find

1. A find parancs

A `find` parancs segítségével állományokat tudunk keresni a fájlrendszerben.

1. A `find` parancs szintaxisa (a `man`-ban is nézzünk utána):

   ```
   find keresesi_utvonalak kifejezes
   ```

   A `find` parancs a megadott útvonalak mindegyikét rekurzív módon bejárja. Ha egy érintett fájlra teljesül a kifejezés, akkor ezen a fájlon végre tudunk hajtani egy műveletet. Vegyük például a következő parancsot:

   ```
   find $HOME -name '*.txt' -print
   ```

   Vagyis: a saját HOME könyvtárunkban (s ennek minden egyes alkönyvtárában) nyomtassunk ki (`print`) minden `txt` kiterjesztésű fájlt.

   Vegyük észre, hogy a `find` parancs a fájlok *teljes elérési útvonalát* írja ki.

2. Próbáljuk ki a következő parancsokat:

   ```
   find $HOME
   find $HOME -name hello.sh -print
   find $HOME -name '*.txt' -print
   ```

   Hány darab szöveges állomány van a saját könyvtárunkban?

3. A `-o` operátor jelentése: logikai `vagy`.

   ```
   find $HOME -name '*.txt' -o -name 'd*' -print
   ```

   Vigyázat! Ebben a példában a `print` művelet csak a második feltételhez van hozzárendelve. Továbbá a logikai kifejezések rövidzár módon kerülnek kiértékelésre (a C nyelvhez hasonlóan). Például a `proba.txt` fájl megfelel az első feltételnek, így az ehhez tartozó művelet fog végrehajtódni, ami viszont üres. Vagyis ezen fájl nem fog megjelenni a kimenetben. Ha mind a `*.txt`, mind pedig a `d*` fájlok neveit ki szeretnénk íratni, akkor két dolgot tehetünk:

   - Vagy minden feltételhez megadunk egy műveletet:

     ```
     find $HOME -name '*.txt' -print -o -name 'd*' -print
     ```

   - Vagy pedig összefogjuk a feltételeket. Egyszerűbb ezt használni ha a művelet ugyanaz:

     ```
     find $HOME \( -name '*.txt' -o -name 'd*' \) -print
     ```

     A `(` és `)` karakterek speciális jelentéssel rendelkeznek a shell számára. Ahhoz, hogy ezeket a karaktereket a `find` parancs kapja meg, eléjük kell tenni egy backslash-t. Ennek hatására a shell nem fogja speciális karakterekként kezelni őket.

     Figyeljünk arra, hogy a `\(` és `\)` jelek előtt és után legyen egy-egy szóköz, különben hibaüzenetet fogunk kapni.

4. Hány darab C forrásfájl található a saját könyvtárunkban?

   ---

5. Hány darab `*.o` fájl található a HOME könyvtárunkban?

   ---

6. A `print`-en kívül természetesen más műveleteket is végre lehet hajtani, például:

   ```
   find $HOME -name '*.txt' -exec ls -l {} \;
   ```

   Ennek jelentése:

   - Egyetlen Unix parancs végrehajtását kérjük (`exec`), mely jelen esetben az `ls -l` (részletes információ egy fájlról). Ezt a parancsot az `sh` shell hajtja végre, vagyis a bash-ben használt alias-aink itt nem fognak látszani. A parancs továbbá nem tartalmazhatja sem a pipe „`|`”, sem pedig a parancsok elválasztására szolgáló „`;`” jeleket.
   - A kapcsos zárójelek `{}` a `find` által éppen feldolgozott fájlt jelentik. Vagyis az aktuális fájlt ennek a segítségével lehet átadni paraméterként a végrehajtandó Unix parancsnak.
   - Az `exec` által végrehajtott parancsot egy „`;`”-vel kell lezárni. Mivel ez is a `find`-hoz tartozik s nem a shell-hez, ezért ez elé is backslash-t kell tenni.
   - A fájlnevek megadásakor aposztróf helyett idézőjel is használható. Emlékezzünk: az aposztrófok között a shell minden speciális karaktert ignorál, míg idézőjelek között minden speciális karaktert ignorál *kivéve* a `$` (dollár), ‘ (backtick, „visszafele aposztróf”), és \ (backslash) jeleket.

