GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
Projektträger Jülich
Forschungszentrum Jülich

de.NBI
GERMAN NETWORK FOR BIOINFORMATICS INFRASTRUCTURE

Introduction to Linux and the Command Line with SimpleVM

Peter Belmann
David Weinholz
Viktor Rudko

27.11.2023

JÜLICH Forschungszentrum

SimpleVM

Hands-On Session

- **Part 1: SimpleVM Introduction**
- Part 2: Linux Command Line

SimpleVM

- SimpleVM is a multi-cloud application that eases the access to computational resources of the federated de.NBI Cloud.
- It takes care of any necessary network or volume setup.
- Automatically installed software
- No background knowledge in cloud computing necessary
- Additional Project Modes:
  - Workshop Mode
  - SimpleVM Cluster
- New SimpleVM Portal since October 2023

de.NBI Cloud - A Solution for (almost) Every Use Case

Infrastructure-, Platform- and Workflows- as-a-Service

openstack

- High configurability, infrastructure virtualization
- API access, e.g. for use with Terraform or Ansible
- Any software of the cloud ecosystem

kubernetes

- High configurability, service and container orchestration
- API access, e.g. for use with kubectl
- Any containerized software / service, Helm charts

SimpleVM

- Beginner-friendly, preconfigured Research Envs
- ELIXIR guarded interactive browser sessions
- “One-click” solution for setting up a workshop or cluster

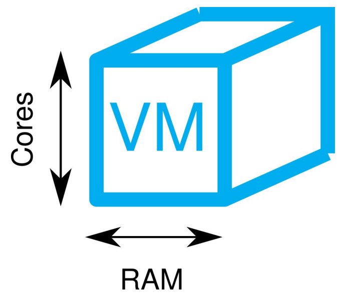
useGalaxy.eu

- Fully established bioinformatics tools and workflows, maintained by the community
- Point-and-click GUI for composition of bioinformatics workflows
- Interactive tours and comprehensive training library

Cloud Components

- Main building block of the cloud is a **virtual machine**.
- A virtual machine **instance** is a running instance of the virtual machine image with resource parameters (Cores and RAM) assigned to it.

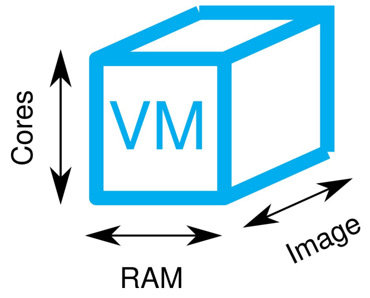
1. Create your blueprint



Cloud Components

- Main building block of the cloud is a **virtual machine**.
- A virtual machine **instance** is a running instance of the virtual machine image with resource parameters (Cores and RAM) assigned to it.
- A virtual machine **image** is a virtual disk that has a bootable operating system installed on it.

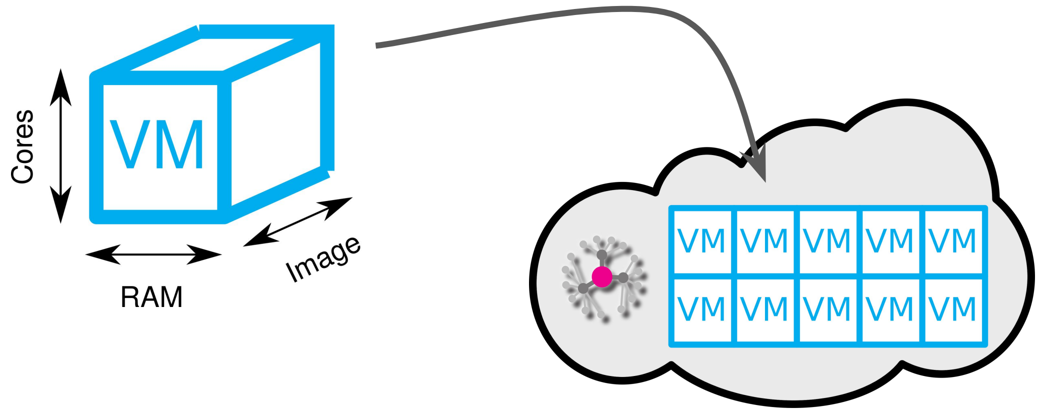
1. Create your blueprint



Cloud Components

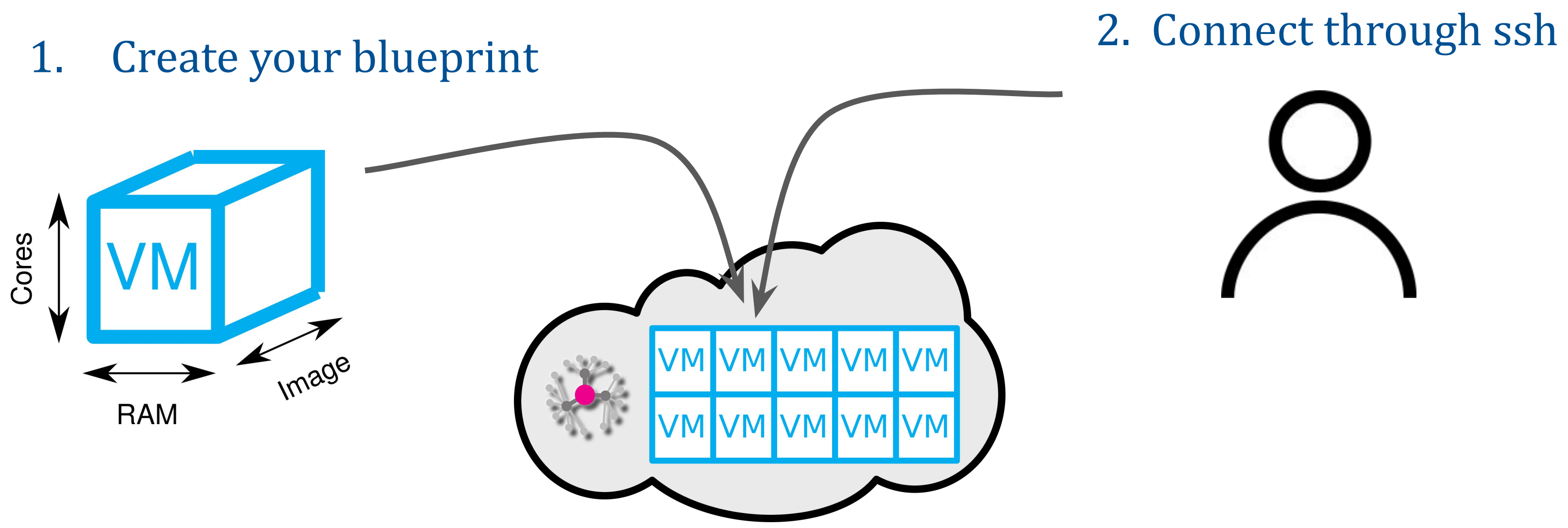
- Main building block of the cloud is a **virtual machine**.
- A virtual machine **instance** is a running instance of the virtual machine image with resource parameters (Cores and RAM) assigned to it.
- A virtual machine **image** is a virtual disk that has a bootable operating system installed on it.

