

The Console Toolkit

Nils Leuzinger

TheAlternative, SSC | ETHZ and UZH

HS 2020

Why the console? It's already current year!

- People asked for it
- Still an excellent interface for complex tasks
- Many advanced tools only available on the command line
- Can be much faster than GUI tools
- Allows easy remote access to other computers
- Skills portable to every Unix-like system

Course outline

Part 1

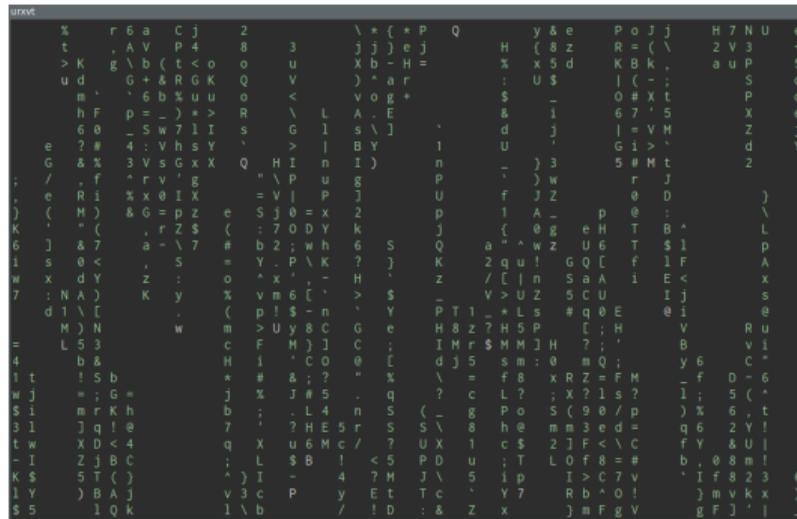
- Getting used to the console
- Navigating the file system
- Modifying files
- Working with text files
- Users & permissions

Part 2

- Managing software
- SSH
- Git
- Backups
- Scripting

What is the console

- Keyboard-only interface to your computer
- Related terms: *console*, *terminal*, *shell*, *bash*, *command prompt*, *command line interface*

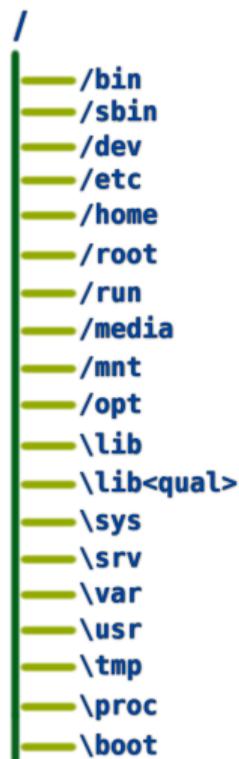
A screenshot of a terminal window displaying a complex ASCII art graphic. The graphic is a dense grid of characters, including letters, numbers, and symbols, arranged in a way that suggests a stylized face or abstract pattern. The terminal prompt is visible at the top left, and the cursor is positioned at the end of the last line of the art.

The 70s were interesting times, or so I'm told

Unix dogma: *Everything is a file!*

- Data files
- Directories (or “folders”)
- Storage devices
- Keyboards
- Printers
- Cameras
- *But not network sockets ...*

File system



[1]

- File system organized as tree
- Everything under /, the root directory
- In the console, you will be at some point in the tree, the *working directory*

Working directory

- Where am I? → `pwd`
- Present working directory
- Also sometimes directly shown in the prompt

```
[luke@host ~]$ pwd
/home/luke
[luke@host ~]$
```

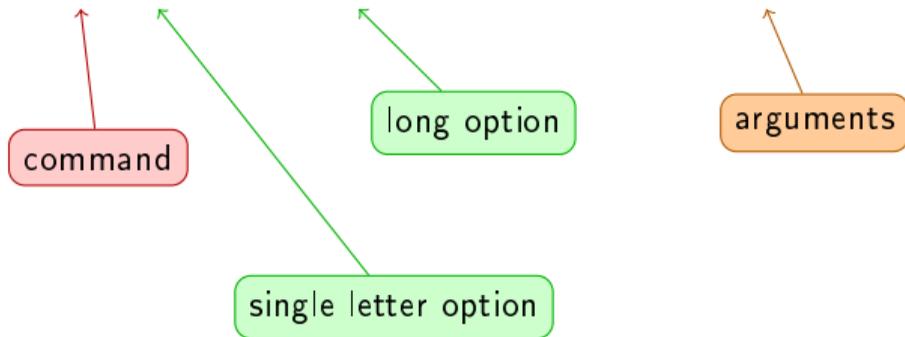
Listing files

- What is in here? → `ls`
- *list*

```
[luke@host ~]$ ls
Desktop
Documents
Downloads
Music
Pictures
Videos
cat1.jpg
cat2.jpg
[luke@host ~]$
```

