# Parancssor használata

Kovács Kristóf, Magyar András, Simon András

BME TTK Matematika Intézet

2022. december 6.

# Mi várható?

A cél az, hogy használni tudják az intézeti számítástechnikai környezetet (ami lényegében Linux).

- `Username: XXX`
  `Passwd: XXX`
- `http://wiki.math.bme.hu/view/Informatika1-2021/Gyakorlat1`
  Rengeteg gyakorlati tanács, pl. hogyan jelentkezzünk be otthonról, hogyan férhetünk hozzá az intézeti wi-fi hálózathoz.

Részlet egy Linux file-rendszerből:

```
/usr
    /usr/bin
    /usr/games
    /usr/include
        /usr/include/arpa
        /usr/include/asm
        /usr/include/asm-generic
        /usr/include/atk-1.0
            /usr/include/atk-1.0/atk
        /usr/include/at-spi-2.0
            /usr/include/at-spi-2.0/atspi
        /usr/include/at-spi2-atk
            /usr/include/at-spi2-atk/2.0
        /usr/include/bits
            /usr/include/bits/platform
            /usr/include/bits/types
        /usr/include/blkid
        /usr/include/brotli
         ...
```

**pwd** aktuális könyvtár (amire .-al is lehet hivatkozni) nevének kiíratása

**ls** könyvtár tartalmának listázása. Pl.:

```
$ ls
```

vagy

```
$ ls /dev
```

vagy

```
$ ls -hl /usr/bin
```

**cd** könyvtárváltás. Ha paraméter nélkül hívjuk meg, akkor a saját („home”) könyvtárunkba visz. Pl.:

```
$ cd /mnt
```

vagy

```
$ cd
```

vagy

```
$ cd ..
```

**pushd/popd** `pushd somedir` olyan, mint `cd somedir`, kivéve, hogy utána `popd` visszavisz minket bárhol is voltunk; ezek egymásba ágyazhatók

**mkdir** könyvtár létrehozása. Pl.:

```
$ mkdir newdir
```

**cp** file(ok) másolása. Pl.:

```
$ cp mit.txt hova.txt
```

vagy

```
$ cp mit.txt hova/
```

Itt `hova/`, ami egy directory neve, lehet `.` vagy `..` is. `cp -r` rekurzívan másol, így lehet egy teljes directory-strukturát másolni

mv mozgatás/átnevezés (az eredeti nem marad meg). Pl.:

```
$ mv mit.txt hova.txt
```

vagy

```
$ mv mit.txt hova/
```

rm törlés, Pl.:

```
$ rm alma.txt
```

Rekurzívan töröl mindent a könyvtárral együtt (tehát *veszélyes*):

```
$ rm -r somedir
```

quota A felhasznált/megmaradt helyet írja ki. Ha a kvótánk betelik, akkor nem kapjuk meg a nekünk címzett leveleket és a grafikus terminálra sem tudunk belépni; ilyenkor a parancssoros terminálra belépve (Ctrl-Alt-F5) tudjuk kitakarítani a home-unkat.

df, du lemezen szabad / felhasznált helyet írja ki. A `-h` kapcsolóval olvasható GB, MB, kB értékeket kapunk, de lehet fix byte-okban is kiírni. Pl.:

```
$ df -h
```

filekezelők `mc`, emacs dired módja, ...

**tab completion** enélkül mozdulni sem lehet: ha nem tudjuk, mit akarunk, nyomjuk meg a Tab billentyűt! Kiegészíti a file-neveket, parancsok neveit, néha még a parancsok paramétereinek neveit is.

**history** a fel/le kurzornyilakkal tudunk közlekedni a korábban kiadott parancsok között; kereshetünk is közöttük visszafelé `Ctrl-r`-el.

**job control** `Ctrl-c`-vel általában le tudjuk lőni a parancssorból indított programot; `Ctrl-z`-vel felfüggeszteni, aztán `bg`-vel a háttérbe helyezni (ahol tovább fut), `fg`-al ismét az előtérbe. Eleve a háttérben indíthatunk egy programot a neve (és esetleges argumentumai) után írt `&` karakterrel. Utóbbiaknak hosszan futó, nem interaktív programoknál van értelme, bár sokszor egyszerűbb egy új terminál-ablakot nyitni.

**cat** `cat` az argumentuma (egy file neve) tartalmát a terminálra másolja. Ez alkalmassá teszi (rövid) szövegfile-ok megnézésére, de nem ez a legfőbb haszna; pl. file-okat lehet vele összefűzni[1] (lesz rá példa rövidesen)

**less** szövegfile-ok megnézése; Pl.:

```
$ less mit.txt
```

Ideális esetben a fel/le kurzornyilakkal, `PgUp`/`PgDn`-nal tudunk közlekedni.

**tail** egy szövegfile végét lehet vele megnézni; a `-f` („follow”) kapcsolóval folyamatosan lehet nézni, hogy mi kerül a file végére. Ez akkor érdekes, ha egy program időnként ír egy új sort a fileba, mi pedig szeretnénk követni az eseményeket.

[1] a neve is ebből (conCATenate) jön

Példa az előzőre:

```
$ for i in {1..10}; do echo $i » /tmp/logfile; sleep 10; done &
$ tail -f /tmp/logfile
```

wc statisztika szövegfile-okról: byte-okat (`wc -m` karaktereket, ami sokkal hasznosabb), szavakat, sorokat számol; alapértelmezésben mindhármat, de pl.
`$ wc -l szoveg.txt`
csak a `szoveg.txt` sorait

grep minta szerint szűri egy szövegfile sorait. Pl.
`$ grep torta szoveg.txt`
a `szoveg.txt`-beli, „torta”-t tartalmazó sorokat adja vissza. (Ha arra vagyunk kíváncsiak, hogy hány ilyen van, akkor
`$ grep torta szoveg.txt | wc -l`
Rövidesen látni fogjuk, hogy ez hogy működik.) A `grep` első argumentuma *reguláris kifejezés* is lehet.

> kimenet átirányítása egy file-ba. Pl.:
> `$ ls -l > ittalista`
> az aktuális directory tartalmát az `ittalista` nevű file-ba teszi (és felülírja, ami addig esetleg ott volt). Pl.
> $ cat file1 file2 > file12
> után file12-ben file1 és file2 tartalma lesz egymás után fűzve

» mint >, de nem írja felül, hanem a végére biggyeszti

| `parancs1 | parancs2` az első parancs outputja második inputja lesz. Így működik a fenti `grep torta szoveg.txt | wc -l`. A `grep` kimenete a „torta” szót tartalmazó sorok, `wc -l` ezeket számolja meg. Más példa:
| `$ ls -l | less`.

Egy további példa innen:
https://datascienceatthecommandline.com/2e/index.html

```
$ curl -s "https://www.gutenberg.org/files/11/11-0.txt" |
  grep " CHAPTER"
 CHAPTER I.      Down the Rabbit-Hole
 CHAPTER II.     The Pool of Tears
 CHAPTER III.    A Caucus-Race and a Long Tale
 CHAPTER IV.     The Rabbit Sends in a Little Bill
 CHAPTER V.      Advice from a Caterpillar
 CHAPTER VI.     Pig and Pepper
 CHAPTER VII.    A Mad Tea-Party
 CHAPTER VIII.   The Queen's Croquet-Ground
 CHAPTER IX.     The Mock Turtle's Story
 CHAPTER X.      The Lobster Quadrille
 CHAPTER XI.     Who Stole the Tarts?
 CHAPTER XII.    Alice's Evidence
```