1. Create your blueprint



Cloud Components

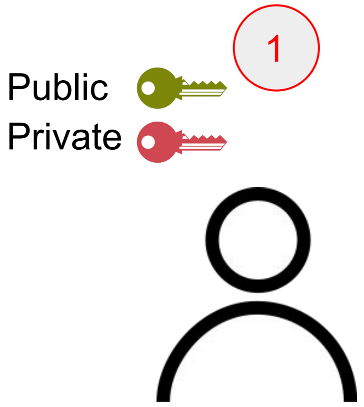
- Main building block of the cloud is a **virtual machine**.
- A virtual machine **instance** is a running instance of the virtual machine image with resource parameters (Cores and RAM) assigned to it.
- A virtual machine **image** is a virtual disk that has a bootable operating system installed on it.



Cloud Components - Secure Shell Protocol (SSH)

First Time Usage

1. Generate a key pair locally or let the Portal generate one for you

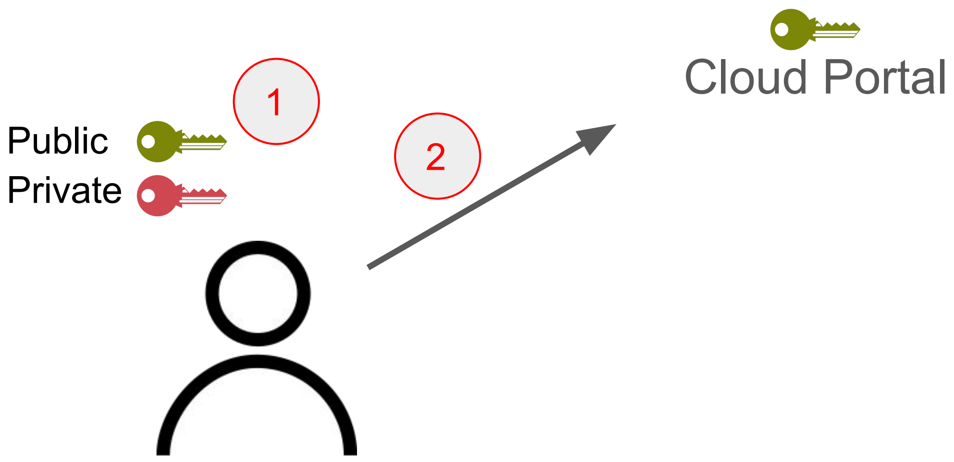


Cloud Components - Secure Shell Protocol (SSH)

First Time Usage

1. Generate a key pair locally or let the Portal generate one for you
2. Upload your public key to the de.NBI Cloud Portal/OpenStack

Connect to your VM

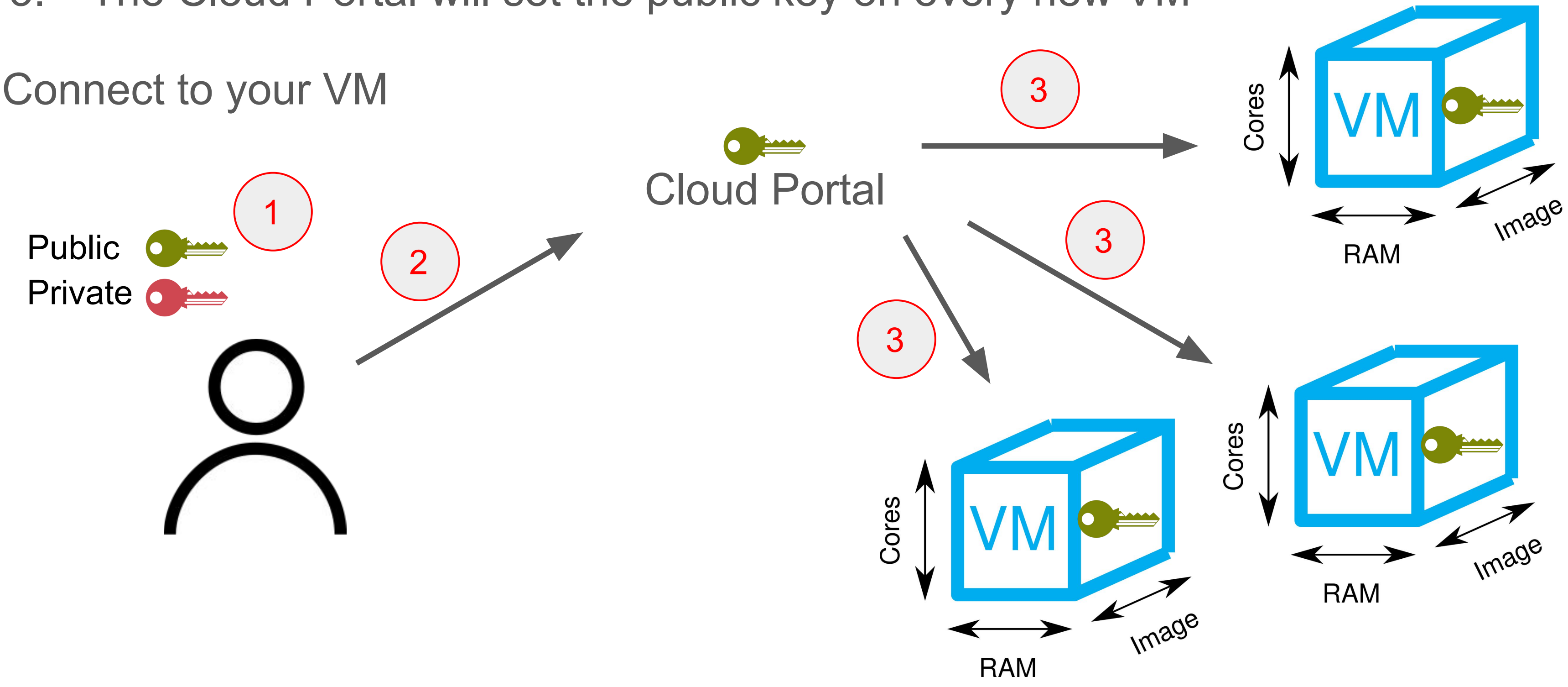


Cloud Components - Secure Shell Protocol (SSH)

