Bash cheatsheet*

További olvasgatásra: http://tldp.org/guides.html.

- `echo`: kiírja az argumentumokat

Dokumentáció

- `man`: az argumentumként megadott parancs dokumentációja
- `apropos`: keresés a dokumentációk között

Fájlmegjelenítés

- `cat`: az argumentumokként megadott fájlok tartalmát kiírja
- `less`: az argumentumban megadott fájl tartalmát kiírja, görgethető
  - kurzormozgató billentyűk működnek
- `file`: fájltípus információ
- `diff`: két argumentumában megadott fájl közötti különbség
- `hexdump`: (főleg bináris) fájlok tartalmának hexadecimális formában való kiírása
  - javaslom a `-C` kapcsolót

Szövegszerkesztő

- `nano`
- elszántaknak: `vi`, `emacs`
- hibátlanoknak: `echo`, `cat`

Fájl- és könyvtárműveletek

- `pwd`: kiírja az aktuális könyvtárat
- `cd`: változtatja az aktuális könyvtárat

*Utoljára módosítva: 2019. november 7.

  - abszolút vs. relatív elérési út
  - `~`
- `ls`: kilistázza a könyvtárban található fájlokat és alkönyvtárakat
  - `-a`: minden fájlt (rejtett: `.`-al kezdődőeket is)
  - `-l`: részletes
  - `--color=auto` vagy `--color=none`: színezés
  - `-h`: emberi mértékegységek
  - `-lha`: kombó
- `rm`: az argumentumként megadott fájlok törlése
  - `-r`: könyvtárat is töröl rekurzív módon (alkönyvtárat, alalkönyvtárat, stb.)
- `rmdir`: az argumentumként megadott (üres) könyvtárak törlése
- `mkdir`: az argumentumként megadott könyvtárak létrehozása
- `cp`: fájlok és könyvtárak másolása
  - argumentumok: mit mivé
    - ha a „mivé” könyvtár, akkor „hová” lesz
  - `-r`: rekurzívan másolja a könyvtárakat is
- `mv`: fájlok és könyvtárak áthelyezése/átnevezése
  - mint a `cp`, de alapértelmezetten rekurzív
- `find`: keresés a fájlrendszerben
  - első argumentum: hol keressen
  - utána tesztek
    - `-name`, `-iname`: fájl neve (`iname` esetén a kis/nagybetű nem számít legyen a következő argumentum
      - pl.: `-name notes.txt`
      - `?`: joker-karakter, bármilyen karaktert jelöl
      - `*`: joker-karakter, 0 vagy több bármilyen karaktert jelöl
    - ha egy teszt sincs, minden fájlt és könyvtárat kilistáz
  - `-exec`: a következő argumentumokat (`';'`-ig) parancsként értelmezi, és végrehajtja minden találatnál
    - `'{}'`: behelyettesíti a találat nevét
    - `';'`: a parancs végét jelzi, utána már ismét a `find` argumentumait lehet írogatni
    - pl.: `-exec echo '{}' ':' ';' -exec echo "==========" ';' -exec cat '{}' ';'`
- `ln`: linket hoz létre
  - argumentumok: link célja és (opcionálisan) link neve
  - `-s`: soft linket hoz létre
- `mc`: Midnight Commander

Szövegmanipuláció

- `egrep`: kiszűri a sorokat, amelyekben nem található meg a *minta*
  - első argumentum: minta, ezt keresi
  - többi argumentum: fájlok, ahol keres
    - ha nincs, akkor a bemenetén keres
  - `-i`: kis/nagybetű nem számit
  - `-v`: megfordítja a keresést (kiszűri a sorokat, amelyekben ott van a minta)
  - `-c`: sorok kiírása helyett a találatok számát írja ki
  - Minta:
    - `.` Bármilyen karakter, kivéve újsor
    - `\w \d \s` betű, szám, whitespace
    - `\W \D \S` nem betű, nem szám, nem whitespace
    - `[abc]` a, vagy b, vagy c
    - `[^abc]` nem a és nem b és nem c
    - `[a-g]` karakter a és g között
    - `^ $` szó eleje, szó vége
    - `\.  \*` speciális karakterek jelentésének megvonása (escaped)
    - `(abc)` csoportosítás
    - `a* a+ a?` 0 vagy több a, 1 vagy több a, 0 vagy 1 a
    - `a{5} a{2,} a{2,5}` pontosan 5 a, 2-nél több a, 2 és 5 közötti a
    - `ab|cd` ab vagy cd
- `head`: a bemenete első sorait írja ki
  - `-n`: hány sort írjon ki, pl.: `head -n 11`
- `tail`: a bemenete utolsó sorait írja ki, lásd `head`
- `sort`: a bemenetének sorait sorbarendezi
- `uniq`: az egymás után jövő duplikátum-sorokat kiszedi
- `wc`: megszámolja a bemenetének sorait, szavait és byte-jait
- `cut`: minden sorból csak egy részt ír ki
  - `-d`: az oszlopelválasztó karaktert adja meg, pl.: `cut -d ' ' -f 1`
  - `-f`: adott sorszámú oszlopo(ka)t tartja meg, pl.: `cut -d ' ' -f 1`
    - `2`: második oszlopot
    - `2-4`: második oszloptól a negyedikig
    - `3-`: harmadik oszloptól a sor végéig
    - `-3`: első három oszlopot
  - `-b`: kiválaszt adott byte-okat
    - `2`: második byte-ot
    - `2-4`: második byte-tól a negyedikig
    - `3-`: harmadik byte-tól a sor végéig
    - `-3`: első három byte-ot
- `tr`: karaktereket átalakít
  - pl.: `tr a b` átalakítja az `a`-kat `b`-vé
  - `-d` törli a karaktereket
    - pl.: `tr -d '\n'` törli a sorok végét, tehát összefűzi a sorokat

- `iconv`: karakterkódolás megváltoztatása
- haladó: `sed`, `awk`

Ezeknél meg lehet adni egy fájlnevet, akkor annak tartalmán működnek, ennek hiányában a standard bemenetükön.

