Prep-Course Informatics for Life Scientists

Introduction to Unix and the Command Line Interface

Philipp Thiel

April 4, 2022

License and Contributors

Main Author
Léon Kuchenbecker

Additional Contributors
Alexander Seitz, Philipp Thiel, 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

Basic Usage

- 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 [ARGUMENTS]
COMMAND [OPTIONS] [OPERANDS]
```

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
```

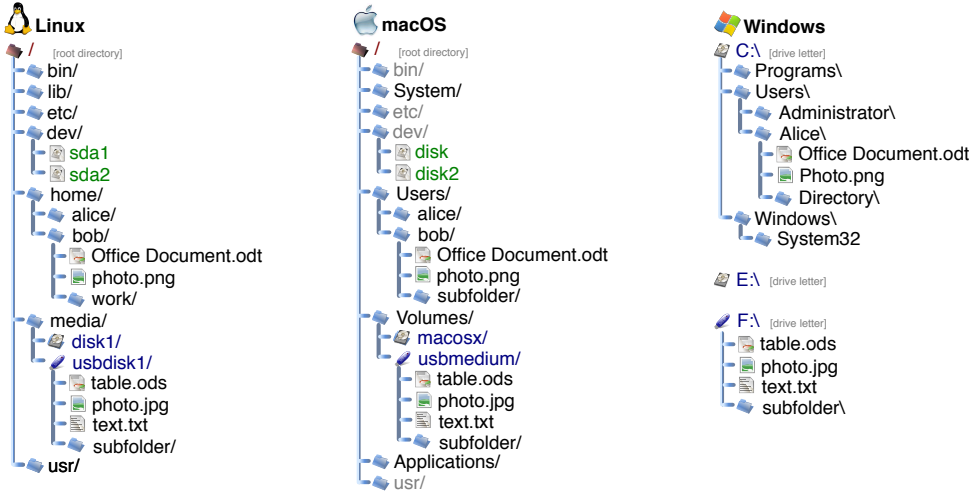
- Multiple switches can be specified together

```
$ 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



File systems

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

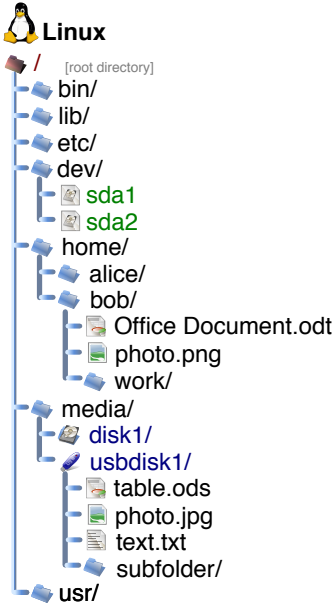
File systems

- 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”.

File systems

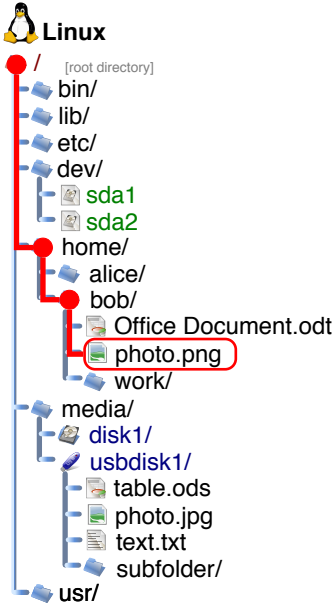
- Every running computer program, i.e. every process, including the shell, refers to one directory inside the file system tree as its working directory.
- When using the shell, one often says “I am in the directory xzy”.
- The working directory is not static, it can be changed throughout the runtime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

File systems



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

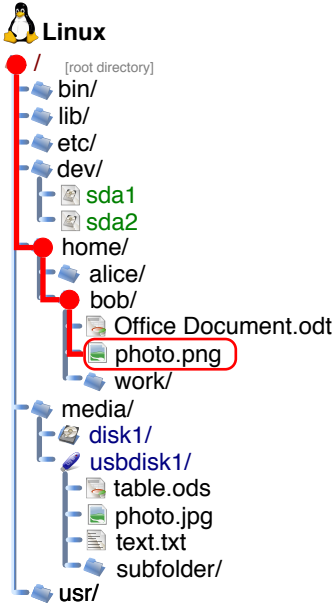
File systems



- 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



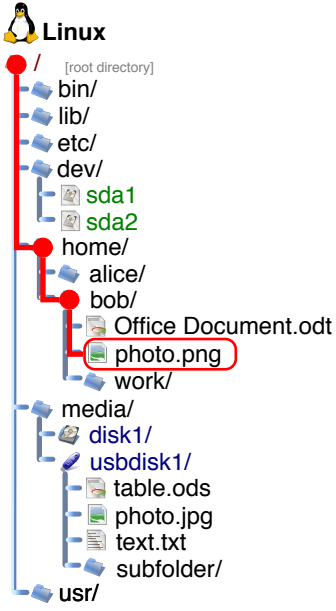
- Every file or directory in the file system can be described by a path
- Absolute paths start in the root directory, for example

  `/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

Working with the file system

- **`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.