First Time Usage

1. Generate a key pair locally or let the Portal generate one for you
2. Upload your public key to the de.NBI Cloud Portal/OpenStack
3. The Cloud Portal will set the public key on every new VM

Connect to your VM



Hands-On Session

- Go to the first part of the workshop:

  https://github.com/deNBI/simpleVMWorkshop/blob/main/part1.md

  - **Please start with part 1 of the SimpleVM workshop**

- ~ 10 Minutes

Internet:

WiFi Name:

LAN1-JBB

WiFi PWD:

jbb45_45

Hands-On Session

- Part 1: SimpleVM Introduction
- **Part 2: Linux Command Line**

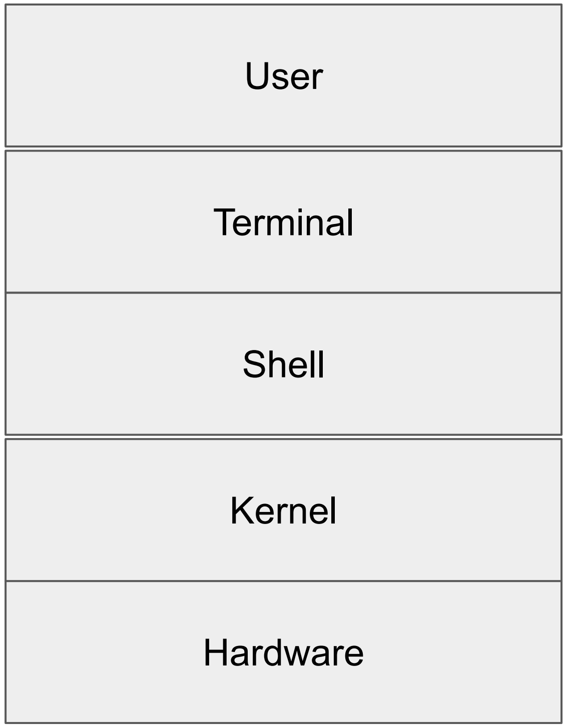
Part 2: Linux Command Line Introduction

Linux Command Line Introduction - Unix/Linux?

- Linux is a flavour of **Unix**
- **Unix:** set of programs to interact with the computer
- Main Properties:
  - **Multitasking environment:** Multiple processes can run in parallel
  - **Multiuser system:** Multiple users can use the same Linux system at the same time

Linux Command Line Introduction - Unix/Linux?

- The heart of a Linux system is the **kernel**
- The kernel allocates system resources
- You, as a user, use a **shell** to interact with the kernel
- The **terminal** is a graphical interface that lets you interact with the shell



Linux Command Line Introduction - Part 1

- Almost every command has a usage and manual **man page**

Get help and usage message: **--help / -h / -? / --usage**

```
ls --help
```

Read the manual: **man**

```
man ls
```

Space	one page down
b	one page up
q	quit

If **man** does not give any result, try:
info
help

Don't know what command to use? Try:
apropos

Linux Command Line Introduction - Part 1

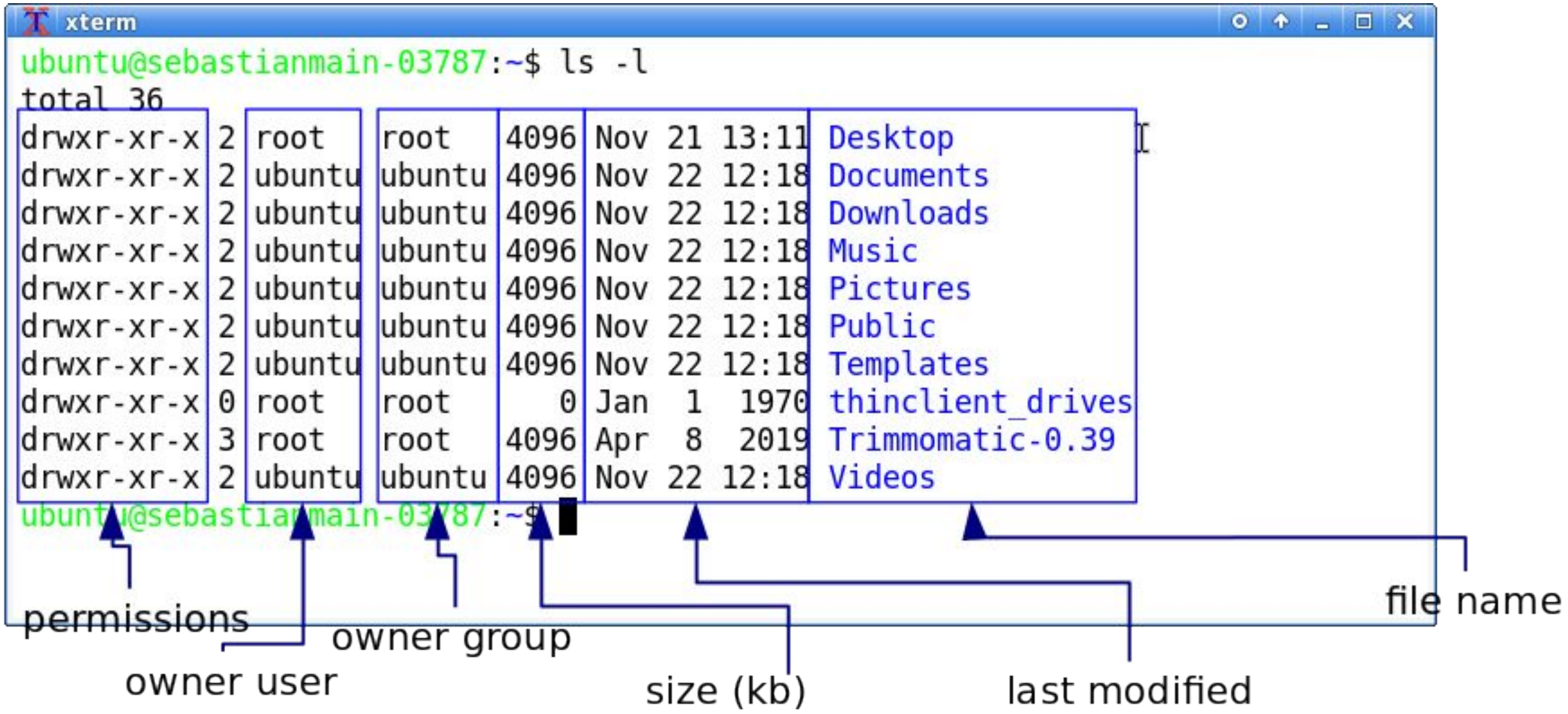
- Data in Unix is organized in files
    - Three types of files:
        - ordinary files (e.g. a program or text),
        - directories,
        - special files (e.g. shortcuts)
- Directories organized in hierarchical, tree like structure
    - with “/” as its root.
        - /home/ubuntu/projects/unix-course
        - /home/ubuntu/Downloads/test.txt

