Kódolás és IT biztonság (VIHIBB01)
Laboratóriumi gyakorlat

Engedélyezés, hozzáférés-szabályzás (Linux operációs rendszeren)

Szerző:
Ládi Gergő

Utoljára frissült: 2020. november 1.

Tartalomjegyzék

1. A labor célja

A labor során megismerheted, hogyan működik a felhasználók és csoportok kezelése, valamint a Discretionary Access Control Linux rendszerek esetében. A labor célja, hogy a gyakorlatban is kipróbálhassátok, hogyan történik a felhasználók és csoportok létrehozása, módosítása és törlése, továbbá lássátok, hogyan működik és hogyan lehetséges a fáljokhoz való hozzáférés szabályzása a hagyományos Unix engedélyek segítségével.

A feladatok sikeres megoldása után képes leszel alapvető felhasználó- és csoportkezeléssel kapcsolatos műveletek önálló elvégzésére valamint egyszerűbb jogosultságkezeléssel kapcsolatos feladatok megoldására Linux alapú környezetekben.

2. Elméleti háttér

2.1. Felhasználókezelés

A Linux alapú operációs rendszerek esetében minden felhasználó rendelkezik egy szöveges felhasználónévvel és egy számszerű azonosítóval (UID), melyek közül mindkettő rendszerszinten egyedi[1]. A 0 és 999 közötti UID-k általában a rendszer és az azon futó szolgáltatások számára fenntartottak, a tényleges emberi felhasználók UID-jei 1000-től kezdődnek. A 0-s UID-jű felhasználó egy különleges felhasználó, a *root*. A root felhasználó a lehető legmagasabb szintű jogosultságokkal rendelkezik, minden jogosultságellenőrzésen automatikusan átmegy[2]. Éppen ezért kiemelten fontos, hogy megfelelően erős jelszava legyen, valamint hogy korlátozzuk a fiókhoz való közvetlen távoli hozzáférést (pl. SSH).

A felhasználókhoz kapcsolódó összes információt a mindenki által olvasható (szakkifejezéssel: world-readable) */etc/passwd* fájlban tárolja a rendszer, kivéve a jelszavakat és a jelszavakkal kapcsolatos információkat. Ezek a */etc/shadow* fájlban találhatók, amelyhez csak a root felhasználó tud hozzáférni. (A teljesség kedvéért megjegyzem, hogy a rendszer felhasználói a passwd fájl mellett származhatnak külső forrásból is, például LDAP-ból, viszont ez már túlmutat a labor anyagán.)

[1] Technikailag lehetséges, hogy több felhasználó is ugyanazt a UID-t használja, ez azonban általánosságban nem ajánlott.

[2] Néhány kivételtől eltekintve, amellyel most nem foglalkozunk.

```
gergo.ladi@demovm:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
gergo.ladi:x:1001:1001:Gergő Ládi,,,:/home/gergo.ladi:/bin/bash
mysql:x:107:112:MySQL Server,,,:/nonexistent:/bin/false
redis:x:108:113::/var/lib/redis:/bin/false
```

1. ábra. Részlet egy /etc/passwd fájból.

A passwd fájl felépítése az 1. ábrán látható. Minden sor egy felhasználót jelöl, és minden sor mezőkből áll, melyeket kettőspontok választanak el.

A mezők jelentése a következő:

1. A felhasználó neve.
2. Ez a mező egykoron a felhasználó jelszavát tartalmazta (hashelve). Mivel a fájl mindenki által olvasható, bármelyik helyi felhasználó megnézhette a hasheket, majd megpróbálhatta feltörni azokat, ezáltal adott esetben hozzáférve olyan, magasabb jogosultságokkal rendelkező fiókokhoz, melyek jelszava gyenge volt. A modern Linux rendszereken ezen mező tartalma fixen egy x, és a tényleges jelszóhashek a /etc/shadow fájlban találhatók.
3. A felhasználó UID-ja.
4. A felhasználó elsődleges csoportjának azonosítója (erről bővebben később).
5. A GECOS mező. Ebben tárolhatjuk a felhasználó emberi nevét, szobaszámát, otthoni és irodai telefonszámát, valamint egyéb kapcsolódó információkat.
6. A felhasználó home könyvtára. Ide mutat a ~/, belépés után ide nyílik a shell (fura magyar szóval rendszerhéj). A legtöbb esetben ez egy olyan könyvtárra mutat, amelyhez csak az adott felhasználónak van hozzáférése.
7. A felhasználó alapértelmezett shellje. Ez a program indul el és kapja meg a vezérlést, miután sikeresen belépett a felhasználó. A szolgáltatások fiókjainak esetében (amelyeknek nincs szükségük shellre), ez biztonsági okokból általában a */usr/sbin/nologin* vagy a */bin/false*.

```
gergo.ladi@demovm:~$ sudo cat /etc/shadow
root:!:17804:0:99999:7:::
daemon:*:17804:0:99999:7:::
bin:*:17804:0:99999:7:::
sys:*:17804:0:99999:7:::
sync:*:17804:0:99999:7:::
_apt:*:17804:0:99999:7:::
sshd:*:17804:0:99999:7:::
gergo.ladi:$6$Qfz(...)YZ$sQMCjdFnL.d1o2P(...)NWbu60:17804:0:99999:7:::
mysql:!:17804:0:99999:7:::
redis:*:17805:0:99999:7:::
```

2. ábra. Részlet egy /etc/shadow fájlból. Bizonyos részek törölve lettek.

A shadow fájl (2. ábra) formátuma hasonló a passwd fájléhoz. Az egyes sorokban szerepel egy felhasználó neve, $algorithm_identifier$salt$salted_hash formátumban a jelszava, valamint az utolsó jelszóváltás és a lejárat dátuma pár egyéb kapcsolódó mező mellett. Ha a jelszó mező egy csillagot (*) vagy felkiáltójelet (!) tartalmaz, akkor az adott felhasználó jelszóval nem fog tudni belépni, de más módon, például SSH kulccsal, adott esetben beléphet.

A /etc/passwd és /etc/shadow fájlok kézzel történő szerkesztése erősen ellenjavallott, helyette a felhasználókezelő parancsokat érdemes használnunk. Ugyanis, ha véletlenül megsértjük a fájlok struktúráját, érvénytelen adatot viszünk fel ezekbe, akár olyan szinten használhatatlanná tehetjük a rendszert, hogy senki nem fog tudni többet belépni arra.

2.1.1. Felhasználókezeléssel kapcsolatos parancsok

Íme néhány Linux parancs, amely hasznosnak bizonyulhat felhasználókezeléssel kapcsolatos feladatok megoldásánál. A lista nem teljes, és az egyes parancsokhoz is csak egy-egy lehetséges (a leggyakoribb) paraméterezés szerepel. Az egyes parancsok részletes leírása, lehetséges paraméterei megtekinthetők az adott parancs *man page*-én.

- adduser *username* – létrehoz egy új felhasználót *username* néven. *root*ként kell futtatni.
- passwd *[username]* – megváltoztatja a jelszavad, vagy ha root vagy és megadtál egy felhasználónevet is, akkor az ő jelszavát változtatja meg.

- deluser *username* – törli a *username* nevű felhasználót.
  *root*ként kell futtatni.
- id *username* – kiírja a megadott felhasználó UID-ját és GID-ját, valamint a hozzá tartozó, csoporttagsággal kapcsolatos információkat.
- chage *[options] username* – a *username* jelszavának lejáratával kapcsolatos beállításait változtatja meg. *root*ként kell futtatni.
- chfn *[username]* – a GECOS meződ tartalma szerkeszthető vele. A *teljes név* értéket csak a root változtathatja. Ha a *username* paraméter adott és rootként futtatod, más felhasználók adatait is tudod szerkeszteni.
- su *[-] username* – felveheted vele egy másik felhasználó identitását. Tudnod kell a jelszavát vagy rootként kell futtatnod. Ha a - kapcsolót is használod, az adott felhasználó környezeti változóit is megkapod.
- sudo *command* – lefuttatja a *command* parancsot egy másik (általában a root) felhasználó nevében a /etc/sudoers fájlban megadottak szerint.

