Zawiki

Release v0.11

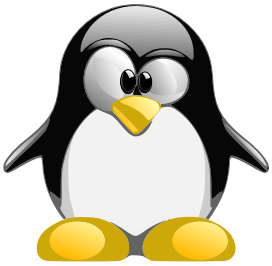
tschinz

Feb 02, 2020

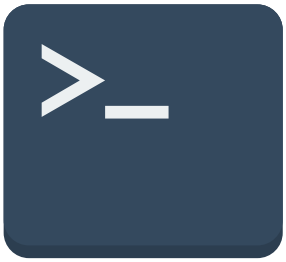
Contents

Chapter 1

Linux



1.1 Commandline



1.1.1 Cheatsheet

- *Admin rights*
- *Quit*
- *Mounting*
- *Wipe Disk*
- *Environment variables*
- *User*

Admin rights

Note: In order to gain administrative rights, for a session or only for a command

Listing 1: admin

```
sudo su
sudo "command"
sudo -s           # Longtime root
su -              # root preserved env
sudo !!           # execute previous command as root
chsh -s /bin/zsh  # change login shell to zsh
```

Quit

Listing 2: quit

```
sudo halt          # Sleep
sudo reboot        # Reboot
sudo shutdown now  # Shutdown
```

Mounting

For permanent mount see: */etc/fstab*

Listing 3: mounting

```
sudo vim /etc/fstab             # To edit default mount drives
sudo fdisk -l           # Drive info
ls -l /dev/disk/by-uuid         # get Drive UUID's
mkdir -p /media/d       # make folder for HD
mount -t vfat -o iocharset-utf8, umask=000 /dev/sda3 /media/d

mkdir -p /mnt/mountplace
mount /dev/sda1 /mnt/mountplace

unmount /mnt/mountplace

mount -U <UUID>                 # mount drive according to fstab definition

df -k                           # check partitions and the available space
```

Wipe Disk

Listing 4: wipe disk

```
# unmount disk
sudo umount /dev/sdXY -l

# use /dev/random to write Zeros on entire disk§
sudo dd if=/dev/urandom of=/dev/sdX bs=10M
```

Environment variables

They can be set permanently system wide */etc/profile* or per user shell */etc/.bashrc* */etc/.zshrc*

Listing 5: environment variables

```
# Licenses
export LM_LICENSE_FILE=$LM_LICENSE_FILE:portnumber@serverip

# Print Environmemt variables
printenv
echo $name_env_var

# Set env var
setenv name value
```

User

Listing 6: user

```
# Access to different PC with unknown Password
chroot path/of/new/systemroot    # change root of FileSys
# Useful for hacking another PC
# 1. with LiveUSB / CD login
# 2. mount HD
# 3. chroot to his filessystem
# 4. Change user / password and everything else you want

# User information
who                              # returns all users logged in
whoami                           # return actual username
id <username>                    # return groups & id's of user

# Change to user
sudo -u user2 bash               # open bash of user2

# Send info
write <username> <tty>           # write to a logged user
                                 # see command who output

# Add user
sudo useradd -d /home/<username> -m <username>

# Add user to group
usermod -a -G <groupname> <username>

# Change user password
sudo passwd <username>
```

Alias

Listing 7: alias

```
# Set up aliases
alias <aliasname>="<command>"
alias ll="ls -la"
```

Permissions

Listing 8: permissions

```
chmod xxx file|folder                # xxx = rwx|xnumber
chmod -x  file|folder                # add only executable Flag
chown -R user:group file|folder      # change owner recursively

find . -type d -exec chmod 755 {} \; # find dir's and set 755
find . -type f -exec chmod 644 {} \; # find files and set 644
```

Rights			
read	write	execute	Abreviation
•	•	•	0
•	•	x	1
•	x	•	2
•	x	x	3
x	•	•	4
x	•	x	5
x	x	•	6
x	x	x	7

Threads

PID = Process ID

Listing 9: threads

```
ps -x                                # view executed threads
ps -ax | grep name                   # search for specific process name
kill <pidnumber>                     # kill thread with given PID
kill signal <pidnumber>              # kill with a signal type see table below
```

Signal Name	Single Value	Effect
SIGHUP	1	Hangup
SIGINT	2	Interrupt from keyboard
SIGKILL	9	Kill signal
SIGTERM	15	Termination signal
SIGSTOP	17, 19, 23	Stop the process

General

Listing 10: general

```
uname -a                             # Distribution & Kernel informations
whereis command                      # returns location of command

mkdir /existing/path/dirname         # creates a directory
mkdir -p /non/existing/path/name     # creates a directory path
mkdir -p project/{lib/ext,bin,src,doc/{html,info,pdf},demo/stat/a}
                                     # creates a tree structure
pwd                                  # print working directory
ls                                   # list content
ls -la                               # list flags
ll                                   # short list flags
cd                                   # change dir
rm name                              # remove file
rm -r                                # remove directory with content
rm -R name                           # remove recursively folder with its content
rm !(delete_all_but_this)            # delete all except !()
cp source/path /dest./path           # copy file
cp -R source/path dest./path         # copy directory with content
cp -R --preserve=mode,ownership,timestamp  source/path dest/path
                                     # copy with preserving owner and permission and␣
↪time

df                                   # show disk sizes
df -H                                # show disk sizes in KB, MB, GB


diff path/to/file1 path/to file2     # compare file1<->file2 and shows the difference
sdiff path/to/file1 path/to file2    # compare file1<->file2 and merge directly
```

Find / Which

Listing 11: find and which

```
# finding and delete all folder with <foldername> and it's content
find -type d -iname "<foldername>" -exec rm -rf {} \;
# finding and delete all files with <filename> and it's content
find -type f -iname "<filename>" -exec rm -rf {} \;
# finding all files and directories within a directory
find /etc
# finding all files within a directory
find /etc -type f
# finding all files with a suffix
find /etc -type f -name "*.conf"

# Find location of a program
which zsh
```

Grep

Grep let you search for word in files and outputs the line it was found.

Listing 12: grep

```
grep boo /etc/passwd          # search boo in for /etc/passwd
grep -r "192.168.1.5" /etc/   # search recursivaly in /etc for 192.168.1.5
grep -w "boo" /path/to/file   # search for word "boo" only
```

grep is also often uses in pipes to search within the output of an other command

Listing 13: grep pipe

```
cat /proc/cpuinfo | grep -i 'Model' # display CPU Model name
ps -x | grep vnc
```

Note: Flags

- `-r` : search recursively in all files \
- `-n` : display line number \
- `-c` : count number of times found \
- `--color` : colors the word searched in the results

Links

Listing 14: links

```
ln target-filename symbolic-filename    # create hardlink
ln -s target-filename symbolic-filename # create softlink
```

Note: Hard Link vs Softlink

Symbolic links are different from hard links. Hard links may not normally point to directories, and they cannot link paths on different volumes or file systems. Hard links always

refer to an existing file.

Compression

Tar, bz2, gz

Listing 15: compress

```
tar cfv name.tar /path/to/folder      # Compression tar
tar xfv tarfile                       # Decompression tar

tar cfvz name.tar.gz /path/to/folder  # Compression tar.gz
tar xfvz tarfile                      # Decompression tar.gz

tar cfvj name.tar.bz2 /path/to/folder # Compression tar.bz2
tar xfvj tarfile                      # Decompression tar.bz2
```

Note: Flags

- `c` = Compression | `x` = eXtraction
- `f` = file/folder
- `v` = Verbose
- `j` = bz2 | `z` = gz
- `p` = Preserve (keep permissions)

RAR

Listing 16: compress rar

```
# compress and split in files of 700MB
rar a -m5 -v700m rarname folder_or_file_to_compress

# uncompress, if a split rar uncompress the first
rar e rarname
```

Note: Flags

- `m5` = highest compression `m0` = lowest compression
- `e` = extract in current folder
- `a` = append to rar
- `v<SIZE>m` = size of split files

In Outputs

Tail

Listing 17: tail

```
tail file folder                # give end of a file
./executable > output.txt       # redirect output to a file
./executable > output.txt 2< 1  # redirect output to a file output 2 & 1
                                # 2 = Error output
                                # 1 = Std output
```

Cat

Listing 18: cat

```
cat > file1.txt                         # To Create a new file
cat >> file1.txt                        # To Append data into the file
cat file1.txt                           # To display a file
cat file1.txt file2.txt                 # Concatenate several files and display
cat file1.txt file2.txt > file3.txt     # To cat several files and transfer output␣
↪to another file
```

PGP Pretty Good Privacy & GPG

see also *GnuPg*

Files

Listing 19: pgp files

```
/home/user/.ssh
pgp                             # private key
pgp.pub                         # public key
gpg_fingerprint.txt             # Infos for the gpg fingerprint
```

Create PGP files

Listing 20: create key's

```
ssh-keygen -t dsa -f filename   # Create private and public key
gpg --gen-key                   # Create gpg fingerprint
```

GPG Privacy

Listing 21: gpg

```
gpg --gen-key                # Create a key
gpg --export -a "User Name" > public.key# Export a public key
gpg --export-secret-key -a "User Name" > private.key# Export private key
gpg --import public.key         # Import public key
gpg --allow-secret-key-import --import private.key# Import private key
gpg --delete-key "User Name"        # Delete public key
gpg --delete-secret-key "User Name" # Delete private key
gpg --list-keys              # List key in public key ring
gpg --list-secret-keys          # Lsit key in private key ring
gpg --fingerprint > fingerprint     # Short list of numbers to verify public key
gpg -e -u "Sender User Name" -r "Receiver User Name" somefile # Encrypt data
gpg -d mydata.tar.gpg           # Decrypt data
```

SSH

Listing 22: ssh_config

```
# Edit config file
sudo vim /etc/ssh/sshd_config

# start, stop, restart SSH
sudo /etc/init.d/ssh start
sudo /etc/init.d/ssh stop
sudo /etc/init.d/ssh restart
```

Listing 23: ssh

```
ssh-agent bash          # start new bash agent
ssh-add privatekey      # key you want to use for that session
             # without a given key he search for ~/.ssh/id_rsa
```

Connect to another station by ssh by default a password is needed or if configured no password but with rsh keys

Listing 24: ssh connection

```
ssh -p <portnumber> -l <username> server.address.com

# or
ssh -p <portnumber> user@server.address.com

# or with port forward and no commandline
ssh -N -T -L <port>:localhost:<port> <user>@<hostname>
-N = No Output
-T = No Terminal access
-L = Port Forwarding
```

(continues on next page)

(continued from previous page)

```
# or with port forward and commandline
ssh -L <port>:localhost:<port> <user>@<hostname>
```

How to set up ssh with rsa keys

Listing 25: ssh keys

```
# Generating RSA Key pair
ssh-keygen -t rsa

# Copy key
ssh-copy-id -i ~/.ssh/id_rsa.pub "user@remote.machine.com -p <portnumber>"
# OR
scp id_rsa.pub user@host:~/.ssh/machine.pub

# Append key to file authorized_keys
cat ~/.ssh/*.pub | ssh admin@server.machine.com -p <portnumber> 'umask 077; cat >>.
↪ssh/authorized_keys'
```

SCP

Transferring file through SSH For these command you have to use either the PW or already bash started

Listing 26: scp

```
# Synchronising directories
rsync -avr -P --rsh='ssh -p YYYY' /localpath/to/dir user@host:/remotepath/to/dir

# Host -> Remote
scp sourceFile user@host:/directory/targetFile
scp -R sourceFolder user@host:/directory/targetFolder
scp -P port sourceFile user@host:/directory/targetFolder

# Remote -> Host
scp user@host:/directory/sourceFile targetFile
scp -R user@host:/directory/sourceFolder targetFolder
scp -P port user@host:/directory/sourceFolder targetFolder
```

1.1.2 Crontab

Crontab is used to regularly execute some task e.g. shell scripts

Listing 27: crontab

```
crontab -l    # List crontab's for current user
crontab -r    # Del  crontab's for current user
crontab -e    # Edit crontab's for current user
```

Crontab consist of 2 columns per entry

Column	Description
1	Minutes (0-59)
2	Hours (0-23)
3	Day of the month (1-31)
4	Month of the Year (1-12)
5	Day of the Week (0-6, 0 is Sunday)
6	Absolute path to script

Crontab entry examples

Listing 28: crontab entries

```
0 * * * * /home/user/backupServerA.sh  # At Noon each day
0 0 * * * /home/user/backupServerB.sh  # At Midnight each day
0 1 * * * /home/user/backupServerC.sh  # At 1 o'clock each day
0 * * * 1 /home/user/backupServerD.sh  # At Noon each Monday
@reboot /home/user/Documents/x11vnc_start.bash
@reboot nohup airsonos &
@weekly /home/user/script.bash > /home/user/scriptoutput.log
```

1.1.3 General Shell Commands

- *Change permissions on type*
- *SSH relia*

Change permissions on type

```
sudo find /var/www -type d -print0 | sudo xargs -0 chmod 0755
sudo find /var/www -type f -print0 | sudo xargs -0 chmod 0644
```

SSH relia

```
ssh -p 2222 -L 5900:localhost:5900 -L 19999:localhost:19999 zas@relia.zapto.org
```

1.1.4 Network

Interfaces

Listing 29: interface

```
ifconfig

ifup <if_name>
ifup eth0

ifdown <if_name>
ifdown eth0

ifquery -l
```

NMAP

Find open ports on a ip subnet range

Listing 30: nmap

```
# Finding ssh server in ip range 192.168.0-192.168.0.255
nmap -p 22 --open 192.168.1.0/24
```

ARP-SCAN

Finding a machine on your local subnet using DHCP.

Listing 31: arp-scan

```
# Finding ssh server in ip range 192.168.0-192.168.0.255
sudo apr-scan --interface=eth0 --localnet | grep aa:bb:cc:dd:ee:ff
```

1.1.5 RSync

Synchronizing directories local and remotely

Listing 32: rsync

```
# Synchronising directories to distant computer
rsync -avzP --rsh='ssh -p YYYY' /localpath/to/dir user@host:/remotepath/to/dir

# Synchronizing local left to right
rsync -avP --delete --stats /source/folder/* /destination/folder

# Synchronizing and exclude a folder or file
rsync -avP --delete --stats --exclude 'excl_folder' --exclude 'folder/excl_file.txt
↪' /source/folder/* /destination/folder
rsync -avP --delete --stats --exclude-from '/home/user/exclude.txt' /source/folder/
↪* /destination/folder
```

Listing 33: exclude.txt

```
sources
public_html/database.*
downloads/test/*
```

Note: Flags

```
-a = archive (recursion & preserve rights, time, owner, group)\\
-v = verbose \\
-z = compress during transfer \\
-P = display progress \\
--delete = delete files on destination folder \\
--stats = display some statisics at the end \\
--exclude <PATTERN> = exclude file or folder \\
--exclude-from <file> = exclude list defined in file
--dry-run = show what would have been done
```

1.2 Scripts

1.2.1 Config Files

- */etc/profile*
  - *Add Program to"PATH"*
- *~.bashrc .zshrc*
  - *Create a cmd alias*
- *Add custom functions*
- *etc/fstab*

- My linux configfiles

/etc/profile

/etc/profile contains Linux system wide environment and startup programs. It is used by all users with bash, zsh, sh shell. Usually used to set PATH variable, user limits, and other settings for user. It only runs for login shell. If you wanted to make large changes or application specific changes use /etc/profile.d directory.

My profile

Add Program to“PATH“

```
export PATH=$PATH:/opt/sublime_text
```

~.bashrc .zshrc

Execute commands at start of a shell instance for a given users only

.bashrc .zshrc

Create a cmd alias

Listing 34: alias

```
# Common home locations
alias home='cd ~'
alias root='cd /'
alias dtop='cd ~/Desktop'
alias dwld='cd ~/Downloads'
alias docs='cd ~/Documents'
alias www='cd /var/www/html'
# Common data directories
# Common commands
alias o=open
alias ..='cd ..'
alias ...='cd ..; cd ..'
alias ....='cd ..; cd ..; cd ..'
# Common command shortcuts
alias cls=clear
alias ll='ls -la'
alias owner-wwwdata='sudo chown -R www-data:www-data ./'
alias permission-file='sudo find . -type f -exec chmod 644 {} \;'
alias permission-folder='sudo find . -type d -exec chmod 755 {} \;'
# commands
alias backup='~/Documents/backup.bash'
```

Add custom functions

Listing 35: function

```
# Draw Mandelbrot Factral
function mandelbrot_zsh {
  local lines columns colour a b p q i pnew
  ((columns=COLUMNS-1, lines=LINES-1, colour=0))
  for ((b=-1.5; b<=1.5; b+=3.0/lines)) do
    for ((a=-2.0; a<=1; a+=3.0/columns)) do
      for ((p=0.0, q=0.0, i=0; p*p+q*q < 4 && i < 32; i++)) do
        ((pnew=p*p-q*q+a, q=2*p*q+b, p=pnew))
      done
      ((colour=(i/4)%8))
      echo -n "\\e[4${colour}m "
    done
    echo
  done
}
```

etc/fstab

There's a file called /etc/fstab in your Linux system. Learn what its contents mean and how it's used in conjunction with the mount command. When you learn to understand the fstab file, you'll be able to edit its contents yourself, too.

My fstab

1. column - Device
   - `UUID=...`
   - `/dev/hda2`
2. column - Default mount point
   - `/`
   - `mnt/data`
   - `media/disk`
3. column - Filesystem type
   - `ext2`
   - `ext4`
   - `ntfs`
   - `vfat`
   - `auto`
4. column - Mount options
   - `auto` and `noauto` - mounted automatically at bootup
   - `user` and `nouser` - allows normal user to mount the device
   - `exec` and `noexec` - lets execute binaries from that partition
   - `ro` and `rw` - **R** ead- **O** nly and **R** ead- **W** rite
   - `sync` and `async` - data can be writte synchron or asynchron
   - `default` - means `rw,suid,dev,exec,auto,nouser,async`
5. column - Dump options

- In most cases 0

6. column - fck options

- In most cases 0

Listing 36: fstab

```
UUID=3d3920bb-91c7-4632-8fd0-1d87b110a496 /                 ext4    errors=remount-ro 0       1
/swapfile                                 none              swap    sw                ␣ 0       0

# internal WD 1TB Harddisk on /dev/sda1
#UUID=377d6d5c-3d62-4155-b7f1-3f07fe09a0c2 /mnt/data2        ext4    defaults          ␣ 0       0


# external Lacie Rugged 2TB Harddisk on /dev/sda1
UUID=0c6f2eed-3ec0-493e-9ab8-e954a9e3a25d /media/zas_backup ext3    nofail,nobootwait 0       0

# external WD Passport 1TB Harddisk on /dev/sde1
UUID=20F605D47F5FE7AC                     /media/zas_media  ntfs    nofail,nobootwait 0       0
```

1.2.2 Scripts

- *Shell Bang*
- *Script End*
- *Variables*
- *Command line arguments*
- *Functions*
- *Console prints*
- *User Inputs*
- *Check and create folder*
- *Lockfile*
- *Samples*

Here you can dowload some example files for Linux. It can be neither Scripts or Links or config files

A lot of scripts and configurations can be found in my config repo:

- Shell Scripts

Shell Bang

At the beginning of a file there need to be a line to indentify the program or the file. `#!<path of the program executable>`

```
#!/bin/sh
```

```
#!/bin/bash
```

```
#!/usr/bin/env python
```

Script End

```
exit 0
```

Variables

Listing 37: variables

```
# Var
SEPARATOR='------------------------------------------------------------------------
↪--------'
INDENT='  '

# Array
MOUNT_POINTS=( 'elem1' 'elem2')

# Use Env var
Linux_user="$USER"
```

Command line arguments

Listing 38: cli arguments

```
usage='Usage: script.bash [-v] [-h]'
usage="$usage\n\t[-n input_n] [-u input_u]"

while getopts "n:u:vh" options; do
  case $options in
    n ) var_n=$OPTARG;;
    u ) var_u=$OPTARG;;
    v ) verbose=1;;
    h ) echo -e $usage
          exit 1;;
    * ) echo -e $usage
          exit 1;;
  esac
done

if [ -n "$verbose" ] ; then
  echo "Verbose"
fi
```

Functions

Listing 39: functions

```
# Define function
function test () {
  local arg1=$1 ; local arg2=$2

  $result = $arg1 + $arg2

  return 1
}

# Usage function
test 1 2
```

Console prints

Display message welcome on screen

Listing 40: echo

```
# Console print
echo 'Welcome'

# Write message File deleted to /tmp/log.txt
echo 'File has been deleted' > /tmp/log.txt

# Append message File deleted /tmp/log.txt
echo 'File has been deleted' >> /tmp/log.txt

# Append message and command output on screen, print variable
echo "Today's date is $(date)"
```

User Inputs

Listing 41: user inputs 1

```
echo -n "Please enter: "

stty -echo
read user_text
stty echo

echo ""         # force a carriage return to be output
```

Listing 42: user inputs 1

```
read -n1 -r -p "Press space to continue..." key
if [ "$key" = '' ]; then
    # Space pressed, do something
    # echo [$key] is empty when SPACE is pressed # uncomment to trace
else
    # Anything else pressed, do whatever else.
    # echo [$key] not empty
fi
```

Check and create folder

Listing 43: check and create folder

```
if [ ! -d "/folder/location" ]; then
  sudo mkdir /folder/location
fi
```

Lockfile

Lockfiles you can wait until another process is finished.

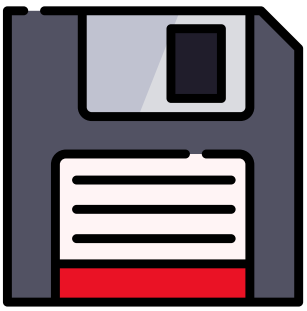
Listing 44: check and create folder

```
# Define path and lockfile
lockDir="/path/to/lock_files"
lockFilePath="$lockDir/lockfile.lock"
# Loop until file no longer exist
while [ -e "$lockFilePath" ]
do
    exit
done

# Create new lockfile
touch $lockFilePath

TO SOMETHING THE LOCK IS YOURS

# Remove lockfile
rm -f $lockFilePath
```

Samples

```
DIR="$( cd "$( dirname "$0" )" && pwd )" # get dir of executed file
```

1.3 Tools



1.3.1 dd and ddfldd

Use for creating and copying iso files from and to a medium.

The `dd` command doesn't has a output during copy but `dcfldd` does. It gives an output all X blocks written. This means in the commands below you can also just replace `dd` with `dcfldd`.

Install

Listing 45: dcfldd install

```
sudo apt-get install dcfldd
```

On Linux

Listing 46: dd usage

```
# Create usb stick or sdcard => image
fdisk -l                                    # get disk info
unmount /dev/sdX                            # unmount disk
dd if=/dev/sdX of=/location/for/image.iso bs=1M conv=noerror,sync # copy usb stick␣
↪to image

# Copy image => usb stick or sdcard
fdisk -l                                    # get disk info
unmount /dev/sdX                            # unmount disk
dd if=/location/of/image.iso of=/dev/sdX bs=1M conv=noerror,sync # copy image to␣
↪usb stick
```

dd has no output normally, if you want to watch the status of the copy then open a new Terminal and try one of the following commands

Listing 47: watch dd

```
sudo kill -USR1 $(pgrep '^dd$')                             # dd will display␣
↪status once
sudo watch -n <interval in sec> kill -USR1 $(pgrep '^dd$') # dd will display␣
↪status continously
```

On MacOs

Listing 48: dd usage

```
# Create usb stick or sdcard => image
diskutil list                                # get disk info
diskutil unmountDisk /dev/diskX              # unmount disk
dd if=/dev/diskX of=/location/for/image.iso bs=1m # copy usb stick to image

# Copy image => usb stick or sdcard
diskutil list                               # get disk info
diskutil unmountDisk /dev/diskX             # unmount disk
dd if=/location/of/image.iso of=/dev/diskX bs=1m conv=noerror,sync # copy image to␣
↪usb stick
```

1.3.2 Let's Encrypt

- *Version*
- *Renew Certificates*

Version

```
certbot --version
```

Renew Certificates

```
# Stop Webserver Service
sudo service apache2 stop

# Update Certificates
sudo certbot renew
sudo certbot renew --dry-run

# Restart Webserver Service
sudo service apache2 start
```

1.3.3 Systemd

- *List services*
- *Status service*
- *Start Stop Service*
- *Add Service*
- *Add in vim file*
- *Start Service manually*
- *Start Service on boot*

List services

```
systemctl --type=service
```

Status service

```
systemctl status firewalld.service
```

Start Stop Service

```
systemctl stop firewalld.service
systemctl status firewalld.service
```

Add Service

```
cd /etc/systemd/system
sudo vim jupyterlab.service
```

Add in vim file

```
[Unit]
Description = Jupyterlab service
After = network.target
StartLimitIntervalSec=0

[Service]
Type=simple
User=zas
ExecStart=/home/zas/Documents/jupyterlab_start.bash

[Install]
WantedBy = multi-user.target
```

Start Service manually

```
systemctl start jupyterlab
```

Start Service on boot

```
systemctl enable jupyterlab
```

1.4 Ubuntu

1.4.1 Installation and Config

- *Installation*
  - *Default Tools*
  - *ZSH*

Installation

This installation is based on Ubuntu 18.4 LTS and ROS Melodic Morenia.

