak a fejlesztők, így csak az olvasás, az írás, illetve a futtatás jogait rendelték hozzá a fájlokhoz, valamint meghatároztak egy tulajdonost és egy csoportot is. A teljes jogrendszer tárolására elég volt néhány bájt adatterület, és ez nagyon jó kompromisszum volt a fejlesztők részéről. Azt is mondhatnánk erre a kialakításra, hogy több a semminél, de mindenre nem elég, s idővel a felhasználható tárterület is hatalmasra duzzadt, így ez a jogrendszer egyre inkább szűkös lett. Nézzük egy egyszerű példát, ahol van néhány felhasználónk: `tomy`, `noobi`, `joe` és `franko`. Ők mind egy munkahelyen dolgoznak, s néhány munkát meg kell osztaniuk egymás között. Például `tomy` dolgozik egy áramkör tervezésén, amelybe `noobi` is belesegít, viszont `joe` meg sem nézheti ezt a munkát, mert egy titkos fejlesztése a cégnek, miközben `franko` felügyelheti a tervezést. Ugyanakkor `franko` egy új programot fejleszt, amelyet `joe` és `noobi` csak megnézhet, viszont `tomy` ki van zárva a fejlesztők köréből.

Név	Munka1	Munka2
tomy	rwx	---
noobi	rwx	r--
joe	---	r--
franko	r--	rwx

A legjobb megoldás egy táblázatba foglalni a hozzáférések körét, amelyből hamar kiolvashatjuk, hogy ki és milyen jogosultságokkal képes hozzáférni a megadott állományokhoz. Megpróbálhatjuk ezeket a jogokat hozzárendelni a két állományhoz (`munka1` és `munka2`), de csoportok nélkül ez nem fog menni. Triviális az a tény, hogy a két munkaállomány csak kettő tulajdonossal jöhet létre, és a többiek jogait már csak a csoportjogokkal befolyásolhatjuk. Az is hamar észrevehető, hogy van mindkét munka esetén legalább egy olyan felhasználó, akinek egyetlen jogot nem tudunk adni. A második munka esetén létre tudunk hozni csoportot, amelynek tagjai (`noobi` és `joe`) csak olvasás joggal rendelkeznek, és elő is állt ennek a megoldása:

```
$ ls -l munka2
-rwxr-----  1 franko  munka2_csoport 0 Feb 22 13:35
➥munka2
```

Az első munkaállomány azonban szinte megoldhatatlan feladatot jelent, mivel nem tudunk kettő tulajdonost meghatározni, vagy kettő (egy `rwx` és egy `r--` jogú) csoportot rendelni az állományhoz. Tulajdonképpen ez a „kis” probléma az oka annak, hogy ***UNIX*** (vagy ***UNIX***-szerű) rendszerek nem tudtak betörni a fájlszerverek piacára, s nem tudták máig megtörni a ***Novell NetWare*** vagy a ***Microsoft Windows Server*** vezető szerepét ezen a területen.

Az ACL bekapcsolása

A ***FreeBSD 5.x*** az ***UFS2*** fájlrendszeren támogatja az ***ACL*** használatát (a ***FreeBSD 4.x*** esetén új rendszermagot kell fordítanunk), így képesek leszünk megoldani a jogosultsággal kapcsolatos problémáinkat. A kompatibilitás okán megmaradtak a megszokott jogok is, amelyeket azonos módon tudunk kezelni. Az ***ACL*** azonban alapértelmezés szerint nem aktív, bekapcsolásához a ***/etc/fstab*** megfelelő sorában meg kell adnunk az ***acls*** tulajdonságot is:

```
/dev/ad0s1f     /usr    ufs     rw,acls         1 1
```

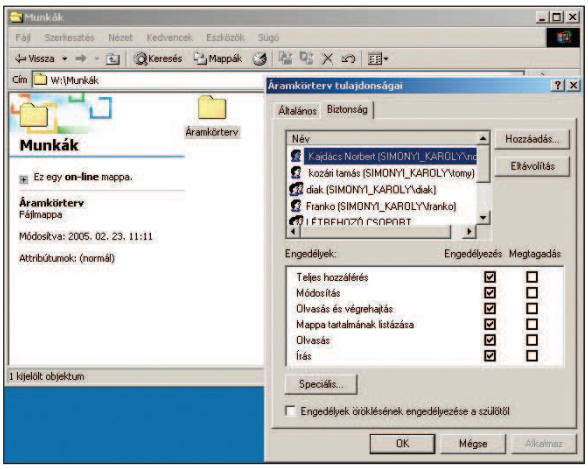
Természetesen bekapcsolhatjuk a `tunefs` programmal is, amelynél a `-a` kapcsoló `enabled` állásával tehetjük ezt meg:

```
$ tunefs -a enabled /dev/as0s1f
```

Ha mindezen túl vagyunk, akkor (elvileg) véget is ért a ***FreeBSD*** specifikus rész, s minden figyelmünket az ***ACL*** listák működésének megértésére tudjuk fordítani.

