

Bioinformatics Preparatory Course

Basic Unix

Léon Kuchenbecker and Nantia Leonidou

April 7, 2021



Attribution 4.0 International (CC BY 4.0)

Main Author

Léon Kuchenbecker

Additional Contributors

Alexander Seitz, Samuel Wein

Why bother with the shell?

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

Why bother with the shell?

- 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 remotely
 - SSH (secure shell) interface to access a Unix-like system remotely
 - Standard to interact with larger computer infrastructures (e.g. a compute server or cluster)

Why bother with the shell?

- 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 remotely
 - SSH (secure shell) interface to access a Unix-like system remotely
 - Standard to interact with larger computer 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 \Rightarrow compute server, cluster
 - Most file formats established in Bioinformatics are plain text

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

```
$ ls ↵
```

```
Desktop Documents Exercise
```

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

```
$ ls --help ↵
```

```
$ man ls ↵
```

- The basic command syntax often follows this structure:

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

Basic Usage

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

```
$ ls ↵
```

```
Desktop  Documents  Exercise
```

```
$ ls -r ↵
```

```
Exercise  Documents  Desktop
```

```
$ ls --reverse ↵
```

```
Exercise  Documents  Desktop
```

Basic Usage

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

```
$ ls ↵
```

```
Desktop  Documents  Exercise
```

```
$ ls -r ↵
```

```
Exercise  Documents  Desktop
```

```
$ ls --reverse ↵
```

```
Exercise  Documents  Desktop
```

- Multiple switches can be specified together

```
$ ls -F -r ↵
```

```
Exercise/  Documents/  Desktop/
```

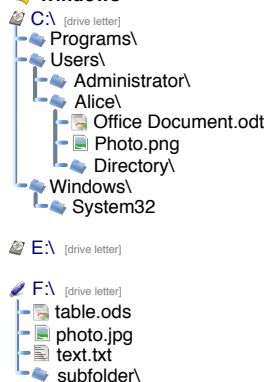
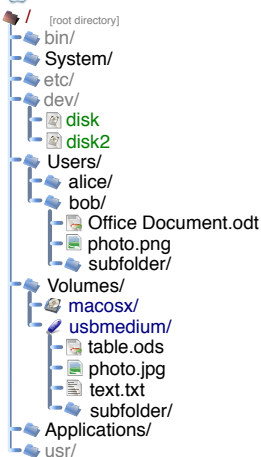
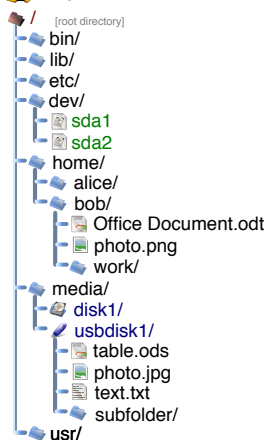
```
$ ls -Fr ↵
```

```
Exercise/  Documents/  Desktop/
```


File systems

File systems

A typical computer **file system** is structured like a **tree**

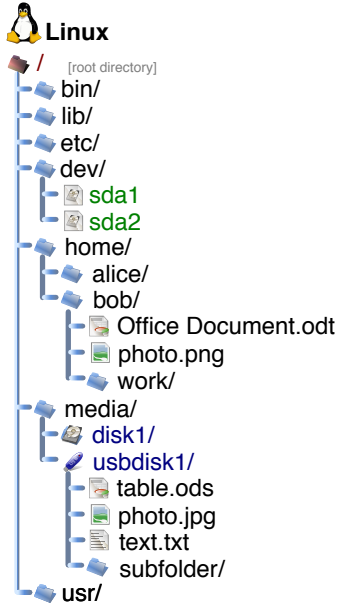


- Every running computer program, i.e. every process, including the [shell](#), refers to one directory inside the file system tree as its [working directory](#).

- Every running computer program, i.e. every process, including the [shell](#), refers to one directory inside the file system tree as its [working directory](#).
- When using the [shell](#), one often says “I am in the directory xzy”.

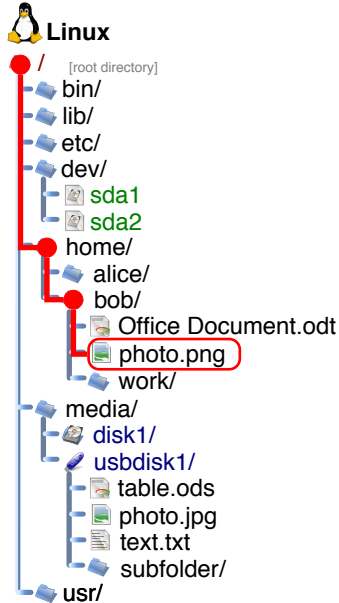
- Every running computer program, i.e. every process, including the **shell**, refers to one directory inside the file system tree as its **working directory**.
- When using the **shell**, one often says “I am in the directory xzy”.
- The **working directory** is not static, it can be changed throughout the runtime of a process.