Default Tools

```
sudo apt-get install git curl vim openssh-server krename rar unrar kget diffutils␣
↪kate x11vnc
echo "Configure Firewall and Port for ssh"
sudo ufw allow ssh
sudo ufw enable
sudo ufw status
sudo service ssh restart
```

ZSH

```
sudo apt-get install zsh
sudo chsh -s /bin/zsh $USER
```

Oh My ZSH

```
cd ~/Downloads
sh -c "$(curl -fsSL https://raw.github.com/robbyrussell/oh-my-zsh/master/tools/
↪install.sh)"
```

SublimeText 3

```
wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add -
sudo apt-get install apt-transport-https
echo "deb https://download.sublimetext.com/ apt/stable/" | sudo tee /etc/apt/
↪sources.list.d/sublime-text.list
sudo apt-get update
sudo apt-get install sublime-text
```

SublimeMerge

```
wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add -
sudo apt-get install apt-transport-https
echo "deb https://download.sublimetext.com/ apt/stable/" | sudo tee /etc/apt/
↪sources.list.d/sublime-text.list
sudo apt-get update
sudo apt-get install sublime-merge
```

Krusader

```
sudo apt-get install krusader
```

Yakuake

```
sudo apt-get install yakuake
```

FSearch

```
sudo add-apt-repository ppa:christian-boxdoerfer/fsearch-daily
sudo apt update
sudo apt-get install fsearch-trunk
```

Anaconda

```
cd ~/Downloads
wget https://repo.anaconda.com/archive/Anaconda3-2019.10-Linux-x86_64.sh
bash Anaconda3-2019.10-Linux-x86_64.sh
```

QT-Creator

```
cd ~/Downloads
wget http://download.qt.io/official_releases/qt/5.13/5.13.1/qt-opensource-linux-
↪x64-5.13.1.run
chmod +x qt-opensource-linux-x64-5.13.1.run
./qt-opensource-linux-x64-5.13.1.run
sudo apt-get install build-essential
sudo apt-get install libfontconfig1
sudo apt-get install mesa-common-dev
sudo apt-get install libglu1-mesa-dev -y
```

Visual Studio Code

```
curl https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > packages.
↪microsoft.gpg
sudo install -o root -g root -m 644 packages.microsoft.gpg /usr/share/keyrings/
sudo sh -c 'echo "deb [arch=amd64 signed-by=/usr/share/keyrings/packages.microsoft.
↪gpg] https://packages.microsoft.com/repos/vscode stable main" > /etc/apt/sources.
↪list.d/vscode.list'

sudo apt-get install apt-transport-https
sudo apt-get update
sudo apt-get install code # or code-insiders
```

Configuration

Oh My ZSH Config

Listing 49: ~/.zshrc additions

```
echo"#-----------------------------------------------------------------------------
↪-" >> ~/.zshrc
echo "# Program in Path" >> ~/.zshrc
echo "#" >> ~/.zshrc
echo "#-----------------------------------------------------------------------------
↪--" >> ~/.zshrc
echo "# Special zsh config" >> ~/.zshrc
echo "# Show hidden files and folders" >> ~/.zshrc
echo "setopt globdots" >> ~/.zshrc
echo "#-----------------------------------------------------------------------------
↪--" >> ~/.zshrc
echo "# Goto Alias" >> ~/.zshrc
echo "# Common home locations" >> ~/.zshrc
echo "alias home='cd ~'" >> ~/.zshrc
echo "alias root='cd /'" >> ~/.zshrc
echo "alias dtop='cd ~/Desktop'" >> ~/.zshrc
echo "alias dwld='cd ~/Downloads'" >> ~/.zshrc
echo "alias docs='cd ~/Documents'" >> ~/.zshrc
echo "alias www='cd /var/www/html'" >> ~/.zshrc
echo "alias workspace='cd ~/Workspace'" >> ~/.zshrc
echo "alias aptlock-rm='sudo rm /var/lib/dpkg/lock && sudo rm /var/lib/dpkg/lock-
↪frontend'" >> ~/.zshrc
echo "# Common commands" >> ~/.zshrc
echo "alias o=open" >> ~/.zshrc
echo "alias ..='cd ..'" >> ~/.zshrc
echo "alias ...='cd ..; cd ..'" >> ~/.zshrc
echo "alias ....='cd ..; cd ..; cd ..'" >> ~/.zshrc
echo "# Common command shortcuts" >> ~/.zshrc
echo "alias cls=clear" >> ~/.zshrc
echo "alias ll='ls -la'" >> ~/.zshrc
```

SublimeText 3 Config

Listing 50: ~/.zshrc additions

```
echo "# Sublime Text" >> ~/.zshrc
echo "export PATH=$PATH:/opt/sublime_text" >> ~/.zshrc

echo "# Sublime Text" >> ~/.bashrc
echo "export PATH=$PATH:/opt/sublime_text" >> ~/.bashrc

cp ./../config/sublimetext/Package Control.sublime-settings ~/.config/sublime-text-
↪3/Packages/User/
```

SublimeMerge Config

Listing 51: ~/.zshrc additions

```
echo "#Sublime Merge" >> ~/.zshrc
echo "export PATH=$PATH:/opt/sublime_merge" >> ~/.zshrc

echo "#Sublime Merge" >> ~/.bashrc
echo "export PATH=$PATH:/opt/sublime_merge" >> ~/.bashrc
```

How To Use Ubuntu Tools

SSH

SSH connection without password

```
# On your local machine generate a RSA Key pair
ssh-keygen -t rsa

# Copy your local public key to the remote machine safely
ssh-copy-id -i ~/.ssh/id_rsa.pub "<user>@<remoteip> -p <portnumber>"
# OR
scp id_rsa.pub <user>@<remoteip>:~/.ssh/machine.pub

# Append key to file authorized_keys
cat ~/.ssh/*.pub | ssh <user>@<remoteip> -p <portnumber> 'umask 077; cat >>.ssh/
↪authorized_keys'
```

Open SSH Connection

```
# Just ssh
ssh <user>@<remoteip>

# ssh with portforwarding
ssh -L <local-port>:localhost:<remote-port> <user>@<remoteip>
# ssh with vnc port forwarding
ssh -L 5900:localhost:5900 spl@<remoteip>
```

VNC

On remote PC x11vnc needs to be installed and launched. Prefereable add to startup commands

Create password

Only needed if not only localhost used.

```
x11vnc -storepasswd
```

Launch x11vnc

```
# Command with all options
x11vnc -usepw -forever -display :0 -safer -bg -o ~/Documents/log/vnc/x11vnc.log -
↪localhost

# Minimal command but still restricted to localhost
x11vnc -forever -display :0 -safer -bg -localhost
```

1.4.2 Introduction

- *Additional Informations*

Additional Informations

- https://ubuntu.com/ - Ubuntu Webpage
  - https://ubuntu.com/#download - Ubuntu Download
- https://www.osboxes.org/ubuntu/ - Virtual Box images
- Additional Tools
  - ZSH
    - Oh My ZSH
  - Sublime Text
  - Sublime Merge
  - Krusader
  - Yakuake
  - FSearch
  - Anaconda
  - QT Creator
  - Visual Studio Code
  - Hitachi SDK
    - Hitachi LiDaR SDK

    * Hitachi LiDaR ROS Driver
  - ROS Installation

1.5 Filesystem

1.5.1 Important Files

```
/boot/grub             # Grub files
/etc/fstab             # Drive mounts
/etc/profile           # EnvVar for all users
/etc/group             # All groups available
/home/username/.bashrc # User-script @startup
/etc/init.d/           # Global start-scripts
/etc/ssh/sshd.config   # SSH config file
~.bashrc               # bash shell init scripts
~.zshrc                # zsh shell init scripts
```

1.5.2 Folders

Foreword

Like UNIX, Linux chooses to have a single hierarchical directory structure.

Everything starts from the root directory, represented by `/`, and then expands into sub-directories instead of having so-called 'drives'. In the Windows environment, one may put one's files almost anywhere: on C drive, D drive, E drive etc. Such a file system is called a hierarchical structure and is managed by the programs themselves (program directories), not by the operating system. On the other hand, Linux sorts directories descending from the root directory / according to their importance to the boot process.

Root directory /

As we all know Linux file system starts with `/`, the root directory. All other directories are 'children' of this directory. The partition which the root file system resides on is mounted first during boot and the system will not boot if it doesn't find it. On our reference system, the root directory contains the following sub-directories:

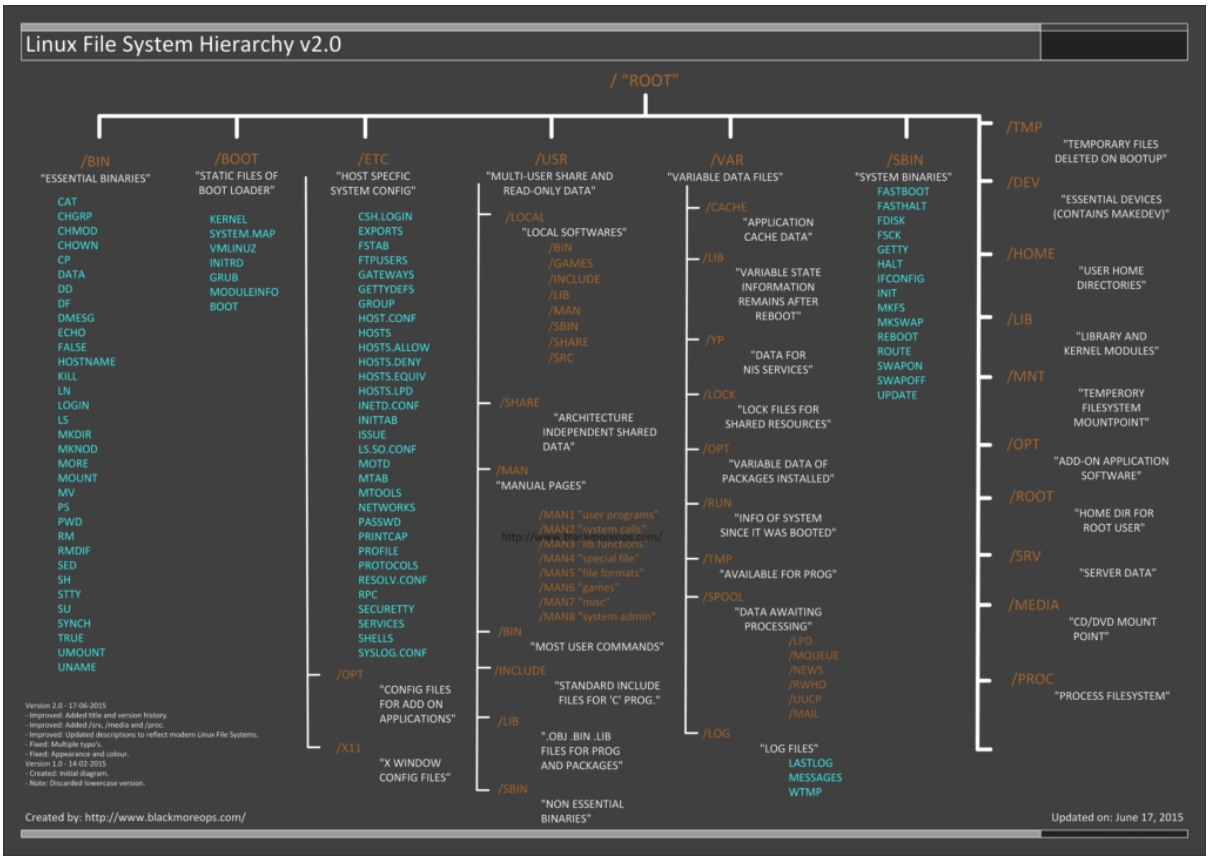


Figure1: https://www.blackmoreops.com/

Folder	Description
/bin	Essential command binaries
/boot	Static files of the boot loader
/dev	Device files
/etc	Host-specific system configuration
/root	Home directory for the root user (optional)
/home	User home directories (optional)
/lib	Essential shared libraries and kernel modules
/lib<qual>	Alternate format essential shared libraries (optional)
/media	Mount point for removeable media
/mnt	Mount point for mounting a filesystem temporarily
/opt	Add-on application software packages
/sbin	Essential system binaries
/srv	Data for services provided by this system
/tmp	Temporary files
/usr	Secondary hierarchy
/var	Variable data

/bin

The bin directory contains several useful commands that are of use to both the system administrator as well as non-privileged users. It usually contains the shells like bash, csh, etc. and commonly used commands like cp, mv, rm, cat, ls. For this reason and in contrast to `/usr/bin`, the binaries in this directory are considered to be essential. The reason for this is that it contains essential system programs that must be available even if only the partition containing `/` is mounted. This situation may arise should you need to repair other partitions but have no access to shared directories (ie. you are in single user mode and hence have no network access). It also contains programs which boot scripts may depend on.

/boot

This directory contains everything required for the boot process except for configuration files not needed at boot time (the most notable of those being those that belong to the GRUB boot-loader) and the map installer. Thus, the `/boot` directory stores data that is used before the kernel begins executing user-mode programs. This may include redundant (back-up) master boot records, sector/system map files, the kernel and other important boot files and data that is not directly edited by hand. Programs necessary to arrange for the boot loader to be able to boot a file are placed in `/sbin`. Configuration files for boot loaders are placed in `/etc`. The system kernel is located in either `/` or `/boot` (or as under Debian in `/boot` but is actually a symbolically linked at `/`.

/boot/map

Contains the location of the kernel.

/boot/vmlinuz /boot/vmlinuz-kernel-version

Normally the kernel or symbolic link to the kernel.

/boot/grub

This subdirectory contains the GRUB configuration files including boot-up images and sounds. GRUB is the GNU GRand Unified Bootloader, a project which intends to solve all bootup problems once and for all. One of the most interesting features, is that you don't have to install a new partition or kernel, you can change all parameters at boot time via the GRUB Console, since it knows about the filesystems.

/boot/grub/grub.conf /boot/grub/menu.lst

Grub configuration file.

/dev

/dev is the location of special or device files. It is a very interesting directory that highlights one important aspect of the Linux filesystem - everything is a file or a directory. Look through this directory and you should hopefully see hda1, hda2 etc.... which represent the various partitions on the first master drive of the system. /dev/cdrom and /dev/fd0 represent your CD-ROM drive and your floppy drive. This may seem strange but it will make sense if you compare the characteristics of files to that of your hardware. Both can be read from and written to. Take /dev/dsp, for instance. This file represents your speaker device. Any data written to this file will be re-directed to your speaker. If you try cat /boot/vmlinuz > /dev/dsp (on a properly configured system) you should hear some sound on the speaker. That's the sound of your kernel! A file sent to /dev/lp0 gets printed. Sending data to and reading from /dev/ttyS0 will allow you to communicate with a device attached there - for instance, your modem.

3 Informations are relevant:

- Kind of the Access
  - bloc oriented (b) - buffered access e.g. Harddisks
  - characteroriented (c) - non buffered access e.g. Screen, Printer
- Major device number
  - Specify the driver to be used.
- Minor device number
  - To describe the actual instance of a device. In case multiple devices of the same driver are used

Some common device files as well as their equivalent counterparts under Windows that you may wish to remember are:

Listing 52: deviceslist.txt

```
/dev/ttyS0 (First communications port, COM1)
    First serial port (mice, modems).

/dev/psaux (PS/2)
    PS/2 mouse connection (mice, keyboards).

/dev/lp0 (First printer port, LPT1)
    First parallel port (printers, scanners, etc).

/dev/dsp (First audio device)
    The name DSP comes from the term digital signal processor, a specialized␣
↪processor chip optimized for digital signal analysis. Sound cards may use a␣
↪dedicated DSP chip, or may implement the functions with a number of discrete␣
↪devices. Other terms that may be used for this device are digitized voice and␣
↪PCM.

/dev/usb (USB Devices)
    This subdirectory contains most of the USB device nodes. Device name␣
↪allocations are fairly simplistic so no elaboration is be necessary.

/dev/sda (C:\, SCSI device)
    First SCSI device (HDD, Memory Sticks, external mass storage devices such as␣
↪CD-ROM drives on laptops, etc).

/dev/scd (D:\, SCSI CD-ROM device)
    First SCSI CD-ROM device.
```

(continues on next page)

(continued from previous page)

```
/dev/js0 (Standard gameport joystick)
    First joystick device.

/dev/hd*
    Mounted Harddisk Partitions
```

/etc

Contains all local configurationfiles.

/root

Home folder for the root user. In most systems it was eliminated

/home

Home folder for all system users. Each user has a own RWX folder with his name inside /home/

/lib

Needed systemlibraries and kernelmodules

/lib<qual>

Needed systemlibraries and kernelmodules

/media

Debian automatic mountpoint for Plug&Play programms

/mnt

Temporary mountpoint

/opt

Place for optional Software installed be the user.

/sbin

Important systemprogramms. They are used for the startup of the system. Only executable by root user.

/tmp

All users can use this foder to store temporary files

/usr

User Data

/var

Variable data

Chapter 2

Mac

2.1 Geektool

Geektool is a wonderful program to display information on your desktop. Here you can see my “Control Center”.

2.1.1 Date & Time

All command geeklets

```
date +%d                # Day number
date +%B                # Month
date +%A                # Weekday
```

```
date "+%H"              # Hour
date "+%M"              # Minutes
```

Figure1: Geektool Example

2.1.2 Location & Weather

Actual Weather

All command geeklets Get the weather of a certain location from Yahoo weather.

```
WEATHER=`curl --silent "http://weather.yahooapis.com/forecastrss?p=SZXX0035&u=c" |
↪grep -E '(Current Conditions:|C<BR)' | tail -n1 | sed -e 's/<BR \/>//' -e 's/ C$/
↪ °C/'`
echo "Sion " $WEATHER
```

Moonphase

Get the image for the actual Moonphase from lexiyoga. Directly the image will be fetch, therefore a image geeklet is needed.

```
http://www.lexiyoga.com/images/moon/moon16.png
```

2.1.3 System information

Uptime

```
uptime | awk '{sub(/[0-9]|user\,|users\,|load/, "", $6); sub(/mins,|min,/, "min",
↪$6); sub(/user\,|users\,/, "", $5); sub(",", "min", $5); sub(":", "h ", $5);␣
↪sub(/[0-9]/, "", $4); sub(/day,/, " day ", $4); sub(/days,/, " days ", $4); sub(/
↪mins,|min,/, "min", $4); sub("hrs,", "h", $4); sub(":", "h ", $3); sub(",", "min
↪", $3); print "Uptime: " $3$4$5$6}'
```

Networking

```
# Internal Wireless IP
myen0=`ifconfig en0 | grep "inet " | grep -v 127.0.0.1 | awk '{print $2}'`
if [ "$myen0" != "" ]
then
echo "Wireless: $myen0"
else
echo "Wireless INACTIVE"
fi

# Internal Ethernet IP
myen1=`ifconfig en1 | grep "inet " | grep -v 127.0.0.1 | awk '{print $2}'`
if [ "$myen1" != "" ]
then
echo "Ethernet: $myen1"
else
echo "Ethernet INACTIVE"
fi

# External IP
wip=`curl --silent http://checkip.dyndns.org | awk '{print $6}' | cut -f 1 -d "<"`
echo "External IP: $wip"
```

Battery status

```
BATTERY=`ioreg -l | awk '$3~/Capacity/{c[$3]=$5}END{OFMT="%.f %%";max=c["\
↪"MaxCapacity\""];print(max>0?100*c["\"CurrentCapacity\""]/max:"?")}'`
echo $BATTERY '\n\n\n'
```

HDD usage

```
DISK=`df -hl | grep 'disk0s2' | awk '{print $5 "  ("$4"/"$2")"}'`
echo $DISK '\n\n\n'
```

Ram usage

```
MEM=`top -l 1 | grep PhysMem: | awk '{print int($2/($2+$6)*100)}'`
echo $MEM'%\n\n\n'
```

CPU load

```
SYSTEM=`top -l3 | grep "CPU usage" | tail -1 | awk '{print int(($3+$5))}' | sed 's/
↪\%//'`
echo $SYSTEM '%\n\n\n'
```

Process manager

```
ps -amcwwwxo "command %mem %cpu" | grep -v grep | head -13
```

2.2 Keyboard Layout

All Mac have a keyboard were many symbols are missing. In programming this is very unpractical, so either you know on the tip of your head where which symbol is; or you look at the layout below ;-).

The layout is accordingly to the Mac Retina Book Swiss keyboard.

Figure2: Mac keyboard swiss layout

2.3 Macport

https://www.macports.org

```
sudo port install <programname>
sudo port uninstall <programname>

port list

port info <programname>
```

2.3.1 Useful Macports

- Krusader
- kdiff3
- md5deep
- unrar
- pzip
- putty
- tightvnc
- …

Chapter 3

Windows

3.1 Firewall

- *SSH Over FTP Port*

3.1.1 SSH Over FTP Port

Ff FTP Port is used for SSH connections disable `Statefulftp` in the Windows firewall

```
netsh advfirewall set global statefulftp disable
```

3.2 Group Policies

- *Modify Policies*
- *See all modified Group Policies*

3.2.1 Modify Policies

Search for `Edit group policy`

3.2.2 See all modified Group Policies

Search for `rsop.msc`

3.3 Registry

- *Login*
- *DateTime*
- *Shell Overlay Icons*
- *Context Menu*
- *New Context Menu*
- *SAP Shortcut Password*
- *PowerPoint Options*

3.3.1 Login

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
```

3.3.2 DateTime

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\DateTime\Servers
```

3.3.3 Shell Overlay Icons

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\
↪ShellIconOverlayIdentifiers
```

3.3.4 Context Menu

```
Computer\HKEY_CLASSES_ROOT\*\shellex\ContextMenuHandlers
```

3.3.5 New Context Menu

```
Computer\HKEY_CLASSES_ROOT\
```

3.3.6 SAP Shortcut Password

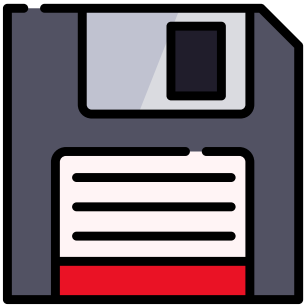
```
Computer\HKEY_CURRENT_USER\Software\SAP\SAPShortcut\Security
```

3.3.7 PowerPoint Options

```
Computer\HKEY_CURRENT_USER\Software\Microsoft\Office\16.0\PowerPoint\Options
  * ExportBitmapResolution => DWORD 32bit => 300 (ppi)
  * AutomaticPicturesCompressionDefault => DWORD => 0
```

Chapter 4

Useful Tools



Quick unfinished List of useful tools

4.1 Multimedia

- VLC - Video Player
- Gimp - Image Editor
- Inkscape - Vector Graphic Editor
- pdftk
- PlantUML
- Graphviz
- Wavedrom
- *Latex*

4.2 Internet

- Madsonic
- X11VNC

4.3 Commandline

- Yakuake
- Total Terminal
- Cmder

4.4 Managment

- Total Commander
- Krusader
- DCommander
- Hyperdock
- iStat Menu
- Virtualbox
- Keepass

4.5 Programming

- Sublime Text
- Sublime Merge
- git
- Mentor HDL Designer
- Mentor Modelsim
- Xilinx ISE
- Intel Quartus
- Intellij IDEA

- Intellij PyCharm
- Jupyterlab
- SpinalHDL

Chapter 5

Git

5.1 Git Commands

- *Start a working area*
- *Work on the current change*
- *Examine the history and state*
- *Grow, mark and tweak your common history*
- *Collaborate*

5.1.1 Start a working area

Command	Description
`clone`	Clone a repository into a new directory
`init`	Create an empty Git repository or reinitialize an existing one

5.1.2 Work on the current change

Command	Description
`add`	Add file contents to the index
`mv`	Move or rename a file, a directory, or a symlink
`reset`	Reset current HEAD to the specified state
`rm`	Remove files from the working tree and from the index

5.1.3 Examine the history and state

Command	Description
`log`	Show commit logs
`show`	Show various types of objects
`status`	Show the working tree status

5.1.4 Grow, mark and tweak your common history

Command	Description
`branch`	List, create, or delete branches
`checkout`	Switch branches or restore working tree files
`commit`	Record changes to the repository
`diff`	Show changes between commits, commit and working tree, etc
`merge`	Join two or more development histories together
`rebase`	Reapply commits on top of another base tip
`tag`	Create, list, delete or verify a tag object signed with GPG

5.1.5 Collaborate

Command	Description
`fetch`	Download objects and refs from another repository
`pull`	Fetch from and integrate with another repository or a local branch
`push`	Update remote refs along with associated objects

5.2 Git Flow

- *Branches*

5.2.1 Branches

- `master` - protected branch - Production releases
- `develop` - protected branch - main developement merge of all `feature` branches
- `feature/*` - for each feature a separate feature branch is created fork from `develop`
- `release` - preparing `development` branch for release on `master` branch, mainly for bugfixes
- `hotfix` - quick and dirty hotfix directly into `develop` and `master` branch

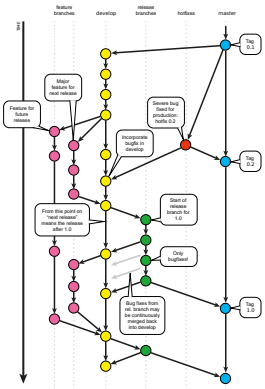


Figure1: Git Flow

5.3 Git General

- *Global setup*
  - *Check setup*
  - *Save Credentials*
  - *Not verify https Certificates*
- *Git Repo Creation / Cloning*
  - *Create new repository*
  - *Clone existing repository*
  - *Existing folder*
  - *Switch to new Remote*
  - *Get Remote Information*
  - *Change Push Remote URL*
- *Git Repo information*
- *Add Files*
- *Checkout*
- *Push*
- *Branch*
  - *Merge*

5.3.1 Global setup

```
git config --global user.name "Silvan Zahno"
git config --global user.email "silvan.zahno@hevs.ch"
```

Check setup

```
git config --list
```

Save Credentials

```
git config credential.helper store
```

Not verify https Certificates

```
git config --global http.sslVerify false
```

5.3.2 Git Repo Creation / Cloning

Create new repository

```
git init
```

Clone existing repository

```
git clone git@gitlab.hevs.ch:course/ElN/eln_labs.git
cd eln_labs
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Existing folder

```
cd existing_folder
git init
git remote add origin git@gitlab.hevs.ch:course/ElN/eln_labs.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Switch to new Remote

```
cd existing_repo
git remote rename origin old-origin
git remote add origin git@gitlab.hevs.ch:course/ElN/eln_labs.git
git push -u origin --all
git push -u origin --tags
```

Get Remote Information

```
git remote show origin
```

Change Push Remote URL

```
git remote set-url --push <new_repo_push_url>
```

5.3.3 Git Repo information

```
# Status about current files ion folder
git status

# Status about last commits
git log --oneline
```

5.3.4 Add Files

```
# Stage a File
git add README.md

# Commit file
git commit —m "Initial commit, add README file"
```

5.3.5 Checkout

```
# Checkout certain commit
git checkout e006db0 -b inspectingPrev

# Checkout given branch
git chekout master
```

5.3.6 Push

```
git push origin master
```

5.3.7 Branch

```
# Create new branch
git branch dev_branch_1

# List all existing branches
git branch

# Checkout certain branch
git branch dev_branch_1

# Delete certain branch
git branch -d dev_branch_1
```

Merge

```
# Checkout branch you want to merge into
git checkout master
# Merge the two branches
git merge dev_branch_1
```

5.4 Git Submodules

- *Clone Repo with submodules*
- *Pull changes*
  - *Pull all changes in the repo including changes in the submodules*
  - *Pull all changes for the submodules*
- *Add submodule and define the master branch as the one you want to track*
- *Move Submodule*

5.4.1 Clone Repo with submodules

```
git clone --recursive [URL to Git repo]
```

5.4.2 Pull changes

Pull all changes in the repo including changes in the submodules

```
git pull --recurse-submodules
```

Pull all changes for the submodules

```
git submodule update --remote
```

5.4.3 Add submodule and define the master branch as the one you want to track

```
git submodule add -b master [URL to Git repo]

git submodule init
```

5.4.4 Move Submodule

```
git mv a b
```

Chapter 6

Jupyter

6.1 Common Functions

- *Common Jupyterlab and Nodejs functions*
  - *install nvm*
  - *Install nodejs via conda*
  - *update npm*
  - *Rebuild Jupyterlab*
  - *Remove nodejs and npm*
- *Auto import of Libraries*
- *Check*

6.1.1 Common Jupyterlab and Nodejs functions

install nvm

```
https://github.com/creationix/nvm
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.34.0/install.sh | zsh
### Install latest nodejs
nvm install node # "node" is an alias for the latest version
```

Install nodejs via conda

```
conda install -c conda-forge nodejs
```

update npm

```
sudo npm install -g npm
```

Rebuild Jupyterlab

```
jupyter lab build
```

Remove nodejs and npm

```
conda remove nodejs npm
```

6.1.2 Auto import of Libraries

- Navigate to `~/.ipython/profile_default`
- Create a folder called `startup` if it's not already there
- Add a new Python file called `start.py`
- Put your favorite imports in this file
- Launch IPython or a Jupyter Notebook and your favorite libraries will be automatically loaded every time!

Example `start.py`

```
import pandas as pd
import numpy as np

# Pandas options
pd.options.display.max_columns = 30
pd.options.display.max_rows = 20

from IPython import get_ipython
ipython = get_ipython()

# If in ipython, load autoreload extension
if 'ipython' in globals():
    print('\nWelcome to IPython!')
    ipython.magic('load_ext autoreload')
    ipython.magic('autoreload 2')

# Display all cell outputs in notebook
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'

# Visualization
import plotly.plotly as py
import plotly.graph_objs as go
from plotly.offline import iplot, init_notebook_mode
```

(continues on next page)

(continued from previous page)

```
init_notebook_mode(connected=True)
import cufflinks as cf
cf.go_offline(connected=True)
cf.set_config_file(theme='pearl')

print('Your favorite libraries have been loaded.')
```

6.1.3 Check

Confirm that Libraries are loaded with

```
globals()
```

6.2 Extensions

- *Installed extensions*

6.2.1 Installed extensions

```
jupyter labextension list
```

6.3 General

- *Anaconda / Conda Update*
- *nbconvert*
  - *Convert to python for linting*
  - *Convert to html*
  - *Convert to pdf*

6.3.1 Anaconda / Conda Update

```
# Update all Conda packages
conda update --all

# Update Anaconda only
conda update conda
conda update anaconda
```

6.3.2 nbconvert

Converts jupyter notebook to other formats

```
jupyter nbconvert --to <format> notebook.ipynb
```

formats are:

- `--to html` - HTML
  - `--template full` (default)
  - `--template basic`
- `--to latex` - LaTeX
  - `--template article` (default)
  - `--template report`
  - `--template basic`
- `--to pdf` - PDF
  - `--template article` (default)
  - `--template report`
  - `--template basic`
- `--to sildes` - Reveal.js HTML slideshow
- `--to markdown` - Markdown
- `--to rst` - reStructuredText
- `--to script` - executable script (`.py`)
- `--to notebook` -

Convert to python for linting

```
jupyter nbconvert --to script test.ipynb
```

Convert to html

```
jupyter nbconvert --to html test.ipynb
```

Convert to pdf

needs Latex installed see: *LaTeX*

```
jupyter nbconvert --to latex test.ipynb
```

6.4 Plotly Dash

6.4.1 Install Plotly Dash

```
# The core dash backend
pip install dash==0.43.0
# DAQ components (newly open-sourced!)
pip install dash-daq==0.1.0
```

Checking Versions

```
import dash_core_components
print(dash_core_components.__version__)
```

Getting help

```
help(dcc.Dropdown)
```

Jupyter integration install

```
git clone https://github.com/plotly/jupyterlab-dash
cd jupyterlab-dash
npm install
npm run build
jupyter labextension link .
~/anaconda3/bin/./python -m pip install -e .
```

To rebuild the package and the JupyterLab app

```
npm run build
jupyter lab build
```

Additional Packages

```
pip install aiohttp
pip install django_plotly_dash
pip install jupyter_plotly_dash
```

Install Dash DAQ

```
pip install dash_daq
```

6.5 Installation

- *My Extension list*
  - *All in one install*
  - *Add install R to jupyter*
  - *Add install pandoc and inkscape to conda*
- *Install Python Additional Stuff*
  - *Graphviz*
  - *Install python Libraries*
- *Problems*
  - *Anaconda Navigator not starting*
- *Install Plotly and Plotly Express*
- *Better PDF Export*

see also jupyter config

6.5.1 My Extension list

```
jupyter labextension install @jupyter-widgets/jupyterlab-manager
jupyter labextension install @jupyterlab/statusbar-extension
jupyter labextension install @jupyterlab/geojson-extension
jupyter labextension install @jupyterlab/git
pip install -e git+https://github.com/jupyterlab/jupyterlab-git.git#egg=jupyterlab_
↪git
jupyter serverextension enable --py jupyterlab_git --sys-prefix
jupyter labextension install @jupyterlab/plotly-extension
jupyter labextension install @jupyterlab/toc
jupyter labextension install @deathbeds/jupyterlab_graphviz
```

(continues on next page)

(continued from previous page)

```
jupyter labextension install @ryantam626/jupyterlab_sublime
jupyter labextension install jupyter-matplotlib
jupyter labextension install jupyterlab_bokeh
jupyter labextension install @mflevine/jupyterlab_html
jupyter labextension install jupyterlab-drawio
jupyter labextension install jupyterlab-flake8
# jupyter labextension install jupyterlab_nbmetadata
jupyter labextension install jupyterlab_hidecode
jupyter labextension install @krassowski/jupyterlab_go_to_definition
jupyter labextension install @lckr/jupyterlab_variableinspector
```

All in one install

```
jupyter labextension install @lckr/jupyterlab_variableinspector @krassowski/
↪jupyterlab_go_to_definition @jupyter-widgets/jupyterlab-manager @jupyterlab/
↪statusbar-extension @jupyterlab/geojson-extension @jupyterlab/plotly-extension␣
↪@jupyterlab/toc @deathbeds/jupyterlab_graphviz jupyterlab_hidecode @ryantam626/
↪jupyterlab_sublime jupyter-matplotlib jupyterlab_bokeh @mflevine/jupyterlab_html␣
↪jupyterlab-drawio jupyterlab-flake8
pip install -e git+https://github.com/jupyterlab/jupyterlab-git.git#egg=jupyterlab_
↪git
jupyter serverextension enable --py jupyterlab_git --sys-prefix
```

Add install R to jupyter

```
conda install -c r r-essentials
```

Add install pandoc and inkscape to conda

```
conda install -c conda-forge pandoc
conda install -c conda-forge inkscape
```

6.5.2 Install Python Additional Stuff

Graphviz

Install Graphviz from https://graphviz.gitlab.io/download/ put Graphviz/bin in your PATH

```
pip install graphviz
```

Install python Libraries

```
pip install pixiedust
pip install SchemDraw
pip install nbwavedrom
pip install flake8
pip install pyflakes
pip install nbconvert
pip install watermark
```

oneline

```
pip install pixiedust SchemDraw nbwavedrom flake8 pyflakes nbconvert graphviz
```

6.5.3 Problems

Anaconda Navigator not starting

When starting anaconda-navigator produces the follwowing error.