Linux Command Line Introduction - Part 1

- Commands for navigation: **`pwd`**, **`ls`**, **`cd`**
- Places to navigate (current: /home/ubuntu/dir, aim: /home/ubuntu):
  - Absolute path: **`cd /home/ubuntu/`**
  - Relative path: **`cd ..`**
  - Home directory: **`cd ~`**
- Commands to manipulate the filesystem tree:
  - **`mkdir`**, Syntax: mkdir directory_to_create
  - **`rm -r`**, Syntax: rm -r directory_to_remove
- View a file:
  - **`less`** file_to_view
- Other:
  - **`file`**: Returns you the type of the file
  - **`df`**: Allows you check the usage of your disk. (size of the free and used space)

Linux Command Line Introduction - Part 1

- Permissions:
    - You can view the permissions of file with "**`ls -l`**"



```
ubuntu@sebastianmain-03787:~$ ls -l
total 36
drwxr-xr-x 2 root   root   4096 Nov 21 13:11 Desktop
drwxr-xr-x 2 ubuntu ubuntu 4096 Nov 22 12:18 Documents
drwxr-xr-x 2 ubuntu ubuntu 4096 Nov 22 12:18 Downloads
drwxr-xr-x 2 ubuntu ubuntu 4096 Nov 22 12:18 Music
drwxr-xr-x 2 ubuntu ubuntu 4096 Nov 22 12:18 Pictures
drwxr-xr-x 2 ubuntu ubuntu 4096 Nov 22 12:18 Public
drwxr-xr-x 2 ubuntu ubuntu 4096 Nov 22 12:18 Templates
drwxr-xr-x 0 root   root      0 Jan  1  1970 thinclient_drives
drwxr-xr-x 3 root   root   4096 Apr  8  2019 Trimmomatic-0.39
drwxr-xr-x 2 ubuntu ubuntu 4096 Nov 22 12:18 Videos
ubuntu@sebastianmain-03787:~$
```

Linux Command Line Introduction - Part 1

- Permissions:
  - You can view the permissions of file with "`ls -l`"
  - Modify permissions by using **chmod**
    - -rw-r-- --- = (user, group, others)
    - r = readable, w = writable, x = executable
    - Example: `chmod g+w file` (make file writable for the group)
- Execute a program:
  - You can execute a local program by providing the path of the file: "/path/to/my_program_to_execute"
  - You can cancel any command with "Ctrl+c"

Linux Command Line Introduction - Part 1

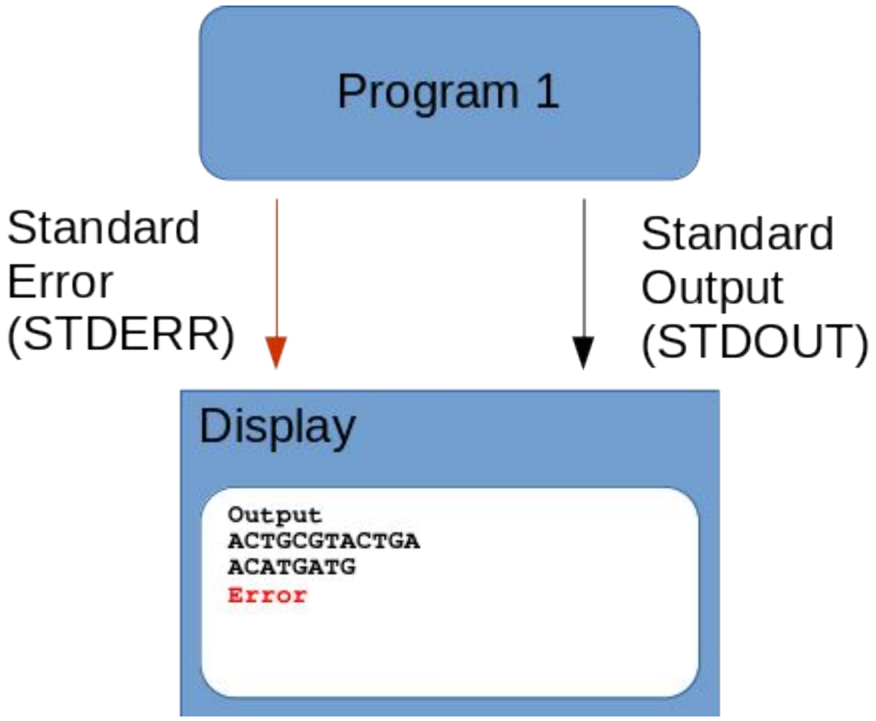
- Go to the second part of the SimpleVM workshop:

  https://github.com/deNBI/simpleVMWorkshop

- Once you are done, continue with the actual Linux Course:
- Your task is to **identify the correct commands** by using the list of commands and execute them. Feel free to experiment. Take a look at the solution **if absolutely necessary.**
- Tutorial: https://github.com/deNBI/unix-course
- Please just do part 1 of the tutorial!
- ~20 Minutes

