4. Laborgyakorlat

Linux fájlrendszerek.

Előző gyakorlaton, már volt szó a fájlrendszerekről, mikor a mount parancs -t kapcsolójáról volt szó. Linux alatt, az egyes fájlokhoz való hozzáférések miatt, a fájlokhoz tulajdonost, csoportot és a hozzájuk tartozó jogosultságokat rendelünk.
Az ls -l parancs használatakor, a kapott lista a következő információkat írja ki a fájlokról:

1. oszlop fájlok típusa, jogosultságok
2. oszlop ún. inode szám
3. oszlop fájl tulajdonosa
4. oszlop fájl csoportja
5. oszlop fájl mérete
6, 7. oszlop utolsó módosítás ideje
8. oszlop fájl neve

A fájlokról ezeket az adatokat, a fájlrendszer tárolja. Számunkra az 1, 3, 4. oszlopok lesznek az érdekesek.

chmod, chown, chgrp

A jogosultságok a fájl tulajdonosára, csoportjára, és mindenki másra vonatkoznak:

fájltípus	jogok		
-	tulajdonosi	csoport	más felhasználók
	rwx	r-x	r-x
	111	101	101
	7	5	5
	olvasás, írás, futtatás	olvasás, futtatás	olvasás, futtatás

Fájltípusok például a következőek lehetnek:

- -: reguláris fájl (teljesen egyszerűbináris vagy szöveges állomány)
- d: könyvtár típusú állomány
- c: karakteres típusú eszközfájl (konzol is ilyen például: /dev/tty)
- b: blokk típusú eszközfájl (winchesterek: /dev/hda)
- l: link típusú fájl

Jogok fájlok esetén:

- r: olvasási jog
- w: írási jog
- x: futtatási jog

Készítette: Sinkó Gergely

Jogok könyvtárak esetén:

- r: olvasási
- w: írási
- x: könyvtár-hozzáférési

A fájlok jogait a legegyszerűbb 3 oktális szám segítségével megadni, mint az alábbi példákon is fogjuk látni:

chmod 600 /tmp/saját_fájlom.txt
chmod 640 /tmp/csoportolvashatja
chmod 660 /tmp/csoportirhatjais

Lehetőségünk van könyvtárakban, alkönyvtárakban lévő fájlok jogainak rekurzív módosítására:

chmod -R 644 /tmp/probakonyvtar

Ezzel ugyanakkor óvatosan kell bánnunk, hiszen ha rekurzívan változtatunk egy olyan könyvtárat melyben alkönyvtár van, akkor az alkönyvtár jogait is módosítja!
A tulajdonost, és csoportot is van lehetőségünk módosítani. Tulajdonost a chown paranccsal, ezt csak a root tud, míg csoportot, - a chgrp paranccsal - a felhasználó is, ha tagja annak a csoportnak melyhez hozza szeretné rendelni a fájlt. Lehetőségünk van egyszerre megváltoztatni a tulajdonost és a csoportot megfelelő jogosultságok esetén a chown parancs segítségével.

Feladat

1. Lépjen be root felhasználóként az első terminálon, majd diak felhasználóként a második terminálon.
2. Lépjen be mindkét terminálon a /tmp könyvtárba!
   (`cd /tmp`)
3. Hozza létre root-ként a /tmp/proba.txt fájlt!
   (`touch /tmp/proba.txt`)
4. Rendelje hozzá root felhasználóként, a létrehozott /tmp/proba.txt fájlt a diak felhasználóhoz!
   (`chown diak /tmp/proba.txt`)
5. Rendelje hozzá root felhasználóként, a /tmp/proba.txt fájlt, a diak csoporthoz!
   (`chgrp diak /tmp/proba.txt`)
6. Hozza létre root felhasználóként, a /tmp/proba2.txt fájlt!
   (`touch /tmp/proba2.txt`)
7. Rendelje hozzá root felhasználóként, a létrehozott /tmp/proba2.txt fájlt a diak felhasználóhoz és csoporthoz egy paranccsal!
   (`chown diak.diak /tmp/proba2.txt`)
8. Hozzon létre egy proba3.txt nevű fájlt root felhasználóként, a /tmp könyvtárban!
   (`touch /tmp/proba3.txt`)
9. A /tmp/proba3.txt jogait állítsa be úgy, hogy a tulajdonos és a csoport tudja írni olvasni, a többi felhasználó csak olvasni tudja!
   (`chmod 664 /tmp/proba3.txt`)
10. Hozza létre a /tmp/proba/elso könyvtárstruktúrát root felhasználóként!
    (`mkdir -p /tmp/proba/elso`)
11. A /tmp/proba könyvtár jogait állítsa be úgy, hogy a tulajdonos tudja írni, olvasni, böngészni, a csoport tudja böngészni és olvasni, a többi felhasználó csak böngészni tudja!
    (`chmod 751 /tmp/proba`)