```
$ anaconda-navigator.exe
Traceback (most recent call last):
  File "c:\Users\silvan.zahno\AppData\Local\Continuum\anaconda3\lib\site-packages\qtpy\__init__.py", line 202, in <module>
    from PySide import __version__ as PYSIDE_VERSION  # analysis:ignore
ModuleNotFoundError: No module named 'PySide'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "c:\Users\silvan.zahno\AppData\Local\Continuum\anaconda3\Scripts\anaconda-navigator-script.py", line 6, in <module>
    from anaconda_navigator.app.main import main
  File "c:\Users\silvan.zahno\AppData\Local\Continuum\anaconda3\lib\site-packages\anaconda_navigator\app\main.py", line 22, in <module>
    from anaconda_navigator.utils.conda import is_conda_available
  File "c:\Users\silvan.zahno\AppData\Local\Continuum\anaconda3\lib\site-packages\anaconda_navigator\utils\__init__.py", line 15, in <module>
    from qtpy.QtGui import QIcon
  File "c:\Users\silvan.zahno\AppData\Local\Continuum\anaconda3\lib\site-packages\qtpy\__init__.py", line 208, in <module>
    raise PythonQtError('No Qt bindings could be found')
qtpy.PythonQtError: No Qt bindings could be found
```

```
pip uninstall PyQt5
conda update conda
conda update anaconda-navigator
anaconda-navigator.exe
```

6.5.4 Install Plotly and Plotly Express

```
conda install -c plotly plotly_express plotly-orca
```

6.5.5 Better PDF Export

```
sudo apt-get install texlive-xetex
pip install jupyter_contrib_nbextensions
pip install cite2c
```

Chapter 7

Pandoc

7.1 Pandoc

- *Additional Arguments*
  - *Highlight Styles*
  - *PDF Output*
    - *For my template needed packages*
  - *Template*
    - *Windows*
    - *Linux*

If you need to convert files from one markup format into another, pandoc is your swiss-army knife.

- Pandoc Online
- Pandoc Download

7.1.1 Additional Arguments

Highlight Styles

```
# List all Highlight Styles
pandoc --list-highlight-styles
pygments
tango
espresso
zenburn
kate
monochrome
breezedark
haddock

## Pandoc Argument
--highlight-style breezedark
```

PDF Output

```
--pdf-engine=xelatex
```

For my template needed packages

- cm-super
  - Error no Scalable font
- koma-script
  - ! LaTeX Error: File `scrartcl.cls` not found.

Template

Latex Template needs to be in the following folders

Windows

```
C:\\Users\\<username>\\AppData\\Roaming\\pandoc\\templates
```

Linux

```
~/.pandoc/templates/
```

```
--template=<template>.latex
```

Chapter 8

AutoHotKey AHK

8.1 Tips & Tricks

- *Comment*
- *Performance and Compatility*
- *Warnings*
- *Enable Regex for Title mach Mode*
- *Tray Icon and ToolTip*
- *Examples*
  - *For Win10 Hibernate*
  - *For Win10 Sleep*
  - *Home and End Hotkey*
- *Check for AHK Version and output message*
- *Supend a script via Hotkey*

My ahk scripts can be found in the config repo

8.1.1 Comment

```
;------------
;-- Comment
;--
```

8.1.2 Performance and Compatility

```
; Recommended for performance and compatibility with future AutoHotkey releases
#NoEnv
```

8.1.3 Warnings

```
; Enable warnings to assist with detecting common errors
#Warn
```

8.1.4 Enable Regex for Title mach Mode

```
SetTitleMatchMode,RegEx ; then
IfWinExist, Total Commander.*
```

8.1.5 Tray Icon and ToolTip

```
Menu, TRAY, Icon, Favicon.ico
Menu, TRAY, Tip, Tooltip Text
```

8.1.6 Examples

For Win10 Hibernate

```
; Wait for Hotkey
;   Ctrl + Win + Alt + l
; Send Hotkey
;   Ctrl + Win + x + u + s
^#<!l::Send #xuh
```

For Win10 Sleep

```
; Wait for Hotkey
;   Ctrl + Win + l
; Send Hotkey
;   Win + x + u + s
^#l::Send #xus
```

Home and End Hotkey

```
; Ctrl + Left
^Left::Send {Home}
; Ctrl + Right
^Right::Send {End}
```

8.1.7 Check for AHK Version and output message

```
If (A_AhkVersion < "1.0.39.00")
{
    MsgBox,20,,This script may not work properly with your version of AutoHotkey.␣
↪Continue?
    IfMsgBox,No
    ExitApp
}
```

8.1.8 Supend a script via Hotkey

```
f1::suspend
```

8.2 Key Definitions

- *Raw Keys*
- *Double Keypress Detection*

8.2.1 Raw Keys

```
^  ; Ctrl
#  ; Win
<# ; Left Win
># ; Right Win
!  ; Alt
>! ; Right Altt
<! ; Left Alt
+  ; Shift
>+ ; Right Shift
<+ ; Left Shift
^  ; Control
<^ ; Left Control
>^ ; Right Control
; Multimedia
{Volume_Up}
{Volume_Down}
{Volume_Mute}
```

8.2.2 Double Keypress Detection

Alt Key in the example

```
~Alt::
DoubleAlt := A_PriorHotkey ="~Alt" AND A_TimeSincePriorHotkey < 400
Sleep 0
KeyWait Alt  ; This prevents the keyboard's auto-repeat feature from interfering.
return
```

8.3 Tips & Tricks

- *Comment*
- *Performance and Compatility*
- *Warnings*
- *Enable Regex for Title mach Mode*
- *Tray Icon and ToolTip*
- *Examples*
  - *For Win10 Hibernate*
  - *For Win10 Sleep*
  - *Home and End Hotkey*
- *Check for AHK Version and output message*
- *Supend a script via Hotkey*

My ahk scripts cna be found in the config repo

8.3.1 Comment

```
;------------
;-- Comment
;--
```

8.3.2 Performance and Compatility

```
; Recommended for performance and compatibility with future AutoHotkey releases
#NoEnv
```

8.3.3 Warnings

```
; Enable warnings to assist with detecting common errors
#Warn
```

8.3.4 Enable Regex for Title mach Mode

```
SetTitleMatchMode,RegEx ; then
IfWinExist, Total Commander.*
```

8.3.5 Tray Icon and ToolTip

```
Menu, TRAY, Icon, Favicon.ico
Menu, TRAY, Tip, Tooltip Text
```

8.3.6 Examples

For Win10 Hibernate

```
; Wait for Hotkey
;   Ctrl + Win + Alt + l
; Send Hotkey
;   Ctrl + Win + x + u + s
^#<!l::Send #xuh
```

For Win10 Sleep

```
; Wait for Hotkey
;   Ctrl + Win + l
; Send Hotkey
;   Win + x + u + s
^#l::Send #xus
```

Home and End Hotkey

```
; Ctrl + Left
^Left::Send {Home}
; Ctrl + Right
^Right::Send {End}
```

8.3.7 Check for AHK Version and output message

```
If (A_AhkVersion < "1.0.39.00")
{
    MsgBox,20,,This script may not work properly with your version of AutoHotkey.␣
↪Continue?
    IfMsgBox,No
    ExitApp
}
```

8.3.8 Supend a script via Hotkey

```
f1::suspend
```

Chapter 9

Python

9.1 Docstring

- *Python begin file*
- *Variables*
- *Functions*
  - *Function with types*
  - *Function with pep484 type annotations*
  - *Function modules level*
  - *Function - other examples*
- *Class*

9.1.1 Python begin file

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

"""Example NumPy style docstrings.

This module demonstrates documentation as specified by the `NumPy
Documentation HOWTO`_. Docstrings may extend over multiple lines. Sections
are created with a section header followed by an underline of equal length.
```

(continues on next page)

(continued from previous page)

```
Example
-------
Examples can be given using either the ``Example`` or ``Examples``
sections. Sections support any reStructuredText formatting, including
literal blocks::

    $ python example_numpy.py


Section breaks are created with two blank lines. Section breaks are also
implicitly created anytime a new section starts. Section bodies *may* be
indented:

Notes
-----
    This is an example of an indented section. It's like any other section,
    but the body is indented to help it stand out from surrounding text.

If a section is indented, then a section break is created by
resuming unindented text.

Attributes
----------
module_level_variable1 : int
    Module level variables may be documented in either the ``Attributes``
    section of the module docstring, or in an inline docstring immediately
    following the variable.

    Either form is acceptable, but the two should not be mixed. Choose
    one convention to document module level variables and be consistent
    with it.


.. _NumPy Documentation HOWTO:
   https://github.com/numpy/numpy/blob/master/doc/HOWTO_DOCUMENT.rst.txt

"""
```

9.1.2 Variables

```
module_level_variable1 = 12345

module_level_variable2 = 98765
"""int: Module level variable documented inline.

The docstring may span multiple lines. The type may optionally be specified
on the first line, separated by a colon.
"""
```

9.1.3 Functions

Function with types

```
def function_with_types_in_docstring(param1, param2):
    """Example function with types documented in the docstring.

    `PEP 484`_ type annotations are supported. If attribute, parameter, and
    return types are annotated according to `PEP 484`_, they do not need to be
    included in the docstring:

    Parameters
    ----------
    param1 : int
        The first parameter.
    param2 : str
        The second parameter.

    Returns
    -------
    bool
        True if successful, False otherwise.

    .. _PEP 484:
        https://www.python.org/dev/peps/pep-0484/

    """
```

Function with pep484 type annotations

```
def function_with_pep484_type_annotations(param1: int, param2: str) -> bool:
    """Example function with PEP 484 type annotations.

    The return type must be duplicated in the docstring to comply
    with the NumPy docstring style.

    Parameters
    ----------
    param1
        The first parameter.
    param2
        The second parameter.

    Returns
    -------
    bool
        True if successful, False otherwise.

    """
```

Function modules level

```
def module_level_function(param1, param2=None, *args, **kwargs):
    """This is an example of a module level function.

    Function parameters should be documented in the ``Parameters`` section.
    The name of each parameter is required. The type and description of each
    parameter is optional, but should be included if not obvious.

    If \*args or \*\*kwargs are accepted,
    they should be listed as ``*args`` and ``**kwargs``.

    The format for a parameter is::

        name : type
            description

            The description may span multiple lines. Following lines
            should be indented to match the first line of the description.
            The ": type" is optional.

            Multiple paragraphs are supported in parameter
            descriptions.

    Parameters
    ----------
    param1 : int
        The first parameter.
    param2 : :obj:`str`, optional
        The second parameter.
    *args
        Variable length argument list.
    **kwargs
        Arbitrary keyword arguments.

    Returns
    -------
    bool
        True if successful, False otherwise.

        The return type is not optional. The ``Returns`` section may span
        multiple lines and paragraphs. Following lines should be indented to
        match the first line of the description.

        The ``Returns`` section supports any reStructuredText formatting,
        including literal blocks::

            {
                'param1': param1,
                'param2': param2
            }

    Raises
    ------
    AttributeError
        The ``Raises`` section is a list of all exceptions
        that are relevant to the interface.
    ValueError
        If `param2` is equal to `param1`.

    """
    if param1 == param2:
```

(continues on next page)

(continued from previous page)

```
        raise ValueError('param1 may not be equal to param2')
    return True
```

Function - other examples

```
def example_generator(n):
    """Generators have a ``Yields`` section instead of a ``Returns`` section.

    Parameters
    ----------
    n : int
        The upper limit of the range to generate, from 0 to `n` - 1.

    Yields
    ------
    int
        The next number in the range of 0 to `n` - 1.

    Examples
    --------
    Examples should be written in doctest format, and should illustrate how
    to use the function.

    >>> print([i for i in example_generator(4)])
    [0, 1, 2, 3]

    """
    for i in range(n):
        yield i
```

```
class ExampleError(Exception):
    """Exceptions are documented in the same way as classes.

    The __init__ method may be documented in either the class level
    docstring, or as a docstring on the __init__ method itself.

    Either form is acceptable, but the two should not be mixed. Choose one
    convention to document the __init__ method and be consistent with it.

    Note
    ----
    Do not include the `self` parameter in the ``Parameters`` section.

    Parameters
    ----------
    msg : str
        Human readable string describing the exception.
    code : :obj:`int`, optional
        Numeric error code.

    Attributes
    ----------
    msg : str
        Human readable string describing the exception.
    code : int
        Numeric error code.

    """
```

(continues on next page)

(continued from previous page)

```
    def __init__(self, msg, code):
        self.msg = msg
        self.code = code
```

9.1.4 Class

```
class ExampleClass(object):
    """The summary line for a class docstring should fit on one line.

    If the class has public attributes, they may be documented here
    in an ``Attributes`` section and follow the same formatting as a
    function's ``Args`` section. Alternatively, attributes may be documented
    inline with the attribute's declaration (see __init__ method below).

    Properties created with the ``@property`` decorator should be documented
    in the property's getter method.

    Attributes
    ----------
    attr1 : str
        Description of `attr1`.
    attr2 : :obj:`int`, optional
        Description of `attr2`.

    """

    def __init__(self, param1, param2, param3):
        """Example of docstring on the __init__ method.

        The __init__ method may be documented in either the class level
        docstring, or as a docstring on the __init__ method itself.

        Either form is acceptable, but the two should not be mixed. Choose one
        convention to document the __init__ method and be consistent with it.

        Note
        ----
        Do not include the `self` parameter in the ``Parameters`` section.

        Parameters
        ----------
        param1 : str
            Description of `param1`.
        param2 : :obj:`list` of :obj:`str`
            Description of `param2`. Multiple
            lines are supported.
        param3 : :obj:`int`, optional
            Description of `param3`.

        """
        self.attr1 = param1
        self.attr2 = param2
        self.attr3 = param3  #: Doc comment *inline* with attribute

        #: list of str: Doc comment *before* attribute, with type specified
        self.attr4 = ["attr4"]

        self.attr5 = None
```

(continues on next page)

(continued from previous page)

```
        """str: Docstring *after* attribute, with type specified."""

    @property
    def readonly_property(self):
        """str: Properties should be documented in their getter method."""
        return "readonly_property"

    @property
    def readwrite_property(self):
        """:obj:`list` of :obj:`str`: Properties with both a getter and setter
        should only be documented in their getter method.

        If the setter method contains notable behavior, it should be
        mentioned here.
        """
        return ["readwrite_property"]

    @readwrite_property.setter
    def readwrite_property(self, value):
        value

    def example_method(self, param1, param2):
        """Class methods are similar to regular functions.

        Note
        ----
        Do not include the `self` parameter in the ``Parameters`` section.

        Parameters
        ----------
        param1
            The first parameter.
        param2
            The second parameter.

        Returns
        -------
        bool
            True if successful, False otherwise.

        """
        return True

    def __special__(self):
        """By default special members with docstrings are not included.

        Special members are any methods or attributes that start with and
        end with a double underscore. Any special member with a docstring
        will be included in the output, if
        ``napoleon_include_special_with_doc`` is set to True.

        This behavior can be enabled by changing the following setting in
        Sphinx's conf.py::

            napoleon_include_special_with_doc = True

        """
        pass

    def __special_without_docstring__(self):
        pass
```

(continues on next page)

(continued from previous page)

```
    def _private(self):
        """By default private members are not included.

        Private members are any methods or attributes that start with an
        underscore and are *not* special. By default they are not included
        in the output.

        This behavior can be changed such that private members *are* included
        by changing the following setting in Sphinx's conf.py::

            napoleon_include_private_with_doc = True

        """
        pass

    def _private_without_docstring(self):
        pass
```

9.2 General

- *flake8*
  - *.flake8*

Python samples

9.2.1 flake8

```
python -m flake8 test.py
```

.flake8

Flake8 configuration file is formated at `ini` File. and located at:

- Linux - `~/.config/flake8`
- Windows - `%userprofile%\.flake8`

see my config .flake8

```
[flake8]
max-line-length = 200

ignore =
  #E501: Line too long
  E501

  #E722 do not use bare 'except'
  E722

  #W504 line break after binary operator (one has to disable one of the W503/W504␣
↪pair)
```

(continues on next page)

(continued from previous page)

```
  W504

  #W391 blank line at end of file
  W391

exclude =
    .git,
    __pycache__,
    docs/source/conf.py,
    old,
    build,
    dist
```

9.3 Flake 8

- *.flake8*

Python samples

```
python -m flake8 test.py
```

9.3.1 .flake8

Flake8 configuration file is formated at `ini` File. and located at:

- Linux - `~/.config/flake8`
- Windows - `%userprofile%\.flake8`

see my config .flake8

```
[flake8]
max-line-length = 200

ignore =
  #E501: Line too long
  E501

  #E722 do not use bare 'except'
  E722

  #W504 line break after binary operator (one has to disable one of the W503/W504␣
↪pair)
  W504

  #W391 blank line at end of file
  W391

exclude =
    .git,
    __pycache__,
    docs/source/conf.py,
    old,
    build,
    dist
```

9.4 PIP

- *Admin*
- *Package*
- *Create `requirements.txt`*

9.4.1 Admin

```
# Show pip help
pip --help

# Show installed pip version
pip --version

# Update pip (Linux)
pip install --upgrade pip

# Update pip (Windows)
python -m pip install --upgrade pip
```

9.4.2 Package

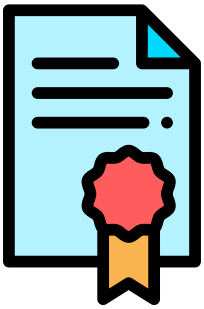
```
# Search a package
pip search <packagename>

# See package version
pip show <packagename>

# See all installed packages
pip list

# Install
pip install <packagename>
pip install -I <packagename>==<package version>
pip install -I ipython==5.4.0

# Uninstall
pip uninstall <packagename>
```

9.4.3 Create `requirements.txt`

```
pip freeze > requirements.txt
```

Chapter 10

Licenses



10.1 All rights reserved

```
All Rights Reserved

Copyright (c) 2019 - tschinz

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

10.2 MIT

```
MIT License

Copyright (c) 2019 tschinz

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
```

(continues on next page)

(continued from previous page)

```
The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

10.3 WTFPL

```
            DO WHAT THE FUCK YOU WANT TO PUBLIC LICENSE
                    Version 2, December 2004

Copyright (C) 2004 Sam Hocevar <sam@hocevar.net>

Everyone is permitted to copy and distribute verbatim or modified
copies of this license document, and changing it is allowed as long
as the name is changed.

            DO WHAT THE FUCK YOU WANT TO PUBLIC LICENSE
  TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
```

Chapter 11

ROS - Robot Operating System

11.1 Introduction

- *Philosophy*

ROS aka Robotic Operating System is not a OS itself but a framework and middleware.

- Software Framework for programming robots
- Prototype from Standfort AI Research Institute and created by Willow Garage in 2007
- Since 2013 maintained by the Open Source Robotics Foundation (OSRF)
- Consists of infrastrucutre, tools, capabilities and a ecosystem

Table1: Source : ROS Tutorial #1 - https://www.youtube.com/watch?v=9U6GDonGFHw&t=1s

Advantages	Disadvantages
Provides lots of infrastructure, tools and capabilities	Approaching maturity, but still changing
Easy to try other people's work and shar your own	Security and scalability are not first-class concerns
Large community	OSes other than Ubuntu Linux are not well supported
Free, open source, BSD license	
Great for open-source and researchers	**Not great for mission-critical tasks**

Figure1: ROS Equation

Plumbing	Tools	Capabilities	Ecosystem
Process management	Simulation	Control	Package organization
Inter-process communication	Visualization	Planning	Software distribution
Device drivers	Graphical user interface	Perception	Documentation
	Data logging	Mapping	Tutorials
		Manipulation	

11.1.1 Philosophy

- **Peer to peer** - Individual programs communicate over defined API (ROS messages, services, etc.).
- **Distributed** - Programs can be run on multiple computers and communicate over the network.
- **Multi-lingual** - ROS modules can be written in any language for which a client library exists (C++, Python, MATLAB, Java, etc.).
- **Thin** - The ROS conventions encourage contributors to create standalone libraries and then wrap those libraries so they can send and receive messages to and from other ROS modules.
- **Free and open source** - The core of ROS is released under the permissive BSD license, which allows commercial and noncommercial use.

11.2 Basics

- *Coding Rules*
- *Standard Unit in ROS*
- *Master*
- *Publisher and Subscribers*
- *Catkin Overview*
  - `src/` *Folder*
  - `build/` *Folder*
  - `devel/` *Folder*

- *install/ Folder*
- *Messages*

11.2.1 Coding Rules

The following rules apply when writing code with ROS.

Table2: ROS Robot Programming by TurtleBot3 Developers, section 7.1.3

Type	Naming Rule	Example
Package	under_scored	`first_ros_package`
Topic, Service	under_scored	`raw_image`
File	under_scored	`turtlebot3_fake.cpp`
Namespace	under_scored	`ros_awesome_package`
Variable	under_scored	`string table_name;`
Type	camelCased	`typedef int32_t PropertiesNumber;`
Class	camelCased	`class UrlTable`
Structure	camelCased	`struct UrlTableProperties`
Enumeration Type	camelCased	`enum ChoiceNumber`
Function	camelCased	`addTableEntry()`
Method	camelCased	`void setNumEntries(int32_t_num_entries)`
Constant	ALL_CAPITALS	`const uint8_t DAYS_IN_A_WEEK = T;`
Marco	ALL_CAPITALS	`#define PI_ROUNDED 3.0`

11.2.2 Standard Unit in ROS

Table3: Source : ROS Robot Programming by TurtleBot3 Developers, section 7.1.1

Quantity	Unit
Length	Meter
Mass	Kilogram
Time	Second
Current	Ampere
Angle	Radian
Frequency	Hertz
Force	Newton
Power	Watt
Voltage	Volt
Temperature	Celsius

11.2.3 Master

ROS `master` is a Server tracking all network addresses of all nodes. In addition to network addresses it also tracks other information like parameters. All `nodes` must know the network address of the master on startup `ROS_MASTER_URI`.

A `master` can be started with the `roscore` command or a `roslaunch` will also start a `master` if it doesn't exists already.

```
roscore
```

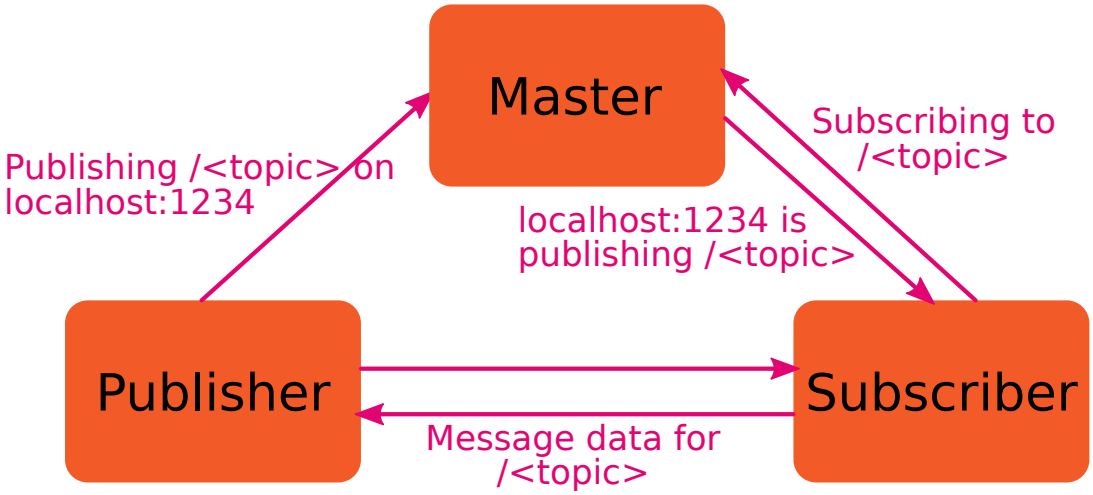


Figure2: ROS Master Publisher Slave

11.2.4 Publisher and Subscribers

With help of the `master`, `publisher` and `subscriber` establish a peer-to-peer connection. All `nodes` must know the network address of the master on startup `ROS_MASTER_URI`.