File systems



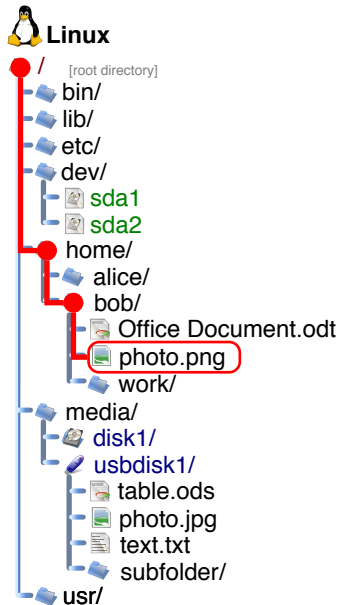
- Every file or directory in the file system can be described by a **path**

File systems



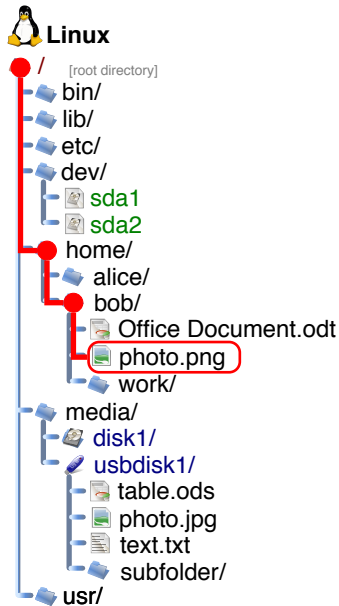
- Every file or directory in the file system can be described by a **path**
- **Absolute paths** start in the root directory, for example
`/home/bob/photo.png`

File systems



- Every file or directory in the file system can be described by a **path**
- **Absolute paths** start in the root directory, for example
`/home/bob/photo.png`
- **Relative paths** start in the current working directory, for example
`./bob/photo.png`
given that the current working directory is `/home`.

File systems



- Every file or directory in the file system can be described by a **path**
- **Absolute paths** start in the root directory, for example
`/home/bob/photo.png`
- **Relative paths** start in the current working directory, for example
`./bob/photo.png`
given that the current working directory is `/home`.
- Every character sequence that does not start with a slash `/` character is interpreted as a relative path.

Working with the file system

- **pwd** - print working directory
Prints the current working directory

Working with the file system

- **pwd** - print working directory
Prints the current working directory
- **ls** - list
Displays the contents of folders (directories)
 - **ls -l** show details
 - **ls -lh** human readable file sizes
 - **ls -a** show hidden files and folders

Working with the file system

- **pwd** - print working directory
Prints the current working directory
- **ls** - list
Displays the contents of folders (directories)
 - **ls -l** show details
 - **ls -lh** human readable file sizes
 - **ls -a** show hidden files and folders
- **cd** - change directory
Changes the current working directory

- `mkdir <path>` - make directory

Creates a new directory

Working with the file system

- **mkdir <path>** - make directory

Creates a new directory

- **cp <source path> <destination path>** - copy

Copies files and folders. **Overwriting targets cannot be undone!**

- **cp -r** - recursive mode

Required to copy directories with their contents.

Working with the file system

- **mkdir <path>** - make directory

Creates a new directory

- **cp <source path> <destination path>** - copy

Copies files and folders. **Overwriting targets cannot be undone!**

- **cp -r** - recursive mode

Required to copy directories with their contents.

- **mv <source path> <destination path>** - move

Move files and folders **Overwriting targets cannot be undone!**

Working with the file system

- **mkdir <path>** - make directory

Creates a new directory

- **cp <source path> <destination path>** - copy

Copies files and folders. **Overwriting targets cannot be undone!**

- **cp -r** - recursive mode

Required to copy directories with their contents.

- **mv <source path> <destination path>** - move

Move files and folders **Overwriting targets cannot be undone!**

- **rmdir <path>** - remove directory

Removes directories, but only if they are empty. Safe to use.

- `rm <path>` - remove

Remove files and folders. **Removing cannot be undone!**

- `rm -r` - recursive mode

Required to remove directories with their contents.

Getting started with plain text files

- **more <path>**

Page through text one page at a time

Inspecting plain text files

- **more <path>**

Page through text one page at a time

- **less <path>** - the opposite of more

More powerful than more, bidirectional, provides searching

Inspecting plain text files

- **more <path>**

Page through text one page at a time

- **less <path>** - the opposite of more

More powerful than more, bidirectional, provides searching

- **cat <path> [<path> ...]** - concatenate

Concatenates input files and prints them to the standard output (more on that later).

- **head <path>**

Output the first n lines of a plain text file.

Inspecting plain text files

- **head <path>**

Output the first n lines of a plain text file.

- **tail <path>**

Output the last n lines of a plain text file.