Készítette: Sinkó Gergely

12. Váltson át a második terminálra -ahol diak felhasználóként jelentkezett be -, majd lépjen be a /tmp/proba könyvtárba!
    (`cd /tmp/proba`)
13. Szintén diak felhasználóként, listázza ki a könyvtár tartalmát! Mit tapasztal?
    (`ls`)

Ha mindent jól csinált, akkor a rendszer hozzáférés megtagadva hibaüzenettel válaszol.

14. Váltson vissza az 1. terminálra, majd root felhasználóként hozzon létre egy fájlt, file néven, a /tmp/proba könyvtárban „nem ures” tartalommal!
    (`echo 'nem ures' > /tmp/proba/file`)
15. Most próbálja meg cat paranccsal megnézni diak felhasználóként a /tmp/proba/file tartalmát! Mit tapasztal?
    (`cat /tmp/proba/file`)
16. Törölje le a /tmp könyvtárból, a proba kezdetű fájlokat és könyvtárakat egy paranccsal! (`rm -rf /tmp/proba*`)

A könyvtáron, az olvasási jog hiánya, nem befolyásolja, a könyvtárban lévő fájlokhoz való hozzáférést.

A vim megismerése

A vim szövegszerkesztő a régi vi szövegszerkesztő „modernebb” utódja. Használata nehézkes, de nem lehetetlen. A vim vagy normál vagy szerkesztő üzemmódban dolgozik. Azt, hogy épp melyikben vagyunk egyértelműen nem mindig lehet eldönteni, viszont majd minden esetben az <ESC> billentyű használatával a normál módba jutunk.
Normál módban van lehetőségünk kilépni, menteni, kijelölni szövegrészletet, másolni ezeket. Ebből a módból többféle szerkesztő módba juthatunk. Általában a kiegészítő (insert) módot használjuk, melybe a normál módból az <i> billentyű lenyomásával juthatunk. Az insert módban van lehetőségünk új szöveget begépelni, a <BACKSPACE> segítségével törölni – a <DEL> tapasztalataim szerint problémákat okozhat(!). Lehetőségünk van felülíró szerkesztő módba lépni (ez egyébként a régi vi alapértelmezett szerkesztő módja). Két felülíró mód van. Ha csak egy karaktert szeretnénk cserélni, akkor normál módban az <r> billentyű lenyomásával juthatunk ebbe a módba. Itt a cserélni kívánt karakterre állunk, a kurzormozgató billentyűkkel, majd a kívánt karakter beírásával cseréljük a „hibás” karaktert. Ha normál módban a <SHIFT>+<r> billentyű kombinációval váltunk felülíró módba, akkor addig tudunk karaktereket cserélni, amíg ki nem lépünk ebből a módból.
Minden szerkesztő módnál igaz, hogy <ESC> hatására normál módba jutunk.
A dokumentum mentése normál módban történik, a <:> <w> karaktereket beírva (ez alul megjelenik :w -ként), majd <enter>. A :w után megadhatunk fájl nevet is, akkor azon a néven menti el a vi a szerkesztett fájlt. A :q kilépésre szolgál. Akkor tudunk kilépni hibaüzenetek nélkül, ha nem szerekesztettük a fájlt, vagy kilépés előtt mentettük a változtatásokat vagy ha kilépéskor mentjük a változtatásokat a :wq paranccsal. Ha szerkesztettük a fájlt, de nem kívánjuk menteni a változásokat, a :q! utasítást kell használnunk.
Szükségünk lehet arra, hogy egyes sorokat karaktereket töröljünk. Ehhez normál módra kell váltanunk, majd a törölni kívánt karakter fölé mozgatjuk a kurzort, és az <x> segítségével törlünk. Ha egész sorokat szeretnénk törölni, a :<sorszám>d utasítást kell használnunk szintén normál módban.