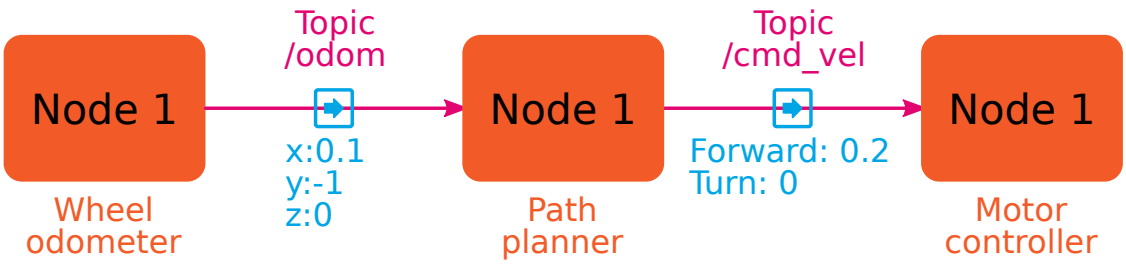


Figure3: ROS Publisher Slave

- Any node can publish a message to any topic
- Any node can subscribe to any topic

- Multiple nodes can publish to the same topic
- Multiple nodes can subscribe to the same topic
- A node can publish to multiple topics
- A node can subscribe to multiple topics

11.2.5 Catkin Overview

`src/` Folder

Location for create and clone new packages

The command `catkin_make` searches only in the `src/` folder for packages and builds them

It is a good practice to clone the ros packages into a different folder e.g. `~/git/<package_name>` and create a symlink into you catkin workspace

```
ls -s ~/git/<package_name>/ ~/catkin_ws/src/
```

`build/` Folder

`catkin_make` create buixld files and intermediate cache CMake files inside the `build/` folder.

`devel/` Folder

`catkin_make` builds each package, if successful, the target executable le is created. Executables are stored inside the `devel/` folder. Current workspace packages can be access by the command line if the following command is used:

```
# for bash
source ~/<workspace_name>/devel/setup.bash

# for zsh
source ~/<workspace_name>/devel/setup.zsh
```

It is beneficial to add this the the `~/.bashrc` or `~/.zshrc` file.

In addtion there is the `catkin_tools` program which simplifies the use.

See dedicated page: *Catkin Tools*

`install/` Folder

After building the executables in the `devel/` folder, this executables can be install by:

```
catkin_make install
```

See also:

- http://wiki.ros.org/catkin/workspaces#Catkin_Workspaces

11.2.6 Messages

- Serialization format for structured data
- Defined in a `.msg` file
- Compiled to C++/Python classes before using them
- more info https://wiki.ros.org/Messages

geometry_msgs/Point.msg

```
float64 x
float64 y
float64 z
```

sensor_msgs/Image.msg

```
std_msgs/Header header
  uint32 seq
  time   stamp
  string frame_id
uint32  height
uint32  width
string  encoding
uint8   is_bigendian
uint32  step
uint8[] data
```

geometry_msgs/PoseStamped.msg

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Pose pose
  geometry_msgs/Point position
    float64 x
    float64 y
    float64 z
  geometry_msgs/Quaternion orientation
    float64 x
    float64 y
    float64 z
    float64 w
```

11.3 Books summary

- *Topics*
- *SLAM (Simultaneous localization and modeling)*
- *TF (Transform Frames)*
- *QR code reader*
- *3D*
- *BAG recording*
- *Odometry and navigation*
- *Point Clouds*
- *OpenCV*

11.3.1 Topics

Basic topics such as workspace description, packages and nodes creation can be found in most of the book mentioned in this summary. They are not part of this summary since it focuses on more advanced topics. Tutorials to understand those topics are available in books or on the ROS wiki.

This summary lists all the books we have related to ROS, and some more specific PDF documents. Storage of the referenced documents :

- books : ros/books/
  - Effective_Robotics_Programming_with_ROS_3E.pdf
  - Learning_ROS_for_Robotics_Programming_2E.pdf
  - Mastering_ROS_for_Robotics_Programming.pdf
  - Programming_Robots_with_ROS.pdf
  - Programming_Robots_with_ROS-A_Practical_Introduction_to_the_Robot_Operating_System.pd
  - Robot_Operating_System_for_Absolute_Beginners.pdf
  - ROS_Robot_Programming.pdf
  - ROS_Robotics_By_Example.pdf
  - ROS_Robotics_By_Example_2E.pdf
  - Teach_ROS_with_No_Hassle_2E.pdf
- other documents : ros/slides/
  - octomap.pdf
  - ros-ethz-1.pdf
  - ros-ethz-2.pdf
  - ros-ethz-3.pdf
  - ros-ethz-4.pdf
  - ros-ethz-5a.pdf
  - ros-ethz-5b.pdf
  - ros-ethz-5c.pdf
  - ros-misc.pdf
  - ros-tf.pdf
  - ros-tf-2.pdf

11.3.2 SLAM (Simultaneous localization and modeling)

- Mastering_ROS_for_Robotics_Programming.pdf page 146

11.3.3 TF (Transform Frames)

- Effective_Robotics_Programming_with_ROS_3E.pdf page 171
- Learning_ROS_for_Robotics_Programming_2E.pdf page 305

11.3.4 QR code reader

- TODO

11.3.5 3D

- Effective_Robotics_Programming_with_ROS_3E.pdf page 120
- Learning_ROS_for_Robotics_Programming_2E.pdf page 143
- Mastering_ROS_for_Robotics_Programming.pdf page 265

11.3.6 BAG recording

- Effective_Robotics_Programming_with_ROS_3E.pdf page 128
- Learning_ROS_for_Robotics_Programming_2E.pdf page 120

11.3.7 Odometry and navigation

- Effective_Robotics_Programming_with_ROS_3E.pdf page 179
- Learning_ROS_for_Robotics_Programming_2E.pdf page 303
- Mastering_ROS_for_Robotics_Programming.pdf page 140

11.3.8 Point Clouds

- Effective_Robotics_Programming_with_ROS_3E.pdf page 394
- Learning_ROS_for_Robotics_Programming_2E.pdf page 231
- Mastering_ROS_for_Robotics_Programming.pdf page 251

11.3.9 OpenCV

- Effective_Robotics_Programming_with_ROS_3E.pdf page 359
- Mastering_ROS_for_Robotics_Programming.pdf page 250

11.4 Catkin Tools

11.4.1 Catkin build system

This Python package provides command line tools for working with the catkin meta-buildsystem and catkin workspaces. These tools are separate from the Catkin CMake macros used in Catkin source packages. It has to be installed seperately.

- https://catkin-tools.readthedocs.io/

Installation Catkin Tools

```
# Add ROS Repositories
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu `lsb_release -sc` main" >␣
↪/etc/apt/sources.list.d/ros-latest.list'
wget http://packages.ros.org/ros.key -O - | sudo apt-key add -

# Install via apt-get
sudo apt-get update
sudo apt-get install python-catkin-tools

# Install via pip
sudo pip install -U catkin_tools
```

11.4.2 Cheat Sheet

This is a non-exhaustive list of some common and useful invocations of the `catkin` command. All of the commands which do not explicitly specify a workspace path (with `--workspace`) are assumed to be run from within a directory contained by the target workspace. For thorough documentation, please see the chapters on each verb.

Initialize Workspaces

Initialize a workspace with a default layout (`src/build/devel`) in the *current* directory:

```
catkin init
catkin init --workspace .
catkin config --init
mkdir src && catkin build
```

... with a default layout in a *different* directory:

```
catkin init --workspace /tmp/path/to/my_catkin_ws
```

... which explicitly extends another workspace:

```
catkin config --init --extend /opt/ros/indigo
```

Initialize a workspace with a **source space** called `other_src`:

```
catkin config --init --source-space other_src
```

... or a workspace with **build**, **devel**, and **install space** ending with the suffix `_alternate`:

```
catkin config --init --space-suffix _alternate
```

Configuring Workspaces

View the current configuration:

```
catkin config
```

Setting and unsetting CMake options:

```
catkin config --cmake-args -DENABLE_CORBA=ON -DCORBA_IMPLEMENTATION=OMNIORB
```

```
catkin config --no-cmake-args
```

Toggle installing to the specified **install space**:

```
catkin config --install
```

Building Packages

Build all the packages:

```
catkin build
```

... one at a time, with additional debug output:

```
catkin build -p 1
```

... and force CMake to re-configure for each one:

```
catkin build --force-cmake
```

Build a specific package and its dependencies:

```
`catkin build <package_name>
```

... or ignore its dependencies:

```
catkin build <package_name> --no-deps
```

Build the package containing the current working directory:

```
catkin build --this
```

... but don't rebuild its dependencies:

```
catkin build --this --no-deps
```

Build packages with additional CMake args:

```
catkin build --cmake-args -DCMAKE_BUILD_TYPE=Debug
```

... and save them to be used for the next build:

```
catkin build --save-config --cmake-args -DCMAKE_BUILD_TYPE=Debug
```

Build all packages in a given directory:

```
catkin build $(catkin list -u /path/to/folder)
```

... or in the current folder:

```
catkin build $(catkin list -u .)
```

Cleaning Build Products

Blow away the build, devel, and install spaces (if they exist):

```
catkin clean
```

... or just the **build space**:

```
catkin clean --build
```

... or just clean a single package:

```
catkin clean PKGNAME
```

... or just delete the build directories for packages which have been disabled or removed:

```
catkin clean --orphans
```

11.5 Commandline Commands

11.5.1 Commandline Variables

```
echo $<variable_name>        # To display value

ROS_DISTRO                   # Distro name e.g. melodic
ROS_ETC_DIR                  #
ROS_LISP_PACKAGE_DIRECTORIES # common-lisp folder e.g. ~/catkin_ws/devel/share/
↪common-lisp
ROS_HOSTNAME                 # ros hostname e.g. localhost
ROS_MASTER_URI               # ros master url e.g. http://localhost:11311
ROS_PACKAGE_PATH             # package path's e.g. ~/catkin_ws/src:/opt/ros/$(ROS_
↪DISTRO)/share
ROS_PYTHON_VERSION           # python version 2 or 3 e.g. 2
ROS_ROOT                     # ros installation e.g. /opt/ros/$(ROS_DISTRO)/share/
↪ros
ROS_VERSION                  # ros version 1 or 2 e.g. 1
```

11.5.2 Useful commands

ROS tools

roscore

Launch ROS `master` core

```
roscore
```

rosversion

```
rosversion -d                              # Print ROS distro name
rosversion <package_name>                  # Print package vrosnode
```

rosparam

Nodes use the parameter server to store and retrieve parameters at runtime.

http://wiki.ros.org/rosparam

```
rosparam list                              # list parameter names
rosparam set /<parameter_name> <value>     # set parameter
rosparam get /<parameter_name>             # get parameter
rosparam delete /<parameter_name>          # delete parameter
rosparam dump <file>                       # dump parameter to file
rosparam load                              # load parameter from file
```

rosnode

Work with `nodes`

```
rosnode list                               # list all nodes
rosnode ping /<node_name>                  # check node connectivity
rosnode info /<node_node>                  # print information about node
rosnode machine                            # list nodes running on a particular␣
↪machine
rosnode kill /<node_name>                  # kill a running node
```

rostopic

Work with `topics`

```
rostopic list                              # list all topics
rostopic info /<topic_name>                # print information about active topic
rostopic echo /<topic_name>                # print messages to screen
rostopic pub /<topic_name> msg/MessageType "data:value" # pubish data to topic

rostopic type /<topic_name>                # print topic or field type
rostopic find <type>                       # find topics by type
rostopic bw /<topic_name>                  # display bandwidth used by topic
rostopic hz /<topic_name>                  # display publishing rate of topic
```

roslaunch

To start a launch file which can contain multiple nodes.

```
roslaunch <ros_pkg_name> <launch_file_name> # Launch ros launch file
```

rosrun

To run a node

```
rosrun <ros_pkg_name> <node_name>         # Start a ros node
rosrun <PACKAGE_NAME><NODE_NAME> __name:=<INSTANCE_NAME> # Start another instance␣
↪of a node, the parameter *INSTANCE_NAME* can be whatever you want, but it must␣
↪be unique.
```

rosservice

Work with services

```
rosservice list                           # list active services
rosservice info <service_name>            # print information about service
rosservice uri <service_name>             # print service ROSRPC
```

rosbag

ROS offers the possibility to record the data published on topics into bag files :

1. create a directory to store the bag files:

   ```
   ~/ mkdir ros_bag_files && cd ros_bag_files
   ```

2. run the *record* command :

   ```
   rosbag record -O <bag_name>.bag <topic_name> <topic_name>
   ```

3. play the bag file :

   ```
   rosbag play <bag_name>.bag
   ```

Many options are available for the *rosbag* command, see this page for more details.

Note : to play a bag with point clouds, it is required to have the following topics :

- /cloud
- /tf_static

The TF transformation is required, otherwise RViz can't display the point clouds.

```
rosbag record -O cloud.bag /cloud /tf_static
...
rosbag play cloud.bag
```

rosmsg

Display information about ros `messages`.

```
rosmsg list                      # List all messages
rosmsg info <message_name>       # Show message description
rosmsg package <package_name>    # List messages in a package
rosmsg packages <package_name>   # List packages that contain messages
```

Other Commands

```
roscd <PKG_NAME>                  # move to the folder of the package

rosinstall <PKG_NAME>             # install a ROS package

rosdep <PKG_NAME                  # install all the dependencies of a package

rqt                               # tool with many plug-ins available such as topic␣
↪publisher, service caller, …

rqt_graph                         # display the connections between nodes

rviz                              # launch the graphical tool to visualize robots,␣
↪point clouds, …

view_frames                       # create a PDFcalled ``frames.pdf`` with the TF␣
↪frames that are active
evince frames.pdf                 # show with evince the generated frames.pdf
```

Catkin

More info:

- http://wiki.ros.org/catkin/Tutorials

Create Package

1. new terminal
2. navigate to the source folder of the catkin workspace : `.../catkin_ws/src`
3. run : `catkin_create_pkg <PACKAGE_NAME> <DEPENDENCIES>`
4. update both CMakeLists.txt and package.xml note : *run_depend* has to be replaced by the *exec_depend*
5. write source code in the source folder of the package :
6. build the catkin workspace with the alias command : `cm`
7. launch the master as explained [here](ros-commands.md#roscore).
8. now launch the node as explained [here](#roslaunch) and [here](rosrun).

```
catkin_create_pkg <PKG_NAME> <PKG_DEPENDENCIES> # create a package, must be called␣
↪inside a catkin workspace
```

Build

```
cm
catkin_make                                     # build the whole workspace
catkin_make <PKG_NAME>                          # build a single package
```

Install

```
catkin_make install                             # installs all executables
catkin_make install <PKG_NAME>                  # installs single executables
```

Python modules

Tips :

- put the script in a folder called *scripts*
- make sure to run `chmod +x <script_name>.py` so that the script is recognized as an executable by ROS

11.5.3 Update services with RQT

1. launch *RQT* from a new terminal : run `rqt`
2. Search for the plugin *Service Caller*
3. Choose the service that you want to update
4. Fill the *expression* field with an expected parameter of this service
5. Call the service and the response is displayed

11.6 Installation

11.6.1 How to install ROS

This installation is based on Ubuntu 18.4 LTS and ROS Melodic Morenia.

Prerequisites

Some tools are not mandatory.

NTP

Only needed in a multi-pc system.

```
echo "Install Chrony and ntpdate"
sudo apt-get install -y chrony ntpdate
sudo ntpdate -q ntp.ubuntu.com
```

Sources

ROS Ubunbtu apt-get packages sources.

```
echo "Add ROS Package Sources"
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
↪ /etc/apt/sources.list.d/ros-latest.list'
```

Ubuntu 18.04 LTS (Bionic Beaver)

```
echo "Add ROS Package Sources for Ubuntu 18.04 LTS Bionic Beaver"
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
↪ /etc/apt/sources.list.d/ros-latest.list'
```

Keys

- ROS Kinetic
- ROS Melodic

```
echo "Add ROS Package Key"
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key␣
↪C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

ROS Base

```
echo "Install ROS Base Desktop Full"
sudo apt-get install ros-melodic-desktop-full
```

ROS Additional Packages

RQT

```
echo "Install ROS R-QT"
sudo apt-get install ros-melodic-rqt*
```

Individual ROS packages

Search & install individual ROS packages

```
echo "Install ROS R-QT"
apt-cache search ros-melodic
sudo apt-get install ros-melodic-[NAME_OF_PACKAGE]
```

Setup ROS Environment

Initialise rosdep

```
echo "[Initialize rosdep]"
sudo sh -c "rosdep init"
rosdep update
```

Environment setup

Differs depending if it's zsh or bash

```
echo "[Environment setup and getting rosinstall]"
if [ -n "$ZSH_VERSION" ]; then
   # assume Zsh
  source /opt/ros/$name_ros_version/setup.zsh
elif [ -n "$BASH_VERSION" ]; then
  # assume Bash
  source /opt/ros/$name_ros_version/setup.sh
fi
```

ROS Install

```
sudo apt install -y python-rosinstall python-rosinstall-generator python-wstool
```

Create catkin workspace

```
echo "[Make the catkin workspace and test the catkin_make]"
mkdir -p $HOME/$name_catkin_workspace/src
cd $HOME/$name_catkin_workspace/src
catkin_init_workspace
cd $HOME/$name_catkin_workspace
catkin_make
```

Shell Scripts

All the above can be done with help of the `ros-melodic-install.bash`

Additional Install

Hitachi SDK

```
cd ~/Downloads
echo "$INDENT Manually download http://hlds.co.jp/download/tofsdk/v2.3.0/
↪HldsTofSdk.2.3.0ubuntu16_x64.zip into your Downloads/ folder"
echo ""
echo "PRESS [ENTER] WHEN YOU'RE FINISHED AND TO CONTINUE THE INSTALLATION"
read
mkdir HldsTofSdk.2.3.0ubuntu16_x64
unzip HldsTofSdk.2.3.0ubuntu16_x64.zip -d ./HldsTofSdk.2.3.0ubuntu16_x64
sudo apt install HldsTofSdk.2.3.0ubuntu16_x64/libtof-dev_2.3.0-4ubuntu16_amd64.deb
```

11.6.2 Configuration

ROS Configuration

.bashrc

```
echo "[Set the ROS evironment in ~/.bashrc]"
echo "alias eb='vim ~/.bashrc'" >> ~/.bashrc
echo "alias sb='source ~/.bashrc'" >> ~/.bashrc
echo "alias gs='git status'" >> ~/.bashrc
echo "alias gp='git pull'" >> ~/.bashrc
echo "alias cw='cd ~/$name_catkin_workspace'" >> ~/.bashrc
echo "alias cs='cd ~/$name_catkin_workspace/src'" >> ~/.bashrc
echo "alias cm='cd ~/$name_catkin_workspace && catkin_make'" >> ~/.bashrc

echo "source /opt/ros/$name_ros_version/setup.bash" >> ~/.bashrc
echo "source ~/$name_catkin_workspace/devel/setup.bash" >> ~/.bashrc

echo "export ROS_MASTER_URI=http://localhost:11311" >> ~/.bashrc
echo "export ROS_HOSTNAME=localhost" >> ~/.bashrc
```

.zshrc

```
echo "[Set the ROS evironment in ~/.zshrc]"
echo "alias eb='vim ~/.zshrc'" >> ~/.zshrc
echo "alias sb='source ~/.zshrc'" >> ~/.zshrc
echo "alias gs='git status'" >> ~/.zshrc
echo "alias gp='git pull'" >> ~/.zshrc
echo "alias cw='cd ~/$name_catkin_workspace'" >> ~/.zshrc
echo "alias cs='cd ~/$name_catkin_workspace/src'" >> ~/.zshrc
echo "alias cm='cd ~/$name_catkin_workspace && catkin_make'" >> ~/.zshrc

echo "source /opt/ros/$name_ros_version/setup.zsh" >> ~/.zshrc
echo "source ~/$name_catkin_workspace/devel/setup.zsh" >> ~/.zshrc

echo "export ROS_MASTER_URI=http://localhost:11311" >> ~/.zshrc
echo "export ROS_HOSTNAME=localhost" >> ~/.zshrc
```

ROS Test

```
roscore
```

11.7 Launch

- *Launcher*
  - *Launch file*
    - *Arguments*
    - *Including other launch files*
  - *Create a launcher in a new package*
  - *Include another launcher inside this launcher*
  - *Parameters in launcher*
    - *Get the value of a parameter at run time*
    - *Public vs Private parameters*
- *Rviz configuration*

11.7.1 Launcher

- launch os a tool for launchine multiple nodes (as well as setting parameters)
- Are written in XM as `*.launch` files
- If not yet running, launch atuomatically stars a roscore

Browse to the folder and start a launch file with

```
roslaunch <file_name>.launch
```

Start a launch file from a package with

```
roslaunch <package_name> <file_name>.launch
```

Launch file

Listing 1: talker_listerner.launch

```
<launch>
<node name="listener" pkg="roscpp_tutorials" type="listener" output="screen"/>
<node name="talker" pkg="roscpp_tutorials" type="talker" output="screen"/>
</launch>
```

launch: Root element of the launch file

- **node**: Each `<node>` tag specifies a node to be launched
- **name**: Name of the node (free to choose)
- **pkg**: Package containing the node
- **type**: Type of the node, there must be a corresponding executable with the same name
- **output**: Specifies where to output log messages (screen: console, log: log file)

More Info

- http://wiki.ros.org/roslaunch/XML
- http://wiki.ros.org/roslaunch/Tutorials/Roslaunch%20tips%20for%20larger%20projects

Arguments

- Create re-usable launch files with `<arg>` tag, which works like a parameter (default optional)

  ```
  <arg name="arg_name" default="default_value"/>
  ```

- Use arguments in launch file with

  ```
  $(arg arg_name)
  ```

- When launching, arguments can be set with

  ```
  roslaunch launchf_file.launch arg_name:value
  ```

Example:

Listing 2: range_world.launch

```
<?xml version="1.0"?>
  <launch>
    <arg name="use_sim_time" default="true"/>
    <arg name="world" default="gazebo_ros_range"/>
    <arg name="debug" default="false"/>
    <arg name="physics" default="ode"/>

    <group if="$(arg use_sim_time)">
      <param name="/use_sim_time" value="true" />
    </group>
```

(continues on next page)

(continued from previous page)

```
    <include file="$(find gazebo_ros) /launch/empty_world.launch">
    <arg name="world_name" value="$(find gazebo_plugins)/ test/test_worlds/$(arg␣
↪world).world"/>
    <arg name="debug" value="$(arg debug)"/>
    <arg name="physics" value="$(arg physics)"/>
  </include>
</launch>
```

More info http://wiki.ros.org/roslaunch/XML/arg

Including other launch files

- Include other launch files with `<include>` tag to organize large projects

```
<include file="package_name" />
```

- Find the system path to other packages with

```
$(find package_name)
```

- Pass arguments to the included file

```
<arg name="arg_name" value="value"/>
```

Listing 3: range_world.launch

```
<?xml version="1.0"?>
<launch>
  <arg name="use_sim_time" default="true"/>
  <arg name="world" default="gazebo_ros_range"/>
  <arg name="debug" default="false"/>
  <arg name="physics" default="ode"/>

    <group if="$(arg use_sim_time)">
    <param name="/use_sim_time" value="true" />
  </group>

    <include file="$(find gazebo_ros) /launch/empty_world.launch">
    <arg name="world_name" value="$(find gazebo_plugins)/test/test_worlds/
↪$(arg world).world"/>
    <arg name="debug" value="$(arg debug)"/>
    <arg name="physics" value="$(arg physics)"/>
  </include>
</launch>
```

More info: http://wiki.ros.org/roslaunch/XML/include

Create a launcher in a new package

1. move to the folder of the package
2. run : `mkdir launch && cd launch`
3. run : `gedit <LAUNCHER_NAME>.launch`
4. fill the launcher file, for example:

```
<launch>
  <node pkg="<PACKAGE1_NAME>" type="<NODE1_NAME>" name="<INSTANCE0>"/>
  <node pkg="<PACKAGE2_NAME>" type="<NODE2_NAME>" name="<INSTANCE1>"/>
  <node pkg="<PACKAGE2_NAME>" type="<NODE2_NAME>" name="<INSTANCE2>"/>
  <node pkg="<PACKAGE2_NAME>" type="<NODE2_NAME>" name="<INSTANCE3>"/>
</launch>
```

Include another launcher inside this launcher

Add the include directive :

```
<launch>
  <include file="$(find <PKG_NAME>)/launch/<LAUNCHER_NAME>.launch" />
  </launch>
```

This is very useful to combine launcher together, or complete a first launcher :

- the first launcher is responsible to launch a driver
- the second launcher that includes the first one launches also a graphical tool on top of that

The advantage being that it is not necessary to copy paste all the code of the first launcher into the second one to use them together.

Parameters in launcher

Parameters can be set in the launcher and get by the node at run time. This is a convenient way to avoid rebuilding the code each time it is necessary to change the value of a variable, for example a path to a file.

The syntax is the following one :

```
<param name="<PARAM_NAME>" type="<TYPE>" value="<VALUE>" />
```

Get the value of a parameter at run time

It can be used in the node at run time with this C++ code :

```
ros::NodeHandle nh;
std::string iniPath;
nh.getParam("ini_path", iniPath);
```

The node handler gets the parameter called *ini_path* in the launcher and will store it in the variable *iniPath*. If the parameter is public, therefore accessible by all the nodes, this is sufficient to get its value. If the parameter is private to a node, then the node handler needs to know the name of the node :

```
ros::NodeHandle nh;
std::string iniName;
nh.getParam("tof_driver_1/ini_name", iniName);
```

To get the name of the node at run time, it is possible to use this line :

```
std::string nodeName = ros::this_node::getName();
```

Public vs Private parameters

Depending of where the parameter is declared in the launcher, the parameter will be either private to a node, or accessible by all the nodes. If the parameter is declared outside of a `<node></node>` tag, it is public and accessible to all the nodes. At the opposite, if the parameter is declared inside a `<node></node>` tag, it will only be accessible by the node, with the specific method described above.

In this example :

- The parameter *ini_path* is public and accessible by all the nodes only with its name.
- The parameter *ini_name* is private to each node and is accessible with the name of the node and its name, concatenated together. This allows to declare two time the same parameter with different value, as long as they are declared inside different nodes.

```
<launch>
      <!-- Public parameters for both nodes -->
      <param name="ini_path" type="str"
             value="$(find ros_driver_for_multiple_tof_sensors)/launch/" />

      <!-- Call the driver node for sensor 1 (IP = 192.168.0.105)-->
      <node pkg="ros_driver_for_multiple_tof_sensors"
            type="ros_driver_multiple_sensors_node" name="tof_driver_1"
            args="" required="true" output="screen" >

            <!-- Private parameter for node 1 -->
            <param name="ini_name" type="str" value="tof_sensor1.ini" />
      </node>

      <!-- Call the driver node for sensor 2 (IP = 192.168.1.105)-->
      <node pkg="ros_driver_for_multiple_tof_sensors"
            type="ros_driver_multiple_sensors_node" name="tof_driver_2"
            args="" required="true" output="screen" >

            <!-- Private parameter for node 2 -->
            <param name="ini_name" type="str" value="tof_sensor2.ini" />
      </node>
</launch>
```

11.7.2 Rviz configuration

After setting up the display configuration in Rviz, you can save it with the tab `File -> Save config as -> ...`

Then you can call it directly in the launch file by adding :

```
<node pkg="rviz" type="rviz" name="rviz"
    args="-d <PATH_TO_FILE>/<CONFIG_NAME>.rviz"/>
```

This will open Rviz with the saved configuration when the *launch* file is launched.

11.8 Lidar Driver

- *Install the SDK*

11.8.1 Install the SDK

run in a new terminal:

```
sudo dpkg -i libtof-dev_<version_number>ubuntu16_amd64.deb
```

11.9 Packages

- *Package Structure*
- *Package Files*
  - *file `package.xml`*
  - *file `CMakeLists.txt`*
- *Eclipse integration*
- *C++ Client Library*
  - *Example*
  - *Node Handle*
  - *Logging `ROS_INFO`*
    - *Severity Levels*
  - *Subscriber*
  - *Publisher*
  - *OOP*
  - *Parameter Server*
    - *C++ API*

11.9.1 Package Structure

ROS software is organized into packages, which can contain source code, launch files, configuration files, message definitions, data, and documentation. A package can depend on other packages called *dependencies*.

```
catkin_create_pkg <package_name> {dependencies}
```

A package need two things, its source code and the message definition. It is encouraging to place message definition into a separate folder.

