

The Console Toolkit - Part 2

Alexander Schoch

TheAlternative, SSC | ETHZ and UZH

25th of October, 2019

Recall

- `cd [dst]` to change the working directory
 - `ls` to list the content of a directory
 - `chmod [opt] [file]` to change permissions
 - `cp [src] [dst]` to copy, `mv [src] [dst]` to move/rename
 - `man [command]` to learn more about a command
- ⇒ The slides from monday can be found at
<https://gitlab.ethz.ch/thealternative/courses>

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

```
[alex@theAlt: ~]$ ps -e | head -n 3
PID TTY          TIME CMD
1 ?            00:00:04 systemd
2 ?            00:00:00 kthreadd
[alex@theAlt: ~]$ kill 16740
[alex@theAlt: ~]$ pkill -9 firefox
```


sudo

- sudo stands for superuser do
- run any command as root user
 - ▶ does not work for scripts
 - ▶ scripts can be made executable using `sudo chmod +x [file]`
- run `sudo su` to change the user to root
- Handy trick: use `sudo !!` to run the last command with `sudo`

```
[alex@theAlt: ~]$ touch /etc/file.txt
touch: cannot touch '/etc/file.txt':
  Permission denied
[alex@theAlt: ~]$ sudo touch /etc/file.txt
[sudo] password for alex:
[alex@theAlt: ~]$
```

The legendary sudo warning

```
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:
```

- #1) Respect the privacy of others.
- #2) Think before you type.
- #3) With great power comes great responsibility.

```
root's password:
```

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)

```
[alex@theAlt: ~]$ cat numbers | sort  
five  
four  
one  
three  
two  
zero
```

```
[alex@theAlt: ~]$ 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

```
[alex@theAlt: ~]$ cat numbers | tail -n 2  
nine  
ten
```

```
[alex@theAlt: ~]$ 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

```
[alex@theAlt: ~]$ 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

```
[alex@theAlt: ~]$ echo "Hello!" >
hello.txt
[alex@theAlt: ~]$ cat hello.txt
Hello World!
[alex@theAlt: ~]$ cat < hello.txt
Hello World!
```

Redirection

Redirection is useful!

- Store final or intermediate results
- Append output to files

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

Redirection

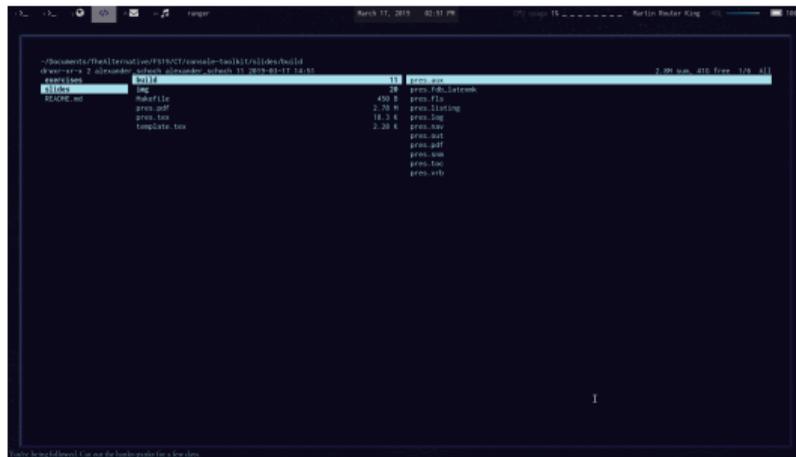
Redirection is useful! (cont.)

- Store final or intermediate results
- Append output to files

```
[alex@theAlt: ~]$ ./logger_script >> log.txt  
[alex@theAlt: ~]$ 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



