# Command line

Computers:

- launch programs
- store data
- communicate with other computers
- communicate with people

Most often graphical interfaces (GUI) are used to control personal computers. In order to increase the speed, accuracy of communication and most importantly to automate the control we use command line interfaces (CLI).

## Launch terminal

- Press start, search for terminal and open it.
- Terminal starts in the home directory. Users home directory is /home/username. Universal shortcut for home directory is `~` . We can see that by executing command:

In [ ]:
```
pwd
```

## Navigation

- To see the contents of current directory execute:

In [ ]:
```
ls
```

- Files starting with a dot `.` are hidden in Linux. To see hidden files add additional argument to ls command (flag)

In [ ]:
```
ls -a
```

- To see other files in other directories (for example ~/Documents)

In [ ]:
```
ls -a ~/Documents
```

- Press TAB when writing directory names (or other commands) for autocompletion
- To get more information on the ls command check documentation

In [ ]:
```
man ls
```

- Documentation is opened with vi/less programs and navigation is controlled with hjkl keys, search /, exit with q

---

> **TASK**
>
> - Find and describe options: l h G g B t
> - Write command that would sort files by their size

---

- By default terminal opens in home directory. Change directory to Documents directory

In [ ]:
```
cd ~/Documents
```

---

> **TASK**
>
> - test commands, navigate around home directory

In [ ]:
```
cd ~/Documents/       # go to Documents direcotry
cd /                  # go to root directory
cd ..                 # go one directory up
cd .                  # current directory
cd ../../             # go two diretories up (relative path)
cd -                  # go to previuos location
ls                    # view contents of directory
pwd                   # get path of current directory
```

## Create and remove

In [ ]:
```
touch filename          # create empty file named filename
rm filename             # remove file named filename
mkdir dirName           # create directory named dirName
rm -rf dirName          # remove direcotry recursively
cp -r folderName place  # copy folder to place
mv file newfile         # move file to other place or rename it
```

**TASK**

- go to Documents directory
- create your personal directory
- test file and folder creation commands

- using bash programming language it is easy to create whole directory tree

In [ ]:
```
cd ~/Documents
mkdir -p Biodata/{lectures,labwork/{images,code/{py,jpynb,md,sh,m,cpp,config}},
data{xlsx,txt,csv,mat,img}}
```

**TASK**

- Change previous command to create separate folders for all data types

## View

In [ ]:
```
ls            # view contents of home directory
cat filename  # view contents of filename
less filename # view contents of filename in less
head filename # view first 5 rows of file
tail filename # view last 5 rows of file
```

**TASK**

- go to your folder and copy .bashrc file from home directory to your new directory
- examine the file with cat, less, head and tail commands

## Search

- Find sequence of letters `the` in file test.txt

```
In [ ]: grep the test.txt
```

- Line number of `the` occurrence shows option -n, case insensitivity -i

```
In [ ]: grep -n -i the test.txt
```

---

> **TASK**
>
> - find WORD `the` in test.txt

---

- find all files in current directory

```
In [ ]: find .
```

- find all directories

```
In [ ]: find . -type d
```

- find files with specific name elements (find all pdfs)

```
In [ ]: find . -name '*.pdf'
```

## Wild cards

- asterisk (*) and question mark (?) symbols allow for quick filtering in many commands
- asterisk goes instead of 0 or more symbols. So the command bellow will find all files that end with .pdf

```
In [ ]: find . -name '*.pdf'
```

- ? means any 1 symbol. So the command bellow will find task.pdf or ttask.pdf but will not find ask.pdf

In [ ]:
```
find . -name '?ask.pdf'
```

## Pipe

- Pipe | transfers one command output as text to other command standard input
- `command > filename` writes command output to new file named 'filename'
- `command >> filename` appends command output to file named 'filename'
- command bellow `curl` will download contents of the website pipe them to command `wc` (word count) to count the line statistics and write `wc` results to new file called linenumbers.txt

In [ ]:
```
curl -s https://docs.python.org/3/ | wc -l >> linenumbers.txt
```

- command2 $(command1) will catch output of command1 and give it to command2 as its arguments

---

**TASK**

What is the difference between these two lines:

In [ ]:
```
wc -l $(find . -name '*.txt')
```

```
find . -name '*.txt' | wc -l
```

## Writing scripts

- All executed commands are stored in history file ~/.bashrc_history
- To get last 5 commands execute command `history` and to create a scrip write the output to *.sh file

In [ ]:
```
history | tail -n 5 > scriptFile.sh
```

- To exclude command from history enter space before it
- Review the script, check for errors and run it using `bash` command

In [ ]:
```
bash scriptFile.sh
```

### Configuration

- Depending on unique work-flow people use same commands very often. To automate and to type less shortcuts can be created (aliases).
- ~/.bashrc file contains commands that run every time terminal starts.