- `package_name`
  - `config` - parameter files (`YAML`)
  - `include/package_name` - C++ include headers
  - `launch` - `*.launch` files
  - `src` - Source files
  - `test` - Unit and or ROS Tests
  - `CMakeList.txt` - CMake build file
  - `package.xml` - Package information
- `package_name_msgs`
  - `action` - Action definitions
  - `msg` - Message definitions
  - `src` - Service definitions
  - `CMakeList.txt` - CMake build file
  - `package.xml` - Package information

More info

- http://wiki.ros.org/Packages

11.9.2 Package Files

`package.xml`

- The `package.xml` file defines the properties of the package
  - Package name
  - Version number
  - Authors

- Dependencies on other packages
- ...

Listing 4: package.xml

```
<?xml version="1.0"?>
<package format="2">
    <name>ros_package_template</name>
    <version>0.1.0</version>
    <description>A template for ROS packages.</description>
    <maintainer email="user@email.ch">Firstname Lastname</maintainer>
    <license>BSD</license>
    <url type="website">https://github.com/link/ros_</url>
    <author email="user@email.ch">Firstname Lastname</author>

    <buildtool_depend>catkin</buildtool_depend>

    <depend>roscpp</depend>
    <depend>sensor_msgs</depend>
</package>
```

More info

- http://wiki.ros.org/catkin/package.xml

CMakeLists.txt

The `CMakeLists.txt` is the input to the CMake build system

1. Required CMake Version (`cmake_minimum_required`)
2. Package Name (`project()`)
3. Find other CMake/Catkin packages needed for build (`find_package()`)
4. Message/Service/Action Generators (`add_message_files()`, `add_service_files()`, `add_action_files()`)
5. Invoke message/service/action generation (`generate_messages()`)
6. Specify package build info export (`catkin_package()`)
7. Libraries/Executables to build (`add_library()/add_executable()/target_link_libraries()`)
8. Tests to build (`catkin_add_gtest()`)
9. Install rules (`install()`)

Listing 5: CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(husky_highlevel_controller)
add_definitions(--std=c++11)

find_package(catkin REQUIRED COMPONENTS roscpp sensor_msgs )

catkin_package(
  INCLUDE_DIRS include
  # LIBRARIES
  CATKIN_DEPENDS roscpp  sensor_msgs
  # DEPENDS
)
```

(continues on next page)

(continued from previous page)

```
include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(${PROJECT_NAME} src/${PROJECT_NAME}_node.cpp src/
↪HuskyHighlevelController.cpp)

target_link_libraries(${PROJECT_NAME} ${catkin_LIBRARIES})
```

More info

- http://wiki.ros.org/catkin/CMakeLists.txt

11.9.3 Eclipse integration

- Build the Eclipse project files with additional build flags

```
catkin build package_name --cmake-args -G"Eclipse CDT4 - Unix Makefiles" -D__
↪cplusplus=201103L D__GXX_EXPERIMENTAL_CXX0X__=1
```

- To use flags by default in your catkin environment, use the *catkin config* command.
- The Eclipse project files will be generated in *~/catkin_ws/build*

11.9.4 C++ Client Library

- http://wiki.ros.org/roscpp
- http://wiki.ros.org/roscpp/Overview

Example

```
#include <ros/ros.h>

int main(int argc, char** argv)                    // ROS main head file
{
    ros::init(argc, argv, "hello_world");          // has to be called before ROS␣
↪func's
    ros::NodeHandle nodeHandle;                    // access poiunt for␣
↪communication
    ros::Rate loopRate(10);                        // ros:Rate runs loops at␣
↪desired freq e.g. 10 = 10 Hz

    unsigned int count = 0;
    while (ros::ok()) {                            // checks if a node should␣
↪continue running
        ROS_INFO_STREAM("Hello World " << count); // ROS_info() logs messages from␣
↪fs
        ros::spinOnce();                           // processes incommind msg via␣
↪callbacks
        loopRate.sleep();
        count++;
    }
    return 0;
}
```

Node Handle

http://wiki.ros.org/roscpp/Overview/NodeHandles

```
// Default (public) node handle:      // Recommended
nh_ = ros::NodeHandle();              // /namespace/topic

// Private node handle:               // Recommended
nh_private_ = ros::NodeHandle("~");   // /namespace/node/topic

// Namespaced node handle:
nh_eth_ = ros::NodeHandle("hevs");    // /namespace/hevs/topic

// Global node handle:                // NOT Recommended
nh_global_ = ros::NodeHandle("/");    // /topic
```

Logging `ROS_INFO`

- http://wiki.ros.org/rosconsole
- http://wiki.ros.org/roscpp/Overview/Logging

Send text to log files and console. Instead of `std::cout`, use e.g. `ROS_INFO`.

Severity Levels

	Debug	Info	Warn	Error	Fatal
stdout	✓	✓			
stderr			✓	✓	✓
Log file	✓	✓	✓	✓	✓
/rosout	✓	✓	✓	✓	✓

Formatting Style

```
ROS_INFO("Result: %d", result);           // printf style
ROS_INFO_STREAM("Result: " << result);    // stream style
```

Launchfile

To see the output in the console set configuration to *screen* in the launch file.

```
<launch>
    <node name="listener" more="stuff" output="screen"/>
</launch>
```

Subscriber

http://wiki.ros.org/roscpp/Overview/Publishers%20and%20Subscribers

Start listening to a topic by calling the method subscribe() of the node handle

```
ros::Subscriber subscriber = nodeHandle.subscribe(topic, queue_size, callback_
↪function);
```

Example

Listing 6: listener.cpp

```
#include "ros/ros.h"
#include "std_msgs/String.h"

// callback function when a message is received
void chatterCallback(const std_msgs::String& msg) {
    ROS_INFO("I heard: [%s]", msg.data.c_str());
}

int main(int argc, char **argv) {
    ros::init(argc, argv, "listener");
    ros::NodeHandle nodeHandle;
    // Subscript to topic with a queue size of 10 (1-10 is recommended)
    ros::Subscriber subscriber = nodeHandle.subscribe("chatter", 10,
↪chatterCallback);
    ros::spin(); // stay's here forever
    return 0;
}
```

Publisher

http://wiki.ros.org/roscpp/Overview/Publishers%20and%20Subscribers

Create a publisher with help of the node handle

```
ros::Publisher publisher = nodeHandle.advertise<message_type>(topic, queue_size);
```

Example

```
:caption: talker.cpp
#include <ros/ros.h>
#include <std_msgs/String.h>

int main(int argc, char **argv) {
    ros::init(argc, argv, "talker");
    ros::NodeHandle nh;
    // Node handle queue size of 1
    ros::Publisher chatterPublisher = nh.advertise<std_msgs::String>("chatter", 1);
    ros::Rate loopRate(10);

    unsigned int count = 0;
    while (ros::ok()) {
        std_msgs::String message;
        // Cretae message content
        message.data = "hello world " + std::to_string(count);
        ROS_INFO_STREAM(message.data);
        chatterPublisher.publish(message);
        ros::spinOnce();
        loopRate.sleep();
```

(continues on next page)

(continued from previous page)

```
        count++;
    }
    return 0;
}
```

OOP

http://wiki.ros.org/roscpp_tutorials/Tutorials/UsingClassMethodsAsCallbacks

Example

```
:caption: my_package_node.cpp
#include <ros/ros.h>
#include "my_package/MyPackage.hpp"
int main(int argc, char** argv) {
    ros::init(argc, argv, "my_package");
    ros::NodeHandle nodeHandle("~");
    // Call
    my_package::MyPackage myPackage(nodeHandle);

    ros::spin();
    return 0;
}
```

class MyPackage	class Algorithm
Main node class providing ROS interface (subscribers, parameters, timers etc.)	Class implementing the algorithmic part of the node **Note: The algorithmic part of the code could be separated in a (ROS-independent) library**

Parameter Server

http://wiki.ros.org/roscpp/Overview/Parameter%20Server

Example Parameter File

```
:caption: config.yaml

camera:
     left:
             name: left_camera
             exposure: 1
     right:
             name: right_camera
             exposure: 1.1
```

Example Launch file

```
<launch>
    <node name="name" pkg="package" type="node_type">
        <rosparam command="load" file="$(find package)/config/config.yaml" />
    </node>
</launch>
```

C++ API

```
ros::NodeHandle nodeHandle("~");
std::string topic;
if (!nodeHandle.getParam("topic", topic)) {
    ROS_ERROR("Could not find topic parameter!");
}
```

Get a parameter in C++ with

```
nodeHandle.getParam(parameter_name, variable)
```

- Method returns `true` if parameter was found, `false` otherwise
- Global and relative parameter access:
  - Global parameter name with preceding `/`

    ```
    nodeHandle.getParam("/package/camera/left/exposure", variable)
    ```

    Relative parameter name (relative to the node handle)

    ```
    nodeHandle.getParam("camera/left/exposure", variable)
    ```
- For parameters, typically use the private node handle

  ```
  ros::NodeHandle("~")
  ```

11.10 External Packages and Nodes

- *Terminology*
- *Overview*
- *3D Mapping*
  - *SLAM*
    - *Octomap_server : +*
    - *Hector slam : +*
    - *REMODE : ~*
  - *LOAM*
    - *RTABMAP : +*
    - *Spin Hokuyo : +*
    - *Lego-LOAM : ~*
  - *Velodyne loam : ~*
  - *Bad solution -*
- *Modbus*
- *Object Tracking*
  - *Multiple objects lidar tracking : ~*
- *Object Detection*

- *QR code readers*

11.10.1 Terminology

- + : interesting topics and hardware abstraction
- ~ : interesting, but quite a lot of work to do for hardware compatibility or mapping
- - : bad solution

11.10.2 Overview

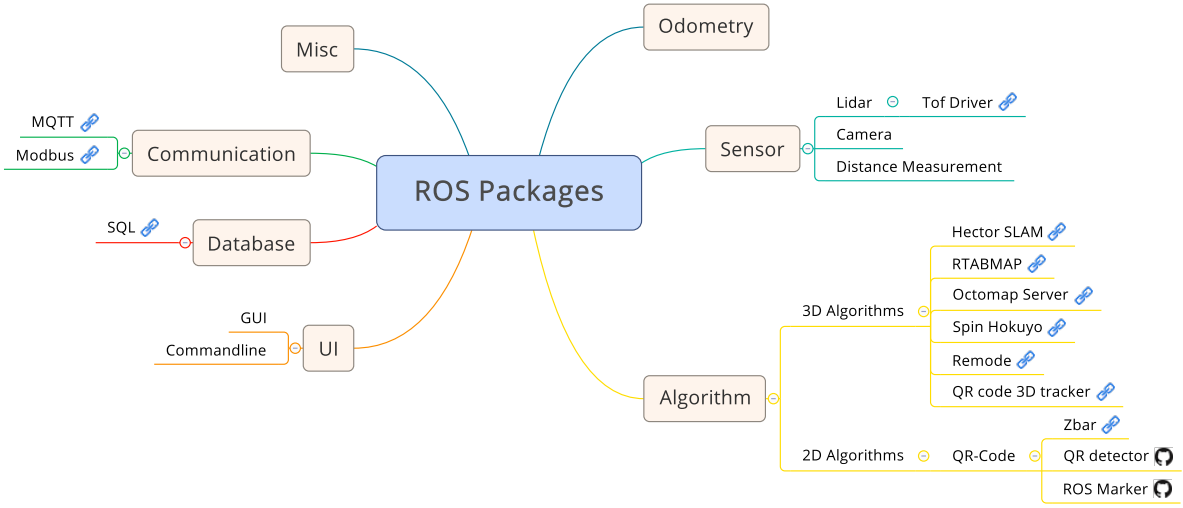


Figure4: ROS Packages Overview

11.10.3 3D Mapping

SLAM

Octomap_server : +

3D occupancy grid mapping, independent from sensor, looks like it does not need odometry

- https://youtu.be/yp0f8-AKvDU
- https://wiki.ros.org/octomap_mapping
- https://wiki.ros.org/octomap_server
- http://octomap.github.io/

Plus :

- maintained
- compatible with melodic
- documentation available as well as many
- no odometry

- independent from hardware (only require the right input topics)

Minus :

- …

Inputs required :

- sensor_msgs/PointCloud2

Hector slam : +

- https://github.com/tu-darmstadt-ros-pkg/hector_slam
- http://wiki.ros.org/hector_slam

Not sure whether we're interested in hector slam itself, or on the

Plus :

- maintained
- not directly compatible with melodic, but easy to build it from source for melodic
- odometry not needed

Minus :

- mostly created for 2D mapping and robot navigation
- not much documentation

Inputs required :

- …

REMODE : ~

- https://www.ros.org/news/2016/02/open-source-release-remode-probabilistic-monocular-dense-rec
  html

modeling of many 3D objects, like rooms, persons, …

Plus :

- noise reduction
- nice rendering

Minus :

- not much documentation and precisions about `hardware/drivers/topics`
- maybe "too much" for our needs ?
- looks like it is not maintained anymore : latest commit was 4 years ago

Inputs :

- …

LOAM

RTABMAP : +

- http://wiki.ros.org/rtabmap_ros

Plus :

- maintained
- compatible with melodic
- real time mapping
- publishes :
  - 3D point clouds
  - 2D occupancy maps
- tutorials and documentation available

Minus :

- oriented towards robot navigation, although "top-down" modeling seems to be possible

Inputs required :

- odometry (not mandatory in all cases)
- scan 2D or 3D

Spin Hokuyo : +

- https://github.com/RobustFieldAutonomyLab/spin_hokuyo
- http://wiki.ros.org/spin_hokuyo

It creates a point cloud with a 2D LiDaR and a servomotor. The interesting node compiles small point clouds to make one big point cloud. Could be very useful to make our digital model.

Plus :

- has a node that compiles point clouds and publish them on a topic
- great rendering

Minus :

- designed for another sensor, but the node that compiles point clouds does not care about that
- need some odometry work

Inputs required :

- laser scan
- odometry

Lego-LOAM : ~

- https://github.com/RobustFieldAutonomyLab/LeGO-LOAM

Plus :

- good rendering

Minus :

- designed for robot navigation, not for "top-down mapping"
- designed for another sensor (velodyne)

Inputs :

- …

Velodyne loam : ~

- http://wiki.ros.org/loam_velodyne

Plus :

- good rendering
- builds 3D maps

Minus :

- for velodyne sensor
- robot navigation

Inputs :

- …

Bad solution -

- https://github.com/koide3/hdl_graph_slam : not what we need. creates maps with corridors and doors, but not "top-down" mapping
- http://wiki.ros.org/robot_pose_ekf : not what we need
- http://wiki.ros.org/ethzasl_icp_mapper : doc not up to date, slowly not maintained anymore, …
- https://github.com/ethz-asl/libpointmatcher/blob/master/doc/index.md

11.10.4 Modbus

- http://wiki.ros.org/modbus

11.10.5 Object Tracking

Multiple objects lidar tracking : ~

- https://github.com/praveen-palanisamy/multiple-object-tracking-lidar

Plus :

- tracks objects in real time
- hardware independent

Minus :

- 2D maps, most likely used for robot navigation

Inputs :

- ...

11.10.6 Object Detection

- https://www.acin.tuwien.ac.at/vision-for-robotics/software-tools/v4r-library/
- https://rgit.acin.tuwien.ac.at/v4r/v4r_ros_wrappers
- http://wiki.ros.org/object_recognition
- https://www.osrfoundation.org/ros2-object-detection-demo/
- http://wiki.ros.org/find_object_2d

11.10.7 QR code readers

- http://wiki.ros.org/zbar_ros
- https://github.com/mdrwiega/qr_detector
- http://wiki.ros.org/visp_auto_tracker

11.11 RViz

- *Overview*
- *Run*
- *Built-In Display Types*

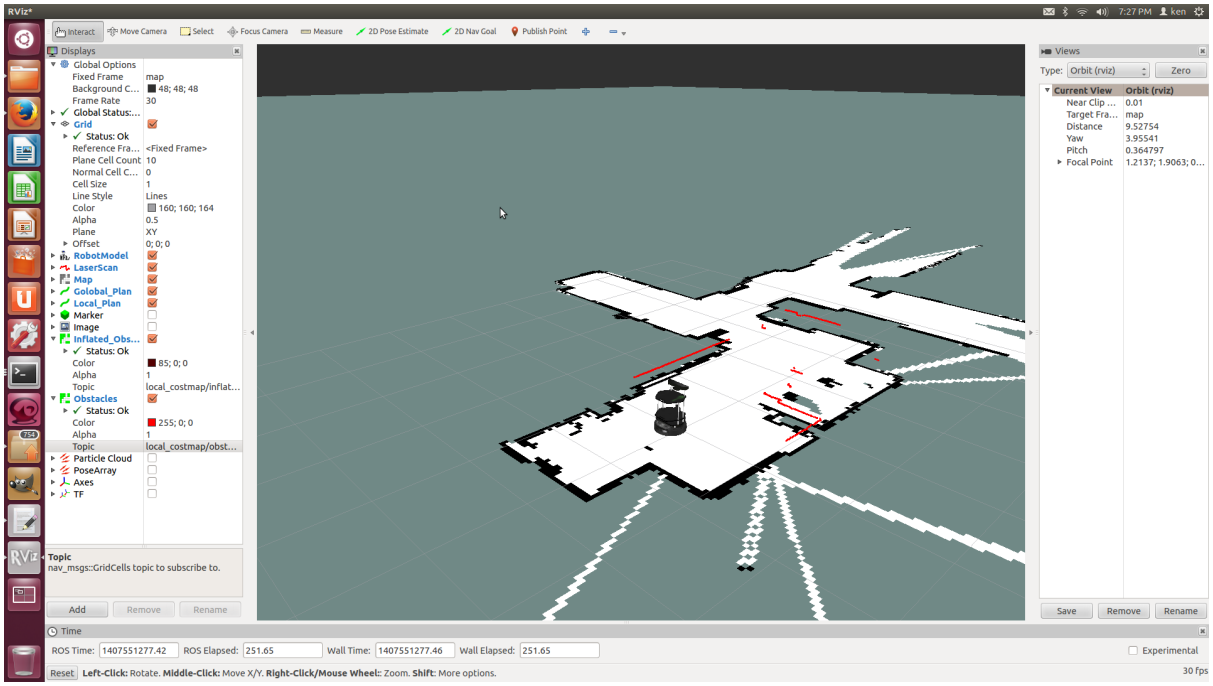
11.11.1 Overview

http://wiki.ros.org/rviz

- 3D visualization tool for ROS
- Subscribes to topics and visualizes the message contents
- Different camera views (orthographic, top-down, etc.)
- Interactive tools to publish user information
- Save and load setup as RViz configuration
- Extensible with plugins

11.11.2 Run

```
rosrun rviz rviz
```



Save configuration with `ctrl+s`

11.11.3 Built-In Display Types

Name	Description	Messages Used
Axes	Displays a set of Axes	
Effort	Shows the effort being put into each revolute joint of a robot.	sensor_msgs/JointStates
Camera	Creates a new rendering window from the perspective of a camera, and overlays the image on top of it.	sensor_msgs/Image, sensor_msgs/CameraInfo
Grid	Displays a 2D or 3D grid along a plane	
Grid Cells	Draws cells from a grid, usually obstacles from a costmap from the navigation stack.	nav_msgs/GridCells
Image	Creates a new rendering window with an Image. Unlike the Camera display, this display does not use a CameraInfo. *Version: Diamondback+*	sensor_msgs/Image
Interactive-Marker	Displays 3D objects from one or multiple Interactive Marker servers and allows mouse interaction with them. *Version: Electric+*	visualization_msgs/InteractiveMarker
Laser Scan	Shows data from a laser scan, with different options for rendering modes, accumulation, etc.	sensor_msgs/LaserScan
Map	Displays a map on the ground plane.	nav_msgs/OccupancyGrid
Markers	Allows programmers to display arbitrary primitive shapes through a topic	visualization_msgs/Marker, visualization_msgs/MarkerArray
Path	Shows a path from the navigation stack.	nav_msgs/Path
Point	Draws a point as a small sphere.	geometry_msgs/PointStamped
Pose	Draws a pose as either an arrow or axes.	geometry_msgs/PoseStamped
Pose Array	Draws a "cloud" of arrows, one for each pose in a pose array	geometry_msgs/PoseArray
Point Cloud(2)	Shows data from a point cloud, with different options for rendering modes, accumulation, etc.	sensor_msgs/PointCloud, sensor_msgs/PointCloud2
Polygon	Draws the outline of a polygon as lines.	geometry_msgs/Polygon
Odometry	Accumulates odometry poses from over time.	nav_msgs/Odometry
Range	Displays cones representing range measurements from sonar or IR range sensors. *Version: Electric+*	sensor_msgs/Range
RobotModel	Shows a visual representation of a robot in the correct pose (as defined by the current TF transforms).	
TF	Displays the ros wiki tf transform hierarchy.	
Wrench	Draws a wrench as arrow (force) and arrow + circle (torque)	geometry_msgs/WrenchStamped
Oculus	Renders the RViz scene to an Oculus headset	

11.12 Transform Frames

A frame in the ROS language is a specific coordinate system in the space. ROS abstracts elements of a robot as coordinates frames. Each physical part of a robot that has a particular meaning will most likely have its own frame :

- a sensor : *laser_frame*
- an arm : *left_arm_frame*

It is up to the programmer to create frames where it is necessary, but some frames are already defined by ROS (see below).

Each frame has it own origin and coordinate system :

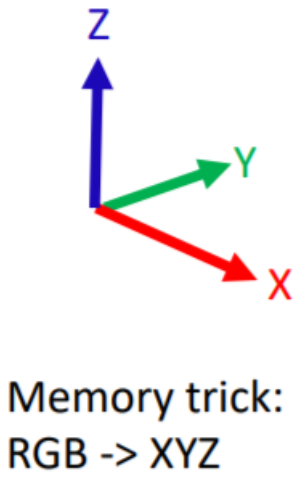


Figure5: coordinate frame axis

To keep trace of the frames in the whole coordinate system, they must all refer to a main frame. Knowing the position of the main frame and the relative positions of all the other frames, ROS is able to know the exact position of each frame all continuously.

The TF2 package tracks the coordinate frames. There are several predefined frames :

- *world* : kind of the parent of all the frames, does not move, there is only one single *world*
- *map* : child of *world*, can be freely fixed in the world frame, does not move compared to the *world*, but it can be several *map* frames in a *world* (usually one *map* per robot)
- *odom* : child of *map*, fixed at the start point of the robot in the *map* frame, does not move compared to *world* and *map*
- *base_link* : kind of the reference frame of a robot, it is moving in *odom*, therefore moving in *map* and *world*
- ...

The TF tree shows the relations between the frames :

One can create coordinate frames for each part of the robot that needs to be tracked, for example :

- *scanner_frame* : position of the scanner on a robot, somehow linked to the *base_link*
- *wheels_frame* : position of the wheels on a robot, somehow linked to the *base_link*

The links between the *base_link* and the other frames can be direct, or they can be relative to it via other frames.

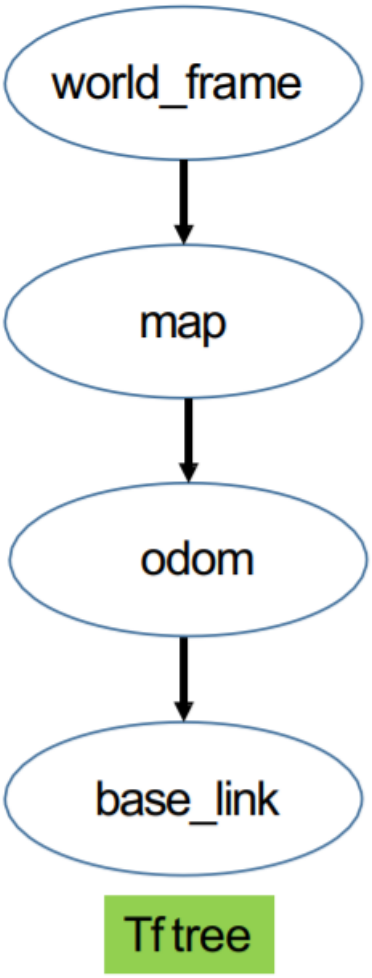


Figure6: tf tree

They are linked together by TF (transform frames). TF can be either static, which means that the relation between two frames will never change (for example two sensors being fixed 1 meter away), or dynamic when the relation evolves in the time (for example the arm of a robot compared to its head).

Let us use the example our two LIDAR sensor : they are oriented in the same way, they are on the same table, the only difference being there is 2.15 meter between them. For this example, they will never move nor rotate. We can use the node *static_transform_publisher* to inform other nodes that will use their data of their relative position. We will also fix them in the *world*, *map* and *base_link* frame.

Since the *base_link* frame will not move neither, it will also be fixed to the *map* by a static transform. The static transformations are called as a node from a launcher :

```
<launch>
      <!-- From world to map, same origin -->
      <node pkg="tf2_ros" type="static_transform_publisher" name="world_to_map"
            args="0 0 0 0 0 0 /world /map" />

      <!-- From map to base_link, fixed in this case -->
      <node pkg="tf2_ros" type="static_transform_publisher" name="map_to_base_link"
            args="0 0 0 0 0 0 /map /base_link" />

      <!-- From base_link to laser_frame1, position of the first lidar -->
      <node pkg="tf2_ros" type="static_transform_publisher" name="base_link_to_
↪lidar1"
            args="0 0 0 0 0 0 /base_link /lidar1_frame" />

      <!-- From base_link to laser_frame2, position of the second lidar -->
      <node pkg="tf2_ros" type="static_transform_publisher" name="base_link_to_
↪lidar2"
            args="-2.15 -0.01 0 0 0 0 /base_link /lidar2_frame" />
</launch>
```

Which will produce the following TF tree :

The arguments are :

- translations in X, Y, Z
- rotations around X, Y, Z
- parent *frame_id*
- child *frame_id*

Each topic has a reference frame. This means that each message being published on a topic kind of contains the position "from where it comes". This is the *frame_id* parameter. The node that will published the data of the LIDAR shall publish them with the right *frame_id*, otherwise the TF tree will not be able to link all the TF together.

Documentation about frames and transformations can be found there :

- tf2

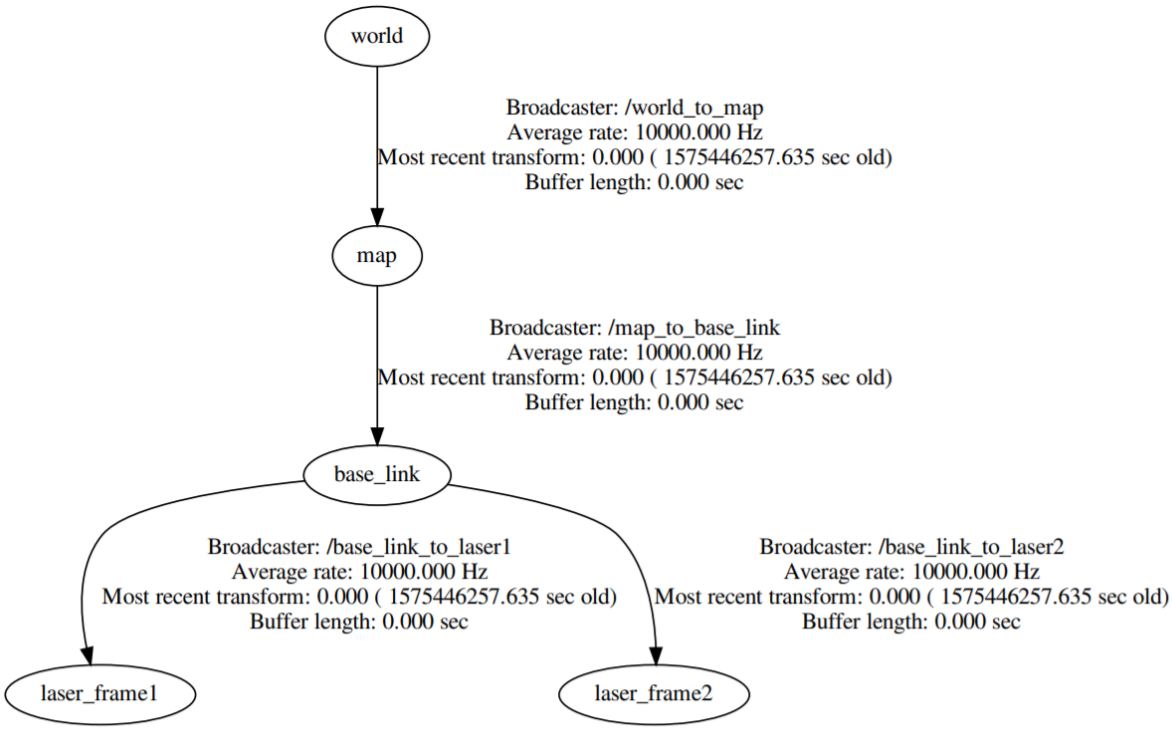


Figure7: lidar tf tree

Chapter 12

LaTeX

12.1 Introduction

- *Some LaTeX helppages*
- *Generate PDF files*

12.1.1 Some LaTeX helppages

- HEI SPL Latex Templates
- Cheatsheet A Guide to Latex
- Tex Stackexchange Forum

12.1.2 Generate PDF files

Latex is best suited to insert images as pdf. In order to convert images or svg into pdf use inkscape Convert `*.svg` images with inkscape to `*.pdf` and `*.pdf_tex`