Hogy láthatjuk meg az ACL jogokat?

Kezdésképpen látni szeretnénk az ***ACL*** listákkal, hiszen – ha már beállítottuk – használnunk is kellene ezt a lehetőséget. Alapvetően a `getfacl` szolgál a hozzáférési listák lekérdezésére, de megvan a módja, hogy – például ***C*** nyelven – saját programból kezelhessük ezeket a listákat.



1. ábra ACL szerkesztése Windows alatt

Igazából bármelyik állományra vagy könyvtárra le tudjuk kérdezni a listát, egyszerűen kiadjuk a

```
$ getfacl munka2
```

parancsot, mire – ha minden jól megy – megkapjuk az állományunk eddigi beállításait:

```
#file:munka2
#owner:1001
#group:1008
user::rwx
group::r--
other::---
```

Ez tulajdonképpen azonos az

```
$ ls -l munka2
```

parancs által kiírt

```
-rwxr-----  1 franko  munka2_csoport 0 Feb 22 13:35
➥munka2
```

sorral, csak az *ACL*-listából nem tudjuk meg, hogy állománnyal vagy könyvtárral van-e dolgunk, illetve csak numerikusan láthatjuk a tulajdonost és a csoportját. A többi sor már az *ACL*-lista része, amelyet (angol nyelven) szövegesen olvasható formában láthatunk, s ennek a formátuma eléggé kötött, mivel csak a tradicionális *UNIX* jogokat láthatjuk. Tetszőleges állomány esetén találkozni fogunk *user*, *group* és *other* jogokkal, mint ahogy ezt meg is szokhattuk. A különbség annyi, hogy látunk kettőspontokat, amelyek elválasztják egymástól a kulcsszavakat és a jogok jelzéseit. A két kettőspont közé azonban olyan információkat tudunk írni, mint egy felhasználónév vagy egy csoport. Sőt, ezekből többet is felsorolhatunk; kezdésképpen oldjuk meg a `munka2` állomány jogait *ACL* használatával.

Hogy adjunk meg ACL jogokat?

A jogok kezelését a `setfacl` parancs végzi, ennek parancssorában meg kell adnunk azon jogot, amelyet hozzá szeretnénk fűzni az eddigi listához. Ehhez először is egy kis módosításra van szükség az állományunk általános jogai környékén:

```
$ chmod 600 munka2
$ chown :franko munka2
$ ls -l munka2
-rw-------  1 franko  franko  0 Feb 22 13:35 munka2
```

Ezzel a `munka2` állomány tulajdonosa és csoportja egyaránt `franko` lesz (a *FreeBSD* esetén minden felhasználónak alapból a saját nevével létrejön egy csoport is), illetve csak a tulajdonosnak van olvasás és írás joga. Ha újra lekérdezzük a *ACL* jogokat, sok változás nem fog történni a fent végrehajtott változásokat leszámítva:

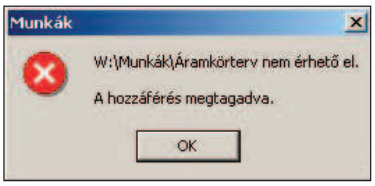
```
$ getfact munka2
#file:munka2
#owner:1001
#group:1003
user::rw-
group::---
other::---
```

A hozzáférést leíró táblázat szerint `noobi` és `joe` férhet hozzá az állományhoz olvasás joggal. Őket egyszerűen csak hozzá kell adni a listához, majd rögtön meg is nézzük, mi változott.

```
$ setfacl -m user:noobi:r-- munka2
$ setfacl -m user:joe:r-- munka2
$ getfacl munka2
#file:munka2
#owner:1001
#group:1003
user::rw-
user:joe:r--
user:noobi:r--
group::---
mask::r--
other::---
$ ls -l munka2
-rw-r-----+ 1 franko  franko  0 Feb 22 20:35 munka2
```