Linux Command Line Introduction - Part 2

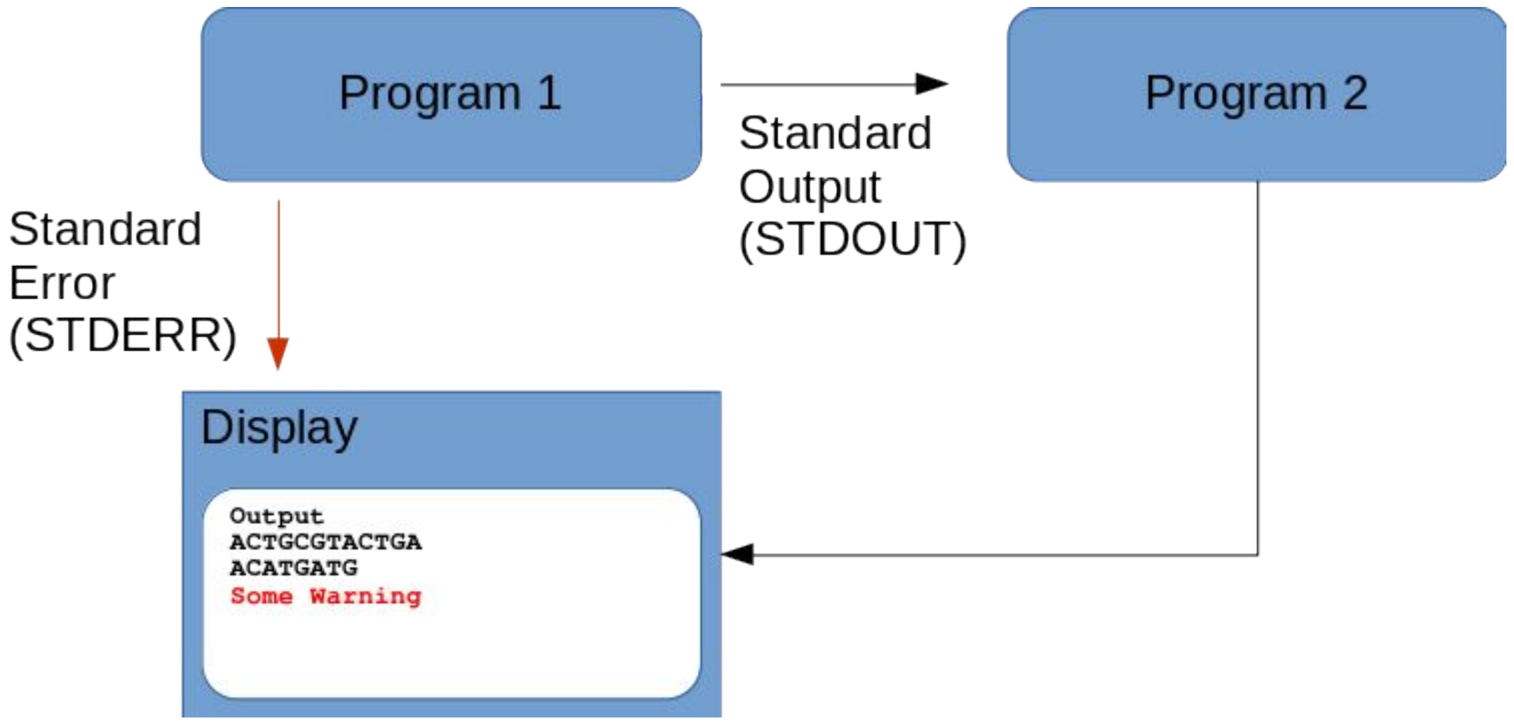
- Many unix commands support streams (files) as input and/or output



```
Output
ACTGCGTACTGA
ACATGATG
Error
```

Linux Command Line Introduction - Part 2

- Many unix commands support streams (files) as input and/or output
- Streams can be **redirected**

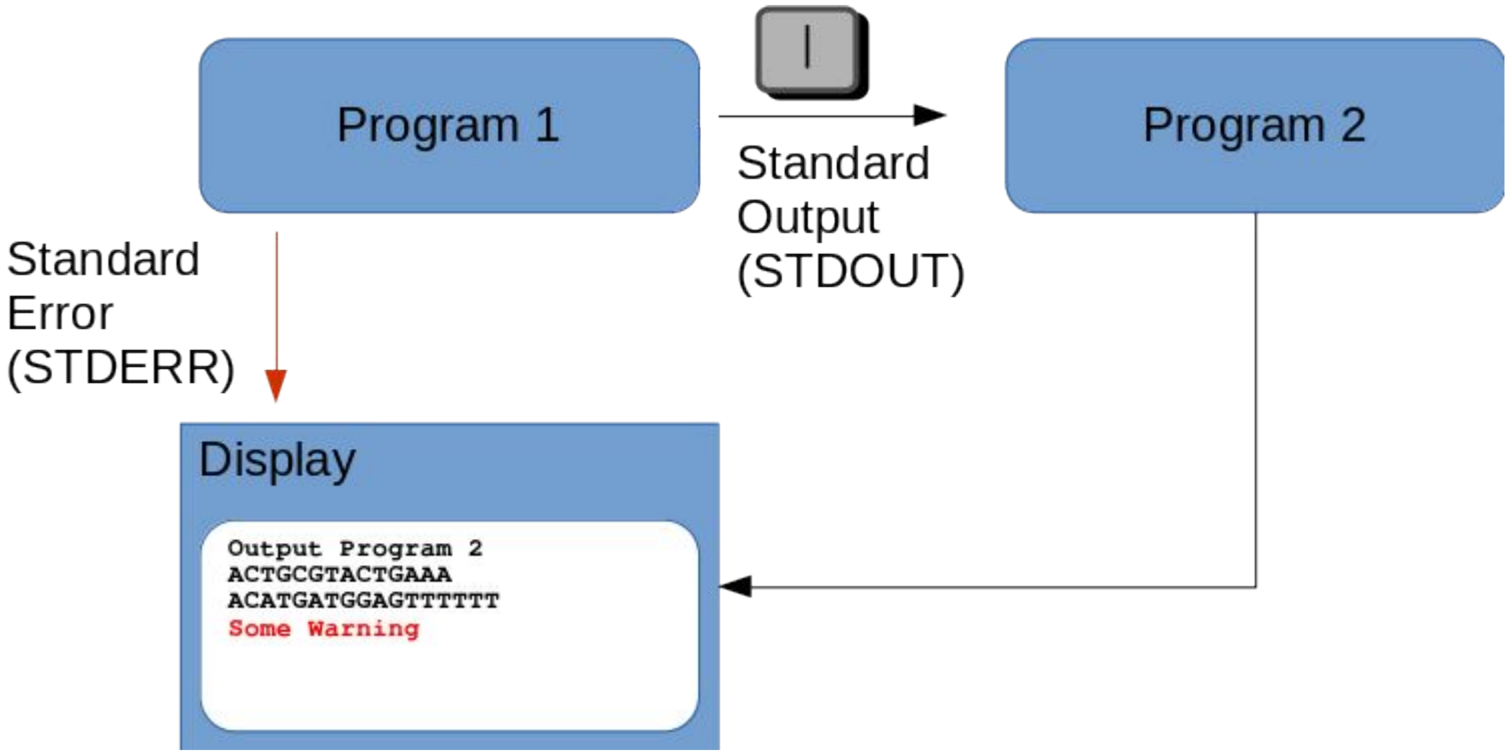


```
Output
ACTGCGTACTGA
ACATGATG
Some Warning
```

How?