Commands & arguments

```
ls -a --human-readable /home/luke/pictures
```



Advanced listing

- `ls` has advanced options
- `-a`: show hidden files
- `-h`: print numbers in human readable format
- `-l`: show the long output format.
- `ls -lah` and `ls -l -a -h` are equivalent

```
[luke@arch-x270 ~]$ ls -lah
total 52K
drwx----- 8 luke luke 4.0K Sep  3 23:27 .
drwxr-xr-x  4 root root 4.0K Sep  3 23:26 ..
-rw-r--r--  1 luke luke   21 Jun  4 10:54 .bash_logout
-rw-r--r--  1 luke luke   57 Jun  4 10:54 .bash_profile
-rw-r--r--  1 luke luke  141 Jun  4 10:54 .bashrc
-rw-r--r--  1 luke luke    0 Sep  3 23:27 cat1.jpg
-rw-r--r--  1 luke luke    0 Sep  3 23:27 cat2.jpg
drwxr-xr-x  2 luke luke 4.0K Sep  3 23:27 Desktop
drwxr-xr-x  2 luke luke 4.0K Sep  3 23:27 Documents
drwxr-xr-x  2 luke luke 4.0K Sep  3 23:27 Downloads
-rw-r--r--  1 luke luke   24 Sep  3 23:28 .hidden_file
drwxr-xr-x  2 luke luke 4.0K Sep  3 23:27 Music
drwxr-xr-x  2 luke luke 4.0K Sep  3 23:27 Pictures
-rw-r--r--  1 luke luke 3.7K Oct 23  2017 .screenrc
drwxr-xr-x  2 luke luke 4.0K Sep  3 23:27 Videos
[luke@arch-x270 ~]$
```

Getting help

- Where can I find out what options are available?
- Manual pages!
- `man`
- E.g.: `man ls`

LS(1)

User Commands

LS(1)

NAME

ls - list directory contents

SYNOPSIS

ls [OPTION]... [FILE]...

DESCRIPTION

List information about the FILEs (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.

-a, --all

do not ignore entries starting with .

-A, --almost-all

do not list implied . and ..

--author

with -l, print the author of each file

-b, --escape

print C-style escapes for nongraphic characters

More on manual pages

- Search by typing `/`
- Quit by typing `q`
- Sometimes there are multiple manuals! → Choose the right section
 - ▶ `man 1 printf` vs. `man 3 printf`

Go somewhere else

- I want to go to some other directory!
→ `cd`
- *Change directory*
- Absolute path: Whole path from the root, like:
`/home/luke/pictures/cat1.png`
- Relative path: Path relative to the current working directory, like
`pictures/cat1.png`

```
[luke@host ~]$ cd
[luke@host ~]$ pwd
/home/luke
[luke@host ~]$ cd /sys
[luke@host sys]$ pwd
/sys
[luke@host sys]$ cd ~
[luke@host ~]$ pwd
/home/luke
[luke@host ~]$ cd pictures/
[luke@host pictures]$ pwd
/home/luke/pictures
```

Special places

- ~ User's home directory
- . The current directory
- .. The directory above in the tree

Copying files

- Copy command: `cp`
- Syntax: `cp source destination`

```
[luke@host ~]$ cp diary diary_copy
[luke@host ~]$ cat diary_copy
Dear diary, today I downloaded
cat pictures from the internet.
[luke@host ~]$
```

Moving files

- Move command: `mv`
- Syntax: `mv source destination`
- Use `mv` to rename files

```
[luke@host ~]$ mv diary secret_diary
[luke@host ~]$ cat secret_diary
Dear diary, today I downloaded
cat pictures from the internet.
[luke@host ~]$
```