---

**TASK**

- add alias to bashrc file, restart terminal and try new shortcut `l`
- what does it do?

```
alias l='ls -GghBd */'
```

- .bashrc file can be edited in many ways:

  `vim ~/.bashrc`

  `nano ~/.bashrc`

```bash gedit ~/.bashrc

or directly in terminal by appending command to file

echo "alias l='ls -GghBd */'" >> ~/.bashrc
```

---

# Most useful commands summary

## System

In [ ]:
```
date         # Show date and time
uptime       # Display system uptime
cal          # Show calendar
w            # Display who is logged in
whoami       # Display effective username
uname -a     # Show kernel info
man cmd      # Show man page for cmd
df           # Display free disk space
du           # Display disk usage stats
free         # Show memory and swap usage
whereis app # Show where app location is
which app    # Show which app
```

## Directories

In [ ]:
```
ls -l           # List current dir contents (long format)
ls              # List current dir contents
ls -a           # List current dir contents including hidden
ls -t           # List current dir contents sorted by modification date
cd              # Change to home dir
cd dir          # Change to directory 'dir'
pwd             # Show current directory
mkdir dir       # Make directory 'dir'
rm -r dir       # Remove directory 'dir'
rm -rf dir      # Remove directory 'dir' (force)
cp -r dir1 dir2 # Copy 'dir1' to 'dir2'
cd -            # Change to previous working dir
```

## Files

In [ ]:
```
rm file           # Remove 'file'
cp file1 file2    # Copy 'file1' to 'file2'
mv file1 file2    # Rename or move file1 to file2
ln -s file1 link1 # Create symbolic 'link1' to 'file1'
touch file        # Create or update 'file'
head file         # Output first 10 lines of 'file'
tail file         # Output last 10 lines of 'file'
```

## Search

In [ ]:
```
grep pattern files  # Search for 'pattern' in 'files'
grep -r pattern dir # Search recursively for 'pattern' in dir
cmd | grep pattern  # Search for 'pattern' in output of cmd
locate file         # Find file names quickly
```

## Terminal shortcuts

ctrl+c # Halt current command ctrl+z # Background current command ctrl+d # Delete char infront of cursor or logout ctrl+u # Erase line ctrl+r # Search recent commands ctrl+a # Move to beginning of line ctrl+e # Move to end of line ctrl+h # Delete char behind cursor (backspace) UP # Move to previous command DOWN # Move to next command

## Processes

In [ ]:
```
ps           # Display your active processes
top          # Display all processes
kill 5       # Terminate process id of 5
kill -9 5    # Terminate (KILL) process id of 5
killall proc # Terminate all processes named 'proc'
bg           # List background jobs
fg           # Bring most recent job to foreground
fg 2         # Bring job 2 to foreground
```

## Command History

In [ ]:
```
!!      # Repeat last command
sudo !! # Repeat last command as root
UP      # Move to previous command
DOWN    # Move to next command
!3      # Execute command 3 in history
history # Show command history
```

# Compression

In [ ]:
```
tar cf file.tar files     # Create a tar 'file.tar' with 'files'
tar xf file.tar           # Extract files from 'file.tar'
tar czf file.tar.gz files # Create tar with gzip compression
tar xzf file.tar.gz       # Extract files from file.tar.gz
gzip file                 # Compress 'file' with gzip
gzip -d file.gz           # Decompress file.gz
```

## Executing scripts

- For future work with python lets download Anaconda package using command line tools

In [ ]:
```
cd ~/Downloads/
curl -o Anaconda.sh https://repo.anaconda.com/archive/Anaconda3-2019.10-Linux-x
86_64.sh
```

- To run Anaconda or other programs we must give execution rights for this file to our user
- chmod edits permissions, add (+) to user (u) execution (x) rights to file Anaconda.sh

In [ ]:
```
cd ~/Downloads/
chmod u+x Anaconda.sh
```

- To run Anaconda or other program append a dot and slash `./` to the name in the command line
- Read installation prompt and when asked type `yes` to continue installation process

In [ ]:
```
./Anaconda3-2019.07-Linux-x86_64.sh
```

# Functions

- Since bash is a programming language we can construct and add to our bashrc function ex to detect the type of compression and extract it with.

In [ ]:

```
ex ()
{
  if [ -f $1 ] ; then
    case $1 in
      *.tar.bz2)   tar xjf $1   ;;
      *.tar.gz)    tar xzf $1   ;;
      *.bz2)       bunzip2 $1   ;;
      *.rar)       unrar x $1     ;;
      *.gz)        gunzip $1    ;;
      *.tar)       tar xf $1    ;;
      *.tbz2)      tar xjf $1   ;;
      *.tgz)       tar xzf $1   ;;
      *.zip)       unzip $1     ;;
      *.Z)         uncompress $1;;
      *.7z)        7z x $1      ;;
      *)           echo "'$1' cannot be extracted via ex()" ;;
    esac
  else
    echo "'$1' is not a valid file"
  fi
}
```