Linux Command Line Introduction - Part 2

- Many unix commands support streams (files) as input and/or output
- Streams can be **redirected**

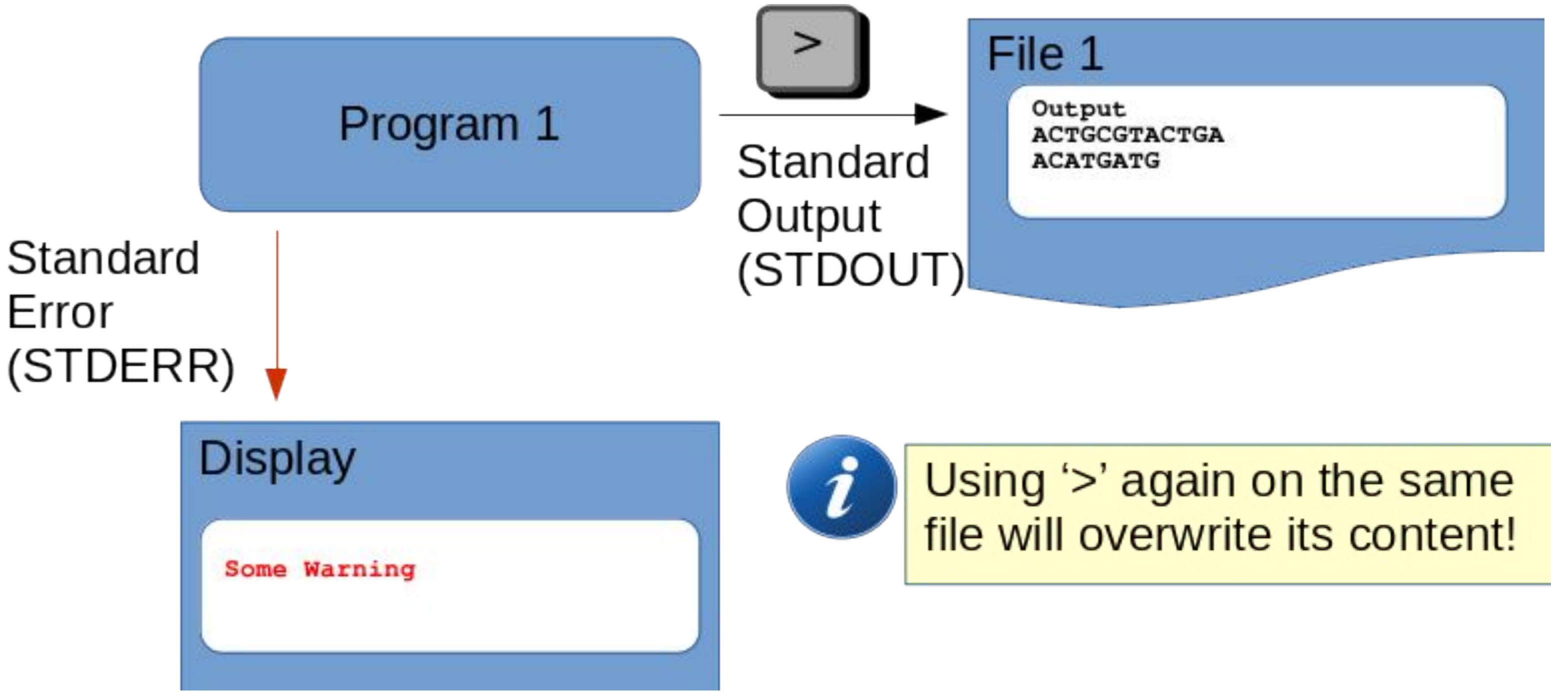


```
Output Program 2
ACTGCGTACTGAAA
ACATGATGGAGTTTTTT
Some Warning
```

Example: `ls -la | less`

Linux Command Line Introduction - Part 2

- Many unix commands support streams (files) as input and/or output
- Streams can be **redirected into files**

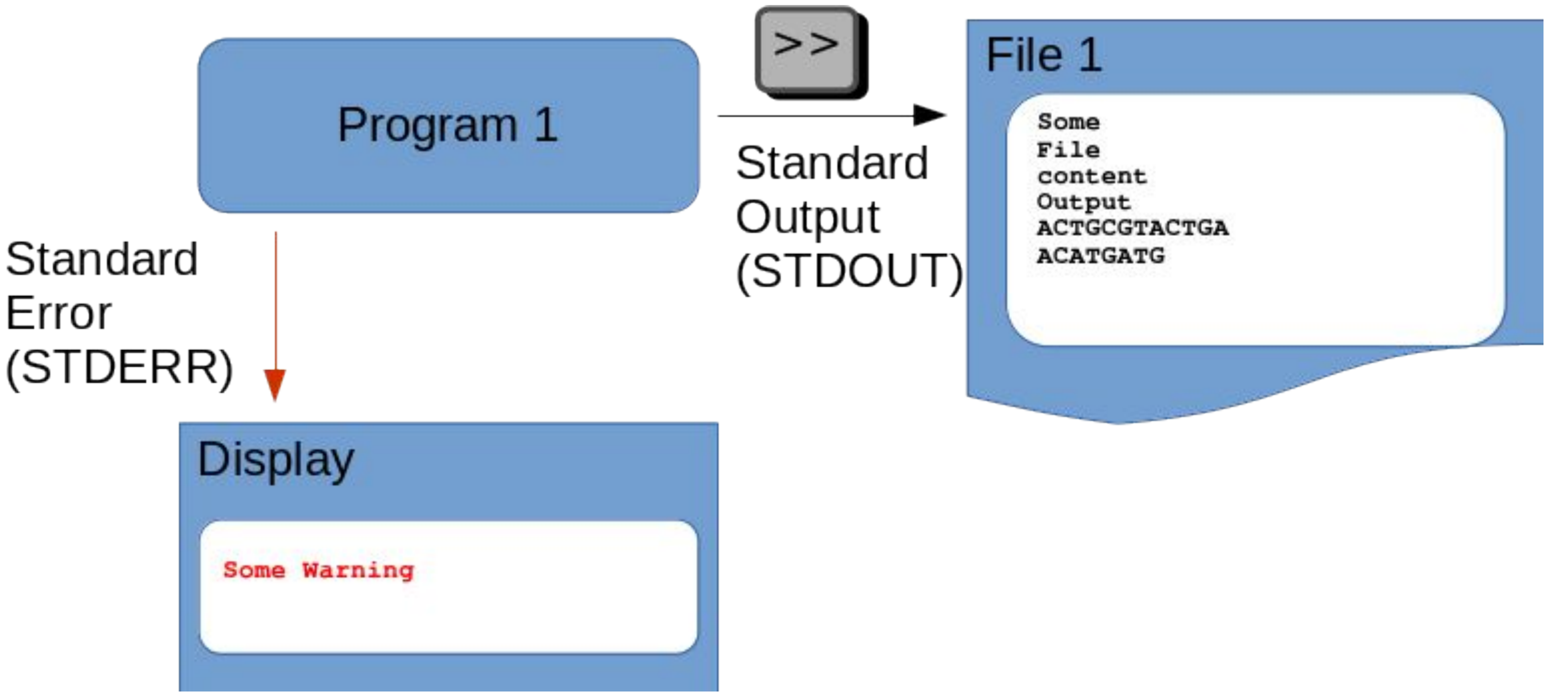


Using '>' again on the same file will overwrite its content!