2.2. Csoportkezelés

A felhasználók csoportokba szerkeszthetők. Ezáltal kényelmesebbé válhat az engedélyek kiosztása, hiszen ahelyett, hogy azokat minden felhasználónak egyesével kellene megadni, az ugyanolyan jogosultságú felhasználókat egy csoportba szervezhetjük, majd ennek a csoportnak adhatunk engedélyeket. Így, ha később egy felhasználótól vissza szeretnénk vonni az engedélyeket, mert például távozik a cégtől, úgy nem kell minden egyes olyan objektumot felkutatni, amelyre volt engedélye, majd ezeket egyesével szerkeszteni, hanem csak egyszerűen törölhetjük a felhasználót a csoportból, melynek hatására el fogja veszteni a csoporton keresztül kapott engedélyeit is.

A felhasználókhoz hasonlóan a csoportokat is egy számmal (GID – group identifier) és egy névvel azonosítjuk, melyeknek szintén egyedieknek kell lenniük (de egy csoportnak és egy felhasználónak lehet ugyanaz a neve vagy az azonosítója). Minden felhasználóhoz létezik egy alapértelmezett csoport (default group), melynek ő az egyetlen tagja. A csoportok ugyanazt a számozási stratégiát követik, mint a felhasználók – a rendszerhez és a szolgáltatásokhoz tartozó csoportok általában 0 és 999 közötti számot kapnak, míg a normál csoportok azonosítói 1000-től kezdődnek. Egy felhasználó több csoportnak is tagja lehet, de mindenkinek csak egy úgynevezett elsődleges csoportja lehet.

```
gergo.ladi@demovm:~$ cat /etc/group
root:x:0:
daemon:x:1:
sudo:x:27:gergo.ladi
www-data:x:33:
gergo.ladi:x:1001:
mysql:x:112:
redis:x:113:
```

3. ábra. Részlet egy /etc/group fájlból.

A csoportokra vonatkozó információkat a /etc/group fájlban tároljuk, a 3. ábrán látható formátumban. Az első mező tartalmazza a csoport nevét, a második a csoport jelszavát (manapság ezt már nem használjuk), a harmadik a csoport azonosítószámát (GID), végül a negyedik a tagokat. Ezt a fájlt kézzel szerkeszteni erősen ellenjavallt, helytte használjuk a csoportkezelő parancsokat.

Megjegyzés: a felhasználók csoporttagságaiban bekövetkező változások nincsenek hatással a már létező munkamenetekre (sessionökre). Ahhoz, hogy ezek érvényre lépjenek, az érintett felhasználóknak ki kell jelentkezniük, majd újra be kell lépniük (vagy valamilyen más egyéb módon új munkamenetet kell nyitniuk).

2.2.1. Csoportkezeléssel kapcsolatos parancsok

Következzék néhány parancs, amelyre csoportok kezelésénél lehet szükség. A lista nem teljes, és az egyes parancsokhoz is csak egy-egy lehetséges (a leggyakoribb) paraméterezés szerepel. Az egyes parancsok részletes leírása, lehetséges paraméterei megtekinthetők az adott parancs *man page*-én.

- addgroup *groupname* – létrehoz egy új csoportot *groupname* néven. *root*ként kell futtatni.
- delgroup *groupname* – törli a *groupname* nevű csoportot. *root*ként kell futtatni.
- groups *[username]* – kiírja azon csoportokat, amelyeknek az adott felhasználó a tagja. Ha nem adunk meg felhasználónevet, akkor a parancsot kiadó felhasználó csoportjait kapjuk meg.
- usermod -g *primarygroup username* – a *username* felhasználó elsődleges csoportja mostantól a *primarygroup* lesz. *root*ként kell futtatni.

- adduser *username groupname* – hozzáadja *username* felhasználót a *groupname* csoporthoz. *root*ként kell futtatni.
- deluser *username groupname* – törli *username* felhasználót a *groupname* csoportból. *root*ként kell futtatni.

2.3. Hagyományos Unix engedélyek (permissions)

A Linux rendszerek esetében minden objektumra fájlként tekintünk. A könyvtárak, a hálózati kapcsolatok, a futó programok, a merevlemez, de még a billentyűzeted is fájlként jelenik meg a fájlrendszerben.