Working with the file system

- **`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

Inspecting 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).

Inspecting plain text files

- **`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.

Creating and editing plain text files

- 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
- A well known graphical editor for Microsoft Windows: Notepad

Creating and editing plain text files

- A text editor is a program used to edit plain text files
- A well known graphical editor for Microsoft Windows: Notepad
- Well known graphical editors for for macOS: TextEdit and Atom

Creating and editing plain text files

- A text editor is a program used to edit plain text files
- A well known graphical editor for Microsoft Windows: Notepad
- Well known graphical editors for for macOS: TextEdit and Atom
- Nano is a text editor that does not require a graphical user interface but works on the command line

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.

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.

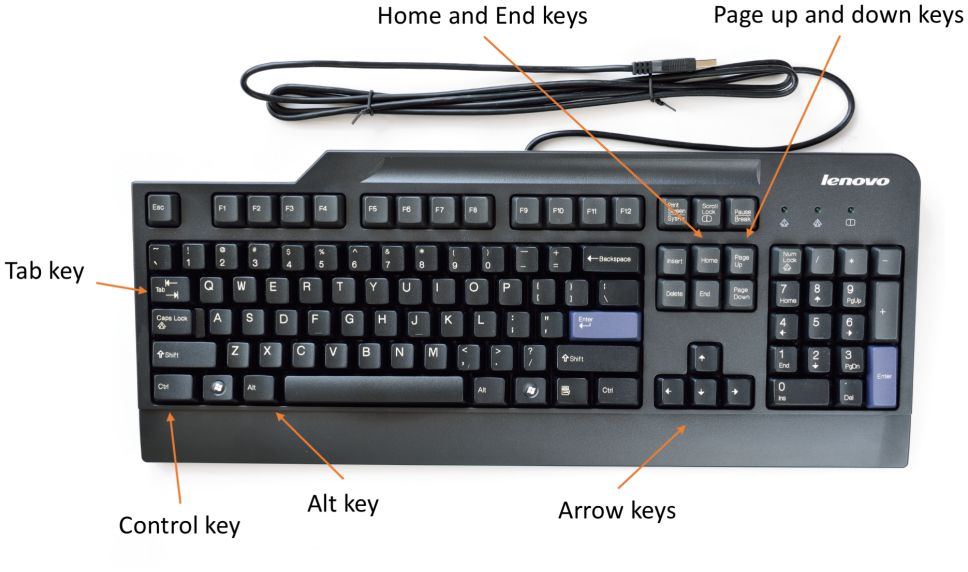
- **`CTRL-O`**
  Saves the current file

- **`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.

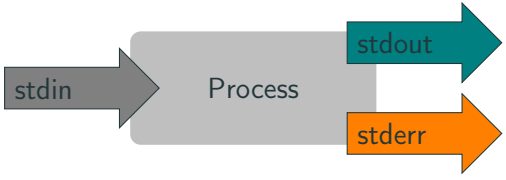
Process streams

Process streams

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

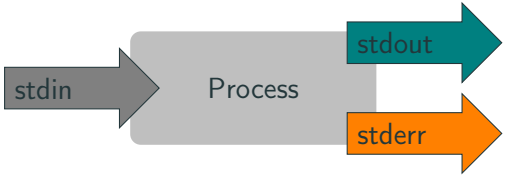
Process streams

- 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.

Process streams

- Normally, we cannot see the differente between stdout and stderr

Process streams

- Normally, we cannot see the differente 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 differente 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
```