Creating and deleting directories

- `mkdir` creates a new directory
- `rmdir` removes a directory
- `rmdir` only works for empty directories!

```
[luke@host ~]$ mkdir new_dir
[luke@host ~]$ ls
new_dir
[luke@host ~]$ rmdir new_dir
[luke@host ~]$ ls
[luke@host ~]$
```

Deleting files

- `rm` removes files and directories
- `rm -r` removes a directory and everything in it (*recursive*)
- `rm` is **irreversible!**

```
[luke@host ~]$ ls
cat1.jpg
cat2.jpg
[luke@host ~]$ rm cat1.jpg
[luke@host ~]$ ls
cat2.jpg
```

`rm` is a shotgun without safety! There is no trashcan. You can delete your entire file system with `sudo rm -rf /`, or your entire home directory with `rm -rf ~!`

Hidden files

- Hidden files start with a dot
- `.hidden_file`
- Show them using `ls -a`

Showing text files

- Output a file's contents to the console with `cat`
- Used to stand for *concatenate*

```
[luke@host ~]$ cat diary
Dear diary, today I downloaded
cat pictures from the internet.
[luke@host ~]$
```

Reading long files

- What if the text doesn't fit on the terminal?
- Use the `less` file viewer
- Scroll up and down with `↑`, `↓`
- Exit with `q`

Editing files

- Need a *text editor*!
- *nano, vim, vis, emacs*
- Simple, intuitive, no learning required? → nano
- Powerful, efficient and extreme nerd cred? → vim
- Obscure, eccentric and even more powerful? → emacs
- Has some advantages to using a big GUI tool
 - ▶ Navigation and editing in the same interface
 - ▶ Quick and efficient
 - ▶ Very powerful tools available
 - ▶ You can talk down on people using Notepad++

Nano

- Syntax: `nano [filename]`
- Key bindings shown on the bottom
- Save: `ctrl` + `o`
- Save: `ctrl` + `x`
- Navigate with arrow keys `←` `↓` `↑`
`→`
- `^` stands for the `ctrl` key (universal)

```
[luke@host ~]$ nano diary.txt
```


Users & permissions

- Linux is a *multi-user operating system*
- There can be many user accounts
- Different users can even use the computer at the same time!
- You usually only use your personal user account

Users

Personal user

- Home directory in `/home/userx`
- Can only access files in home directory
- Can only stop processes started by itself

Root user

- Also called *the superuser*
- "System administrator"
- Can do anything on the system
- Access to all files
- Can kill any process
- Home directory in `/root`

Unix file permissions

drwxr-xr-x	2	luke luke	4.0K Oct 13 14:18	.
drwx--x---+	118	luke luke	36K Oct 13 14:15	..
-rw-r--r--	1	luke luke	39M Oct 13 14:17	cat1.jpg
-rw-r--r--	1	luke luke	45M Oct 13 14:17	cat2.jpg
-rw-r--r--	1	luke luke	36 Oct 13 14:18	diary
-rw-r--r--	1	luke luke	26 Oct 13 14:15	.secret

Permissions

Links

Owner | Group

Size | Last change

Name

Permissions

-rwxr-xr-x

The diagram shows the permissions string **-rwxr-xr-x** with four curly braces underneath. The first brace is under the hyphen and labeled 'Type'. The second brace is under 'rwx' and labeled 'Owner'. The third brace is under 'r-x' and labeled 'Group'. The fourth brace is under 'r-x' and labeled 'World'.

Type Owner Group World

- **r**: Permission to read file
- **w**: Permission to write to file
- **x**: Permission to execute file

Changing permissions

```
chmod {u,g,a}{+,-}{r,w,x} file  
chown OWNER:GROUP file
```

```
chmod +x program.sh  
chmod u-x program.sh  
chmod g+rw file.txt  
chown luke:luke file.txt
```

- chmod: Change permissions
- chown: Change owner, group

- Allow everyone to execute
- Remove execution permission for user
- Allow group to read/write
- Change ownership to user luke, group luke

Octal permissions

```
chmod 744 program.sh
```

- 4: read
- 2: write
- 1: execute

- Octal representation allows setting all permissions in one go
- Desired permissions are added up
- 7 = read + write + execute

Tab completion

- Hit  to automatically complete a word you are typing (Command, file, ...)
- Hit  twice to show all possible options
- Extremely useful terminal feature! Use always!