Jogosultságok

- `chown`: tulajdonos változtatás
  - első argumentum: a tulajdonos felhasználó és csoport kettősponttal elválasztva
  - többi argumentum: a fájlok nevei
  - `-R`: rekurzív
- `chmod`: jogosultságok változtatása
  - első argumentum: a felhasználói csoport és a megfelelő jogosultságok, `+`-al vagy `-`-al elválasztva (rendre a jogosultság megadását vagy elvételét jelezve)
    - felhasználói csoport lehet:
      - `a`: mindenki
      - `u`: tulajdonos
      - `g`: csoport
      - `o`: mindenki más
    - jogosultság lehet:
      - `r`: olvasás
      - `w`: írás
      - `x`: végrehajtás (vagy keresés könyvtárak esetén)
    - vagy meg lehet adni egy oktális számmal is, lásd a man oldalt
  - többi argumentum: a fájlok nevei
  - `-R`: rekurzív

Adminisztráció

- `ps`
  - `ps auxf`, és a kapcsolók bármely részhalmaza
- interaktív: `top`, `htop`
- `who`: ki van még bejelentkezve
- `kill`: folyamat leállítása

IO redirection

- Egy programnak alapértelmezetten van három IO csatlakozása:
  - standard input
  - standard output
  - standard error

- Ezeket át lehet irányítani fájlba/ból:
  - stdin: `<`, pl.: `cat <input.txt`
  - stdout: `>`, pl.: `cat >output.txt`
  - stderr: `2>`, pl.: `cd /nonexistent_directory 2>error.txt`
  - kombó: `cat <input.txt >output.txt 2>error.txt`
  - hozzáfűzés: `>>`
- `<<`: heredoc
  - a `<<` után szereplő szóig várja a bemenetet, utána az lesz a program bemenete
  - pl.:
    ```
    cat >new_file.txt <<EOF
    first line
    second line
    EOF
    ```

Titkosítás

- `gpg`

Shell script

- egyszerű szövegfájl, amelyben shell utasítások vannak, sorvégével, vagy `;`-vel elválasztva
- futtatás:
  - `bash my_shell.sh`
  - vagy közvetlenül:
    - a script első sora: ***#!/bin/bash***
    - a script legyen futtatható (`chmod u+x my_script.sh`)
    - futtatás[1]: `./my_script.sh`
    - vagy: **`source`** `my_script.sh`
      - az `export`-tal deklarált változók megmaradnak a hívó shellben is
- változók
  - nincsenek típusok, minden szöveges
  - szintakszis[2]:
    - `MY_VARIABLE=my_value`
    - `MY_MULTIWORD_VAR="one two three"`
    - scripten kívül, értelmezőben: **`export`** `MY_VARIABLE=my_value`
  - `env`: változók felsorolása
  - használat: **`echo`** `${MY_VARIABLE} "${MY_MULTIWORD_VAR}"`
  - speciális változók
    - `PATH`: könyvtárak `:`-al elválasztott listája, itt keresi a rendszerparancsokat

[1] a `./` nélkül a rendszer parancsai között keresné, nem az aktuális könyvtárban
[2] nincs szóköz az `=` körül

    * `PS1`: promptot változtatja meg, pl.: `PS1='\nI am waiting for your command.\n\u:\h \$'`
- command-substitution: `head -n $(echo "2+3*4"| bc)`
- alias: `alias ll='ls -lah --color=auto'`
- `~/.bashrc`: minden shell indításkor lefut, ide érdemes rakni az aliasokat, és a környezeti változókat

Vezérlő szerkezetek

- Felhasználói input: `read user_inp`: beolvas a standard inputról egy sort, és eltárolja a `user_inp` nevű változóban
- Elágazás[3]:

```
if [[ "a" == "b" ]]
then
  echo "Well, it seems so."
else
  echo 'No way!'
fi
```

  - változók behelyettesítése és command-substitution működik
  - használható logikai/összehasonlító műveletek[4]:
    * szöveges összehasonlítás: `==`, `!=`, `>`, `<`
    * egész szám összehasonlítás: `-eq`, `-ne`, `-lt`, `-lt`, `-gt`, `-ge`
    * logikai műveletek: `&&`, `||`, `!`
- For-ciklus[5]:

```
for i in egy ketto harom
do
  echo ${i}
done
```

  - számsorozat: `for i in range{2..42013}`
- While-ciklus:

```
while [[ "a" != "b" ]]
do
  echo "Still not equal."
done
```

  - mint az elágazás
  - explicit végtelen ciklus: `while true; do`

[3] figyelj a szóközökre a szögletes zárójelek között
[4] további irodalom: http://tldp.org/LDP/abs/html/comparison-ops.html
[5] itt pont, hogy nincsenek idézőjelben a felsorolt elemek, csak ha az egyik elem többszavas; a szóközök (whitespace-ek) alapján választja el egymástól őket

– **`while read line; do`**: a standard inputról (átirányítható, lásd IO redirection) sorokat olvas be, és belerakja a `line` nevű változóba; minden iterációban egy-egy sort