Az első különlegességet az `ls` parancs kimenetében találjuk, mégpedig egy + jel képében, amely azt jelzi, hogy a látott jogoknál sokkal több jogdefinícióra kell számítanunk, mint amennyit látunk. Ezek a jogok a `setfacl` parancs `-m` kapcsolójával kerültek oda, amely a *merge* rövidítése, így annyival több a szimpla hozzáadásnál, hogy nem ismétli meg az azonos sorokat. A `getfacl` listájában találkozunk is ezekkel a sorokkal, pont úgy, ahogy beírtuk őket. Az egyetlen ismeretlen sor a `mask::r--` lesz, amely annyit tesz, hogy a meghatározza a legmagasabb adható jogokat – leszámítva természetesen a tulajdonos saját *ACL* bejegyzését és az egyéb kategória jogait. Ez a maszk arra jó, hogy automatikusan mutatja a legmagasabb jogokat az *ACL* jogok közül, illetve azonos azzal amit az `ls -l` kiír a csoportra vonatkozó jogoknál. Megadhatunk csoportokra is jogokat, viszont törekedjünk arra, hogy tisztán csak felhasználókat adjunk hozzá a listához, bár sok olyan helyzettel találkozhatunk, mikor csoportokat egyszerűbb kezelni. Kipróbálhatjuk, hogy működik-e a megoldásunk, egyszerűen a megnevezett felhasználókkal kell próbálnunk olvasni, írni vagy futtatni a megadott állományt (csodálkoznék, ha nem működne helyesen). A `munka1` állomány jogait is gyorsan beállíthatjuk:

```
$ chmod 600 munka1
$ chown tomy:tomy munka1
$ setfacl -m user:tomy:rwx munka1
$ setfacl -m user:noobi:rwx munka1
$ setfacl -m user:franko:r-- munka1
$ getfacl munka1
#file:munka1
#owner:10636
#group:1008
user::rw-
user:franko:r--
user:noobi:rwx
user:tomy:rwx
group::---
mask::rwx
other::---
```



2. ábra Hozzáférés megtagadva...

```
$ setfacl -d -m user:tomy:rwx
➥Áramkörterv
setfacl: acl_calc_mask() failed:
➥Invalid argument
setfacl: failed to set ACL mask on
➥Áramkörterv/
```

Az ACL jogok másolása

Az egész játszadozás a jogokkal nem ér semmit, ha ezt minden egyes új állományra alkalmaznunk kell. Egyik megoldásunk erre az *ACL* „átmásolása" egy másik állományra:

```
$ getfacl munka1 | setfacl -M - munka2
```

Ezzel a módszerrel a `munka1` állomány *ACL* listája hozzámásolódik a `munka2` állomány *ACL* listájához. Ha ez utóbbi „üres" volt eredetileg, akkor a két állomány listája azonos. Ez látszólag félmegoldás, mivel az új állományokra egyesével meg kell oldani ezt a másolást. Megtehetjük, hogy egyik állomány jogait levesszük egy másik állomány listájáról, ekkor a

```
$ getfacl munka1 | setfacl -X - munka2
```

parancsot kell használnunk.

Az ACL jogok örökítése

Rövid és egyszerű munkákat leszámítva a valóságos projektek több állományt, esetleg egy egész könyvtárstruktúrát jelentenek. Ezeket az állományokat érdemes tehát egy közös könyvtárba tenni, és a könyvtárakkal is eljátszani a jogok kiosztásakor. Maradjunk meg a cikk eleje táján vázolt táblázatnál, csak az eddig használt állományok helyett egy könyvtárstruktúrát építünk ki, amely egy ***Munkák*** főkönyvtárból, benne egy ***Áramkörterv*** és egy ***Program*** alkönyvtárból. Érdemes a projekt könyvtárát arra a felhasználóra átállítani, aki a gazdája annak, így az ***Áramkörterv*** tulajdonosa `tomy` lesz, a ***Program*** könyvtáré pedig `franko`.
A `setfacl -d` kapcsolója felel az alapértelmezett *ACL* jogok kezelésért, amely öröklődik a könyvtárstruktúrában. Ha ezt a kapcsolót is megadjuk, akkor a hozzáfűzött lista lesz az összes – e könyvtár alatt – létrehozott állomány alapértelmezett *ACL* joga. Az alapértelmezett listát a `getfacl -d` kapcsolójával tudjuk lekérdezni:

```
$ getfacl -d Áramkörterv
#file:Áramkörterv
#owner:10636
#group:1008
```