Command History

- Scroll up in your command history by pressing the `↑` key
- Press `ctrl` + `r` to search the history

Killing a running process

- Every process has a unique ID on Linux
- View processes with `ps aux`
- Kill a process with `kill id`
- If ID is unknown, use `pkill name`
- The `-9` flag works like a shotgun

```
[luke@host ~]$ ps -e
PID TTY          TIME CMD
  1 ?             00:00:04 systemd
  2 ?             00:00:00 kthreadd
[luke@host ~]$ kill 16740
[luke@host ~]$ pkill -9 emacs
```

The PATH variable

- How does the shell know where programs are?
- The shell searches the PATH variable
- `ls` → `/usr/bin/ls`

```
[luke@host ~]$ echo $PATH  
/usr/sbin:/usr/bin
```

Adding your own paths

- Let's say you want to add your script directory
- Temporarily: `export PATH=$PATH:~/scripts`
- Permanently: Add the above to `~/.bashrc`

Writing shell scripts

- Scripts are just a sequence of commands
- Very easy automation!
 - ▶ (for more, come to the Bash course)

File `/scripts/music.sh`:

```
#!/usr/bin/env bash
filename="$2.%{(ext)s}"
echo "$1"
youtube-dl -x "$1" -o "$filename"
```

```
[luke@host ~]$ cd scripts
[luke@host ~]$ chmod +x music.sh
[luke@host ~]$ ~/scripts/music.sh "https://www.youtube.com/watch?v=dQw4w9WgXcQ" music
# or:
[luke@host ~]$ ./music.sh "https://www.youtube.com/watch?v=dQw4w9WgXcQ" music
```

Complicated needs

How would you design an interface that can...

- ...delete files larger than 100MB?
- ...show the last 2 lines of a file?
- ...sort files by length?
- ...search calendar entries and create reminders?

Complicated needs

Many possible answers:

- Big GUI that does everything
- A simple tool that users can extend themselves
- Domain specific language that users write queries with
- Many simple and combinable tools

Unix chooses the last two approaches

Piping and redirection

- Unix has small and orthogonal tools
- Piping and redirection are how to combine them

Piping and redirection

- Piping sends output from one command to another command
- Redirection writes to files (streams)

```
[luke@host ~]$ cat numbers | sort  
five  
four  
one  
three  
two  
zero
```

```
[luke@host ~]$ cat numbers > same_numbers
```

Piping

- Uses the pipe symbol: |
- Useful for sequential composition
- Only works in "one direction"
- Internally connects output of one process to output of other process

```
[luke@host ~]$ cat numbers | tail -n 2  
nine  
ten
```

```
[luke@host ~]$ cat numbers | grep "..."  
one  
two  
six  
ten
```

Piping

List unique owners of files in current directory:

- List files in directory
- Omit first two lines
- Truncate whitespace
- Cut (delete) all columns except the third
- Sort alphabetically
- Only show unique entries

```
[luke@host ~]$ ls -l | tail -n +2 | sed 's/\s\s*/ /g' | cut -d ' ' -f 3 | sort | uniq
```

Redirection

- `> file` Write output to file
- `>> file` Append output to file
- `< file` Read input from file

```
[luke@host ~]$ echo "Hello!" >
hello.txt
[luke@host ~]$ cat hello.txt
Hello World!
[luke@host ~]$ cat < hello.txt
Hello World!
```

Redirection

Redirection is useful!

- Store final or intermediate results
- Append output to files

```
[luke@host ~]$ ls -l | tail -n +2 | sed 's/\s\s*/ /g' > result  
[luke@host ~]$ cut -d ' ' -f 3 < result | sort | uniq
```

Redirection

Redirection is useful! (cont.)