```
inkscape -D -z --file=image.svg --export-pdf=image.pdf --export-latex
```

12.2 Installation LaTeX

- *Base Install*
  - *Linux*
  - *Windows*
- *Manual Package install*
  - *Manual Package Linux*
  - *Manual Package Windows*

12.2.1 Base Install

Linux

Package	Archives	Disk Space
texlive-latex-base	59 MB	216 MB
texlive-latex-recommended	74 MB	248 MB
texlive-pictures	83 MB	277 MB
texlive-fonts-recommended	83 MB	281 MB
texlive	98 MB	314 MB
texlive-plain-generic	82 MB	261 MB
texlive-latex-extra	144 MB	452 MB
texlive-full	2804 MB	5358 MB

see also Tex Stack Exchange

```
                      texlive-latex-extra
                                 :   :
       +-----------------+       :   :..................
                                 :                     :
       v                         :                     v
texlive-pictures                 :           texlive-plain-generic
                                 :
            +----------------+   :...............
                                                :
                        texlive                 :
                                                :
                +----------+   +----------+    :
                                                :
       v    v    v                         v    v
texlive-latex-recommended        texlive-fonts-recommended
                                            :
                                            v
                                           tipa

                +----------+   +----------+

                            v v v
                     texlive-latex-base
```

```
sudo apt-get install texlive-latex-extra
```

Windows

- Install MikTeX - https://miktex.org/download
- MikTeX Packages
  - minted

    ```
    pip install pygments
    ```

    add Python Scripts to `PATH` Environment Variable. `%USERPROFILE%\AppData\Local\Continuum\anaconda3\Scripts\`
- Install TeXstudio
  - https://texstudio.org
  - Options => Configure TeXstudio => Commands => add Interpreter Flag `-shell-escape`
  - enable line numbers
  - enable white spaces
- Install Inkscape
  - https://inkscape.org/release/

12.2.2 Manual Package install

For manual installing `*.sty` Packages and `*.cls` Class files.

> **Warning:** For every package create a separate folder

Manual Package Linux

- Find `TEXMFHOME` directory

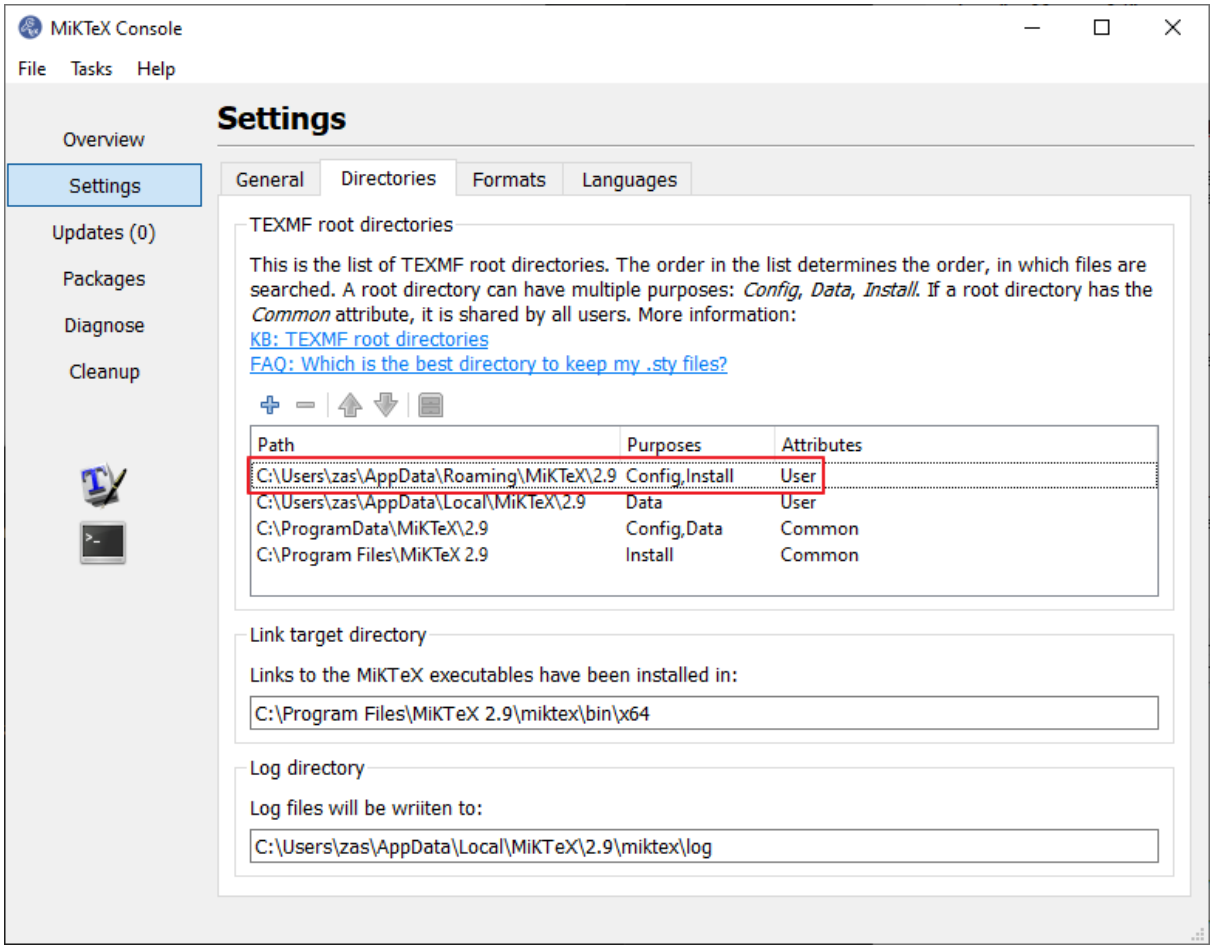
  ```
  kpsewhich -var-value TEXMFHOME
  ```

- Navigate to `$(TEXMFHOME)/tex/latex`
- Copy paste your `*.sty` and `*.cls`
- Update Package index

  ```
  texhash
  ```

Manual Package Windows

- Open MikTeX Console and go to `Settings -> Directories`
- The `Config,Install` and `User` folder is the location of your Packages: `%USERPROFILES%/AppData/Roaming/MikTeX/2.9/`
- Inside you have to navigate to `tex/latex/` folder
- `%USERPROFILES%/AppData/Roaming/MikTeX/2.9/tex/latex/`
- Copy paste your `*.sty` and `*.cls`


MiKTeX Console
File
Tasks
Help
Overview
Settings
Updates (0)
Packages
Diagnose
Cleanup
Settings
General
Directories
Formats
Languages
TEXMF root directories
This is the list of TEXMF root directories. The order in the list determines the order, in which files are searched. A root directory can have multiple purposes: Config, Data, Install. If a root directory has the Common attribute, it is shared by all users. More information:
KB: TEXMF root directories
FAQ: Which is the best directory to keep my .sty files?
Path
Purposes
Attributes
C:\Users\zas\AppData\Roaming\MiKTeX\2.9
Config,Install
User
C:\Users\zas\AppData\Local\MiKTeX\2.9
Data
User
C:\ProgramData\MiKTeX\2.9
Config,Data
Common
C:\Program Files\MiKTeX 2.9
Install
Common
Link target directory
Links to the MiKTeX executables have been installed in:
C:\Program Files\MiKTeX 2.9\miktex\bin\x64
Log directory
Log files will be wriiten to:
C:\Users\zas\AppData\Local\MiKTeX\2.9\miktex\log

- Update Package index

```
texhash
```

Chapter 13

ReStructuredText

13.1 Introduction

- *Some RST Syntax helppages*

13.1.1 Some RST Syntax helppages

- `rst-cheatsheet.pdf`
- Thomas Cokelaer RST Sphinx Syntax
- Docutil Quickref
- Raslina RST Cheatsheet

13.2 RST and Sphinx Cheatsheet

In this page you will get a quick overview about the most used syntax.

13.2.1 Table of content

To include a table of content of all title in a page use

```
.. contents:: :local:
```

13.2.2 Titles

The lines have to be as long or longer than the text.

```
=============
Section Title
=============

Titles
======

Paragraph
---------

Sub-Pragraph
^^^^^^^^^^^^
```

13.2.3 Markup

`*emphasis*`	*emphasis*
`**strong emphasis**`	**strong emphasis**
`` `interpreted text` ``	The rendering and meaning of interpreted text is domain- or application-dependent.
`` ``inline literal`` ``	`inline literal`
`:markup:`	**markup**
`> quote markup`	> quote markup

13.2.4 Links

External Links

Internet

```
`python <http://www.python.org/>`_
`<http://www.python.org/>`_
http://www.python.org/
```

python

http://www.python.org/

http://www.python.org/

Other Repo's

The plugin `'sphinx.ext.extlinks` allows creating shortcuts

```
extlinks = {'config_
↪repo': ('https://github.com/tschinz/config/%s', None),
            'zawiki_
↪repo': ('https://github.com/tschinz/zawiki/%s', None)
            }
```

```
:config_repo:`jupyter config <tree/master/config/jupyter>`
:zawiki_repo:`zawiki link <>`
```

jupyter config zawiki link

Other Sphinx Pages

- absolute link from root *About*
- relative link from document location *About*

```
* absolute link from root
  :doc:`/about/index`

* relative link from document location
  :doc:`../../about/index`
```

In order to link to another subheader in another document you need to use *Internal References*.

In the page to be jumped to add `.. _ref_name:`, and then you can:

```
:ref:`ref_name`
:ref:`link title<ref_name>`
```

Like so:

- *How to use Sphinx Documentation*
- *Sphinx Doc Link*

Internal Links

Link to Titles

Link to titles directly is done with the extension `sphinx.ext.autosectionlabel`.

Important: You need to add the `folder_name` and `subfolder(s)`_name name as well as `file_name` without `.rst` extension in order to reference a section title. This avoids the duplicated label warning.

```
:ref:`Displayname␣
↪<folder_name/subfolder_name/file_name/section_title>`
```

```
:ref:`Back␣
↪to top <writing/rst/cheatsheet:RST and Sphinx Cheatsheet>`

:ref:`writing/rst/cheatsheet:Images`
```

Back to top

Images

Internal References

In any place of the document a reference point can be inserted and later refered to.

```
.. _ref-point:

see :ref:`ref-point`
```

see *Internal References*

File Links

To link to a file within the Sphinx file structure use the Role `:download:`

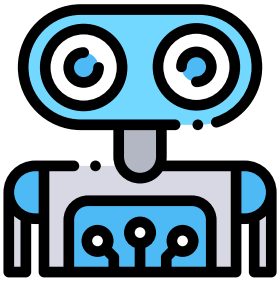
```
:download:`../../coding/
↪ros/books/Mastering_ROS_for_Robotics_Programming.pdf`

:download:`Mastering_
↪ROS_for_Robotics_Programming <../../coding/
↪ros/books/Mastering_ROS_for_Robotics_Programming.pdf>`
```

`../../coding/ros/books/Mastering_ROS_for_Robotics_Programming.pdf`

`Mastering_ROS_for_Robotics_Programming`

13.2.5 Images

```
.. figure:: /img/logo.*
```

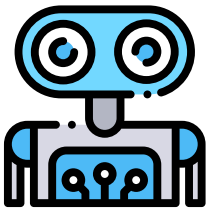


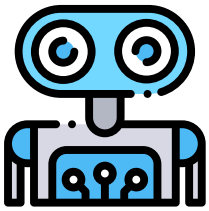
Important: Images should be either in `png` or `svg` format

Important: For `*.svg` files the file ending needs to be changed from `svg` to `*`. That way for html svg is used and pdf or pn for the latex or pdf output.

Image Placement

```
.. figure:: /img/logo.*
   :align: left
   :width: 100px

.. figure:: /img/logo.*
   :align: center
   :width: 100px

.. figure:: /img/logo.*
   :align: right
   :width: 100px

.. figure:: /img/logo.*
   :align: center
   :width: 100px
   :height: 100px
   :scale: 50 %
   :alt: this is the knowhow logo

   Caption of figure
```





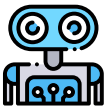


Figure1: Caption of figure

Inline Images

For inline images to work, a substitution needs to be made

```
.. |folder_icon| image:: /img/icons/folder.*

After that the image |folder_icon| can be integrated inline.
```

After that the image  can be integrated inline.

13.2.6 Lists

- item 1
  - item 1.1
  - item 1.2
- item 2
  - item 2.1
    - item 2.1.1

1. auto enumerated list item 1
2. auto enumerated list item 1
3. auto enumerated list item 1
4. auto enumerated list item 1

3. enumerated list start with item 3
4. auto enumerated list item 4
5. auto enumerated list item 5
6. auto enumerated list item 6

13.2.7 Tables

```
+------------+------------+-----------+
| Header 1   | Header 2   | Header 3  |
+============+============+===========+
| body row 1 | column 2   | column 3  |
+------------+------------+-----------+
| body row 2 | Cells may span columns.|
+------------+------------+-----------+
| body row 3 | Cells may  | - Cells   |
+------------+ span rows. | - contain |
| body row 4 |            | - blocks. |
+------------+------------+-----------+
```

Header 1	Header 2	Header 3
body row 1	column 2	column 3
body row 2	Cells may span columns.	
body row 3	Cells may span rows.	• Cells • contain • blocks.
body row 4		

```
=====  =====  ======
Inputs     Output
------------  ------
  A      B    A or B
=====  =====  ======
False  False  False
True   False  True
False  True   True
True   True   True
=====  =====  ======
```

Inputs		Output
A	B	A or B
False	False	False
True	False	True
False	True	True
True	True	True

```
.. list-table::
   :header-rows: 1
   :widths: 1 1 2

   * - Type
     - Literal
     - Description
   * - Boolean
     - true, false
     -
   * - Int
     - 3, 0x32
     - 32 bits integer
   * - Float
     - 3.14f
     - 32 bits floating point
   * - Double
     - 3.14
     - 64 bits floating point
   * - String
     - "Hello world"
     - UTF-16 string
```

Type	Literal	Description
Boolean	true, false	
Int	3, 0x32	32 bits integer
Float	3.14f	32 bits floating point
Double	3.14	64 bits floating point
String	“Hello world”	UTF-16 string

```
.. table:: Table caption

   =====  =====  ======
      Inputs     Output
   ------------  ------
     A      B    A or B
   =====  =====  ======
   False  False  False
   =====  =====  ======
```

Table1: Table caption

Inputs		Output
A	B	A or B
False	False	False

13.2.8 Code

see also: https://build-me-the-docs-please.readthedocs.io/en/latest/Using_Sphinx/ShowingCodeExamplesInSphinx.html

```
.. code-block:: python

   import antigravity

   def main():
       antigravity.fly()
   if __name__=='__main__':
       main()
```

```
import antigravity

def main():
    antigravity.fly()
if __name__=='__main__':
    main()
```

```
.. code-block:: python
   :linenos:
   :caption: Code Blocks can have captions.

   import antigravity

   def main():
       antigravity.fly()
   if __name__=='__main__':
       main()
```

Listing 1: Code Blocks can have captions.

```
import antigravity

def main():
    antigravity.fly()
if __name__=='__main__':
    main()
```

```
.. code-block:: python
   :linenos:
   :lineno-start: 10

   import antigravity

   def main():
       antigravity.fly()
   if __name__=='__main__':
       main()
```

```
10 import antigravity
11
12 def main():
13     antigravity.fly()
14 if __name__=='__main__':
15     main()
```

13.2.9 Infoboxes

```
.. note::
   This is a Note Box
```

Note: This is a Note Box

```
.. warning::
   This is a Warning Box
```

Warning: This is a Warning Box

```
.. important::
   This is a Important Box
```

Important: This is a Important Box

```
.. seealso::
   This is a See Also Box
```

See also:

This is a See Also Box

13.2.10 Special Formatting

```
.. versionadded:: 2.5
   The *spam* parameter.

.. versionchanged:: 2.5
   Feature description

.. deprecated:: 3.1
   Use :func:`spam` instead.
```

New in version 2.5: The *spam* parameter.

Changed in version 2.5: Feature description

Deprecated since version 3.1: Use `spam()` instead.

13.2.11 Math

```
Inline math :math:`a^2 + b^2 = c^2`.
```

Inline math $a^2 + b^2 = c^2$.

```
.. math::

   f(x) &= x^2\\
   g(x) &= \frac{1}{x}\\
   F(x) &= \int^a_b \frac{1}{3}x^3
```

$$f(x) = x^2$$
$$g(x) = \frac{1}{x}$$
$$F(x) = \int_b^a \frac{1}{3}x^3$$

13.2.12 Exclude

In order to exclude some parts for a certain output use the `.. only:: output` directive.

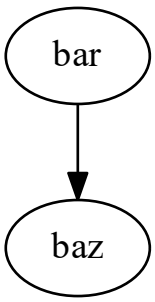
```
.. only:: html
.. only:: draft
.. only:: latex
.. only:: html or draft or latex
.. only:: html and draft
```

Important: This is needed for the all the *Wavedrom* code

13.2.13 GraphViz

Get more samples herer: https://graphviz.gitlab.io/gallery/

```
.. graphviz::

digraph foo {
   "bar" -> "baz";
}
```

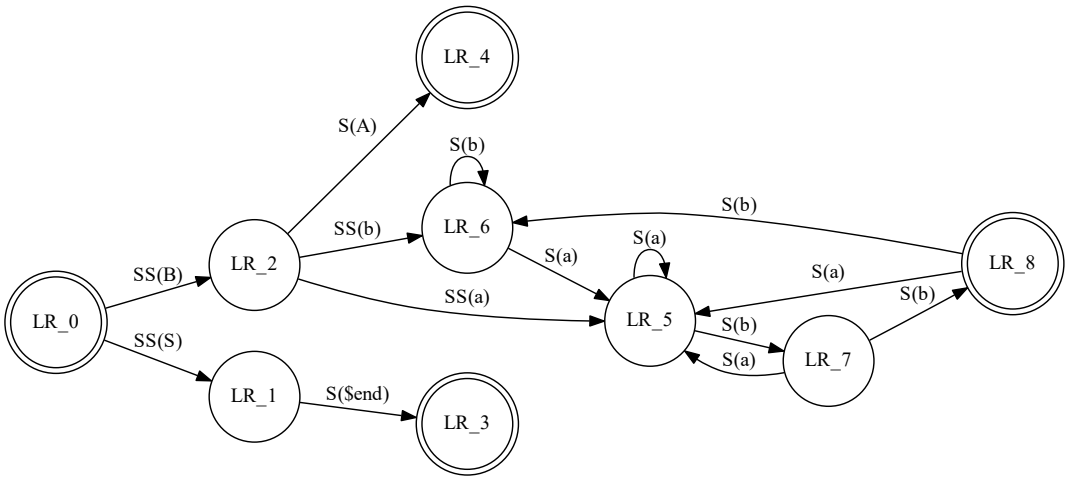



```
.. graphviz::

digraph finite_state_machine {
  rankdir=LR;
  size="8,5"
  node [shape = doublecircle]; LR_0 LR_3 LR_4 LR_8;
  node [shape = circle];
  LR_0 -> LR_2 [ label = "SS(B)" ];
  LR_0 -> LR_1 [ label = "SS(S)" ];
```

(continues on next page)

(continued from previous page)

```
    LR_1 -> LR_3 [ label = "S($end)" ];
    LR_2 -> LR_6 [ label = "SS(b)" ];
    LR_2 -> LR_5 [ label = "SS(a)" ];
    LR_2 -> LR_4 [ label = "S(A)" ];
    LR_5 -> LR_7 [ label = "S(b)" ];
    LR_5 -> LR_5 [ label = "S(a)" ];
    LR_6 -> LR_6 [ label = "S(b)" ];
    LR_6 -> LR_5 [ label = "S(a)" ];
    LR_7 -> LR_8 [ label = "S(b)" ];
    LR_7 -> LR_5 [ label = "S(a)" ];
    LR_8 -> LR_6 [ label = "S(b)" ];
    LR_8 -> LR_5 [ label = "S(a)" ];
  }
```



13.2.14 Wavedrom

For more information see:

- Wavedrom JSON Wiki
- Wavedrom Tutorial

Timing Diagrams

This documentation makes use of the `sphinxcontrib-wavedrom` plugin, So you can specify a timing diagram, or a register description with the `WaveJSON` syntax like so:

```
.. wavedrom::

  { "signal": [
    { "name": "pclk", "wave": 'p.......' },
    { "name": "Pclk", "wave": 'P.......' },
    { "name": "nclk", "wave": 'n.......' },
    { "name": "Nclk", "wave": 'N.......' },
    {},
```

(continues on next page)

(continued from previous page)

```
      { "name": 'clk0', "wave": 'phnlPHNL' },
      { "name": 'clk1', "wave": 'xhlhLHl.' },
      { "name": 'clk2', "wave": 'hpHplnLn' },
      { "name": 'clk3', "wave": 'nhNhplPl' },
      { "name": 'clk4', "wave": 'xlh.L.Hx' },
  ]}
```

and you get:

Note: if you want the Wavedrom diagram to be present in the pdf export, you need to use the "non relaxed" JSON dialect. long story short, no javascript code and use " arround key value (Eg. `"name"`).

Register

you can describe register mapping with the same syntax:

```
{"reg":[
  {"bits": 8, "name": "things"},
  {"bits": 2, "name": "stuff" },
  {"bits": 6},
 ],
 "config": { "bits":16 "lanes":1 }
 }
```

13.2.15 PlantUML

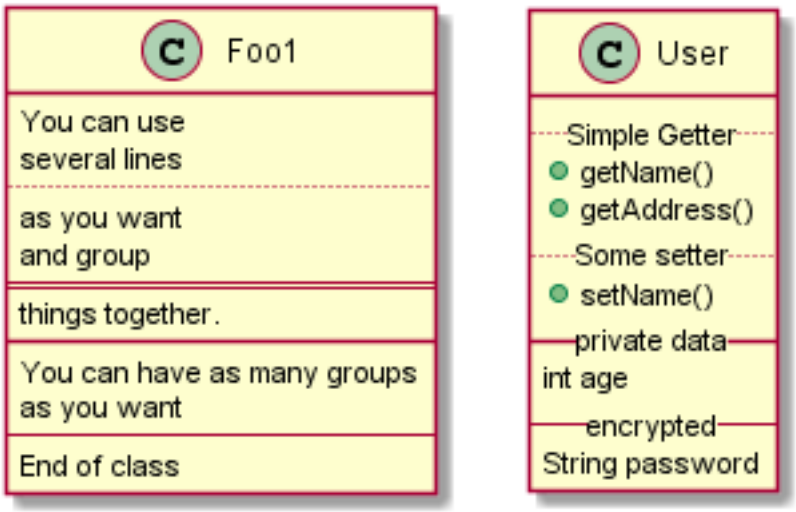
This documentation makes use of the `sphinxcontrib.plantuml` plugin, for more information see the sphinxcontrib.plantuml plugin and the PlantUML Webpage. For a small Cheatsheet for PlantUML see https://ogom.github.io/draw_uml/plantuml/

```
.. uml::

   class Foo1 {
     You can use
     several lines
     ..
     as you want
     and group
     ==
     things together.
     __
     You can have as many groups
     as you want
     --
     End of class
   }

   class User {
     .. Simple Getter ..
     + getName()
     + getAddress()
     .. Some setter ..
     + setName()
```

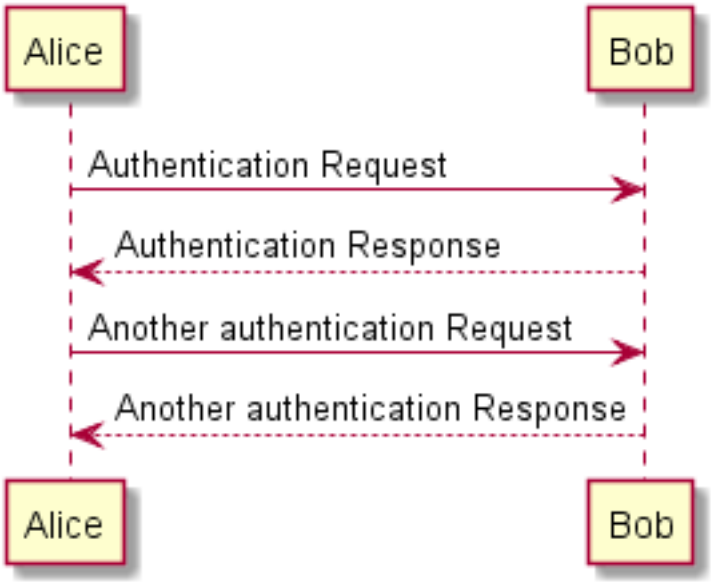
(continues on next page)

(continued from previous page)

```
    __ private data __
    int age
    -- encrypted --
    String password
  }
```




```
.. uml::

  Alice -> Bob: Authentication Request
  Bob --> Alice: Authentication Response

  Alice -> Bob: Another authentication Request
  Alice <-- Bob: Another authentication Response
```

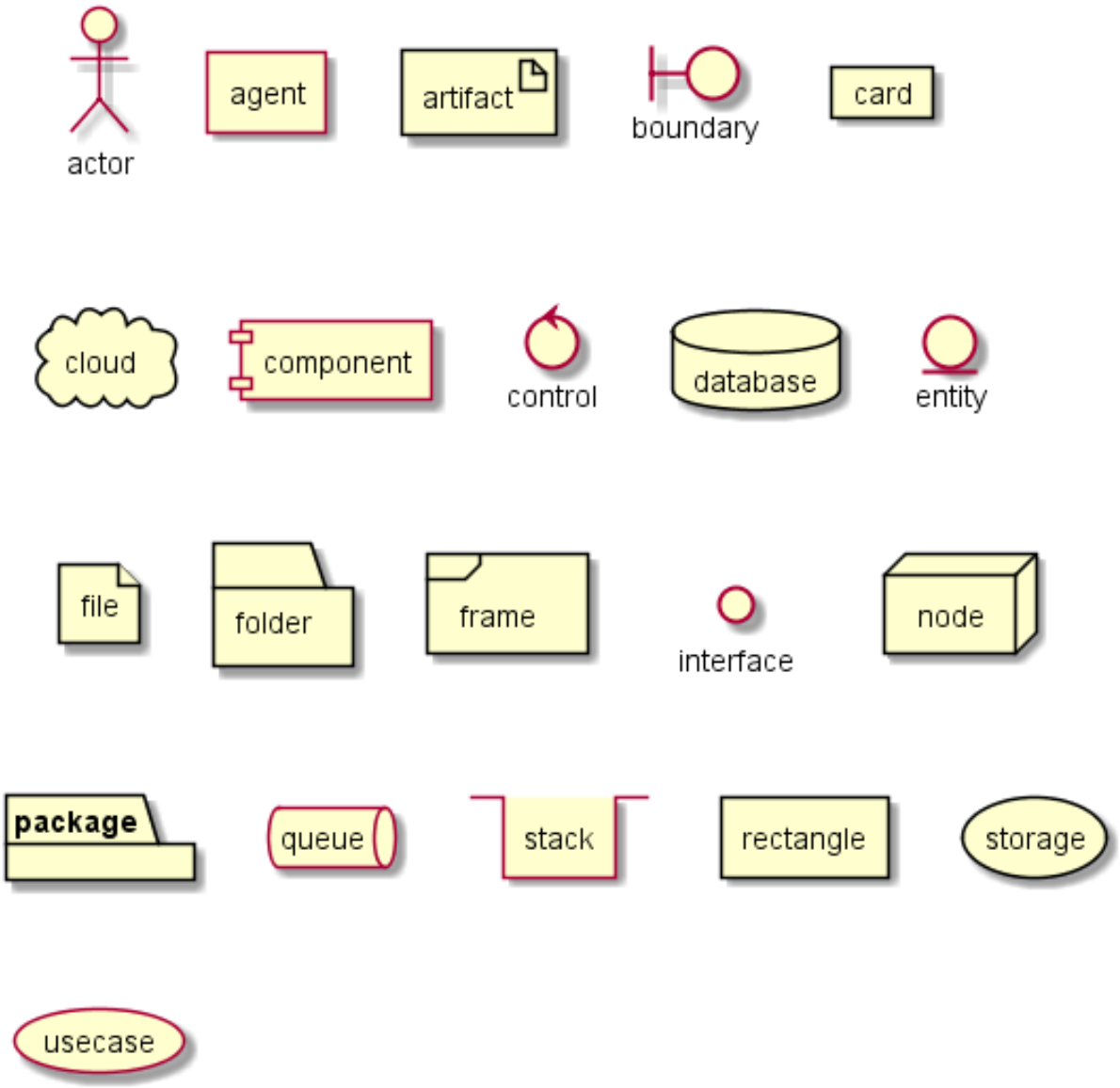



```
.. uml::

  actor actor
  agent agent
  artifact artifact
  boundary boundary
  card card
```

(continues on next page)

(continued from previous page)

```
cloud cloud
component component
control control
database database
entity entity
file file
folder folder
frame frame
interface  interface
node node
package package
queue queue
stack stack
rectangle rectangle
storage storage
usecase usecase
```



Chapter 14

Markdown

14.1 Github Markdown

GFM is a variant of markdown developed by Github.

- https://help.github.com/articles/github-flavored-markdown
- https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet This is intended as a quick reference and showcase. For more complete info, see John Gruber's original spec and the Github-flavored Markdown info page.

14.1.1 Table of Contents

Headers Emphasis Lists Links Images Code and Syntax Highlighting Tables Blockquotes Inline HTML Horizontal Rule Line Breaks YouTube Videos

14.1.2 Headers

```
# H1
## H2
### H3
#### H4
##### H5
###### H6

Alternatively, for H1 and H2, an underline-ish style:

Alt-H1
======

Alt-H2
------
```

14.2 H1

14.2.1 H2

H3

H4

H5

H6

Alternatively, for H1 and H2, an underline-ish style:

14.2.2 Emphasis

```
Emphasis, aka italics, with *asterisks* or _underscores_.