A listánk láthatólag teljesen üres, ezért hozzá kell adnunk néhány jogot. Ha ezt a tevékenységet nem a ***UNIX*** jogokkal kezdjük, akkor az alábbi hibaüzenettel nézünk farkasszemet:

A megoldás egyszerűen annyi, hogy először és lehetőleg egyszerre hozzá kell rendelnünk azokat a jogokat, amelyek az állomány tulajdonosára és a csoportjára vonatkoznak:

```
$ setfacl -d -m u::rwx,g::---,o::---
➥Áramkörterv
```

Ekkor már hozzá tudjuk rendelni egyenként a felhasználók jogait a könyvtárhoz (ne felejtsük el, hogy a `franko` felhasználónak adjunk x jogot is, különben nem fog tudni belépni a könyvtárakba):

```
$ setfacl -d -m user:tomy:rwx Áramkörterv/
$ setfacl -d -m user:noobi:rwx Áramkörterv/
$ setfacl -d -m user:franko:r-x Áramkörterv/
$ setfacl -m user:tomy:rwx Áramkörterv/
$ setfacl -m user:noobi:rwx Áramkörterv/
$ setfacl -m user:franko:r-x Áramkörterv/
```

Az elkészült struktúrában már csak néhány próbát kell elvégezni, hogy láthassuk működés közben:

```
franko Munkák/Áramkörterv $ touch állomány
touch: állomány: Permission denied
franko Munkák/Áramkörterv $ mkdir könyvtár
mkdir: könyvtár: Permission denied
tomy Munkák/Áramkörterv $ touch állomány
noobi Munkák/Áramkörterv $ touch állomány2
tomy Munkák/Áramkörterv $ mkdir könyvtár
noobi Munkák/Áramkörterv $ mkdir könyvtár2
tomy Munkák/Áramkörterv $ touch állomány2
touch: állomány2: Permission denied
```

Hoppá... minden a tervek szerint működne, viszont egy apróságról megfeledkeztünk. Új állomány vagy könyvtár létrehozásakor az operációs rendszer néhány bitjét maszkolja a jogoknak. Alapértelmezésben ez `0022`, amely szerint a csoport és az egyéb felhasználók nem lesznek képesek írni a létrehozott állományokba, s ez az *ACL* listák nélkül tökéletesen megfelelő megoldás. Azonban ez hatással van az *ACL* maszkra is, így az újonnan létrehozott állományok `mask` sora is változni fog:

```
$ getfacl állomány
#file:állomány
#owner:10636
#group:1008
user::rw-
user:franko:r-x          # effective: r--
user:noobi:rwx           # effective: r--
user:tomy:rwx            # effective: r--
group::---
mask::r--
other::---
```

A megoldás egyszerű: vagy ideiglenesen megváltoztatjuk az `umask` parancs segítségével ezt a változót, vagy beépítjük a */etc/login.conf* állományba.
Nézzük meg a változást:

```
tomy Munkák/Áramkörterv $ umask 0002
tomy Munkák/Áramkörterv $ touch állomány3
noobi Munkák/Áramkörterv $ umask 0002
noobi Munkák/Áramkörterv $ touch állomány3
franko Munkák/Áramkörterv $ umask 0002
franko Munkák/Áramkörterv $ touch állomány3
touch: állomány3: Permission denied
```

Vissza vannak még a könyvtárak ellenőrzése, de ez már nem okoz gondot:

```
tomy Munkák/Áramkörterv $ mkdir könyvtár
noobi Munkák/Áramkörterv $ mkdir könyvtár2
franko Munkák/Áramkörterv $ mkdir könyvtár3
mkdir: könyvtár3: Permission denied
```

A könyvtárak létrehozására is működik az alapértelmezett listánk, próbáljuk ki a könyvtárváltást is:

```
tomy Munkák/Áramkörterv $ cd könyvtár2/
noobi Munkák/Áramkörterv $ cd könyvtár
franko Munkák/Áramkörterv $ cd könyvtár
```

Tehát mindenki megkapta a futtatás jogok a létrehozott könyvtárakra. A próbát oldjuk meg úgy, hogy `tomy` és `noobi` felhasználó egymás könyvtárába lépjen bele, ezzel kiderítjük, hogy képesek-e új állományt létrehozni, hogy a másik felhasználó a könyvtár tulajdonosa:

```
franko Munkák/Áramkörterv/könyvtár $ touch állomány
touch: állomány: Permission denied
tomy Munkák/Áramkörterv/könyvtár2 $ touch állomány
noobi Munkák/Áramkörterv/könyvtár $ touch állomány2
```

