Previous - Up - Next

# Tartalomjegyzék



# Bash

Some useful links:

- https://www.codecademy.com/learn/learn-the-command-line
- http://ss64.com/bash/

## Basic commands

- **pwd**: path to the current folder.
- **cd**: change directory
- **ls**: list the contents of the directory: **ls**, *ls -h*, **ls /dev**, **ls -h -l ~**
- **mkdir**: create folder: **mkdir NewFolder**
- **cp**: copy: **cp what.txt where.txt**
- **mv**: move: **mv what.txt where.txt**
- **rm**: remove: **rm alma.txt**, with recursion (delete a whole folder): **rm -r NewFolder**
- **quota**: Shows the currently used storage space and our maximum allowed storage. If this fills up, we won't be able to login to the graphical linux screen. But we can still login to the command line linux screen to clean out our home folder.
- **df**, **du**: disk space used, with the **-h** parameter it will be made more readable: **df -h ~**
- **mc**: midnight commander explorer.
- help yourself
  - **info <parancs>**
  - **man <parancs>** shows detailed help for the given command, navigation with the arrow keys, quit with **q** search with **/** : **man ls**
  - **<command> --help**
  - **help**
- **history**: history of issued commands.
- **apropos**: search for commands: *apropos math*
- **top**: shows running applications.
- **kill**, **killall**, **xkill**: stop the execution of a given or all programs, stopping all programs for a given user: **killall -u username**
- **cat**: prints the contents of the given file: **cat .forward**
- **echo**: prints the argument: **echo Hello World**
- **grep**: searching with regular expressions
- **less**: a smarter form of cat, easier to navigate larger text files: **less something.txt**
- **head** (-n ), **tail** (-n ): print the beginning and the end of the given file: **head -n 100 something.txt**
- **sort** (-g -k): sorting text.

- **wc** (-l): counting bytes/lines/characters.
- **ssh**: secure connection to a remote machine: ssh username@leibniz.math.bme.hu
- **scp**: secure copy from (or to) a remote machine scp from.txt username@leibniz.math.bme.hu:~/to.txt
- **wget**: download files from the internet
- **exit**: exit the terminal (or ctrl+d).
- **sudo**: execute the command as a superuser: **sudo rm /home/someone_I_hate/important_file.txt**
- **halt, reboot**: shuts down, reboots

## Pipe, redirect

Every program (command) can print to the console, or to `stderr` this is what we see on the console. These can be redirected into a file:

```
$ ls -l ~
drwxr-xr-x 8 borbely student 4096 Aug 30 23:24 Desktop
drwxr-xr-x 2 borbely student 4096 Mar 27 2012 Downloads
drwxr-xr-x 2 borbely student 4096 Oct 20 2009 Drives
drwx------ 2 borbely student 4096 Apr 20 10:42 mail
drwxr-xr-x 7 borbely student 4096 Sep 6 13:01 public_html
$ ls -l ~ > folder.txt
$ cat folders.txt
drwxr-xr-x 8 borbely student 4096 Aug 30 23:24 Desktop
drwxr-xr-x 2 borbely student 4096 Mar 27 2012 Downloads
drwxr-xr-x 2 borbely student 4096 Oct 20 2009 Drives
drwx------ 2 borbely student 4096 Apr 20 10:42 mail
drwxr-xr-x 7 borbely student 4096 Sep 6 13:01 public_html
$ _
```

The symbol **>** redirects the output of a command into a file, after this the contents of the file can be printed using `cat`-el. However:

```
$ ls -l /home/algebra/wettl/ > myfile.txt
ls: cannot open directory /home/algebra/wettl/: Permission denied
$ cat myfile.txt
$ _
```

The file **myfile.txt** remains empty, and we get an error message on the console (we didn't have permissions to list the files in the specified folder). This is the case, because the command **ls** did not produce an output onto the stardard output, but to the standard error (output) **stderr**. **stderr** can be redirected using the **2>** symbol:

```
$ ls -l /home/algebra/wettl/ > myfile.txt 2> error.log
$ cat myfile.txt
$ cat error.log
ls: cannot open directory /home/algebra/wettl/: Permission denied
$ _
```

This way the output of the command (**stdout**) and the error messages of the command (**stderr**) can be redirected separately.

Certain programs can read input from the standard input: **stdin**. There are so called non-interactive commands, where we don't need to do any input after issuing a command (most commands we discussed are like this). But there are some commands that can read input after being executed:

- non-interactive command: `grep "http://wiki.math.bme.huneedle"http://wiki.math.bme.hu haystack.txt`
- interactive command: `grep "http://wiki.math.bme.huneedle"http://wiki.math.bme.hu`

On the former command **grep** searches for the keyword *needle* in the given file. On the latter we need to provide input in which the command will search for the word.

```
$ grep "needle"
I hate this
why can't I find anything
why can't I find a needle
why can't I find a needle
oh, a needle
oh, a needle
again
$ _
```

We can stop the input with Ctrl+D. The lines containing *needle* were repeated, every repeated line was the output of grep (it found the word).

Try it like this:

```
$ grep "needle" > needle.txt
I hate this
why can't I find anything
why can't I find a needle
oh, a needle
again
$ cat needle.txt
why can't I find a needle
oh, a needle
$ _
```

With this only the lines containing *needle* were included in the file.

The output of a command can also be made to be the input of another using the **pipe** symbol: `|`.

Example: Print the files in our home directory that contain *info* in their name!

```
$ ls -1 ~
Desktop
Downloads
Drives
info1
info_hf.txt
mail
myfile.txt
myfolder
needle.txt
public_html
regi_info
regi_info_zh.txt
$ ls -1 ~ | grep "http://wiki.math.bme.huinfo"http://wiki.math.bme.hu
info1
info_hf.txt
regi_info
```

```
regi_info_zh.txt
$ _
```

Example: we have a lot of files in this folder, but we would like to look through them:

```
$ ls /home/student/ | wc -l
821
$ ls /home/student/ | less
```

We can also pipe the contents of a file into a command (**<**), these two lines produce the same output:

```
$ cat nevek.txt | sort
$ sort < nevek.txt
```

### Tasks

- Download the text file `williamblake.txt` from `http://math.bme.hu/~kkovacs/info1/williamblake.txt`.
    - `wget`
- Count how many characters it has.
    - `wc`
- Find the lines containing *is*.
    - `grep`
- Count the lines containing *and*.
    - `grep, wc, pipe`
- Write to a file named **the.txt** all the lines containing the word *the*.

## Windows

The terminal of windows DOS didn't go through much improvement since its creation. It's far behind the linux shell.

- Starting the command prompt
    - Start menu -> Command prompt
    - Start menü -> Run -> cmd
    - Search -> cmd
- description
- useful: `help`, `help <command>`
- On Windows 10 new features for the command prompt.
- Since the one year anniversary of Windows 10 bash is included.

Another usable windows shell: PowerShell

### Differences from linux

- While linux is case-sensitive, windows is not.
- Folder- and filenames:
    - `C:\Windows\System32\`
    - `/usr/bin/`
- command options use / instead of -:
    - Windows: `dir /b`
    - Unix: `ls -l`
- script

        - Windows-on: `.bat`
        - Linux-on: `.sh`
    - pipe, redirect is similar

Useful if you want to use linux bash on windows:

- cygwin
- mingw

Previous - Up - Next