7. Írjunk egy `keres` nevű shell-szkriptet, mely a HOME könyvtárunkban megkeresi mindazon fájlokat / könyvtárakat, melyek neve tartalmazza a szkriptnek paraméterként átadott részsztringet.

   Egy kis segítség: a szkriptünkben az első paraméter értéke a `$1` változóban lesz. A szkript eleje a következőképpen néz ki:

   ```
   #!/usr/bin/bash

   echo $1    # irjuk at ezt a sort
   ```

8. Ha csak egy egyszerű keresést akarunk végrehajtani, akkor a `find` kimenete kombinálható a `grep`-pel is. Például keressük meg a HOME könyvtárunkban a `hello.sh` nevű állományt:

   ```
   find $HOME | grep hello.sh
   ```

9. Keressük meg a HOME könyvtárunkban az összes `txt` kiterjesztésű állományt. Használjuk a `find / grep` kombinációt.

   ---

10. Keressük meg, hogy mely felhasználók rendelkeznek `.bashrc` állománnyal. Haladjunk lépésről-lépésre:

    - A felhasználók saját könyvtárai a `/home` könyvtárból nyílnak. Vagyis a rekurzív keresést innen kell indítani.

    - Elképzelhető, hogy egyes felhasználók levédték a könyvtáraikat, nincs rá olvasási jogunk. A `find` által kiírt `Permission denied` hibaüzeneteket rejtsük el.

    - Arra vagyunk kíváncsiak, hogy *mely felhasználók* rendelkeznek a megadott fájllal. A `find` kimenetéből válasszuk ki a felhasználóneveket, s írassuk ki őket ismétlődés nélkül.

    - Hány felhasználónak van `.bashrc` fájlja, magunkat is beleértve?

      ---

11. Írassuk ki a HOME könyvtárunkban mindazon fájlokat (csak a fájlokat), melyekre rajtunk kívül más is rendelkezik írási joggal a csoportunkból (group).

    - Először listázzuk ki a HOME könyvtárunkban lévő fájlokat. A `find` paranccsal ki lehet szűrni a fájlokat, erre a `-type f` feltétel szolgál. Vagyis: `find $HOME -type f`. (A könyvtárakat ehhez hasonlóan, a `-type d` feltétellel lehetne kiszűrni).

    - Az előző listát bővítsük ki úgy, hogy minden egyes fájlról (a rejtett fájlokról is) írassunk ki részletes információt. Tipp: szeretnénk látni a fájlok hozzáférési jogait is.

    - Ha ez megvan, akkor szűrjük ki azon fájlokat, melyekre rajtunk kívül más is rendelkezik írási joggal a csoportunkból (group). Példa: a `-rwxrw----` jogokkal rendelkező fájl például megfelel ennek a feltételnek.

      ---

12. Nagyon sok szövegszerkesztő a szerkesztésre megnyitott állományról biztonsági másolatot készít (a vim is ilyen). Ezen fájlok nevéhez egy `~` jel kerül, pl. `hello.sh~`. Vajon hány ilyen fájllal van „teleszemetelve" a HOME könyvtárunk?

    Ha a `find` segítségével szeretnénk ezeket a fájlokat letörölni, akkor érdemes az `rm -i` parancsot használni, melynek hatására az `rm` rákérdez minden egyes fájlra, hogy valóban le akarjuk-e törölni. Ne feledjük, az `exec` utáni parancsot az `sh` shell futtatja, s ez nem látja a beállított alias-ainkat. Mindig figyeljünk arra, hogy nehogy valamit véletlenül letöröljünk.

    Egy másik biztonsági módszer: ha a `find`-dal mindenképpen törölni akarunk, akkor előbb másoljunk át néhány tesztállományt a `/tmp` könyvtár egy alkönyvtárába s a törlést ezen teszteljük le. Ha olyan állományok is törlődtek, amiket meg kellett volna hagyni, akkor még finomíthatunk a parancson, szerencsére csak egy biztonsági másolaton dolgoztunk.