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 ↵
```

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>**

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!

Process streams

- 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`
- ☞ Write the content of `repetition2.txt` twice into a new 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

Users, groups and permissions

- You all know that there exist so-called users on a computer system

Users, groups and permissions

- You all know that there exist so-called users on a computer system
- All operating systems have this concept to provide individualized accounts

Users, groups and permissions

- You all know that there exist so-called users on a computer system
- All operating systems have this concept to provide individualized accounts
- Every user is part of one or more groups

Users, groups and permissions

- You all know that there exist so-called users on a computer system
- All operating systems have this concept to provide individualized accounts
- Every user is part of one or more groups
- Thus, groups are collections of users

Users, groups and permissions

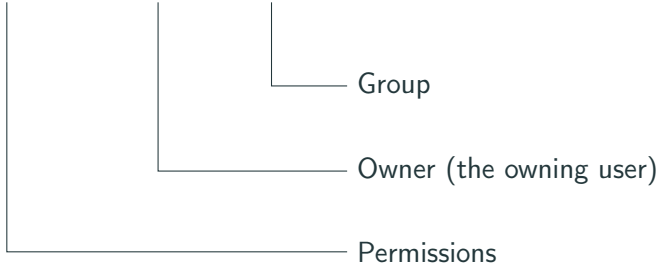
- You all know that there exist so-called users on a computer system
- All operating systems have this concept to provide individualized accounts
- Every user is part of one or more groups
- Thus, groups are collections of users
- Users and and groups can be created and users can be assigned to groups

Users, groups and permissions

- You all know that there exist so-called users on a computer system
- All operating systems have this concept to provide individualized accounts
- Every user is part of one or more groups
- Thus, groups are collections of users
- Users and and groups can be created and users can be assigned to groups
- On Linux systems users and groups are used to manage file permissions

Users, groups and permissions

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



Users, groups and permissions

- Every file has one owner (the user owning the file) and is assigned to one group

Users, groups and permissions

- Every file has one owner (the user owning the file) and is assigned to one group
- Permissions specify file access rights for users based on the following categories: owner, group, and other (world)

Users, groups and permissions

- Every file has one owner (the user owning the file) and is assigned to one group
- Permissions specify file access rights for users based on the following categories: owner, group, and other (world)
- There are three types of permissions with associated symbols:
  - **`read`** (r)
  - **`write`** (w)
  - **`execute`** (x)

Users, groups and permissions

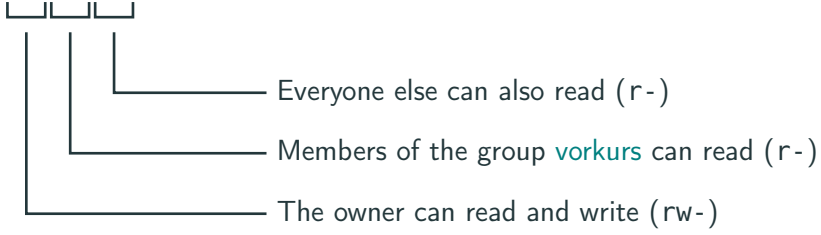
- Every file has one owner (the user owning the file) and is assigned to one group
- Permissions specify file access rights for users based on the following categories: owner, group, and other (world)
- There are three types of permissions with associated symbols:
  - **`read`** (r)
  - **`write`** (w)
  - **`execute`** (x)
- For each category all types are explicitly specified using a triplet of the symbols: (read permission, write permission, execute permission)

Users, groups and permissions

- Every file has one owner (the user owning the file) and is assigned to one group
- Permissions specify file access rights for users based on the following categories: owner, group, and other (world)
- There are three types of permissions with associated symbols:
  - **`read`** (r)
  - **`write`** (w)
  - **`execute`** (x)
- For each category all types are explicitly specified using a triplet of the symbols: (read permission, write permission, execute permission)
- Symbol '-' indicates the absence of the corresponding permission

Users, groups and permissions

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



Users, groups and permissions

- Permissions are also used for directories (in fact they are also just files)
- However, interpretation of permissions for directories is slightly different
  - `r` determines whether the contents of a directory can be seen
  - `w` determines whether files can be created or deleted
  - `x` determines whether a user can change 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>` permissions** Changes the file permissions.
  - Symbol '+' to add permissions
  - Symbol '-' to remove permissions
  - Shortcuts for categories: (u)ser, (g)roup, (o)thers, (a)ll at once
  - Shortcuts for permissions: (r)ead, (w)rite, e(x)ecute

Users, groups and permissions

Some examples:

- Add write permissions to the group:
  `chmod g+w <path>`

Users, groups and permissions

Some examples:

- Add write permissions to the group:
  `chmod g+w <path>`
- Remove read permissions for others (world):
  `chmod o-r <path>`

Users, groups and permissions

Some examples:

- Add write permissions to the group:
  `chmod g+w <path>`
- Remove read permissions for others (world):
  `chmod o-r <path>`
- What is the next command doing?:
  `chmod u+rwx,g+rw,o+r <path>`

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 write permission to everyone:

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 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
```

