EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Interfakultäres Institut für Biomedizinische Informatik

Bioinformatics preparatory course
Basic Unix I

Léon Kuchenbecker and Sam Wein

Why bother?

- **The shell is extremely powerful**
  - Searching, organizing, transferring large numbers of files
  - Exploring and manipulating plain text files
  - Can be programmed

- **The shell can be executed efficiently remotely**
  - SSH (secure shell) interface to access a Unix-Like system remotely
  - Standard to interact with larger compute infrastructures (e.g. a compute server or cluster)

- **The majority of bioinformatics tools have a CLI**
  - We work with a high number of large files
  - Crunching that data requires high compute power → compute server, cluster
  - Most file formats established in Bioinformatics are plain text

Basic usage

- Enter a command and press the Return (⏎) key

```
vorkurs@vorkurs-vm:~$ ls ⏎
Desktop  Documents  Exercise
vorkurs@vorkurs-vm:~$
```

- To get help on how to use a particular command, try

```
vorkurs@vorkurs-vm:~$ ls --help ⏎
vorkurs@vorkurs-vm:~$ man ls ⏎
```

- The basic command structure often follows this structure:

```
COMMAND [OPTIONAL SWITCHES] [MANDATORY ARGUMENTS]
```

Basic usage

- Often there are long (--) and short (-) versions of switches

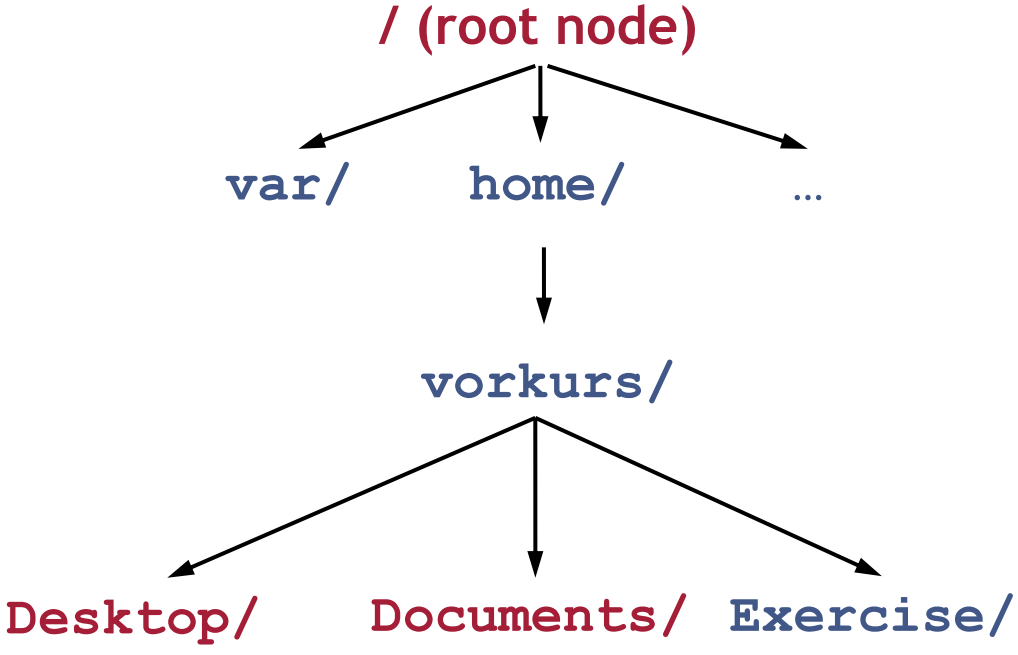
```
vorkurs@vorkurs-vm:~$ ls ⏎
Desktop  Documents  Exercise
vorkurs@vorkurs-vm:~$ ls -r ⏎
Exercise  Documents  Desktop
vorkurs@vorkurs-vm:~$ ls --reverse ⏎
Exercise  Documents  Desktop
```

- Multiple switches can be specified together

```
vorkurs@vorkurs-vm:~$ ls -F -r ⏎
Exercise/  Documents/  Desktop/
vorkurs@vorkurs-vm:~$ ls -Fr ⏎
Exercise/  Documents/  Desktop/
```

- **`pwd`** (***PrintWorkingDirectory***) prints the current working directory
- **`ls`** (***LiSt***) displays the contents of **folders** (**directories**)
  - **`ls␣-l`** show details
  - **`ls␣-lh`** human readable details
  - **`ls␣-a`** show hidden files and folders

- **`cd`** (***Change Directory***) changes the current directory
  - Absolute paths (**`cd␣/home/osboxes/somefolder`**)
  - Relative paths (**`cd␣somefolder`**)
  - Home folder (**`cd␣~/somefolder`**)
  - One Directory up, parent directory (**`cd␣..`**)