Készítette: Sinkó Gergely

Feladat

Lépjen be a rendeszerbe diak felhasználóként!
Lépjen be a /tmp könyvtárba!
Indítsa el a vi-t, váltson insert módba, majd írjon bele több sort!
Váltson normál módra, mentse a fájlt /tmp/proba.txt néven!
Váltson felülíró módba, és írja felül a begépelt szöveg egy részletét!
Lépjen normál módba és töröljön karaktereket, sorokat!
A lépjen ki, hogy a változásokat is elmentse!

gcc

C programok fordítására és ellenőrzésére is van lehetőségünk, unix/linux rendszerek alatt. Mivel mi Linuxszal foglalkozunk, ezért az itt elterjedt gnu C compilert (gcc) és GNU debugger (gdb).
A gcc használata igen egyszerű, ha csak egy C programocskát szeretnénk lefordítani. Szintaxisa:

gcc -o ilyennevreforditsd ezt_a_C_progarmot.c

Az ilyennevreforditsd nevű fájl a fordítás végén, egy futtatható állomány lesz, és a létrehozandó fájl nevének minden esetben a -o után kell állnia. Ha nem adunk meg -o val fájl nevet, az alapértelmezett név az a.out. Az ezt_a_C_programot.c fájlnak .c, vagy .C kiterjesztésűnek kell lennie.

Feladat

1. Lépjen be a /tmp könyvtárba
2. Hozza létre a proba.c fájlt vi segítségével, a következő tartalommal:

```
#include <stdio.h>

int main(void) {
        printf(„nevem: .....\n”);
        printf(„neptun kodom: ....\n”);
        return 1;
}
```

3. Fordítsa le a forrást, majd indítsa el!

```
( gcc -o proba proba.c
./proba )
```

4. Írja be a következő C programot, proba2.c fájlba vi segítségével!

```
#include <stdio.h>

int main(void) {
        printf(„1. sor...\n”);
        printf(„2. sor...\n”);
        printf(„3. sor...\n”);
        printf(„4. sor...\n”);
        printf(„5. sor...\n”);
        printf(„6. sor...\n”);
```


Készítette: Sinkó Gergely

```
        printf(„7. sor...\n”);
        printf(„8. sor...\n”);
        printf(„9. sor...\n”);
        printf(„10. sor.\n”);
        return 1;
}
```

5. Fordítsa le a programot, hogy debugolható (nyom követhető) legyen!
(`gcc -o masodik -g proba2.c`)

A gdb nyomkövető használata

A gnu debugger, a programjainkban való hibakeresést hivatott segíteni. Ehhez a programunkat a már fent említett -g kapcsolóval kell fordítani. Ezzel a kapcsolóval a programunk forrása is része lesz a programnak. A hibakereséshez, a következő parancsot kell kiadnunk:

gdb masodik

A gdb egy shellt fog nekünk adni, amelyben utasításokat adunk ki, melyekkel vezérelhetjük a programunk futását. A help parancs tájékoztat minket, hogy milyen lehetőségeink vannak. Mi a running részben lévő parancsokat fogjuk alkalmazni, ezért a help running fog segíteni.

Az általunk leggyakrabban használt parancsok:

- **r:** program futtatása
- **b <szám>:** brakepoint (megszakítási pont) kijelölése ahol a <szám> a forráskód sorának száma
- **d <brakepoint szám>:** brakepoint törlése
- **l:** progam forráskódjának kiíratása
- **c:** a megszakított futás folytatása
- **n:** egy lépés végrehajtása
- **q:** kilépés

Feladat

1. Indítsa el a masodik nevű programot gdb segítségével! (`gdb masodik`)
2. Állítson be töréspontot a 8. sorra! (`b 8`)
3. Indítsa el a programot! (`r`)
4. A töréspont után léptesse egyesével a programot! (`n amíg le ne fut a program`)
5. Ha lefutott a program, törölje a töréspontot! (`d 1`)
6. Lépjen ki! (`q`)

Készítette: Sinkó Gergely