

Welcome to the Bash Workshop!

- ▶ If you prefer to work on your own, already know programming or are confident in your abilities, please **sit in the back**.
- ▶ If you prefer guided exercises, are completely new to programming, or want to have your hands held, please **sit in the front**.

Bash Workshop

Aline Abler



Bash is a not a programming language.

- ▶ It is a **shell**
- ▶ It is an interface between you and your computer
- ▶ It allows you to access the operating system's services (i.e. run programs)
- ▶ **It is not designed as a programming language**, but can be used as such

Bash is not the only shell there is.

- ▶ sh, ksh, zsh, fish, dash. . .
- ▶ Most commands work in any shell, but **some don't**
- ▶ Usually this doesn't matter too much

Bash scripts are not programming

- ▶ Glue existing programs together to create new ones
- ▶ It's possible to write entire programs, but please don't

Look, a bash script

```
# !/bin/bash

echo 'Hello World'

echo Bash is awesome

# Now some fun stuff:
sudo zypper update
notify-send 'Update complete'
feh --bg-fill 'pictures/fancy_wallpaper.jpg'
youtube-dl -o 'Video.%(ext)s' 'https://www.youtube.com/watch?v=1AIGb11fpBw'
```

What bash can do

- ▶ Automate anything you can do from a console
- ▶ Let several separate programs work together

All about strings

Everything is a string

Strings and word splitting

- ▶ A string is a sequence of characters that is **treated as a unit**
- ▶ Commands are strings, too
- ▶ Strings are split at every space, tab, or newline **unless they're in quotes**

Meaning of strings

```
echo Hello World
```

- ▶ `echo`, `Hello` and `World` are single strings
- ▶ The first string becomes the command, all following become **arguments**

```
echo 'Hello World'
```

- ▶ Here, `Hello World` is just one string

Repeat after me

**Every word is a single argument
unless you use quotes.**

All about commands

**Everything that does something is
a command**

Why is this so important?

- ▶ `if` and `while` are commands
- ▶ `[[` is a command

Example

- ▶ wrong:

```
[[1==3]]
```

- ▶ Bash's answer:

```
bash: [[1==3]]: command not found
```

- ▶ correct:

```
[[ 1 == 3 ]]
```

Repeat after me

If there's brackets, you probably need spaces.

All about return values

Every command returns a value

Return values

- ▶ Every command returns a number, its **return value**
- ▶ 0 means success
Everything else means there was an error
- ▶ Some commands also print to **stdout** - that's what you see.

Return values

▶ What you run

```
echo 'Hello World'
```

▶ What you see

```
Hello World
```

▶ What bash sees

```
0
```

Why is that important?

- ▶ `&&`, `||`, `if` and `while` all act based on the return value of something
- ▶ They disregard that command's actual output

Example

```
if ls -l foo
then
    echo 'File foo exists'
else
    echo 'File foo does not exist'
fi
```

Bash doesn't only run commands

▶ Tilde expansion

`~/files` becomes `/home/alinea/files`

▶ Variable expansion

`$BROWSER` becomes `Firefox`

▶ Arithmetic expansion

`$((1 + 4))` becomes `5`

▶ Command substitution

`$(pwd)` becomes `/home/alinea/scripts`

▶ Pathname expansion (or globbing)

`files/qui*` becomes `files/quicknotes files/quiz`

Bash doesn't only run commands

- ▶ Expansion happens before any command is run
- ▶ Double quotes (") don't prevent expansion, but single quotes (') do.

```
$ echo "$HOME" '$HOME'  
/home/alinea $HOME
```

- ▶ Expansion happens **before word splitting**

The issue with word splitting

```
var='Unimportant File.odt'  
rm $var # Variable expansion
```

becomes

```
rm Unimportant File.odt
```

The issue with word splitting

▶ The correct way

```
rm "$var"
```

Repeat after me:

**If there's a dollar, you probably
need quotes!**

How to write a bash script

- ▶ I want a script that searches youtube and downloads the first video it finds

Google for a program that already does what you need

- ▶ `youtube-dl` can download a video from youtube

```
youtube-dl -o 'Video.%(ext)s' 'https://www.youtube.com/watch?v=lAIGb1lfpBw'
```

Splitting it up even further

- ▶ Search youtube
 - ▶ Download youtube results website
 - ▶ Search website text
 - ▶ Find first youtube video link
- ▶ Download video

Hands on!

Self-driven exercises

- ▶ Self study using a guide
- ▶ Try some of our exercises
- ▶ Choose the exercises you find interesting! No need to go in order.

Guided exercises

- ▶ Solve easy exercises in plenum
- ▶ Tailored to complete beginners
- ▶ Please sit in the front

Course material

- ▶ These slides, exercise sheet and bash guide:
<http://thealternative.ch/index.php?view=knowhow>
- ▶ Please leave some feedback!
<http://feedback.thealternative.ch>
- ▶ Theme by Christian Horea, CC BY