/ (root node)
var/
home/
...
vorkurs/
Desktop/
Documents/
Exercise/

- On Unix-like systems, every user has a personal home folder
  - The folder usually has the name of the user
  - Often, but not always, the folder is located under `/home`
  - The home folder of the user ‘joe’ could for example be `/home/joe`

- If you are using the virtual machine image, your username is `vorkurs` and your home directory is `/home/vorkurs`

- Path specifications starting with „/“ are *absolute*, otherwise they are *relative* to the current directory

```
vorkurs@vorkurs-vm:~$ ls Exercise/ ⏎
data
vorkurs@vorkurs-vm:~$ ls /home/vorkurs/Exercise/ ⏎
data
vorkurs@vorkurs-vm:~$ ls ../vorkurs/Exercise/ ⏎
data
vorkurs@vorkurs-vm:~$ ls ../vorkurs/Exercise/data/ ⏎
clinvar_20180225.vcf.gz
```

- `mkdir` (***MaKeDIRectory***) creates a new folder in the current working directory
- `touch` changes file access and modification times (creates empty file if it does not exist)

- `cp` (***CoPy***) copies files and folders (**overwrite cannot be undone**)
  - `-r` switch required to copy *directories*
- `mv` (***MoVe***) moves files and folders (**overwrite cannot be undone**)

- `rmdir` (***ReMoveDIRectory***) removes directory if empty
- `rm` (***ReMove***) deletes files and folders (**cannot be undone**)
  - `-r` switch required to delete *directories with all of their content*

→ **Think before you type! Use [-i] switch if you are unsure!**

Inspecting plain text files

- **`more`** page through text one page at a time
- **`less`** more powerful than more, bi-directional

- **`cat`** (***ConcAtenaTe***) concatenate files and print on the standard output

- **`head`** output the first part of files
- **`tail`** output the last part of files (`[-f]` continue as file grows)
  - **`tail -f filename`**