Example: `ls -la ~ > myHomeDir.txt`

Linux Command Line Introduction - Part 2

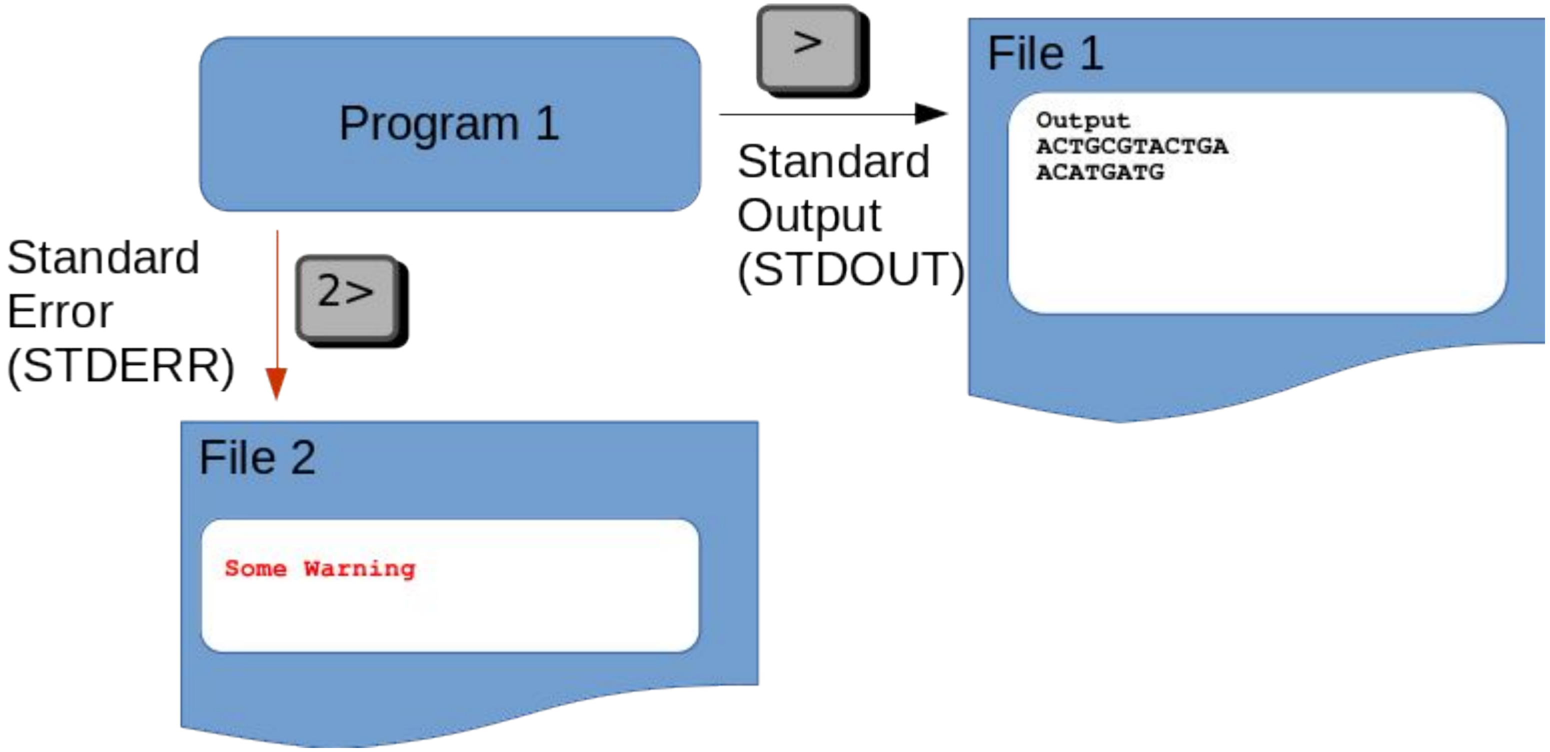
- Many unix commands support streams (files) as input and/or output
- Streams can be appended **to files**



Example: `ls -la ~ >> myHomeDir.txt`

Linux Command Line Introduction - Part 2

- Many unix commands support streams (files) as input and/or output
- You can also **redirect STDERR**

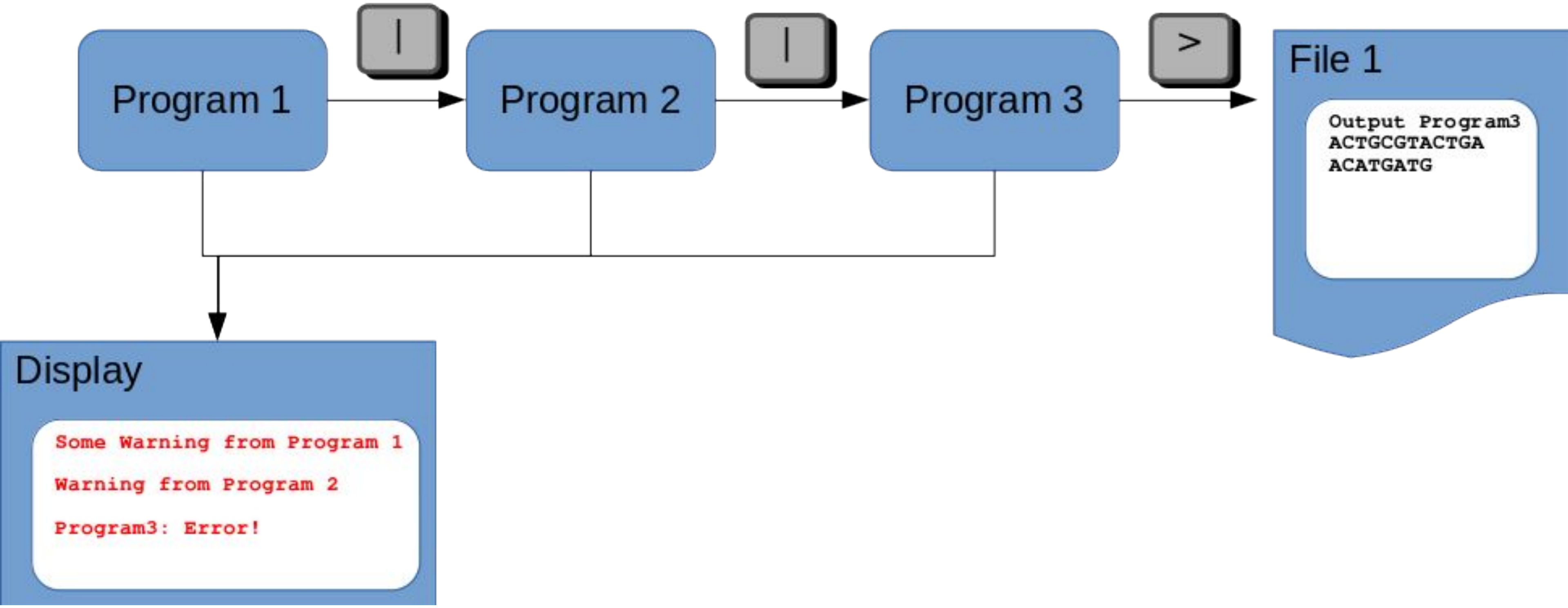


```
Output
ACTGCGTACTGA
ACATGATG
```

```
Some Warning
```