- Store final or intermediate results
- Append output to files

```
[luke@host ~]$ ./logger_script >> log.txt
```

```
[luke@host ~]$ echo "Logging done!" >> log.txt
```

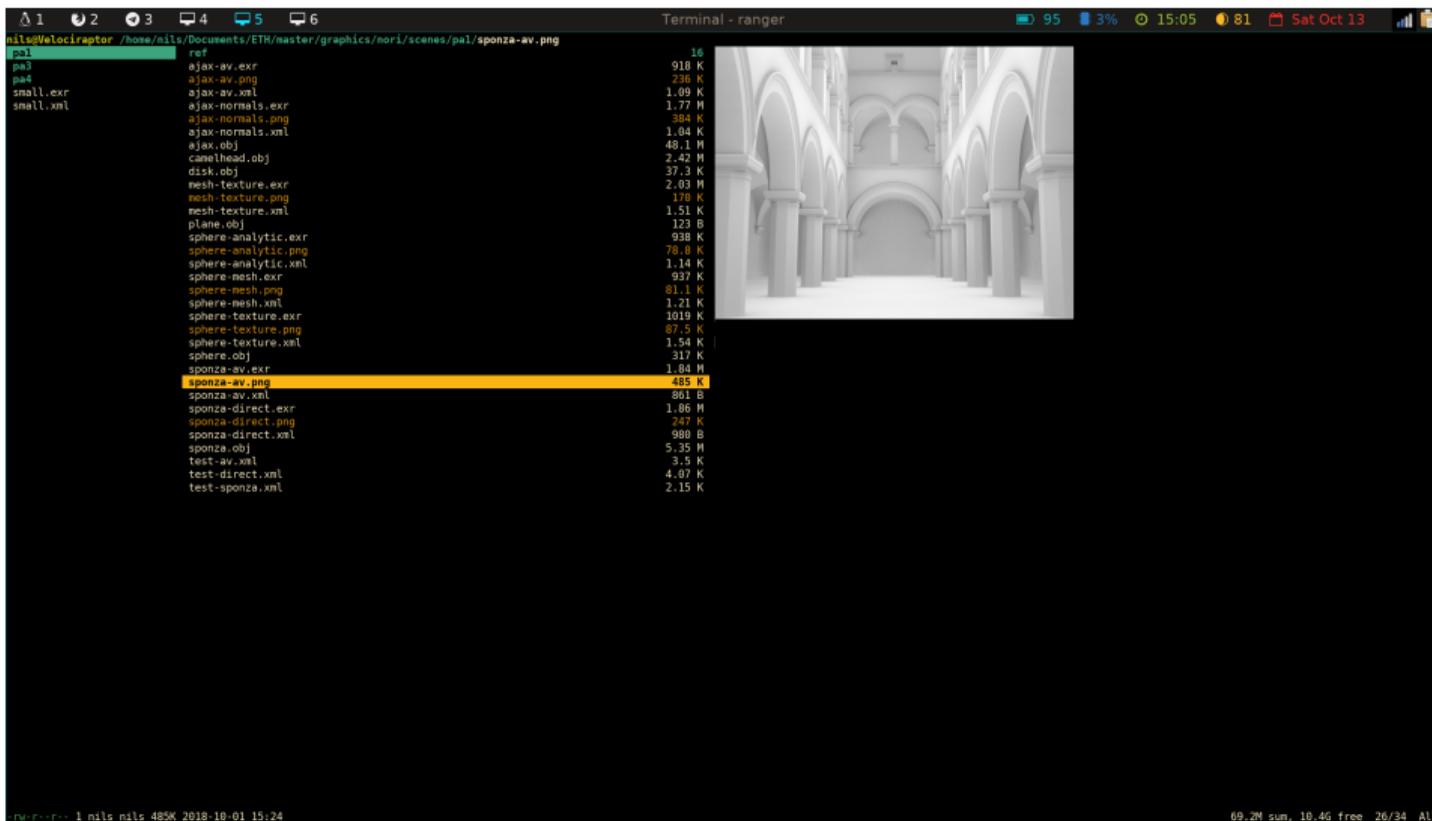
Ranger

- File manager on the console
 - ▶ Usable over SSH
- Can move files, change permissions, bulk rename...
- Bookmarks
- Keyshortcuts for frequent locations
- Plugins
- Preview functionality

```
luke@arch-x270: /home/luke/sync/eth/bachelor-thesis
eth bachelor-thesis 4 adsb
konkret Semester 1 4 docs
scripts Semester 2 5 lutebler_android_adsb
wallpaper Semester 3 4 NOTES.md
Semester 4 5
Semester_5 6
Semester_6 6
Semester_7 2
vin-course 11

drwxr-xr-x 5 luke luke 4 2018-10-15 13:53 0 sum, 58.5G free 1/9 All
```

Ranger image preview



The screenshot shows a terminal window with the Ranger file manager interface. The terminal title is "Terminal - ranger". The current directory is `nils@velociraptor /home/nils/Documents/ETH/master/graphics/mori/scenes/pal/sponza-av.png`. The file listing shows various files and their sizes, with `sponza-av.png` highlighted in yellow. To the right of the listing is a preview window showing a 3D rendering of a classical architectural interior with arches and columns.