S végezetül az illetéktelen felhasználók jogait is érdemes megtekinteni:

```
auth.gabor Munkák $ cd Áramkörterv/
-bash: cd: Áramkörterv/: Permission denied
```

Kapcsolat a Samba programmal

Mindennek a sok szövegnek nincs értelme, ha nem tudnánk ezek után fájlszerverként használni rendszerünket. A *Samba* program képes az *ACL* támogatást kihasználni, csak úgy kell fordítanunk (vagy olyan lefordított binárist telepítenünk), hogy ezt tudatosítjuk is benne. Ezen túl más dolgunk nincs is, mint kipróbálni a megosztáson az említett jogokat.
Ha például egy *Windows* ügyfélen felcsatoljuk a megosztást (a példában `W:` meghajtóként), és lekérdezzük az *Áramkörterv* könyvtár jogait, akkor egyszerűen ugyan azokat a jogokat kell látnunk, mint amit kínkeserves munkával begépeltünk a parancssorba. Egy kicsit látványosabb és könnyebb is egy ilyen megosztáson át piszkálni a hozzáférési jogokat, a parancssori mókát hagyjuk a programocskákra és az esetleges apróbb módosításokra.
Az *ACL* működését ki is próbálhatjuk – szintén *Windows* ügyfél segítségével. A megosztást olyan felhasználó nevével csatoljuk fel meghajtóként, akinek nincs joga olvasni sem az *Áramkörterv* könyvtár tartalmát, majd próbáljunk belépni a megnevezett könyvtárba. Elvileg *„A hozzáférés megtagadva”* üzenetet kell látnunk a képernyőn, demonstrálva ezzel a jól működő rendszerünket.

Kiterjesztett attribútumok

Sokszor jönne jól, ha egy állományhoz megjegyzéseket tudnánk fűzni, például olyan képet hordozó állományhoz, amely formátum nem támogatja a megjegyzésmezőt. A kiterjesztett attribútumok pont erre a célra szolgálnak: hozzá tudunk fűzni kiegészítő adatokat az állományainkhoz.
Az *ACL* kezeléséhez hasonlóan itt a `getextattr`, a `setextattr`, az `lsextattr` és az `rmextattr` parancsok szolgálnak e tulajdonság kezelésére. Egy állományhoz több attribútum is rendelhető, ugyanis meg kell neveznünk az információt, amit tárolni szeretnénk. Ezen túlmenően két névtér közül kell kiválasztanunk a megfelelőt: `user` vagy `system`. Például a lementett nyers képhez hozzáfűztem a felhasználói névterületen egy `tartalom` nevű változóba a kép magyarázó szövegét:

```
$ setextattr user tartalom “Samba – Egy könyvtár
➥jogainak beállítása” Samba01.bmp
$ setextattr user idopont “`date`” Samba01.bmp
```

Meg tudjuk nézni, hogy milyen attribútumok vannak egy állományhoz rendelve:

```
$ lsextattr user Samba01.bmp
Samba01.bmp     tartalom        idopont
```

Illetve le tudjuk kérdezni a megnevezett attribútumot is:

```
$ getextattr user tartalom Samba01.bmp
Samba01.bmp     Samba ACL kezelése
```

Ha már nem szükséges, akkor törölhetjük is:

```
$ rmextattr user tartalom Samba01.bmp
```

Ezt az információs területet sok érdekes dologra felhasználhatjuk, bár a legtöbb formátum esetén maguk a programok oldják meg az ilyen információk kezelését az állományon belül.

Auth Gábor (auth.gabor@enaplo.hu)
Egy pécsi középiskolában informatikát és programozást oktat. Tíz éve botlott először a UNIX rendszerekbe, 7 év Linux használat után kapta el a FreeBSD lázat, amiből máig nem tudott kigyógyulni.

KAPCSOLÓDÓ CÍMEK

A FreeBSD projekt honlapja
➲ http://www.freebsd.org

A magyar FreeBSD honlap
➲ http://www.freebsd.hu

A magyar BSD honlap
➲ http://www.bsd.hu

A kézikönyv magyar fordítása
➲ http://www.enaplo.hu/FreeBSD/handbook/