Linux Command Line Introduction - Part 2

- Many unix commands support streams (files) as input and/or output
- You can build small **pipelines** by ‘connecting’ tools



```
Output Program3
ACTGCGTACTGA
ACATGATG
```

```
Some Warning from Program 1
Warning from Program 2
Program3: Error!
```

Example: `ls ~ | grep .txt | sort`

Linux Command Line Introduction - Part 2

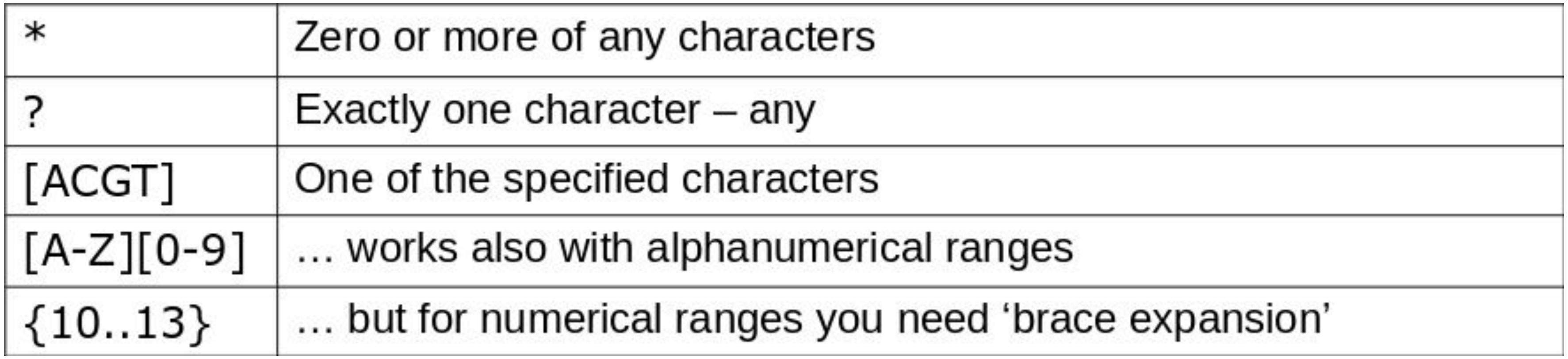
- Many unix commands support streams (files) as input and/or output
- Streams can be **redirected**
- **Into files**
  - Stdout: Text that is produced by the program can be redirected by using “>”.
    - Example: ./myProgram > output.txt
  - Stderr: Error messages are send through stderr and can be redirected using “2>”.
    - Example: ./myProgram 2> error.txt
  - Stdin: The stdin input stream accepts text as input that can be provided by using Linux pipes
    - Example: ./myProgram1 | ./myProgram2
- Linux processes
  - You can list all running programs using “**`ps`**”
  - You can terminate processes by using “**`kill`** id_of_process”.

Linux Command Line Introduction - Part 2

- **Search** for patterns:
    - **`grep`**: Search for specific patterns
        - Example: **`grep “error” output.log`**
        - Search for the term “error” in output.log and displays every line that contains this term
        - Special characters in the pattern:
            - “^”: denotes the beginning of the line
            - “$”: denotes the end of a line
- Search and replace:
    - **`sed`**: search and replaces a pattern in a file (stream)
        - syntax: **`sed ‘s/pattern/replacement/’`**

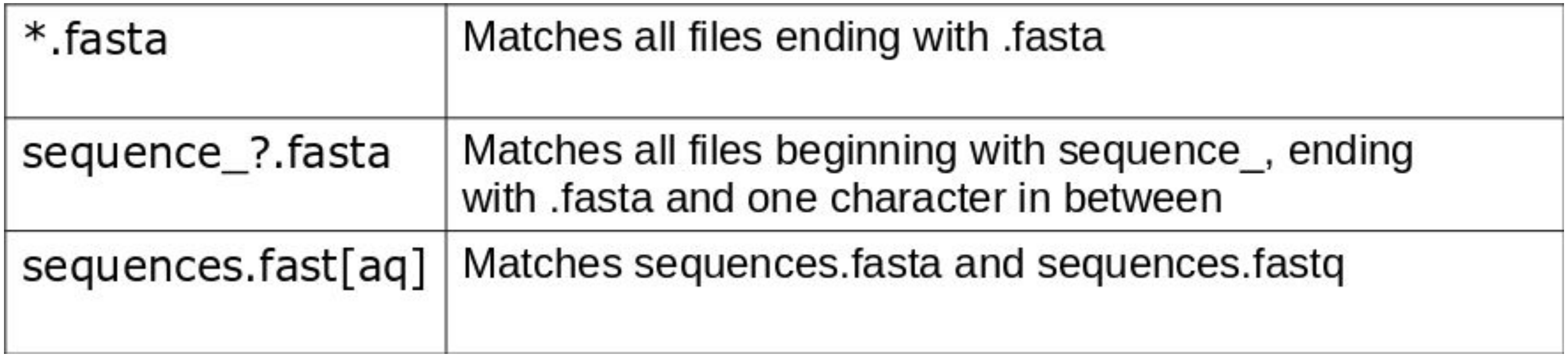
Linux Command Line Introduction - Part 2

- **Wildcards** can be used perform actions on multiple files at the same time
- Common wildcards:



*	Zero or more of any characters
?	Exactly one character – any
[ACGT]	One of the specified characters
[A-Z][0-9]	… works also with alphanumerical ranges
{10..13}	… but for numerical ranges you need ‘brace expansion’

- Example:



*.fasta	Matches all files ending with .fasta
sequence_?.fasta	Matches all files beginning with sequence_, ending with .fasta and one character in between
sequences.fast[aq]	Matches sequences.fasta and sequences.fastq

Linux Command Line Introduction - Part 2

- Your task is to **identify the correct commands** by using the list of commands and execute them. Feel free to experiment. Take a look at the solution **if absolutely necessary.**
- Workshop Material: https://github.com/deNBI/unix-course
- Please just do part 2 of the tutorial!
- ~20 Minutes

Acknowledgements

Cloud Governance Team and Cloud Portal Team:

- David Weinholz
- Peter Belmann
- Nils Hoffmann
- Viktor Rudko