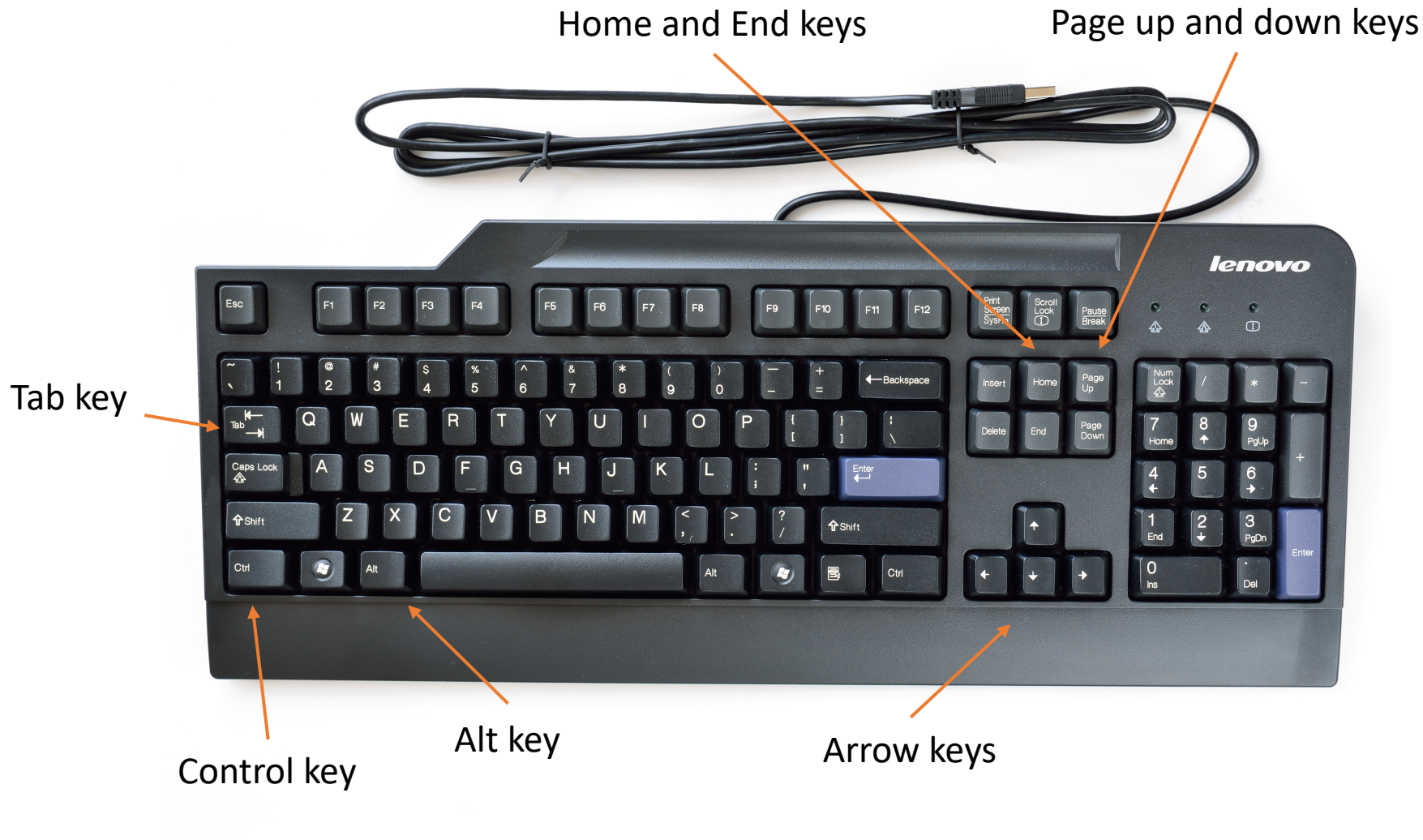
Editing plain text files

- A text editor is a program used to edit plain text files
- Standard text editors are Notepad on Microsoft Windows and TextEdit on Apple macOS
- nano is a text edit that does not require a graphical user interface but works on the command line.
  - `nano <path>`
    Opens the file specified by path
  - CTRL-O
    Saves the current file
  - CTRL-X
    Exit nano

- **`Tab`** Autocomplete on the command line
- **`Tab-Tab`** Show possible completions if not unique

- **`ALT-b / ALT-f`** Move one word (backwards, forwards)

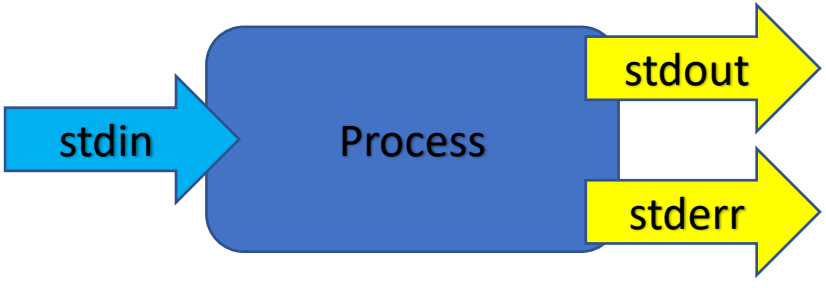
- **`Pos1/Home`** Move to start of line
- **`End`** Move to End of line



- **`Arrow Up`** Browse history backwards in time
- **`Arrow Down`** Browse history forward in time
- **`Ctrl-R`** Search history backwards in time
- **`Ctrl-C`** Interrupts current command

Standard Streams

- Every process has three input / output streams:
  - **Standard Input (stdin)**
  - **Standard Output (stdout)**
  - **Standard Error (stderr)**



- We have so far only worked with one of them: **stdout**

Standard Streams

- Normally, we cannot see the difference between **stdout** and **stderr**

- This is output produced on **stdout**

```
vorkurs@vorkurs-vm:~$ ls -l Exercise/data ⏎
total 13960
-rw-r--r-- 1 vorkurs vorkurs 14293917 Mar 27  2018 clinvar_20180225.vcf.gz
```

- This is output produced on **stderr**

```
vorkurs@vorkurs-vm:~$ ls -l doesnt.exist⏎
ls: cannot access 'doesnt.exist': No such file or directory
```

Standard Streams

- We can redirect the streams into files using the ‘>’ character

```
vorkurs@vorkurs-vm:~$ ls -l Exercise/data 1>stdout.txt 2>stderr.txt
```

```
vorkurs@vorkurs-vm:~$ ls -l doesnt.exist 1>stdout.txt 2>stderr.txt
```

- `1>file` redirects the **standard output** into *file*
- `2>file` redirects the **standard error** into *file*
- ‘>’ is short for ‘1>’

Try the two examples above and check the contents of the two output files each time!

Standard Streams

- The `>` redirection *overwrites* the target file!
- Use `>>` to *append* instead of *overwrite* the target file

- Create a folder named “repetition”
- Create a file in that folder named “repetition.txt”
- Write the text “Hello World” into that file
- Copy the file to a new file named “repetition2.txt”
- Duplicate the content of the file “repetition2.txt” and write it to the file “repetition3.txt”
- Print the content of the three files
- Delete the file “repetition.txt”
- Copy the folder “repetition” to the folder “repetition2” with all its contents
- Rename the folder “repetition” to “repetition_old”
- Remove the contents of the folder “repetition_old”
- Remove the folder repetition_old
- Remove the folder “repetition2”