Alapesetben a fájlokhoz való hozzáférést a hagyományos Unix engedélyekkel szabályozzuk, amely egy Discretionary Access Control (DAC) jellegű hozzáférés-szabályzási modell. Ebben a konkrét modellben minden fájlhoz három entitásnak adhatunk engedélyeket: a tulajdonosnak (owner, **u**), a tulajdonos csoportnak (group owner vagy owner group, g) és mindenki másnak (others, o). Minden entitásnak minden fájlra három különböző engedélye lehet: olvasás (read, r), írás (write, w) és végrehajtás (execute, x), ahol az utóbbi azt jelenti, hogy elindítható, lefuttatható az adott program vagy szkript. Könyvtárak esetében az olvasási engedély lehetővé teszi a tartalom listázását, az írási engedéllyel létrehozhatók új fáljok és könyvtárak benne vagy meglévők törölhetők belőle, a végrehajtási engedély pedig a tartalom olvasásához, írásához szükséges.

A Unix engedélyeket legegyszerűbben az *ls* paranccsal, annak is a *-l* kapcsolójával tudjuk kilistázni egy könyvtárra vonatkozóan, ahogyan ez látható is a 4. ábrán. Minden sorban egy fájl adatai láthatók. Az első betű jelöli a fájl típusát, például a - hagyományos értelemben vett fájlt jelöl, a *d* könyvtárat, az *l* linket, és így tovább. A következő három betű (a képen lilával) jelöli a tulajdonos engedélyeit: ha van olvasási engedélye, akkor az első betű helyén *r* látható, egyébként -; ha van írási engedélye, akkor a második betű helyén *w* látható, egyébként -; és ha van végrehajtási engedélye, akkor a harmadik betű helyén *x* szerepel, egyébként -. A második betűhármas (a képen pirossal) jelöli a tulajdonos csoport engedélyeit, és az utolsó, harmadik betűhármas (a képen kék színnel) jelöli a *mindenki más*ra vonatkozó engedélyeket (azaz azok engedélyeit, akik sem nem a tulajdonosok, sem nem a tulajdonos csoport tagjai). A fájl tulajdonosa (narancssárgával) és a tulajdonos csoportja (zölddel) szintén megjelennek a kimeneten.

Az engedélyek kiértékelésénél mindig csak az adott entitáshoz legközelebb álló engedélyeket vesszük figyelembe, azaz például ha a tulajdonosa szeretne

```
gergo.ladi@demovm:~$ ls -lah
total 40K
drwxr-xr-x 4 gergo.ladi gergo.ladi 4.0K Oct  6 12:56 .
drwxr-xr-x 3 root       root       4.0K Sep 30 21:08 ..
-rw------- 1 gergo.ladi gergo.ladi   25 Oct  6 12:56 .bash_history
-rw-r--r-- 1 gergo.ladi gergo.ladi  220 Sep 30 21:05 .bash_logout
-rw-r--r-- 1 gergo.ladi gergo.ladi 3.5K Sep 30 21:09 .bashrc
drwxrwx--- 2 gergo.ladi inboxusers 4.0K Oct  6 16:23 Inbox
-rw-r--r-- 1 gergo.ladi gergo.ladi  675 Sep 30 21:05 .profile
drwxr-xr-x 4 gergo.ladi gergo.ladi 4.0K Oct  6 16:23 Public
```

4. ábra. Az *ls* parancs kimenete, fájlok engedélyeit mutatva.

megnyitni egy fájlt, akkor csak a tulajdonos engedélyei számítanak, a tulajdonos csoporté és a mindenki másé nem. Ez lehetővé teszi, hogy furcsának tűnő engedélyeket is beállítsunk, például a ----rwxrwx esetében bárki bármit tehet a fájllal, kivéve a tulajdonost, aki nem fogja tudni sem olvasni, sem írni, sem végrehajtani (de persze tulajdonosként megváltoztathatja az engedélyeket).

Az engedélyeket az r, w, x betűkkel történő megadás mellett három számjegyű számokkal is megadhatjuk. Ebben az esetben az első számjegy jelenti a tulajdonos engedélyeit, a második a tulajdonos csoportét, végül a harmadik a mindenki másét. Az olvasási engedély bit 4-et ér, az írás 2-t, a végrehajtás 1-et. A megfelelő engedély megléte esetén a számértékeket összeadva megkapjuk az adott entitás adott fájlra vonatkozó engedélyét számként. Például a -rwxrwxrwx leírható 777-ként is, és a -rwxr-x--- pedig 750-ként.