- **tail -f** - follow

Keeps printing new lines as the file grows.

- A **text editor** is a program used to edit plain text files

Creating and editing plain text files

- A [text editor](#) is a program used to edit plain text files
- Well known graphical editors are [Notepad](#) on Microsoft Windows and [TextEdit](#) on Apple macOS

Creating and editing plain text files

- A [text editor](#) is a program used to edit plain text files
- Well known graphical editors are [Notepad](#) on Microsoft Windows and [TextEdit](#) on Apple macOS
- [Nano](#) is a text editor that does not require a graphical user interface but works on the command line

- **nano <path>**

Opens the file specified by path. Can be used on non-existing paths to create new files.

Creating and editing plain text files

- **nano <path>**

Opens the file specified by path. Can be used on non-existing paths to create new files.

- **CTRL-O**

Saves the current file

Creating and editing plain text files

- **nano <path>**

Opens the file specified by path. Can be used on non-existing paths to create new files.

- **CTRL-O**

Saves the current file

- **CTRL-X**

Exits nano

Keyboard shortcuts

- **TAB**
Autocomplete the command line
- **TAB - TAB**
Show possible completions if not unique
- **ALT-b** and **ALT-f**
Move one word backward (or forward) in the current command line
- **Pos1** or **Home**
Move to the beginning of the current command line
- **End**
Move to the end of the current command line

Keyboard shortcuts



Keyboard shortcuts

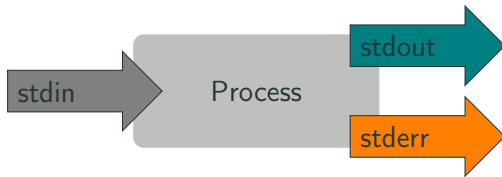
- **Arrow Up**
Browse the shell history backward in time
- **Arrow Down**
Browse the shell history forward in time
- **CTRL-R**
Search the shell history backward in time
- **CTRL-C**
Interrupt the current program.

Process streams

- Every **process** has three input / output streams
 - Standard input (stdin)
 - Standard output (stdout)
 - Standard error (stderr)

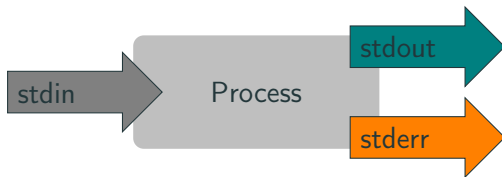
Process streams

- Every **process** has three input / output streams
 - Standard input (stdin)
 - Standard output (stdout)
 - Standard error (stderr)



Process streams

- Every **process** has three input / output streams
 - Standard input (stdin)
 - Standard output (stdout)
 - Standard error (stderr)



- We have so far **only worked with stdout**.

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

- Normally, we cannot see the difference between `stdout` and `stderr`
- This output is produced on `stdout`

```
$ ls -l Exercise/data ↵  
total 13960  
-rw-r--r-- 1 vorkurs vorkurs 14293917 Mar 27  
2018 clinvar_20180225.vcf.gz
```

Process streams

- Normally, we cannot see the difference between `stdout` and `stderr`
- This output is produced on `stdout`

```
$ ls -l Exercise/data ↵  
total 13960  
-rw-r--r-- 1 vorkurs vorkurs 14293917 Mar 27  
2018 clinvar_20180225.vcf.gz
```

- This output is produced on `stderr`

```
$ ls -l doesnt.exist ↵  
ls: cannot access 'doesnt.exist': No such file or directory
```

- We can redirect streams into files using the `>` character

```
$ ls -l Exercise/data 1>stdout.txt 2>stderr.txt ↵
```

```
$ ls -l doesnt.exist 1>stdout.txt 2>stderr.txt ↵
```


- We can redirect streams into files using the **>** character

```
$ ls -l Exercise/data 1>stdout.txt 2>stderr.txt ↵
```

```
$ ls -l doesnt.exist 1>stdout.txt 2>stderr.txt ↵
```

- **1>** **<path>** redirects the **standard output** into **<path>**

Process streams

- We can redirect streams into files using the **>** character

```
$ ls -l Exercise/data 1>stdout.txt 2>stderr.txt ↵
```

```
$ ls -l doesnt.exist 1>stdout.txt 2>stderr.txt ↵
```

- **1> <path>** redirects the **standard output** into **<path>**
- **2> <path>** redirects the **standard error** into **<path>**

Process streams

- We can redirect streams into files using the **>** character

```
$ ls -l Exercise/data 1>stdout.txt 2>stderr.txt ↵
```

```
$ ls -l doesnt.exist 1>stdout.txt 2>stderr.txt ↵
```

- **1> <path>** redirects the standard output into **<path>**
- **2> <path>** redirects the standard error into **<path>**
- **> <path>** is short for **1> <path>**