File Name	Size
ref	16
ajax-av.exr	918 K
ajax-av.png	736 K
ajax-av.xml	1.09 K
ajax-normals.exr	1.77 M
ajax-normals.png	384 K
ajax-normals.xml	1.04 K
ajax.obj	48.1 M
camelhead.obj	2.42 M
disk.obj	37.3 K
mesh-texture.exr	2.03 M
mesh-texture.png	178 K
mesh-texture.xml	1.51 K
plane.obj	123 B
sphere-analytic.exr	938 K
sphere-analytic.png	78.8 K
sphere-analytic.xml	1.14 K
sphere-mesh.exr	937 K
sphere-mesh.png	81.1 K
sphere-mesh.xml	1.21 K
sphere-texture.exr	1019 K
sphere-texture.png	87.5 K
sphere-texture.xml	1.54 K
sphere.obj	317 K
sponza-av.exr	1.84 M
sponza-av.png	485 K
sponza-av.xml	861 B
sponza-direct.exr	1.06 M
sponza-direct.png	247 K
sponza-direct.xml	980 B
sponza.obj	5.35 M
test-av.xml	3.5 K
test-direct.xml	4.07 K
test-sponza.xml	2.15 K

Terminal status bar: `nils@ranger` 95% battery, 3% CPU, 15:05, Sat Oct 13. Bottom status bar: `69.2M` used, `10.4G` free, `26/34` All.

How you get software on Linux

- Don't download installers from the internet!
- Software is managed by the distribution and available through a central repository.
- Software is *packaged*
- Similar to Microsoft's or Apple's app stores



[2]

Installing packages

- Depends on distribution!
- Package Manager is most important feature of a Linux distribution

Debian, Ubuntu, Mint:

```
sudo apt install firefox
```

OpenSUSE:

```
sudo zypper install firefox
```

RedHat, Fedora:

```
sudo dnf install firefox
```

Searching for packages

Debian, Ubuntu, Mint:

```
apt search firefox
```

OpenSUSE:

```
zypper search firefox
```

RedHat, Fedora:

```
dnf search firefox
```

- The basic package search is usually quite limited
- Consult the internet for finding the right programs!

Updating packages

- All packages can be upgraded at once
- Do this every other week!

Debian, Ubuntu, Mint:

```
sudo apt update  
sudo apt upgrade
```

OpenSUSE:

```
sudo zypper update
```

RedHat, Fedora:

```
sudo dnf update
```

Building from source

- Sometimes software is unavailable in the repositories
- Can download sources and compile them manually
- Careful! No automatic updates, malware, package manager conflicts, ...

Building from source

- Download the sources
- Follow the build instructions in the documentation

```
git clone https://github.com/i3/i3
```

```
autoreconf -fi
```

```
mkdir -p build && cd build
```

```
../configure
```

```
make
```

SSH

- *Secure shell*
- SSH allows to log in to another computer over the network
- Server administration, running jobs on supercomputers, log in to your computer at home



Using SSH

- `ssh alice@bob.ch`
- Will ask for user password

```
[luke@host ~]$ ssh alice@bob.ch
alice@bob.ch's password:
[alice@bob ~]$
```

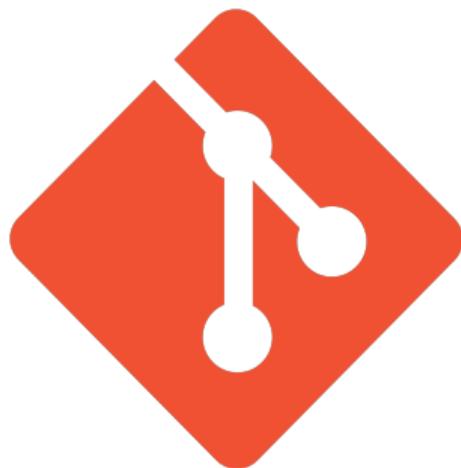
Login without password

Generate an SSH key with `ssh-keygen` and copy the key in `~/.ssh/id_rsa.pub` to `~/.ssh/authorized_keys` on the server.