Valahányszor valaki létrehoz egy fájlt, a fájl tulajdonosa a létrehozója lesz, a fájl *tulajdonos csoportja* pedig a létrehozó elsődleges csoportja. Alapértelmezés szerint az új fájlok 666-os engedélyekkel (-rw-rw-rw-) jönnek létre, míg a könyvtárak 777-tel (drwxrwxrwx). Ez a viselkedés megváltoztatható az *umask* paranccsal. Az umask paranccsal beállíthatunk egy *maszk*ot, amelynek értéke ki lesz vonva minden újonnan létrejövő fájl (és könyvtár) engedélyeiből. Például, a tipikus umask érték 022, ami azt jelenti, hogy minden újonnan létrejövő fájl engedélye $666 - 022 = 644$ (-rw-r--r--) lesz, míg minden újonnan létrejövő könyvtáré $777 - 022 = 755$ (drwxr-xr-x) lesz.

2.3.1. Engedélykezeléssel kapcsolatos alapvető parancsok

A lista nem teljes, és az egyes parancsokhoz is csak egy-egy lehetséges (a leggyakoribb) paraméterezés szerepel. Az egyes parancsok részletes leírása,

lehetséges paraméterei megtekinthetők az adott parancs *man page*-én.

- chmod *permissions filename* – megváltoztatja egy fájl engedélyeit. Csak a fájl tulajdonosa vagy a root felhasználó adhatja ki. Az engedélyeket megadhatjuk számként (pl. 700) vagy műveletek formájában is (pl. +x vagy u+w – az előbbi minden entitásnak ad végrehajtási jogot, míg az utóbbi csak a tulajdonosnak ad írási jogot).
- chown *new_owner filename* – megváltoztatja egy fájl tulajdonosát (és opcionálisan a tulajdonos csoportját). *root*ként kell futtatni.
- chgrp *new_owner_group filename* – megváltoztatja egy fájl tulajdonos csoportját. *root*ként kell futtatni.
- umask *new_mask* – a jelenlegi processzre (és azok esetleges jövőbeli gyerekfolyamataira) vonatkozóan átállítja az umask értéket.

2.3.2. Haladó engedélybitek

A fentebb tárgyaltakon felül létezik három haladó engedélybit, melyeket fájlokhoz és könyvtárakhoz rendelve módosítható azok viselkedése.

- A *setuid* (vagy *suid*) bit – ha beállítjuk egy végrehajtható állományra, akkor valahányszor ezt az állományt valaki elindítja, az a tulajdonos nevében, a tulajdonos engedélyeivel fog futni, függetlenül attól, hogy ki indította el. Ezen bit beállítása biztonsági kockázatot jelenthet, főleg ha a tulajdonos a root felhasználó. Ha a fájl szerkeszthető, mások szerkeszthetik azt, hogy a tulajdonos nevében tetszőleges parancsokat hajthassanak végre (manapság ez már egyre kevésbé működik, mert szerkesztéskor a mai rendszerek törlik a suid bitet). Ha a program biztonsági szempontból sérülékeny, akár írási jog nélkül is futtatható lehet tetszőleges kód a tulajdonos nevében a sérülékenységen keresztül (ez viszont manapság is problémát jelenthet). A suid bitnek nincs hatása könyvtárak vagy scriptek esetén, és szintén nincs hatása, ha a tulajdonosnak vagy a programot elindítani kívánónak nincs végrehajtási joga a fájlon. Tipikusan arra használjuk, hogy nem-root felhasználók számára is elérhetővé tegyünk egy-egy olyan parancsot, amelyhez egyébként root jogosultságok kellenének. A suid bit a *chmod u+s filename* paranccsal állítható be. Az ilyen bittel rendelkező fájlok esetében a jogosultságlistában a tulajdonos engedélyeinél az x helyén s betű fog szerepelni (pl. -rwsr-xr-x).