Process streams

- We can redirect streams into files using the **>** character

```
$ ls -l Exercise/data 1>stdout.txt 2>stderr.txt ↵
```

```
$ ls -l doesnt.exist 1>stdout.txt 2>stderr.txt ↵
```

- **1> <path>** redirects the **standard output** into **<path>**
- **2> <path>** redirects the **standard error** into **<path>**
- **> <path>** is short for **1> <path>**

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

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

Warm up

- 👉 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`

Users, groups and permissions

- File systems on Unix-like systems allow to configure access rights to users and groups of users

- File systems on Unix-like systems allow to configure access rights to users and groups of users
- A user can be a real person or just be an arbitrary logical entity

- File systems on Unix-like systems allow to configure access rights to users and groups of users
- A user can be a real person or just be an arbitrary logical entity
- Groups are sets of users

Users, groups and permissions

- File systems on Unix-like systems allow to configure access rights to users and groups of users
- A user can be a real person or just be an arbitrary logical entity
- Groups are sets of users
- Every file has a single **owner** and is also associated with a single **group**

Users, groups and permissions

```
$ ls -l Exercise/data ↵
```

```
total 13960
```

```
-rw-r--r-- 1 vorkurs vorkurs 14293917 Mar 27 13:42 clinvar_20180225.vcf
```

Group

The file owner

- A set of **permissions** is associated with each file which determine who has what type of access to the file

Users, groups and permissions

- A set of **permissions** is associated with each file which determine who has what type of access to the file
- There are three access types:
 - **read**
 - **write**
 - **execute**

Users, groups and permissions

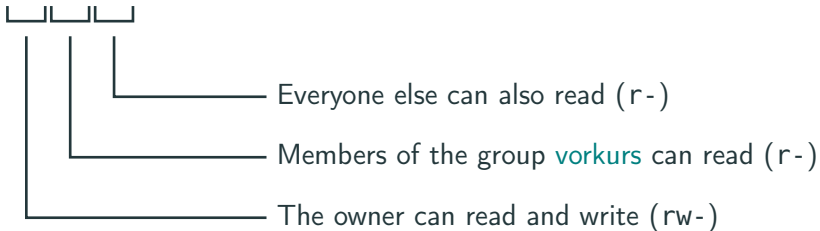
- A set of **permissions** is associated with each file which determine who has what type of access to the file
- There are three access types:
 - **read**
 - **write**
 - **execute**
- This triplet of permissions (**rwX**) is defined for three groups of users:
 - The owner
 - Members of the file's group
 - The rest of the world

Users, groups and permissions

```
$ ls -l Exercise/data ↵
```

total 13960

```
-rw-r--r-- 1 vorkurs vorkurs 14293917 Mar 27 13:42 clinvar_20180225.vcf
```



The interpretation of the permissions for **directories** is slightly different

- **read** determines whether the contents of a directory can be seen
- **write** determines whether files can be created or deleted
- **execute** determines whether a user can change (**cd**) into the directory

- **chown** [owner][:group] <path> - change ownership

Changes the ownership and / or group association of a file or directory. Only the **root** user is allowed to change the owner of a file.

Users, groups and permissions

- **chown** [**owner**][:**group**] **<path>** - change ownership

Changes the ownership and / or group association of a file or directory. Only the **root** user is allowed to change the owner of a file.

- **chmod** **<permissions>** **<path>** - change file mode bits, aka permissions

Changes the file permissions. Examples:

- **chmod g+w <path>**

Add write permissions to the group

- **chmod o-r <path>**

Remove read permissions for the rest of the world

- Shortcuts:

(u)ser, (g)roup, (o)thers, (a)ll at once

(r)ead, (w)rite, e(x)ecute

```
$ ls -l Exercise/data ↵
```

```
total 13960
```

```
-rw-r--r-- 1 vorkurs vorkurs 14293917 Mar 27 13:42 clinvar_20180225.vcf
```

Users, groups and permissions

```
$ ls -l Exercise/data ↵  
total 13960  
-rw-r--r-- 1 vorkurs vorkurs 14293917 Mar 27 13:42 clinvar_20180225.vcf
```

Add the [write](#) permission to everyone:

```
$ chmod a+w Exercise/data/clinvar_20180225.vcf.gz ↵  
$ ls -l Exercise/data ↵  
total 13960  
-rw-rw-rw- 1 vorkurs vorkurs 14293917 Mar 27 13:42 clinvar_20180225.vcf
```

- Permissions can also be specified numerically
 - 4: read (r)
 - 2: write (w)
 - 1: execute (x)

Users, groups and permissions

- Permissions can also be specified numerically
 - 4: read (r)
 - 2: write (w)
 - 1: execute (x)
- To combine permissions, the numbers are added

Users, groups and permissions

- Permissions can also be specified numerically
 - 4: read (r)
 - 2: write (w)
 - 1: execute (x)
- To combine permissions, the numbers are added
- Three digits specify the permissions for all three groups (owner, group, rest of the world) simultaneously:

chmod 750 <path>

- $7=4+2+1$: owner has all permissions (rwx)
- $5=4+1$: group members have read and execute permissions (rx)
- 0: rest of the world has no permissions

Downloading data from the internet

Downloading data from the internet

- **wget <url>**

Downloads a remote file. Supports HTTP(S) and FTP.