Strong emphasis, aka bold, with **asterisks** or __underscores__.

Combined emphasis with **asterisks and _underscores_**.

Strikethrough uses two tildes. ~~Scratch this.~~
```

Emphasis, aka italics, with *asterisks* or *underscores*.

Strong emphasis, aka bold, with **asterisks** or **underscores**.

Combined emphasis with **asterisks and *underscores***.

Strikethrough uses two tildes. ~~Scratch this.~~

14.2.3 Lists

```
1. First ordered list item
2. Another item
  * Unordered sub-list.
1. Actual numbers don't matter, just that it's a number
  1. Ordered sub-list
4. And another item.

   You can have properly indented paragraphs within list items. Notice the blank␣
↪line above, and the leading spaces (at least one, but we'll use three here to␣
↪also align the raw Markdown).

   To have a line break without a paragraph, you will need to use two trailing␣
↪spaces.
   Note that this line is separate, but within the same paragraph.
   (This is contrary to the typical GFM line break behaviour, where trailing␣
↪spaces are not required.)

* Unordered list can use asterisks
- Or minuses
+ Or pluses
```

1. First ordered list item
2. Another item
   - Unordered sub-list.
1. Actual numbers don't matter, just that it's a number
2. Ordered sub-list
3. And another item.

   You can have properly indented paragraphs within list items. Notice the blank line above, and the leading spaces (at least one, but we'll use three here to also align the raw Markdown).

   To have a line break without a paragraph, you will need to use two trailing spaces. Note that this line is separate, but within the same paragraph. (This is contrary to the typical GFM line break behaviour, where trailing spaces are not required.)

- Unordered list can use asterisks
- Or minuses
- Or pluses

14.2.4 Links

There are two ways to create links.

```
[I'm an inline-style link](https://www.google.com)

[I'm an inline-style link with title](https://www.google.com "Google's Homepage")

[I'm a reference-style link][Arbitrary case-insensitive reference text]

[I'm a relative reference to a repository file](../blob/master/LICENSE)

[You can use numbers for reference-style link definitions][1]
```

(continues on next page)

(continued from previous page)

```
Or leave it empty and use the [link text itself].

URLs and URLs in angle brackets will automatically get turned into links.
http://www.example.com or <http://www.example.com> and sometimes
example.com (but not on Github, for example).

Some text to show that the reference links can follow later.

[arbitrary case-insensitive reference text]: https://www.mozilla.org
[1]: http://slashdot.org
[link text itself]: http://www.reddit.com
```

I'm an inline-style link

I'm an inline-style link with title

I'm a reference-style link

```
[I'm a relative reference to a repository file](../../../README.md)
```

You can use numbers for reference-style link definitions

Or leave it empty and use the link text itself.

URLs and URLs in angle brackets will automatically get turned into links. http://www.example.com or http://www.example.com and sometimes example.com (but not on Github, for example).

Some text to show that the reference links can follow later.

14.2.5 Images

```
Here's our logo (hover to see the title text):

Inline-style:
![alt text](https://github.com/adam-p/markdown-here/raw/master/src/common/images/
↪icon48.png "Logo Title Text 1")

Reference-style:
![alt text][logo]

[logo]: https://github.com/adam-p/markdown-here/raw/master/src/common/images/
↪icon48.png "Logo Title Text 2"
```

Here's our logo (hover to see the title text):

Inline-style: ![alt text](https://github.com/adam-p/markdown-here/raw/master/src/common/images/icon48.png "Logo Title Text 1")

Reference-style: ![alt text][logo]

14.2.6 Code and Syntax Highlighting

Code blocks are part of the Markdown spec, but syntax highlighting isn't. However, many renderers – like Github's and *Markdown Here* – support syntax highlighting. Which languages are supported and how those language names should be written will vary from renderer to renderer. *Markdown Here* supports highlighting for dozens of languages (and not-really-languages, like diffs and HTTP headers); to see the complete list, and how to write the language names, see the highlight.js demo page.

```
Inline `code` has `back-ticks around` it.
```

Inline `code` has `back-ticks around` it.

Blocks of code are either fenced by lines with three back-ticks "', or are indented with four spaces. I recommend only using the fenced code blocks – they're easier and only they support syntax highlighting.

```
var s = "JavaScript syntax highlighting";
alert(s);
```

```
s = "Python syntax highlighting"
print s
```

```
No language indicated, so no syntax highlighting in Markdown Here (varies on␣
↪Github).
But let's throw in a <b>tag</b>.
```

14.2.7 Tables

Tables aren't part of the core Markdown spec, but they are part of GFM and *Markdown Here* supports them. They are an easy way of adding tables to your email – a task that would otherwise require copy-pasting from another application.

```
Colons can be used to align columns.

| Tables        | Are           | Cool  |
| ------------- |:-------------:| -----:|
| col 3 is      | right-aligned | $1600 |
| col 2 is      | centered      |   $12 |
| zebra stripes | are neat      |    $1 |

There must be at least 3 dashes separating each header cell.
The outer pipes (|) are optional, and you don't need to make the
raw Markdown line up prettily. You can also use inline Markdown.

Markdown | Less | Pretty
--- | --- | ---
*Still* | `renders` | **nicely**
1 | 2 | 3
```

Colons can be used to align columns.

| Tables | Are | Cool | | ——————- |:——————-:| —–:| | col 3 is | right-aligned | $1600 | | col 2 is | centered | $12 | | zebra stripes | are neat | $1 |

There must be at least 3 dashes separating each header cell. The outer pipes (|) are optional, and you don't need to make the raw Markdown line up prettily. You can also use inline Markdown.

Markdown | Less | Pretty — | — | — *Still* | `renders` | **nicely** 1 | 2 | 3

14.2.8 Blockquotes

```
> Blockquotes are very handy in email to emulate reply text.
> This line is part of the same quote.

Quote break.

> This is a very long line that will still be quoted properly when it wraps. Oh␣
↪boy let's keep writing to make sure this is long enough to actually wrap for␣
↪everyone. Oh, you can *put* **Markdown** into a blockquote.
```

> Blockquotes are very handy in email to emulate reply text. This line is part of the same quote.

Quote break.

> This is a very long line that will still be quoted properly when it wraps. Oh boy let's keep writing to make sure this is long enough to actually wrap for everyone. Oh, you can *put* **Markdown** into a blockquote.

14.2.9 Inline HTML

You can also use raw HTML in your Markdown, and it'll mostly work pretty well.

```
<dl>
  <dt>Definition list</dt>
  <dd>Is something people use sometimes.</dd>

  <dt>Markdown in HTML</dt>
  <dd>Does *not* work **very** well. Use HTML <em>tags</em>.</dd>
</dl>
```

14.2.10 Horizontal Rule

```
Three or more...

---

Hyphens

***

Asterisks

___

Underscores
```

Three or more...

Hyphens

Asterisks

Underscores

14.2.11 Line Breaks

My basic recommendation for learning how line breaks work is to experiment and discover – hit <Enter> once (i.e., insert one newline), then hit it twice (i.e., insert two newlines), see what happens. You'll soon learn to get what you want. "Markdown Toggle" is your friend.

Here are some things to try out:

```
Here's a line for us to start with.

This line is separated from the one above by two newlines, so it will be a␣
↪*separate paragraph*.

This line is also a separate paragraph, but...
This line is only separated by a single newline, so it's a separate line in the␣
↪*same paragraph*.
```

Here's a line for us to start with.

This line is separated from the one above by two newlines, so it will be a *separate paragraph*.

This line is also begins a separate paragraph, but... This line is only separated by a single newline, so it's a separate line in the *same paragraph*.

(Technical note: *Markdown Here* uses GFM line breaks, so there's no need to use MD's two-space line breaks.)

14.2.12 YouTube Videos

They can't be added directly but you can add an image with a link to the video like this:

```
<a href="http://www.youtube.com/watch?feature=player_embedded&v=YOUTUBE_VIDEO_ID_
↪HERE
" target="_blank"><img src="http://img.youtube.com/vi/YOUTUBE_VIDEO_ID_HERE/0.jpg"
alt="IMAGE ALT TEXT HERE" width="240" height="180" border="10" /></a>
```

Or, in pure Markdown, but losing the image sizing and border:

```
[![IMAGE ALT TEXT HERE](http://img.youtube.com/vi/YOUTUBE_VIDEO_ID_HERE/0.
↪jpg)](http://www.youtube.com/watch?v=YOUTUBE_VIDEO_ID_HERE)
```

Referencing a bug by #bugID in your git commit links it to the slip. For example #1.

Chapter 15

Multimedia

15.1 Book Review

A list of my favorite books

15.1.1 Crime

Stieg Larsson

- Millenium Trilogie ⭐⭐⭐⭐⭐
- Verblendung
- Verdammnis
- Vergebung

Jo Nesbø

- Die Fährte ⭐⭐⭐
- Headhunter ⭐⭐⭐

Peter James

- Nicht tot genug ★ 🇨🇭
- Stirb ewig ★ 🇨🇭

Mo Hayder

- Der Vogelmann 🇨🇭 ★★

Arne Dahl

- Gier 🇨🇭 ★

Adler Olsen

- Erlösung 🇨🇭 ★★

15.1.2 Science-Fiction

H.R.Wells

- The Invisible Man 🇬🇧 ★★★

Alan Dean Forster

- The Dig 🇬🇧 ★★★★

Neal Asher

- Departure 🇬🇧 ★★★

Michael Crinchton

- Prey 🇬🇧 ★★★★★
- State of fear 🇬🇧 ★★★★
- Dino Park 🇬🇧 ★★
- Timeline 🇬🇧 ★★★★
- Enthüllung 🇨🇭 ★★

- Gold - Pirate Latitues ★★★
- The Lost Word 🇬🇧 ★★★
- Next 🇬🇧 ★★
- Micro 🇨🇭 ★★★★★

Douglas Adams

- Per Anhalter durch die Galaxis
- Machs gut und Danke für den Fisch
- Restaurant am Edne des Univers

15.1.3 Biography

- Edward Snowden - Permanent Record 🇬🇧 ★★★★
- Steve Jobs - Walter Isaacson 🇬🇧 ★★★
- Leonardo Da Vinci - Walter Isaacson 🇬🇧 ★★★★
- Elon Musk - Alex Whitestone 🇬🇧 ★★

15.2 Programmer Jokes

- *Christmas and Halloween*
- *10 Kind of People*
- *Error Free Programs*
- *Boolean Answer*
- *Programmer Checks*
- *Debugging*
- *HTML Tags*
- *Teacher Punishement*
- *Accelerate a computer*

15.2.1 Christmas and Halloween

Question Why do programmers always mix up Halloween and Christmas?

Answer

- 31 Dec Christmas
- 25 Okt Halloween

15.2.2 10 Kind of People

Question

There are `10` types of people in this world. Those who understand binary and those who don't.

Answer `0b10 = 2`

15.2.3 Error Free Programs

Question There are two ways to write error-free programs; only the third one works.

Answer There is no error free program, therefore the anwer is also wrong

15.2.4 Boolean Answer

Question The best thing about a Boolean is even if you are wrong, you are only off by a bit.

Answer Boolean = `0` or `1` only of by 1bit

15.2.5 Programmer Checks

Question A good programmer is someone who always looks both ways before crossing a one-way street.

Answer Programmers can't make assumptions, they have to check everything

15.2.6 Debugging

Question Debugging: Removing the needles from the haystack.

Answer Debugging is removing bugs form a program. Bugs are hard to find like needles

15.2.7 HTML Tags

Question

Figure1: `<DIV>Q: How to you annoy a web developer? </SPAN`>`

Answer HTML Tags are wrong DIV and SPAN means the same. Above code is wrong.

15.2.8 Teacher Punishement

Question

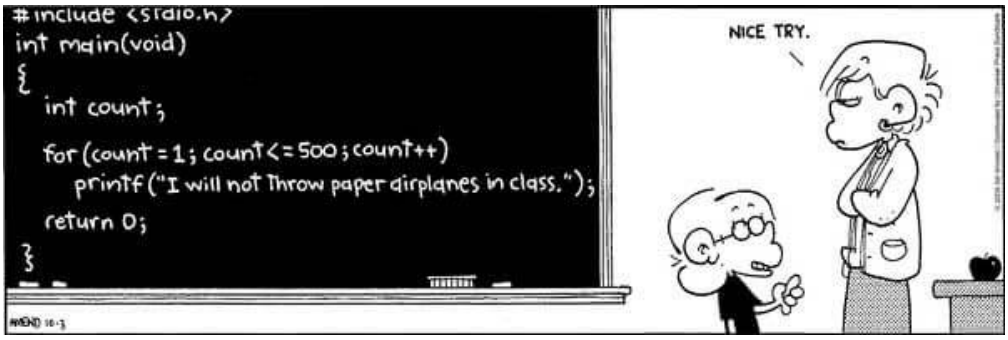


Figure2: teacher_punishement

Answer In this joke, his teacher probably gave him the punishment "Write 'I will not throw paper airplanes in class.' on the board 500 times."

```
#include <stdio.h>
int main(void)
{
  int count;
  for (count = 1; count <= 500; count++)
  printf("I will not throw paper airplanes in class.");
  return 0;
}
```

15.2.9 Accelerate a computer

Question The best method for accelerating a computer is the one that boosts it by 9.8 m/s2

Answer Let it drop. Earth gravity accelerates it by 9.8m/s2

SQL Naming **Question**

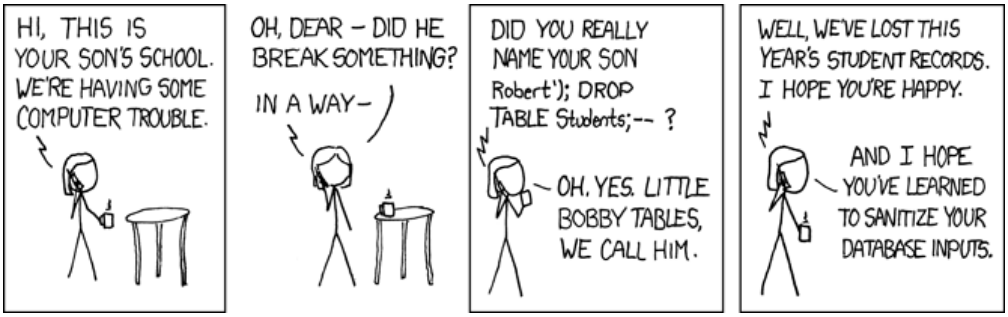


Figure3: sql_name

Answer This joke has to do with SQL, which are commands used to control databases as well as a common hack used against insecure sites, called SQL Injection.

15.3 Inspirational Quotes

Computers are useless. They can only give you answers. *Pablo Picasso*
Data is not information, information is not knowledge, knowledge is not understanding, understanding is not wisdom. *Clifford Stoll*
It's hard to fail. But it's worse never to have tried to succeed. *Theodore Roosevelt*
The good news about computers is that they do what you tell them to do. The bad news is that they do what you tell them to do. *Ted Nelson*
Treat your password like your toothbrush. Don't let anybody else use it, and get a new one every six months. *Clifford Stoll*
By three methods we may learn wisdom: First, by reflection, which is noblest; Second, by imitation, which is easiest; and third by experience, which is the bitterest. *Confucius*
A sense of humor is a major defense against minor troubles. *Mignon McLaughlin*
Humor is perhaps a sense of intellectual perspective: an awareness that some things are really important, others not; and that the two kinds are most oddly jumbled in everyday affairs. *Christopher Morley*
Be as smart as you can, but remember that it is always better to be wise than to be smart. *Alan Alda*
Common sense is not so common. *Voltaire*
Every true genius is bound to be naive. *Friedrich Schiller*
If there are no stupid questions, then what kind of questions do stupid people ask? Do they get smart just in time to ask questions? *Scott Adams*
It's not that I'm so smart, it's just that I stay with problems longer. *Albert Einstein*
The true sign of intelligence is not knowledge but imagination. *Albert Einstein*
Insanity: doing the same thing over and over again and expecting different results. *Albert Einstein*
A man should look for what is, and not for what he thinks should be. *Albert Einstein*
Everything that can be counted does not necessarily count; everything that counts cannot necessarily be counted. *Albert Einstein*
The difference between stupidity and genius is that genius has its limits. *Albert Einstein*
A question that sometimes drives me hazy: am I or are the others crazy? *Albert Einstein*
Even if there is only one possible unified theory, it is just a set of rules and equations. What is it that breathes fire into the equations and makes a universe for them to describe? *Stephen Hawking*

Continued on next page

Table 1 – continued from previous page

I have noticed even people who claim everything is predestined, and that we can do nothing to change it, look before they cross the road *Stephen Hawking*
If we do discover a complete theory, it should be in time understandable in broad principle by everyone. Then we shall all, philosophers, scientists, and just ordinary people be able to take part in the discussion of why we and the universe exist *Stephen Hawking*
Intelligence is the ability to adapt to change *Stephen Hawking*
It is no good getting furious if you get stuck. What I do is keep thinking about the problem but work on something else. Sometimes it is years before I see the way forward. In the case of information loss and black holes, it was 29 years *Stephen Hawking*
It is not clear that intelligence has any long-term survival value *Stephen Hawking*
My goal is simple. It is a complete understanding of the universe, why it is as it is and why it exists at all *Stephen Hawking*
One cannot really argue with a mathematical theorem *Stephen Hawking*
Someone told me that each equation I included in the book would halve the sales *Stephen Hawking*
The usual approach of science of constructing a mathematical model cannot answer the questions of why there should be a universe for the model to describe. Why does the universe go to all the bother of existing? *Stephen Hawking*
The whole history of science has been the gradual realization that events do not happen in an arbitrary manner, but that they reflect a certain underlying order, which may or may not be divinely inspired. *Stephen Hawking*
We are just an advanced breed of monkeys on a minor planet of a very average star. But we can understand the Universe. That makes us something very special. *Stephen Hawking*
Rasender Stillstand *Paul Virilio*
Kommunismus der Gefühle *Paul Virilio*
Sarcasm is highly inefficient against stupid people *Unknown*
There are only 10 types of people: Those that understand binary and those that don't *Unknown*
The day you stop racing is the day you win the race *Bob Marley*
Wenn ich die Menschen gefragt hДtte was sie wollen, hДtten Sie gesagt schnellere Pferde *Henry Ford*
You can't just ask customers what they want and then try to give that to them. By the time you get it built, they'll want something new *Steve Jobs*
Don't get set into one form, adapt it and build your own, and let it grow, be like water. Empty your mind, be formless, shapeless ≈ like water. Now you put water in a cup, it becomes the cup; You put water into a bottle it becomes the bottle; You put it in a teapot it becomes the teapot. Water can flow or it can crash. Be water, my friend *Bruce Lee*

Continued on next page

Table 1 – continued from previous page

Knowing is not enough, we must apply. Willing is not enough, we must do. *Bruce Lee*
Everything in moderation.... including moderation *Keniry Erin*
The glass isn't half empty, it's half full, but of poison. *Woody Allen*
I couldn't help noticing, you noticing me noticing you. *Rango*
Widerstand ist etwas für einzelne, Akzeptanz ist etwas für alle. *Unbekannt*
The optimist claims that we life in the best of all possible worlds, and the pessimists fears that this is true. *Silvan*
Remember less, know more *Silvan*
Everyday is an extention of yesterday *Silvan*
To make the long story short, we thought we had invented bread but we just made them *Guerrino De Luca, Logitech*
Tolle Sache diese Lichtgeschwindigkeit *Unbekannt*
Geocaching, using multibillion dollar technology to find Tupperware hidden in the woods *Unknown*
Lieber haben und nicht brauchen als brachen und nicht haben *Stefan*

15.4 Fonts

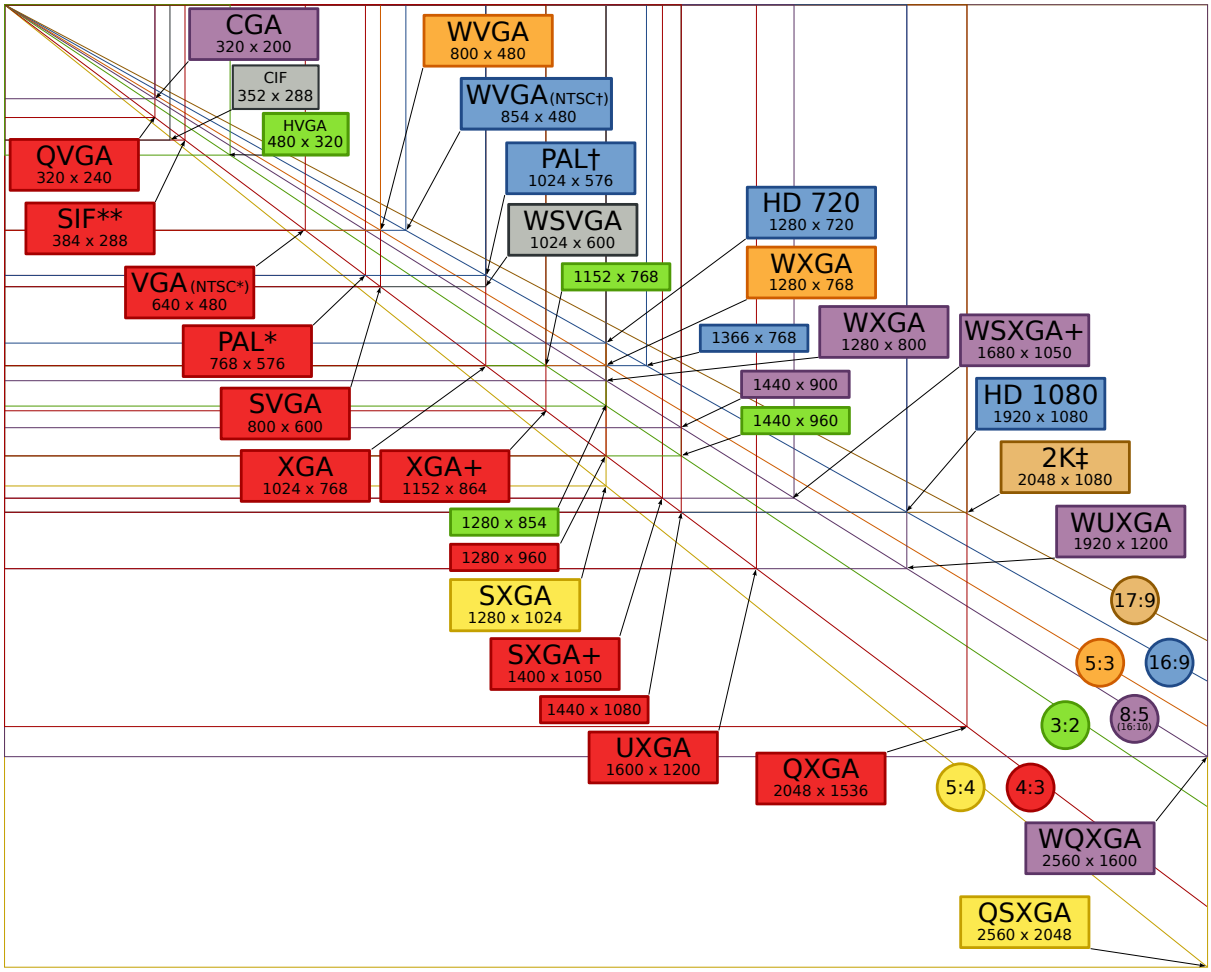
- Lato
- Source Code Pro
- Source Sans Pro
- Source Serif Pro
- Monoid
- NYT Cheltenham
- Fire Code
- Operator Mono

15.5 Icons

15.6 Resolutions

It is important to know the common used display, video and image resolutions.

Acronym	Aspect Ration	Width x Height (px)	Usage
QVGA	4:3	320 x 240	
VGA	4:3	640 x 480	
NTSC	3:2	720 x 480	Television
PAL	4:3	768 x 576	Television
SVGA	4:3	800 x 600	
WSVGA	17:10	1024 x 600	
XGA	4:3	1024 x 768	
XGA+	4:3	1152 x 864	
WXGA	16:9	1280 x 720	HD720
WXGA	5:3	1280 x 768	
WXGA	16:10	1280 x 800	
SXGA-	4:3	1280 x 960	
SXGA	5:4	1280 x 1024	
SXGA+	4:5	1400 x 1050	
HD	~16:9	1360 x 768	
HD	~16:9	1366 x 768	
WXGA+	16:10	1440 x 900	
HD+	16:9	1600 x 900	
UXGA	4:3	1600 x 1200	
WSXGA+	16:10	1680 x 1050	
FHD	16:9	1920 x 1080	HD1080
WUXGA	16:10	1920 x 1200	
2K	17:5	2048 x 1080	
QXGA	4:3	2048 x 1536	
WQHD	16:9	2560 x 1440	
WQXGA	8:5	2560 x 1600	
QFHD	16:9	3840 x 2160	
4K	17:5	4096 x 2160	
18M	3:2	5184 × 3456	Canon 600D


CGA
320 x 200
CIF
352 x 288
HVGA
480 x 320
QVGA
320 x 240
SIF**
384 x 288
VGA (NTSC*)
640 x 480
PAL*
768 x 576
SVGA
800 x 600
XGA
1024 x 768
XGA+
1152 x 864
WVGA
800 x 480
WVGA (NTSC†)
854 x 480
PAL†
1024 x 576
WSVGA
1024 x 600
1152 x 768
1280 x 854
1280 x 960
SXGA
1280 x 1024
SXGA+
1400 x 1050
1440 x 1080
UXGA
1600 x 1200
HD 720
1280 x 720
WXGA
1280 x 768
1366 x 768
WXGA
1280 x 800
1440 x 900
1440 x 960
QXGA
2048 x 1536
WSXGA+
1680 x 1050
HD 1080
1920 x 1080
2K‡
2048 x 1080
WUXGA
1920 x 1200
WQXGA
2560 x 1600
QSXGA
2560 x 2048
17:9
16:9
5:3
8:5
(16:10)
3:2
5:4
4:3

Chapter 16

Security

16.1 GnuPg

16.1.1 Encryption in Linux

To encrypt you need to do the following tasks

1. Install GnuPG
2. Creating a key pair
3. Learn to use public keys
4. Learn Encrypt & Decrypt
5. Learn Sign & Verify

16.1.2 The System

Briefly and without technical detours, the system works as follows. To encrypt and decrypt with GPG, it is necessary to use two different cryptographic keys: a public and a secret key. "Public" keys are used to encrypt and "Private" keys are used to decrypt messages. To send encrypted e.g. e-mails you must have the "public" key of the recipient, which is used to encrypt the message. The recipient then uses his "Private" key to decrypt (and read) the encrypted message. To send encrypted messages to you, senders must first have a copy of your "public" key from your keychain. "Public" keys may be passed on to those who want to send you encrypted messages. For this purpose you can deposit your "Public" Key on a Key Server. "Private" keys may not be passed on.

Note: **Key distributionAllocation of keys** "Private" keys may not be passed on to anyone, "Public" keys must be distributed to everyone.

16.1.3 Installation of GnuPG

Create of a key pair