- A *setgid* (vagy *sgid*) bit – ha beállítjuk egy végrehajtható állományra, akkor valahányszor ezt az állományt valaki elindítja, az a tulajdonos csoport nevében, a tulajdonos csoport engedélyeivel fog futni, függetlenül attól, hogy milyen csoportoknak volt tagja az állományt elindító illető (és az effektív tulajdonos maga nem változik, mint ahogyan változott a setuid esetében). Ha egy könyvtárra állítjuk be (és inkább könyvtárakra szokás beállítani), az abban létrejövő új fájlok tulajdonos csoportja nem az azokat létrehozó illető elsődleges csoportja lesz, hanem a könyvtár tulajdonos csoportja. Tipikusan rendszeren belüli megosztott könyvtárak kialakításakor használjuk. A setgid bit a *chmod g+s filename* paranccsal állítható be. Az sgid bittel rendelkező fájlok, könyvtárak esetében a jogosultséglistában a tulajdonos csoport engedélyeinél az *x* helyén *s* betű fog szerepelni (pl. -rwxrwsrwx).
- A sticky bit – ha beállítjuk egy könyvtárra, a benne lévő fájlokat nem fogja tudni törölni senki, csak a fájlok tulajdonosa (és persze a root felhasználó), még akkor sem, ha egyébként lenne erre jogosultsága az illetőnek. A sticky bit (ma már) nincs hatással fájlokra. Tipikusan olyan, többek által is használt könyvtárakra állítjuk be, ahol nem szeretnénk, hogy egymás fájljait töröljék a felhasználók. A sticky bit a *chmod +t dirname* paranccsal állítható be. Az ezzel a bittel rendelkező könyvtárak esetében a jogosultséglistában a *mindenki más* engedélyeinél az *x* helyén *t* betű fog szerepelni (pl. drwxrwxrwt).

2.4. POSIX ACL-ek

A hagyományos Unix engedélyek egyik nagy hátránya, hogy nem elég granulárisak, azaz segítségükkel nem mindig tudunk minden esetet lefedni. Például, ha szeretnénk *alice* számára írási és olvasási engedélyeket adni egy olyan fájlra, amelynek ő nem a tulajdonosa, és nincs benne a tulajdonos csoportban sem (és ezeken nem is tudunk vagy nem is szeretnénk változtatni), az egyetlen lehetőségünk, hogy *mindenki más*nak adunk olvasási és írási jogosultságot. Ez viszont egy szerencsétlen megoldás, amely kerülendő volna, hiszen ezzel potenciálisan sok más felhasználónak is adnánk engedélyeket, szükségtelenül.

Az ehhez hasonló helyzetekben szolgálhatnak segítségünkre az úgynevezett POSIX ACL-ek. Ezek használatával lehetséges a hagyományos engedélyek megtartása mellett többek között külön jogosultságokat biztosítani az egyes

felhasználóknak és csoportoknak. Az ACL-eket viszont ennél részletesebben nem tárgyaljuk a laborgyakorlat keretein belül.

3. Feladatok

3.1. Vezetett rész

A labor elején megismerkedünk a Linux rendszerek felhasználó-, csoport- és jogosultságkezelési lehetőségeivel, valamint az ezekhez használatos alapvető parancsokkal, azok hatásaival.

Részletes leírás

1. A kerettörténet: egy kis cégnek szüksége van egy fájlszerverre, amire a céges dokumentumokat tudják majd menteni (és ahonnan a többiek el tudják majd érni azokat). Ebből most csak a felhasználókat, csoportokat, és a fájlrendszer szintű jogosultságokat (és még ezeknek is csak egy részét) készítjük most el, állítjuk most be.
2. Nézzük meg a /etc/passwd tartalmát (*cat /etc/passwd*), nézzük át, mi található a fájlban, mit jelentenek az egyes mezők.
3. Nézzük meg a /etc/shadow fájl tartalmát (*sudo less /etc/shadow*). Röviden tekintsük át a mezők felépítését, jelentését.
4. Vegyünk fel egy új felhasználót *alice* usernévvel és tetszőleges jelszóval. A többi adatot most hagyjuk üresen. (*adduser alice*, jelszó kétszer, utána sok enter.)
5. Nézzük meg a /etc/passwd-ben és a /etc/shadow-ban megjelenő egy-egy új sort, amely igazolja, hogy létrejött a felhasználó.
6. Nézzük meg a /etc/group tartalmát (*cat /etc/group*), és gonduljuk át, mit jelentenek az egyes mezők.
7. Vegyünk fel egy új csoportot *finance* néven. (*addgroup finance*)
8. Nézzük meg ismét a /etc/group fájlt. Mi változott? (Válasz: utolsó sorban új felhasználó.)
9. Adjuk hozzá *alice* felhasználót a *finance* csoporthoz. (*adduser alice finance*)
10. Nézzük meg ismét a /etc/group fájlt. Mi változott? (Válasz: utolsó sor módosult, bekerült az új user a csoportba.)