- **wget -c** or **wget --continue**

Continue downloading a file that was already partially downloaded

Downloading data from the internet

- **wget <url>**

Downloads a remote file. Supports HTTP(S) and FTP.

- **wget -c** or **wget --continue**

Continue downloading a file that was already partially downloaded

- **wget** is installed on most Linux systems, but by default not on macOS

Downloading data from the internet

- **wget <url>**

Downloads a remote file. Supports HTTP(S) and FTP.

- **wget -c** or **wget --continue**

Continue downloading a file that was already partially downloaded

- **wget** is installed on most Linux systems, but by default not on macOS

- **curl -LO <url>**

curl is another command line tool to download remote files, similar to **wget**.

Downloading data from the internet

Example **wget** invocation:

```
$ wget https://bioinfprep.github.io/assets/material.zip ↵
--2020-10-21 08:00:04-- https://bioinfprep.github.io/assets/material.zip
Resolving bioinfprep.github.io... 185.199.111.153, 185.199.110.153, 185.199.108.153, ...
Connecting to bioinfprep.github.io|185.199.111.153|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10586 (10K) [application/zip]
Saving to: 'material.zip.3'

material.zip.3          100%[=====>]
10.34K  --.-KB/s    in 0.001s

2020-10-21 08:00:04 (13.8 MB/s) - 'material.zip.3' saved [10586/10586]
```

Downloading data from the internet

Example **curl** invocation:

```
$ curl -LO https://bioinfprep.github.io/assets/material.zip ↵
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload Upload	Total	Spent	Left	Speed
100 10586	100 10586	0 0	57532 0	--:--:--	--:--:--	--:--:--	57532

File compression and file archives

- The most common compressed archive file formats are `.tar.gz` and `zip`
- `tar` is an archive format to reversibly combine multiple files into one file, `gz` (gzip) is a data compression tool
- `zip` is an “all in one” solution, i.e. does archiving and compression

- **unzip -l <path>**

Shows the contents of a zip file

- **unzip <path>**

Unpacks the contents of a zip file

- **zip -r <outputpath> <contentpath> [<contentpath> ...]**

Add content to a zip file recursively. Creates the zip file if it does not exist.

ZIP file handling

```
$ unzip -l material.zip ↵
```

```
Archive: material.zip
```

Length	Date	Time	Name
-----	-----	-----	----
0	03-29-2017	09:36	material/
51	03-22-2017	10:59	material/test_file_1.txt
69	03-22-2017	10:59	material/test_file_2.txt
34753	03-22-2017	12:36	material/large.fasta
67	03-22-2017	12:39	material/small.fasta
595	03-28-2017	08:58	material/duplicated_file.txt
35	03-29-2017	09:29	material/my_diff_2.txt
46	03-29-2017	09:29	material/my_diff_1.txt
35	03-29-2017	09:34	material/my_sort_1.txt
557	03-29-2017	09:36	material/tmp.txt
557	03-29-2017	09:36	material/my_sort_2.txt
-----			-----
36765			11 files

ZIP file handling

```
$ unzip material.zip ↵
```

```
Archive:  material.zip
```

```
  creating: material/
```

```
  inflating: material/test_file_1.txt
```

```
  inflating: material/test_file_2.txt
```

```
  inflating: material/large.fasta
```

```
  inflating: material/small.fasta
```

```
  inflating: material/duplicated_file.txt
```

```
 extracting: material/my_diff_2.txt
```

```
  inflating: material/my_diff_1.txt
```

```
  inflating: material/my_sort_1.txt
```

```
  inflating: material/tmp.txt
```

```
  inflating: material/my_sort_2.txt
```

ZIP file handling

```
$ unzip material.zip ↵
```

```
Archive:  material.zip
```

```
  creating: material/
```

```
    inflating: material/test_file_1.txt
```

```
    inflating: material/test_file_2.txt
```

```
    inflating: material/large.fasta
```

```
    inflating: material/small.fasta
```

```
    inflating: material/duplicated_file.txt
```

```
 extracting: material/my_diff_2.txt
```

```
    inflating: material/my_diff_1.txt
```

```
    inflating: material/my_sort_1.txt
```

```
    inflating: material/tmp.txt
```

```
    inflating: material/my_sort_2.txt
```

```
$ ls -F ↵
```

```
Desktop/  Documents/  Exercise/  material/  material.zip
```

ZIP file handling