Neat key copying trick when password login already works:

```
cat ~/.ssh/id_rsa.pub | ssh username@euler.ethz.ch 'cat - >> .ssh/authorized_keys'
```

Git



[4]

Git

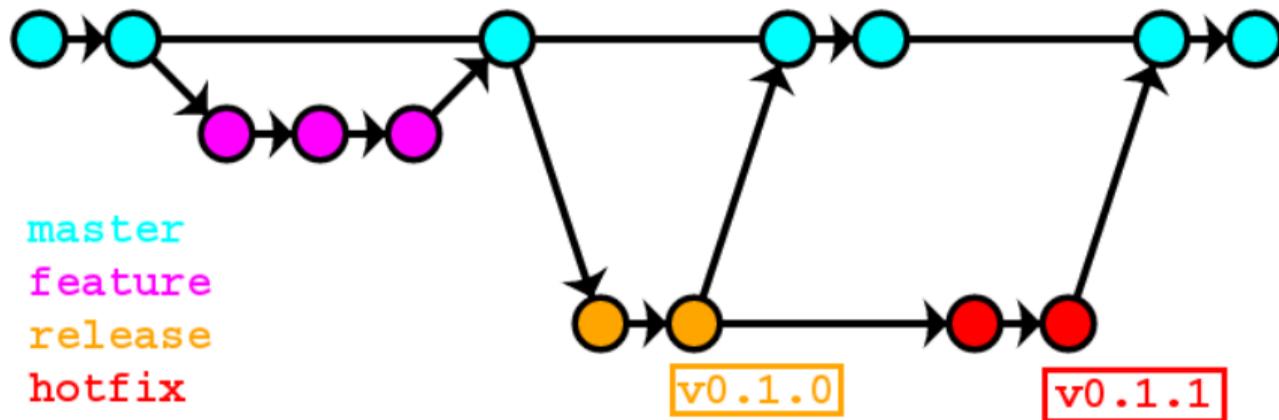
- `thesis.pdf`
- `thesis_old.pdf`
- `thesis_copy.pdf`
- `thesis_finalversion.pdf`
- `thesis_finalversion2.pdf`

Git

Git is a version-control system for tracking changes in computer files and coordinating work on those files among multiple people.

— Wikipedia

Git



[3]

Git

- Track changes to your code
- Comment your changes
- Easily revert back to older versions
- Avoid/manage conflicts when working in teams
- Manage release versions and development versions
- Work on different branches at the same time

Git example

- Initialize a git repository
- Add files you want to include in a commit
- Create a commit for your selected changes
- Push changes to server

```
git init
```

```
git add changed_file.txt
```

```
git commit
```

```
git push
```

Sharing a repository

- You can use services like *Github* or *Gitlab* to collaborate with others on your project
- Or host your repositories yourself!
 - ▶ You can pull from/push to any server you can access via SSH

Borg Backup

- Backup data
- Does compression and deduplication automatically
- Possibility to encrypt
- Runs over ssh, network storage etc.



Borg Example

- Repos are collections of backups
- Create new backups with `borg create`
- Restore files with `borg extract <archive>`

```
[luke@host ~]$ borg init --encryption=repokey /path/to/repo
Enter passphrase:
Repeat passphrase:
[luke@host ~]$ borg create /path/to/repo
[luke@host ~]$ borg create --stats /path/to/repo::Backup1 ~/dir_to_backup
[luke@host ~]$ borg create --stats /path/to/repo::Backup2 ~/dir_to_backup
[luke@host ~]$ borg extract /path/to/repo::Backup1
```

Sources

- 1 https://commons.wikimedia.org/wiki/File:Linux_Filesystem_Hierarchy_Standard.png
- 2 <https://en.wikipedia.org/wiki/File:Gnome-emblem-package.svg>
- 3 https://commons.wikimedia.org/wiki/File:OneFlow_Example.png
- 4 <https://git-scm.com/images/logos/logomark-orange@2x.png>