11. Hozzunk létre egy új könyvtárat a /var-ban, *fileshare* néven. (*mkdir /var/fileshare*).
12. Hozzunk létre egy új fájlt a /var/fileshare-ben, *testfile* néven. (*touch /var/fileshare/testfile*).
13. Lépjünk át a /var/fileshare könyvtárba (*cd /var/fileshare*).
14. Adjuk ki az *ls -la* parancsot. Gondoljunk vissza a jogosultságokról, tulajdonosokról tanultakra.
15. Adjuk ki a *chmod 660 testfile* és *chmod 777 .* parancsokat (direkt van itt egy pont, ez jelenti az aktuális könyvtárat). Gondoljuk át, mit jelentenek ezek.
16. Adjuk ki az *ls -la* parancsot ismét. Mi változott? (Válasz: jogosultságok...)
17. Nyissunk egy másik terminált. Ott adjuk ki a *su - alice* parancsot. Ezzel felvesszük alice identitását és jogosultságait. Lépjünk vele is át a /var/fileshare-be.
18. Alice-ként próbáljuk meg olvasni a /var/fileshare/testfile-t (*cat /var/fileshare/testfile*). Nem fog sikerülni.
19. Rootként/cloudként adjunk olvasási jogot a fájlra *mindenki más*nak (*chmod o+r /var/fileshare/testfile*).
20. Alice-ként próbáljuk meg olvasni a /var/fileshare/testfile-t (*cat /var/fileshare/testfile*). Most sikerülni fog.
21. Alice-ként próbáljuk meg írni a /var/fileshare/testfile-t (pl. *nano /var/fileshare/testfile*). Nem fog sikerülni.
    Azt itt megjelenő hibaüzenet a válasz a Moodle-ben beadandó, vezetett részre vonatkozó kérdésre.
22. Alice-ként próbáljuk meg törölni a /var/fileshare/testfile-t (*rm /var/fileshare/testfile*). Sikerülni fog (mert a könyvtárra felette van írási jogunk, és itt az számít).

3.2. Önálló rész

Az önálló részben folytatjuk a felhasználók, csoportok és jogosultságaik beállítását. Mivel a feladatok többsége egymásra épül, erősen ajánlott sorban haladni.

3.2.1. 1. feladat – Új felhasználó felvétele

Hozz létre *bob* számára is egy felhasználót a rendszerben. Mi lett a felhasználóazonosítója (UID)?

3.2.2. 2. feladat – Felhasználó csoportba felvétele

Vedd fel *bob* felhasználót a korábban létrehozott *finance* csoportba! Milyen csoportoknak tagja most *bob*? Használd az *id* parancsot a megfelelő paraméterezéssel! Beadandó a parancs kimenete.

3.2.3. 3. feladat – Csoport vizsgálata

Kik most a *finance* csoport tagjai? Mellékeld az állítását igazoló sort a megfelelő fájlból!

3.2.4. 4. feladat – Könyvtár felvétele

Készíts egy új könyvtárat a pénzügyeseknek */var/fileshare/finance* néven. Milyen jogosultságokkal jött létre a könyvtár? A választ egy háromjegyű számként add meg!

3.2.5. 5. feladat – Könyvtár tulajdonosának beállítása

Állítsd be, hogy a */var/fileshare/finance* könyvtár tulajdonos csoportja a *finance* legyen. Melyik parancsot használtad ehhez? (A paramétereket is add meg!)

3.2.6. 6. feladat – Könyvtár jogosultságainak beállítása

Állítsd be, hogy a */var/fileshare/finance* könyvtárra a tulajdonosa minden, a tulajdonos csoportja minden, viszont senki más semmi jogosultsággal ne rendelkezzen. Hogyan néz ki most a könyvtár jogosultsági leírója? A választ egy háromjegyű számként add meg!