```
$ unzip material.zip ↵
```

```
Archive:  material.zip
```

```
  creating: material/
```

```
    inflating: material/test_file_1.txt
```

```
    inflating: material/test_file_2.txt
```

```
    inflating: material/large.fasta
```

```
    inflating: material/small.fasta
```

```
    inflating: material/duplicated_file.txt
```

```
 extracting: material/my_diff_2.txt
```

```
    inflating: material/my_diff_1.txt
```

```
    inflating: material/my_sort_1.txt
```

```
    inflating: material/tmp.txt
```

```
    inflating: material/my_sort_2.txt
```

```
$ ls -F ↵
```

```
Desktop/  Documents/  Exercise/  material/  material.zip
```

```
$ ls material ↵
```

```
duplicated_file.txt  my_diff_1.txt  my_sort_1.txt  small.fasta  test_file_2.txt
```

```
large.fasta         my_diff_2.txt  my_sort_2.txt  test_file_1.txt  tmp.txt
```

```
$ zip -r new_material.zip material/ ↵  
adding: material/ (stored 0%)  
adding: material/my_diff_2.txt (stored 0%)  
adding: material/test_file_2.txt (deflated 41%)  
adding: material/duplicated_file.txt (deflated 92%)  
adding: material/test_file_1.txt (deflated 63%)  
adding: material/my_sort_1.txt (deflated 3%)  
adding: material/small.fasta (deflated 45%)  
adding: material/tmp.txt (deflated 52%)  
adding: material/my_sort_2.txt (deflated 51%)  
adding: material/my_diff_1.txt (deflated 4%)  
adding: material/large.fasta (deflated 77%)
```

- **tar tzf <path>**

Shows the contents of a gzipped tar file

- **tar tzf <path>**

Shows the contents of a gzipped tar file

- **tar xvzf <path>**

Unpacks the contents of a gzipped tar file. **Overwrites existing output file paths!**

- **tar tzf <path>**
Shows the contents of a gzipped tar file
- **tar xvzf <path>**
Unpacks the contents of a gzipped tar file. **Overwrites existing output file paths!**
- **tar cvzf <outputpath> <contentpath> [<contentpath> ...]**
Add content to a gzipped tar file recursively. **Overwrites an existing output zip file if it exists.**

```
$ tar tzf material.tar.gz ↵  
material/  
material/test_file_2.txt  
material/tmp.txt  
material/my_sort_1.txt  
material/test_file_1.txt  
material/small.fasta  
material/my_diff_2.txt  
material/my_sort_2.txt  
material/my_diff_1.txt  
material/duplicated_file.txt  
material/large.fasta
```

```
$ tar xvzf material.tar.gz ↵  
material/  
material/test_file_2.txt  
material/tmp.txt  
material/my_sort_1.txt  
material/test_file_1.txt  
material/small.fasta  
material/my_diff_2.txt  
material/my_sort_2.txt  
material/my_diff_1.txt  
material/duplicated_file.txt  
material/large.fasta
```

```
$ tar cvzf new_material.tar.gz material/ ↵  
material/  
material/my_diff_2.txt  
material/test_file_2.txt  
material/duplicated_file.txt  
material/test_file_1.txt  
material/my_sort_1.txt  
material/small.fasta  
material/tmp.txt  
material/my_sort_2.txt  
material/my_diff_1.txt  
material/large.fasta
```

- TAR files are sometimes compressed with compression tools other than `gzip`.

- TAR files are sometimes compressed with compression tools other than `gzip`.
- For `.tar.bz2` files, use the `j` switch instead of the `z` switch

- TAR files are sometimes compressed with compression tools other than `gzip`.
- For `.tar.bz2` files, use the `j` switch instead of the `z` switch
- For `.tar.xz` files, use the `J` switch instead of the `z` switch

- TAR files are sometimes compressed with compression tools other than `gzip`.
- For `.tar.bz2` files, use the `j` switch instead of the `z` switch
- For `.tar.xz` files, use the `J` switch instead of the `z` switch
- In fact, recent versions of `tar` will detect the compression algorithm automatically:

```
$ tar xf material.tar.gz ↵
```


- To compress only a single file, a file archive is not necessary

Standalone GZIP files

- To compress only a single file, a file archive is not necessary
- `gzip` is commonly used to compress all kinds of files

- To compress only a single file, a file archive is not necessary
- `gzip` is commonly used to compress all kinds of files
- **`gzip <path>`**
Compresses the given file into `<path>.gz` and removes the original file

- To compress only a single file, a file archive is not necessary
- `gzip` is commonly used to compress all kinds of files
- **`gzip <path>`**
Compresses the given file into `<path>.gz` and removes the original file
- **`gunzip <path>.gz`**
Decompresses the given file into `<path>` and removes the compressed file

Standalone GZIP files

Download a compressed FASTQ file