Screenshot by Alexander Schoch - CC BY 4.0

Ranger image preview

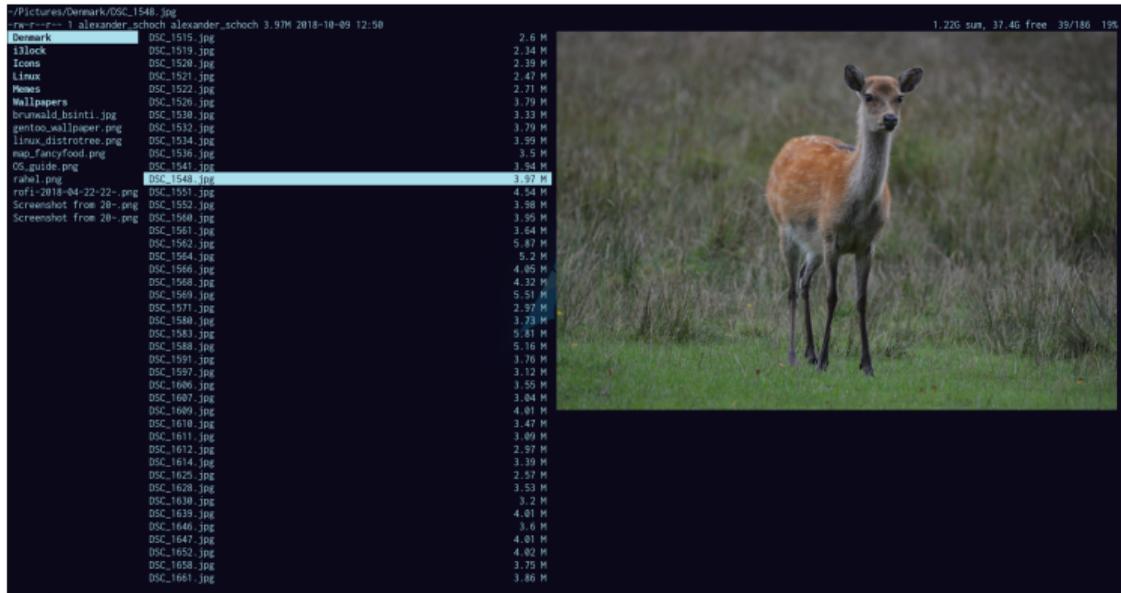


Photo by Philipp Ziegler - CC BY-SA 4.0, Screenshot by Alexander Schoch - CC BY-SA 4.0

youtube-dl & mpv

youtube-dl

- download youtube videos/playlists/channels using `youtube-dl [url]`
- download music from youtube using `youtube-dl [url] --format=bestaudio`
- `youtube-dl 'ytsearch1:basie straight ahead'` will download the first result

mpv

- video/audio player
- `Space` to play/pause, `↶` to skip, `<` `>` to go back/skip, `←` `→` to seek, etc.
- can stream video/audio from youtube and many other sites using `youtube-dl`

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



Picture by GNOME icon artists - CC BY-SA
3.0 via Commons [1]

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
```

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

```
../configure
```

```
make
```

```
sudo make install
```

Raspberry Pi



Image by Michael Henzler - CC BY 4.0 via Commons [2]

But... what is that?

- very minimal computer
- costs about 30 CHF
- normally runs Raspian
- handy to use via command line

What can you do with it?

- pi-hole
- mumble Server
- music Server
- use as an example device for your LinuxDays course :)
- ...

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



Logging in to Euler (cluster)

```
[alex@theAlt: ~]$ ssh schochal@euler.ethz.ch  
schochal@euler.ethz.ch's password:
```



```
Eidgenoessische Technische Hochschule Zuerich  
Swiss Federal Institute of Technology Zurich
```

```
-----  
E U L E R   C L U S T E R
```

```
https://scicomp.ethz.ch  
http://tinyurl.com/cluster-support  
cluster-support@id.ethz.ch
```

```
=====  
[schochal@eu-login-14-ng ~]$
```

Using SSH

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

```
[alex@theAlt: ~]$ ssh alice@bob.ch
alice@bob.ch's password:
[alex@theAlt: ~]$
```

Let's try that!

- log into the Raspberry Pi
 - ▶ IP address: 10.5.88.112
 - ▶ username: pi
 - ▶ password: thealternative
- go to the Desktop folder
- create an empty file with your surname as the filename

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'
```

scp

- stands for "secure copy"
- works like a combination of `ssh` and `cp`
- access files of remote machines with the `ssh` syntax, write a `:` and enter the path
- use `scp -r [source] [destination]` for folders

```
[alex@theAlt: ~]$ scp alice@bob.ch  
:Documents/file.txt Documents/  
alice@bob.ch's password:  
[alice@bob.ch: ~]$
```

X11 forwarding

- display remote GUIs on your machine
- use the `-Y` flag to activate it

```
[alex@theAlt: ~]$ ssh -Y
  schochal@euler.ethz.ch
schochal@euler.ethz.ch's password:
[schochal@eu-login-01-ng: ~]$
  module load matlab
[schochal@eu-login-01-ng: ~]$
  matlab &
```