3.2.7. 7. feladat – Könyvtár jogosultságainak beállítása II.

Szeretnénk, hogy amennyiben a */var/fileshare/finance* könyvtárban fájlokat vagy mappákat hoznak létre, úgy azoknak a tulajdonos csoportja mindig a fi-

nance legyen, s nem pedig a létrehozó elsődleges csoportja. Milyen paranccsal oldhatjuk ezt meg a legegyszerűbben? A paramétereket is add meg!

3.2.8. 8. feladat – Felhasználó adatainak megváltoztatása

Alice felhasználójának létrehozásakor a nagy sietségben elfelejtettük megadni a teljes nevét (Alice Presley). Pótoljuk most ezt a hiányosságot! A megfelelő parancs lefuttatása után add meg a felhasználót reprezentáló sort a felhasználókat tároló rendszerfájlból!

3.3. Szorgalmi feladatok

Az itt található feladatok azon hallgatóknak készültek, akik szeretnének az előzőekhez képest nagyobb kihívásokkal is megküzdeni. **Ezen feladatok megoldása (illetve meg nem oldása) nem befolyásolja a laborra kapott jegyet, így mindenképpen először az önálló feladatokkal érdemes foglalkozni.** A szorgalmi feladatok a félév során bármikor beadhatók, azokat bármennyiszer meg lehet próbálni beadni.

3.3.1. 1. szorgalmi – Mindenki a sajátját...

Hozz létre egy új könyvtárat */var/fileshare/meetingnotes* néven. Ebben a könyvtárban az egyes meetingeken készült feljegyzéseket szeretnénk tárolni. A feljegyzésekhez mindenki férjen hozzá, szerkeszteni is lehessen őket (bárki bárkiét), viszont törölni csak a sajátunkat lehessen. Hogyan néz ki a könyvtár betűvel leírt jogosultsági listája?

3.3.2. 2. szorgalmi – Mennyi? Harminc...

Szeretnénk megtudni, hány felhasználó létezik a rendszerben (beleértve a szolgáltatási és rendszerfiókokat is). A későbbiekben minden reggel szeretnénk a felhasználók számát lekérni és feljegyezni, így a feladat megoldásához ne a *majd kézzel megszámolom* módszert alkalmazza! (Segítség: a feladat megoldásához nem szükséges programot vagy scriptet írni.)

3.3.3. 3. szorgalmi – I am root!

Hozz létre egy új fájlt *prog1.cpp* néven, majd másold bele az alábbi sorokat!

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main()
{
    setuid(0);
    system("id");
    return 0;
}
```

Használd a g++ fordítót az imént létrehozott forráskód lefordítására. (Segítség: *g++ prog1.cpp*.) Az így keletkező *a.out* állományt tedd futtathatóvá. Futtasd le rootként és egy tetszőleges nem-root felhasználóval is. Amennyiben nem root a tulajdonosa az állománynak, úgy mostantól legyen a root a tulajdonos. Állítsd be, hogy mindig a tulajdonos nevében fusson az alkalmazás, függetlenül attól, hogy ki indítja el. Futtasd le újból rootként és egy nem-root felhasználóként is. Mi a különbség a két-két futtatás között?

Milyen háttérszínnel jeleníti meg az *ls -la* parancs az állományt?

3.3.4. 4. szorgalmi – POSIX ACL-ek

Olvass utána a POSIX ACL-ek működésének, kezelésének.

Hozzunk létre egy új felhasználót *charlie* néven. Ő lesz a pénzügyesek főnöke. Ő nem lesz tagja a *finance* csoportnak, de mégis szükséges volna neki teljes hozzáférést adni a */var/fileshare/finance* könyvtárhoz, anélkül, hogy bárki más többletjogosultságokat szerezne. Ügyelj arra, hogy az újonnan létrejövő könyvtárakhoz és fájlokhoz is legyen *charlie*-nak hozzáférése!

Az ACL felvétele után milyen új (eddig nem látott) karakter jelenik meg az *ls -la* kimenetében a */var/fileshare/finance* könyvtárra vonatkozóan?