```
$ curl -LO https://bioinfprep.github.io/assets/sequences.fastq.gz ↵
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current	
			Dload	Upload	Total	Spent	Left	Speed
100	330	100	330	0	0	6346	0	--:--:-- --:--:-- --:--:-- 6470
100	13.5M	100	13.5M	0	0	9403k	0	0:00:01 0:00:01 --:--:-- 11.1M

Standalone GZIP files

Download a compressed FASTQ file

```
$ curl -LO https://bioinfprep.github.io/assets/sequences.fastq.gz ↵
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload	Upload	Total	Spent	Left
100	330	100	330	0	0	6346	0
100	13.5M	100	13.5M	0	0	9403k	0
						0:00:01	0:00:01
							11.1M

Inspect the downloaded file

```
$ ls -lh sequences.fastq.gz ↵
```

-rw-r--r--	1	vorkurs	vorkurs	14M	Apr 3 2018	sequences.fastq.gz
------------	---	---------	---------	-----	------------	--------------------

Standalone GZIP files

Download a compressed FASTQ file

```
$ curl -LO https://bioinfprep.github.io/assets/sequences.fastq.gz ↵
```

% Total		% Received		% Xferd		Average Speed		Time	Time	Time	Current
						Dload	Upload	Total	Spent	Left	Speed
100	330	100	330	0	0	6346	0	--:--:--	--:--:--	--:--:--	6470
100	13.5M	100	13.5M	0	0	9403k	0	0:00:01	0:00:01	--:--:--	11.1M

Inspect the downloaded file

```
$ ls -lh sequences.fastq.gz ↵
```

-rw-r--r--	1	vorkurs	vorkurs	14M	Apr 3 2018	sequences.fastq.gz
------------	---	---------	---------	-----	------------	--------------------

Decompress the downloaded file

```
$ gunzip sequences.fastq.gz ↵
```

Standalone GZIP files

Download a compressed FASTQ file

```
$ curl -LO https://bioinfprep.github.io/assets/sequences.fastq.gz ↵
```

% Total		% Received		% Xferd		Average Speed		Time	Time	Time	Current
						Dload	Upload	Total	Spent	Left	Speed
100	330	100	330	0	0	6346	0	--:--:--	--:--:--	--:--:--	6470
100	13.5M	100	13.5M	0	0	9403k	0	0:00:01	0:00:01	--:--:--	11.1M

Inspect the downloaded file

```
$ ls -lh sequences.fastq.gz ↵
```

-rw-r--r--	1	vorkurs	vorkurs	14M	Apr 3 2018	sequences.fastq.gz
------------	---	---------	---------	-----	------------	--------------------

Decompress the downloaded file

```
$ gunzip sequences.fastq.gz ↵
```

Inspect the decompressed file

```
$ ls -lh sequences.fastq ↵
```

-rw-r--r--	1	vorkurs	vorkurs	89M	Apr 3 2018	sequences.fastq
------------	---	---------	---------	-----	------------	-----------------

File sizes

- **du <path>** - disk usage

Can be used to show the sizes of individual files or the total size of entire directories

- **du -h <path>**

Shows the sizes in a human readable format instead of kb

- **du -a <path>**

When applied on a directory, report sizes of all files, not only folders.

- **du** and **ls** may report different sizes

```
$ du -h material.zip ↵
```

```
12K    material.zip
```

```
$ ls -lh material.zip ↵
```

```
-rw-r--r-- 1 vorkurs vorkurs 11K Apr  3 22:05 material.zip
```

File sizes

- **du** and **ls** may report different sizes

```
$ du -h material.zip ↵
```

```
12K    material.zip
```

```
$ ls -lh material.zip ↵
```

```
-rw-r--r-- 1 vorkurs vorkurs 11K Apr  3 22:05 material.zip
```

- The tools define the size of a file differently
 - **du** reports how much space the file consumes on the underlying storage
 - **ls** reports the size of the content stored inside the file

File sizes

- **du** and **ls** may report different sizes