Git

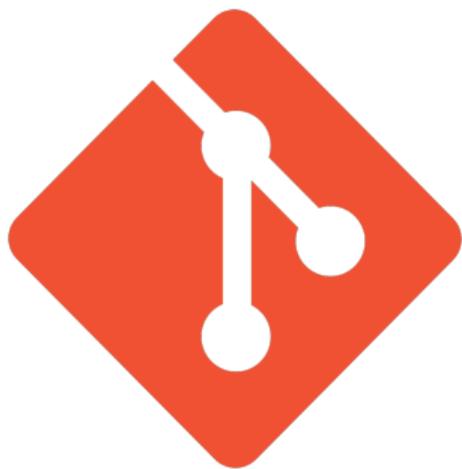


Image by Jason Long - CC BY 3.0 via Commons [3]

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

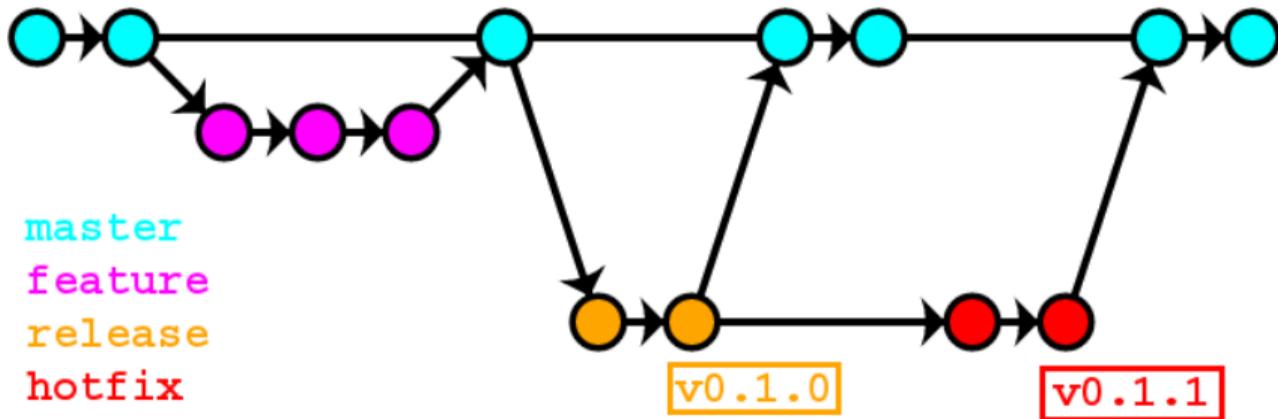


Image by Qeef - CC BY-SA 4.0 via Commons [4]

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>`

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

Some fun to end with

fortune

- Install `fortune-mod`
- `fortune` prints some random quote/joke/fortune/whatever

```
[alex@theAlt: ~]$ fortune  
I'm not laughing with you, I'm laughing at you.
```

- pipe it through `cowsay` and `lolcat` in order to get maximum fun effect

fortune (cont.)

```
[alex@theAlt: ~]$ fortune | cowsay | lolcat
```

```
-----  
/ Apathy is not the problem. It is the \  
\ solution!                               /  
-----
```

```
 \      ^  ^  
  \    --  ^  
   (oo)\_____    
   (__) \         )\/\   
        ||-----w |   
        ||         ||
```

dunnet

- text adventure game
- download emacs and run `emacs -batch -l dunnet`

```
[alex@theAlt: ~]$ emacs -batch -l dunnet
```

```
Dead end
```

```
You are at a dead end of a dirt road. The road goes to the east.
```

```
In the distance you can see that it will eventually fork off. The
```

```
trees here are very tall royal palms, and they are spaced equidistant  
from each other.
```

```
There is a shovel here.
```

```
>
```

Sources

- 1 <https://en.wikipedia.org/wiki/File:Gnome-emblem-package.svg>
- 2 https://en.wikipedia.org/wiki/File:Raspberry_Pi_4_Model_B_-_Side.jpg
- 3 https://commons.wikimedia.org/wiki/File:Git_icon.svg
- 4 https://commons.wikimedia.org/wiki/File:OneFlow_Example.png

Last Slide

⇒ Please fill out the feedback form on <https://feedback.thealternative.ch>

License

- Slides by Lukas Tobler/Alexander Schoch - CC BY-SA 4.0
- Slide Template by Christian Horea - Public Domain

What now?

- Bash scripting course on Tue, 29th of October, 2019 at 05:15 pm at HG F 7
- Join our Stammtisch on Thu, 31st of October, 2019 at 06:00 pm at Aki