Users, groups and permissions

- Permissions also have numeric equivalents
  - 4: read (r)
  - 2: write (w)
  - 1: execute (x)

Users, groups and permissions

- Permissions also have numeric equivalents
  - 4: read (r)
  - 2: write (w)
  - 1: execute (x)
- Permissions can be combined in a single-digit number via summation

Users, groups and permissions

- Permissions also have numeric equivalents
  - 4: read (r)
  - 2: write (w)
  - 1: execute (x)
- Permissions can be combined in a single-digit number via summation
- Triplet of single-digit numbers can specify the permissions for all categories

Users, groups and permissions

- Permissions also have numeric equivalents
  - 4: read (r)
  - 2: write (w)
  - 1: execute (x)
- Permissions can be combined in a single-digit number via summation
- Triplet of single-digit numbers can specify the permissions for all categories
- Example: **`chmod 750 <path>`**
  - Owner: $7 = 4 + 2 + 1$ (all permissions 'rwx')
  - Group: $5 = 4 + 1$ (read and execute permissions 'r-x')
  - Other: 0 (no permissions '---')

Downloading data from the internet

Downloading data from the internet

- **`wget <url>`**

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

  - Option -c (--continue):
    Continue downloading a file that was already partially downloaded
  - Installed on most Linux systems
  - Not installed by default on macOS

Downloading data from the internet

- **`wget <url>`**

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

  - Option -c (--continue):
    Continue downloading a file that was already partially downloaded
  - Installed on most Linux systems
  - Not installed by default on macOS

- **`curl -LO <url>`**
  - Another command line tool to download remote files, similar to wget.
  - Usually installed on macOS

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.tar.gz ↵
  % 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

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
- Often you find the combination of an archive with compression: .tar.gz
- zip is an "all in one" solution, i.e. does archiving and compression

ZIP file handling

- **`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 file handling

```
$ 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%)
```

GZIPed TAR file handling

- `tar tzf <path>`
  Shows the contents of a gzipped tar file

GZIPed TAR file handling

- `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!

GZIPed TAR file handling

- **`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!

GZIPed TAR file handling

```
$ 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
```

GZIPed TAR file handling

```
$ 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
```

GZIPed TAR file handling

```
$ 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
```

GZIPed TAR file handling

- tar files are sometimes compressed with tools other than gzip.

GZIPed TAR file handling

- tar files are sometimes compressed with tools other than gzip.
- bzip2 compression with suffix `.tar.bz2`:
  use option `j` instead of `z`

GZIPed TAR file handling

- tar files are sometimes compressed with tools other than gzip.
- bzip2 compression with suffix `.tar.bz2`:
  use option `j` instead of `z`
- xz compression with suffix `.tar.xz`:
  use option `J` instead of `z`

GZIPed TAR file handling

- tar files are sometimes compressed with tools other than gzip.
- bzip2 compression with suffix `.tar.bz2`:
  use option `j` instead of `z`
- xz compression with suffix `.tar.xz`:
  use option `J` instead of `z`
- Recent versions of tar detect the compression algorithm automatically

GZIPed TAR file handling

- tar files are sometimes compressed with tools other than gzip.
- bzip2 compression with suffix `.tar.bz2`:
  use option `j` instead of `z`
- xz compression with suffix `.tar.xz`:
  use option `J` instead of `z`
- Recent versions of tar detect the compression algorithm automatically
- **`tar xvf <path>`**
  Unpacks the contents of a compressed tar file

Standalone GZIP files

- 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

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
- **`gzip <path>`**
  Compresses the given file into `<path>.gz` and removes the original file

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
- **`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  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
```

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

File sizes

- **`du <path>` - disk usage**
  Can be used to show individual file sizes or the total size of 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.

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
```

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
- **du** invoked with option --apparent-size also reports the size of the content

File comparison

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

File comparison

☞ 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 caracters
  - **`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 duplicate lines only once

Sorting and Counting

- **`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

Sorting and Counting

- `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

Searching patterns in plain text

```
$ 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

Process streams II

- The | (pipe) symbol redirects the standard output stream (stdout) of one process to the standard input stream (stdin) of another process

Process streams II

- The | (pipe) symbol redirects the standard output stream (stdout) of one process to the standard input stream (stdin) of another process
- Remember the common structure of commands that we have seen often today:

```
COMMAND [OPTIONS] [INPUT FILE PATH]
```

Process streams II

- The | (pipe) symbol redirects the standard output stream (stdout) of one process to the standard input stream (stdin) of another process
- Remember the common structure of commands that we have seen often today:

```
COMMAND [OPTIONS] [INPUT FILE PATH]
```

- Most programs we have seen today would process data from stdin 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
```

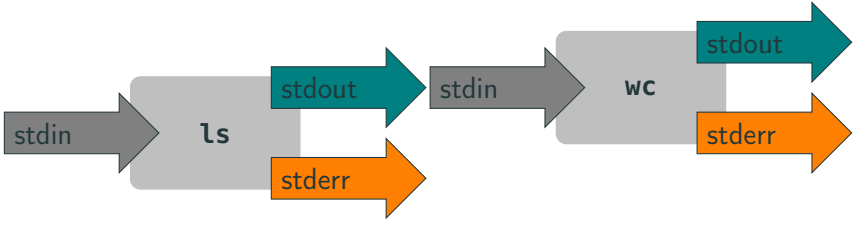
Process streams II

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
```

Process streams II

What is the following command doing and why?

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

Final Task

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