```
$ du -h material.zip ↵
```

```
12K    material.zip
```

```
$ ls -lh material.zip ↵
```

```
-rw-r--r-- 1 vorkurs vorkurs 11K Apr  3 22:05 material.zip
```

- The tools define the size of a file differently
 - **du** reports how much space the file consumes on the underlying storage
 - **ls** reports the size of the content stored inside the file
- When **du** is invoked with the flag **--apparent-size**, it too reports the size of the content

File comparison

- **diff <path1> <path2>**

Identifies differences between two plain text files and reports them as the smallest number of insertions and deletions required to transform one file into the other.

- **diff -i** or **diff --ignore-case**

Ignore case differences in the files contents

- **diff -w** or **diff --ignore-all-space**

Ignore all differences that involve only white space

- **diff -B** or **diff --ignore-blank-lines**

Ignore blank lines

- **diff -y** or **diff --side-by-side**

Output a human readable, side-by-side view instead of machine readable output

👉 Use **diff** to identify the differences between the files `my_diff_1.txt` and `my_diff_2.txt`!

Sorting and Counting

Sorting and Counting

- **sort <path>** - sort plain text

The **sort** tool can be used to sort plain text files line by line.

- **sort -b** or **sort --ignore-leading-blanks**
Ignore leading white space characters
- **sort -n** or **sort --numeric-sort**
Interpret digits numerically instead of lexicographically
- **sort -r** or **sort --reverse**
Sort in reverse order
- **sort -u** or **sort --unique**
Output redundant lines only once

- **uniq <path>**

Collapse consecutive identical lines into one line.

- **uniq -c** or **uniq --count**

Report the number of occurrences with each line

- **uniq -i** or **uniq --ignore-case**

Ignore case differences when comparing lines

- **wc <path>** - word count

Count words (or lines, characters, bytes) in plain text

- **wc -c**

Report the number of characters

- **wc -l**

Report the number of lines

- **wc -w**

Report the number of words

- **wc -C**

Print all three numbers

Sorting and Counting

👉 Sort the file `my_sort_1.txt`!

👉 Sort the file `my_sort_2.txt` in reverse numerical order!

👉 Which lines in `duplicated_file.txt` are duplicated and how often do they occur?

👉 Are all globally redundant lines collapsed by `uniq`?

Searching patterns in plain text

Searching patterns in plain text

- **grep <pattern> <path> [<path> ...]**
Prints all lines of the input that match the given regular expression pattern

Searching patterns in plain text

- **grep <pattern> <path> [<path> ...]**
Prints all lines of the input that match the given regular expression pattern

Searching patterns in plain text

- **grep <pattern> <path> [<path> ...]**

Prints all lines of the input that match the given **regular expression pattern**

- **grep -v <pattern> <path> [<path> ...]**

Inverse mode, print all lines that do *not* match the given pattern

- **grep -l <pattern> <path> [<path> ...]**

Print only the names of files that the pattern matches against

- **grep -n <pattern> <path> [<path> ...]**

Print the line numbers along with the matches

- **grep --color <pattern> <path> [<path> ...]**

Highlight the pattern occurrence

```
$ grep --color -n is sortme.txt ↵
```

```
1:This is an example file with content
```

```
2:that may be sorted. If you sort this file
```

```
4:because the line order is critical. None
```

```
5:of this makes any sense!
```

Process streams II

- The `|` (pipe) character can be used to **redirect** the **standard output** stream of one process to the **standard input** stream of another process

- The | (pipe) character can be used to **redirect** the **standard output** stream of one process to the **standard input** stream of another process
- Remember the common structure of command line interfaces that we have seen often today:

```
COMMAND [OPTIONAL SWITCHES] [INPUT PATH]
```

Process streams II

- The | (pipe) character can be used to **redirect** the **standard output** stream of one process to the **standard input** stream of another process
- Remember the common structure of command line interfaces that we have seen often today:

```
COMMAND [OPTIONAL SWITCHES] [INPUT PATH]
```

- Most programs we have seen today would process data from the **standard input** stream if no input file path is given!

Process streams II

Without stream redirection:

```
$ ls -l 1> temp.txt ↵
```

```
$ wc -l temp.txt ↵
```

```
15 temp.txt
```

Process streams II

Without stream redirection:

```
$ ls -l 1> temp.txt ↵  
$ wc -l temp.txt ↵  
15 temp.txt
```

With stream redirection:

```
$ ls -l | wc -l ↵  
15
```

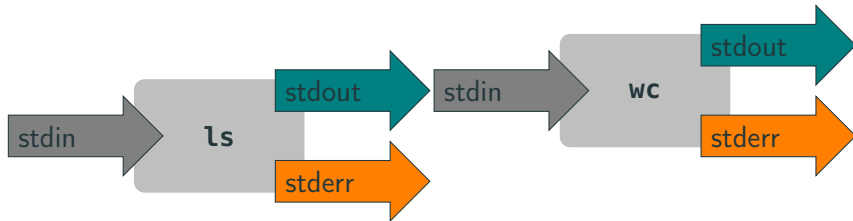

Process streams II

Without stream redirection:

```
$ ls -l 1> temp.txt ↵  
$ wc -l temp.txt ↵  
15 temp.txt
```

With stream redirection:

```
$ ls -l | wc -l ↵  
15
```



Process streams II

Recall:

```
$ uniq -c duplicated_file.txt ↵  
  22 bla  
  28 blabla  
   1 this line is not duplicatedbla  
  21 bla  
  28 blabla
```

Combining **sort** and **uniq**:

```
$ sort duplicated_file.txt | uniq -c ↵  
  43 bla  
  56 blabla  
   1 this line is not duplicatedbla
```

What does this do and why?

```
$ gunzip -c material.tar.gz | tar t ↵
```

<https://bioinfprep.github.io/task/unix1.html>