```
# Generate of the key pair
gpg --gen-key

# View key informations
gpg --list-keys

# Send key to a keyserver
gpg --send-keys --keyserver wwwkeys.pgp.net <key-id>
```

Import of public key

```
# Get public key from keyserver
gpg --recv-keys --keyserver wwwkeys.pgp.net keyid

# import into the keychain
gpg --import
```

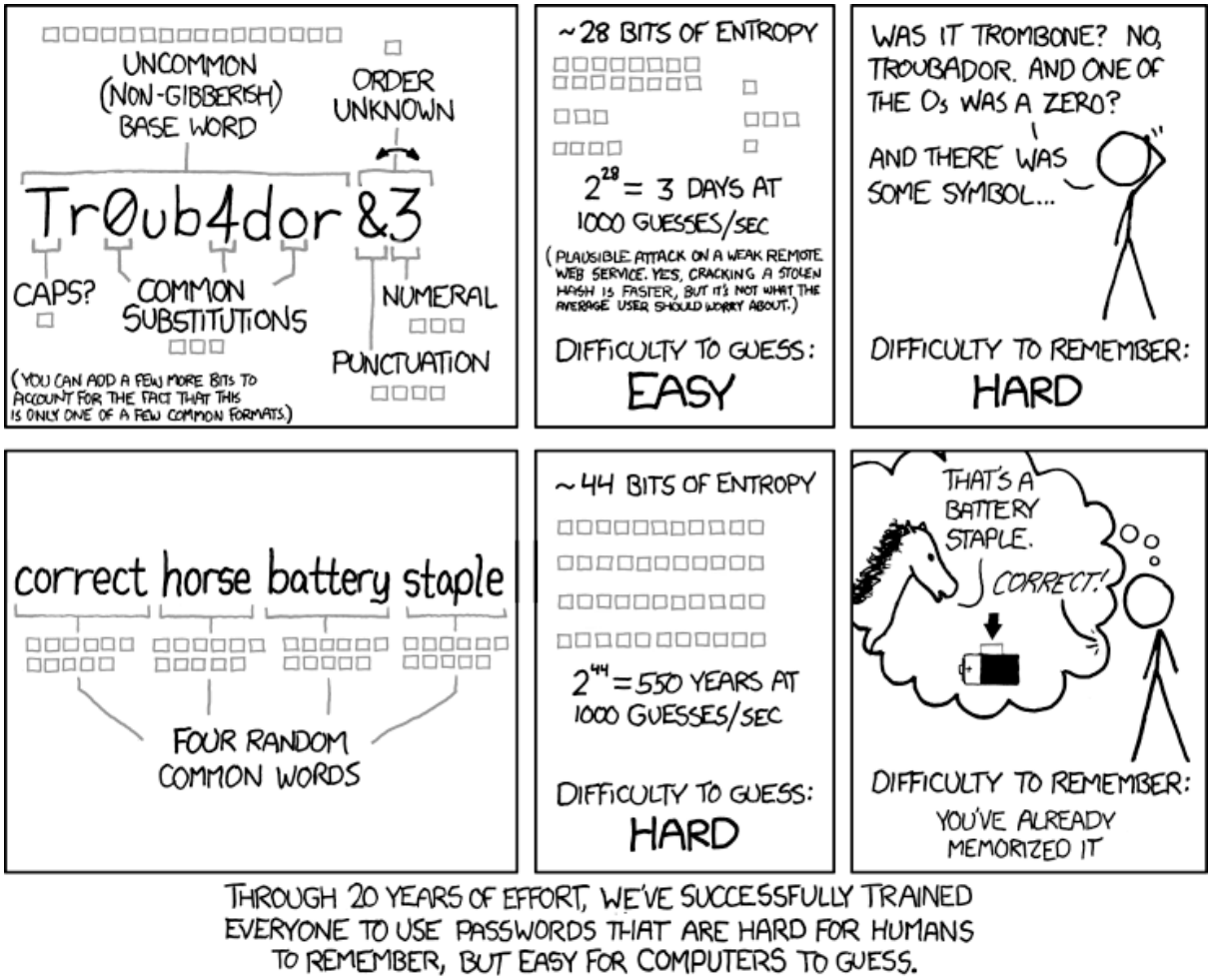
16.1.4 Encryption

```
# Encrypt1
gpg --encrypt filename or gpg -e filename

# Decrypt
gpg --decrypt filename or gpg -d filename

# Sign
gpg --sign filename or gpg -s filename

# Verify
gpg --verify filename or gpg -v filename
```

16.1.5 Links

- Official GnuPG Webpage
- Key Server
- HowTo GnuPG
- Mailvelope enryption for Gmail, Yahoo, GMX, Outlook

16.2 Password



Damit Sie ein sicheres Passwort haben müssen sie folgende Schritte erledigen

1. Ihre Passwörter kontrollieren
2. Neues sicheres Passwort erstellen und sich einprägen
3. Regelmässig Passwort wechseln

16.2.1 Einleitung

Wer kennt das nicht: man hat sich bei einem neuen Service angemeldet, oder auf dem Server steht der monatliche Passwort-Wechsel an, und es fällt und fällt einem kein Passwort ein, das den Anforderungen des Sicherheits-Konzeptes genügt: entweder, es ist zu kurz, oder es ist nicht kryptisch genug, oder es ist zu kryptisch, und man kann es sich nicht merken. Schließlich, weil man keine Lust mehr hat, sucht man sich ein beliebiges, leider meist unsicheres Passwort aus.

Mit diesem kleinen Artikel möchte ich sowohl etwas mehr Sicherheitsbewusstsein wecken, als auch die Wahl eines sicheren Passworts durch das Verständnis, wie so ein Passwort aufgebaut ist, erleichtern. Ich will jedoch gleich vorneweg sagen: die Vergabe von sicheren Passwörtern erfordert viel Disziplin vom User selbst!

16.2.2 Warum ein sicheres Passwort wichtig ist

Viele Leute denken sich NICHTS bei der Vergabe von Passwörtern. Doch dabei ist dieser Vorgang enorm wichtig.

Viel Computer-Kriminalität könnte vermieden werden, wenn mehr Endanwender ihren "inneren Schweinehund" überwinden und vernünftige Passwörter vergeben würden. Doch im gleichen Maß sind die Shop-Betreiber, Sicherheits-Beauftragten und andere, die für Passwortakzeptanz oder Passwortvorschläge zuständig sind, Schuld: ich habe bisher noch keinen Webshop mit Plausiblitätsprüfung des Passwortes gesehen. Ausserdem sollten auch bei Webshops die Generalpasswörter sicher gewählt werden um unerlaubtes Spionieren der Kundenpasswörter zu vermeiden.

Sicher werden jetzt viele sagen "Wer will schon ausgerechnet mir Böses?", oder "Wer will schon ausgerechnet meine E-Mails lesen?". Aber darum geht es ja gar nicht. Dem Angreifer ist es (meist) egal, wessen Account-Passwort er bekommt. Er interessiert sich auch nicht unbedingt für Ihre Mails. Ihn interessiert nur, wie er in das System hinein kommt. Denn hat er erstmal Zugriff, ist im Grunde der Krieg verloren. Es kann Sachen bestellen auf Ihren Namen und E-mail von Ihrem Namen aus senden.

16.2.3 Top 10: Passwörter

Liste der 10 meistbenützten Passwörter

- Platz Nr. 1: Einfache Zahlenkombinationen, wie 12345.
- Platz Nr. 2: Zahlenkombinationen, die an ein Produkt erinnern, wie 4711, 911, X5, A6.
- Platz Nr. 3: Das Wort Passwort selbst.
- Platz Nr. 4: Kosenamen wie Schatz.
- Platz Nr. 5: Das Wort Baby.
- Platz Nr. 6: Jahreszeiten wie Sommer und Winter.
- Platz Nr. 7: Das Wort Hallo.
- Platz Nr. 8: Namen von Großstädten, wie Zürich, Paris oder NewYork.
- Platz Nr. 9: Der eigenen Vornamen.
- Platz Nr. 10: Der Vorname der Frau/FreundinFreund/Mann

16.2.4 Aufbau eines (relativ) sicheren Passworts

Ein sicheres Passwort besteht sinnvollerweise aus Groß- und Kleinbuchstaben sowie aus Ziffern. Es enthält keine (wahrnehmbare) Systematik und ist wenigstens 8 Zeichen lang. Es sollte kein Wort einer bekannten Sprache sein (z. B. Englisch, Deutsch oder Französisch).

Sicherheits-Freaks neigen dazu, sogenannte "Tastatur-Hacks" zum Erzeugen von Passwörtern zu verwenden. Dabei handelt es sich um ein einmaliges, sinnloses und blindes Zehnfinger-Einhacken auf die Tastatur - man merkt sich nicht das Passwort allgemein, sondern nur die Zeichenfolge wie es auf der Tastatur eingegeben wird. Solche Passwörter sind natürlich extrem "sicher". Denn wer z.B. in Anwesenheit anderer Personen ein Passwort eingeben muss, sollte das unauffällig und schnell tun können. Wer in einer solchen Situation auf das "Adlersystem" bei der Eingabe angewiesen ist, erleichtert den Anwesenden nur das unauffällige Mitverfolgen der eingegebenen Zeichenfolge.

Note: Gute Passwörter sollten also einen Mittelweg zwischen nicht erratbaren Zeichenfolgen und merkbaren Zeichenfolgen darstellen. Ein Tastaturhack ist eine einfaches und gutes Password.

Password Generator

Wer zu unkreativ ist sich ein eigenes "sicheres" Passwort auszudenken, kann mit folgender Seite einfach zu merkenden aber denoch sicherePasswörter von verschiedenen Längen, automatisch generieren lassen.

- Keepass
- Safepasswd
- Generista

Password Reuse

Pro Plattform sollte immer ein kompletes einzigartiges Password erstellen. Abwandlungen können relative schnell entdeckt und ausgebeutet werden.

Wie lang sollte ein Passwort sein

Die Frage ist so leicht nicht zu beantworten. Das hängt vom Sicherheitsbereich ab. Generell kann man sagen, eine Mindestlänge von 8 Zeichen ist sinnvoll: 8 Zeichen bedeuten 191707312997281 Kombinationen bei der Zeichenklasse a-zA-Z1-9 (= 61 Zeichen). Das würde bei einer Million Tastenanschläge pro Sekunde eine Maximalzeit von ca. 53252 Stunden (191707312,997281 Sekunden) bedeuten (fast 6 Jahre). Das ist schon mal eine ganz ordentliche Zeit :-)

Zur Einschätzung mal eine kleine Tabelle: Mindestlänge maximal benötigte Zeit (bei angenommener 1 Million Tastaturanschlägen pro Sekunde)

Anzahl Zeichen	Brute Force Zeit
3 Zeichen	0,2 Sekunden
5 Zeichen	14 Minuten
8 Zeichen	53252 Stunden
10 Zeichen	1179469 Wochen
12 Zeichen	84168853 Jahre
15 Zeichen	19104730610573 Jahre

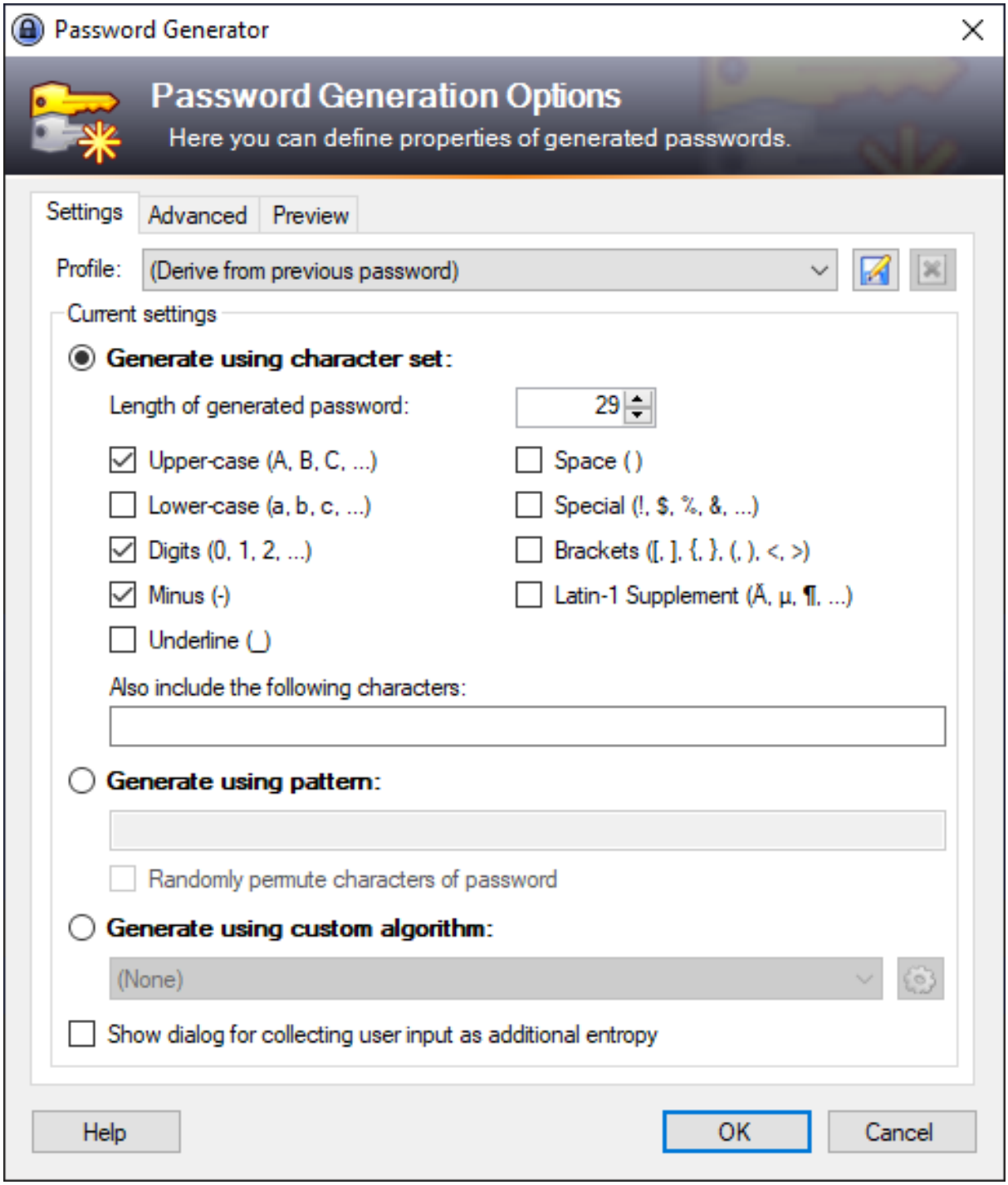
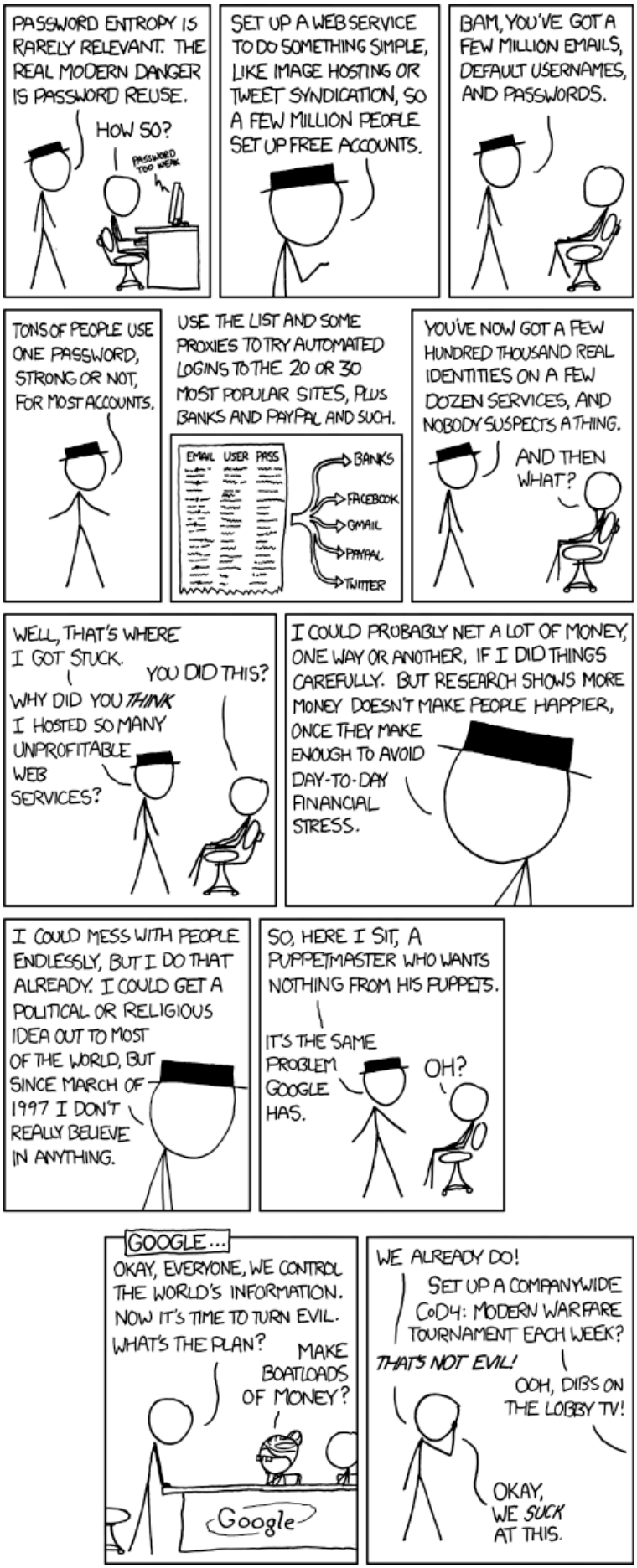


Figure1: Keepass Password Generator


PASSWORD ENTROPY IS RARELY RELEVANT. THE REAL MODERN DANGER IS PASSWORD REUSE.
HOW SO?
PASSWORD TOO WEAK
SET UP A WEB SERVICE TO DO SOMETHING SIMPLE, LIKE IMAGE HOSTING OR TWEET SYNDICATION, SO A FEW MILLION PEOPLE SET UP FREE ACCOUNTS.
BAM, YOU'VE GOT A FEW MILLION EMAILS, DEFAULT USERNAMES, AND PASSWORDS.
TONS OF PEOPLE USE ONE PASSWORD, STRONG OR NOT, FOR MOST ACCOUNTS.
USE THE LIST AND SOME PROXIES TO TRY AUTOMATED LOGINS TO THE 20 OR 30 MOST POPULAR SITES, PLUS BANKS AND PAYPAL AND SUCH.
EMAIL USER PASS
BANKS
FACEBOOK
GMAIL
PAYPAL
TWITTER
YOU'VE NOW GOT A FEW HUNDRED THOUSAND REAL IDENTITIES ON A FEW DOZEN SERVICES, AND NOBODY SUSPECTS A THING.
AND THEN WHAT?
WELL, THAT'S WHERE I GOT STUCK.
YOU DID THIS?
WHY DID YOU THINK I HOSTED SO MANY UNPROFITABLE WEB SERVICES?
I COULD PROBABLY NET A LOT OF MONEY, ONE WAY OR ANOTHER, IF I DID THINGS CAREFULLY. BUT RESEARCH SHOWS MORE MONEY DOESN'T MAKE PEOPLE HAPPIER, ONCE THEY MAKE ENOUGH TO AVOID DAY-TO-DAY FINANCIAL STRESS.
I COULD MESS WITH PEOPLE ENDLESSLY, BUT I DO THAT ALREADY. I COULD GET A POLITICAL OR RELIGIOUS IDEA OUT TO MOST OF THE WORLD, BUT SINCE MARCH OF 1997 I DON'T REALLY BELIEVE IN ANYTHING.
SO, HERE I SIT, A PUPPETMASTER WHO WANTS NOTHING FROM HIS PUPPETS.
IT'S THE SAME PROBLEM GOOGLE HAS.
OH?
GOOGLE...
OKAY, EVERYONE, WE CONTROL THE WORLD'S INFORMATION. NOW IT'S TIME TO TURN EVIL.
WHAT'S THE PLAN?
MAKE BOATLOADS OF MONEY?
Google
WE ALREADY DO!
SET UP A COMPANYWIDE CoD4: MODERN WARFARE TOURNAMENT EACH WEEK?
THAT'S NOT EVIL!
OOH, DIBS ON THE LOBBY TV!
OKAY, WE SUCK AT THIS.

Doch nun kommt die Ernüchterung. Alle diese Angaben sind sogenannte Maximalzeiten! Maximalzeit bedeutet: wenn jemand in der angegebenen Geschwindigkeit versucht, das Passwort zu knacken, und erst die allerletzte eingegebene Zeichenkombination die richtige ist, dann dauert es so lange wie angegeben. Aber theoretisch könnte ja auch schon die allererste eingegebene Zeichenkombination richtig sein. Dann hat es nur eine hunderttausendstel Sekunde gedauert, um das Passwort zu knacken - trotz 15 Zeichen. Es kann also durchaus sein, dass ein Angreifer ein Passwort innerhalb weniger Sekunden herausgefunden hat. Zufall eben. Deshalb sollte man sich bei 8 Zeichen durchaus nicht in Sicherheit wägen. Außerdem kommt es auch auf die Rechenleistung an: hier wurde mit einer Millionen Tanstenanschlägen pro Sekunde gerechnet. Andere, bessere, später gebaute Rechner schaffen Millionenfache. Dies drängt auch immer mehr in den vordergrund mit den Grafikkarten Brute-Force Attacken. Als Beispiel eine neue Nividia Grafikkarte mit CUDA verfügt über 256 Prozessor kerne, die Taktfrequenz eins solchen Kerns kann ungefähr 500MHz betragen. Damit kann die Grafikkarte pro Sekunde 128E9 = 128000000000 Passwörter testen.

Natürlich sollte man auch noch hinzufügen, dass viele Zugangssysteme einen einloggenden Gast nach soundsoviel Fehlversuchen aus dem System werfen. Dann muss sich dieser, wenn er es wieder versuchen will, mit einer neuen Identität, im Internet z.B. manchmal auch mit einer anderen IP-Adresse anmelden. Solche Dinge kann ein Angreifer allerdings bis zu einem gewissen Grad automatisieren.

Links

- Wikipedia Artikel zum Thema Passwort
- Hilfreicher Artikel

16.3 Tor

What is Tor?

Tor (The Onion Router) is free software and an open network that helps you defend against a form of network surveillance that threatens personal freedom and privacy, confidential business activities and relationships, and state security known as traffic analysis.

- Official Tor Webpage
- Check if you're using Tor

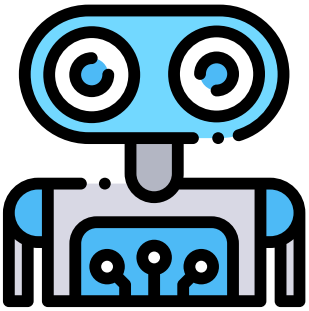
16.3.1 Download

Download the latest **Tor Browser Bundle** from the Tor Webpage

- Tor Bundle Download

Chapter 17

About



17.1 About

17.1.1 Authors

- tschinz - Github Profile

17.1.2 Find me at

- Github
- Flickr
- Twitter @tschinz

17.2 Credits

On this website information, images and documents are used. Hereafter these credits are all listed.

Icons made by Freepik from Flaticon

17.3 How to use Sphinx Documentation

17.3.1 Sphinx Requirements

- make
  - Windows - GnuWin32
  - Linux

```
sudo apt-get install build-essential
```

- Python 3
  - Python
  - Anaconda
- Python Modules (can be installed with pipenv)

```
pip install sphinx
pip install sphinx-rtd-theme
pip install sphinxcontrib-wavedrom
pip install sphinxcontrib-plantuml
pip install recommonmark
```

- Latex Tools (only for latex build)
  - Windows
    - MikTex
    - TexStudio
  - Linux

```
sudo apt install texlive-fonts-recommended texlive-latex-recommended␣
↪texlive-latex-extra
```

- Inkscape (for `.svg` to `.pdf` and to `.png` conversion)
  - Windows - Inkscape
  - Linux

```
sudo apt-get install inkscape
```

17.3.2 How to create a new Sphinxdoc

```
sphinx-quickstart
```

17.3.3 How to Build Sphinxdoc locally

Without pipenv

- Install requirements see: *Sphinx Requirements*
- `cd` to the git folder
- Generate the desired output

```
make          # list all the available output format
make help     # list all the available output format

make html     # for html
make latex    # for latex
make latexpdf # for latex (will require latexpdf installed)

make clean    # cleans all generated file, TODO before commiting
make clean-images # cleans all autogerated png and pdf files
```

With pipenv

- Install requirements *Sphinx Requirements*
- Create a virtual environment with pipenv (will use the Pipfile for installing the necessary packages)

```
pipenv install
```

- then you can build the documentation

```
pipenv run make html
```

- if you want run `make` multiple times, prepone `pipenv run` on each command can be annoying, you can spawn a subshell with

```
pipenv shell
```

- and then you can use `make` the usual way

```
make          # list all the available output format
make help     # list all the available output format

make html     # for html
make latex    # for latex
make latexpdf # for latex (will require latexpdf installed)

make clean    # cleans all generated file, TODO before commiting
make clean-images # cleans all autogerated png and pdf files
```

all the outputs will be in `_build` folder

- html: `_build/html`
- pdf & tex: `_build/latex`

Continuous Build

During developement or creation of a page, the script `build-loop.bash` will rebuild the webpage every X seconds. In this way a constant preview of the page can be shown.

17.3.4 Commit to Repository

Before performing a commit the following steps are required:

- Verify the `html` documentation local *How to Build Sphinxdoc locally*

```
make html
```

- Solve all build `Warnings` and `Errors` display during build in the commandline
- Generate `pdf`

```
make latexpdf
```

- Clean the repo from generated files

```
make clean
```

- Commit and push the changes *SPL Knowhow CI*

17.3.5 Continuous Integration(CI)

The `.travis.yml` will run on each `master` commit and create a `_build/` folder which will be pushed onto the branch gh-pages and consequently be used by github to displayed static html pages.

17.4 HACK this documentation

- *New Documentation Section*
- *Example Section*
  - *Section Images*
  - *Write the contents*

If you want to add your page to this documentation you need to add your source file in the appropriate section. Every main section has its own folder structure and its own `img/` folder containing all images for this section.

This documentation uses a recursive index tree: every folder have a special `index.rst` file that tell sphinx witch file, and in what order put it in the documentation tree.

If you don't have enough knowledge about ReStructuredText then you can also use the pandoc translator or use the internal *Cheatsheet*

17.4.1 New Documentation Section

If you want to add a new section, you need to specify in the `main index.rst`, the `section/index.rst` file of the new section.

```
.. toctree::
   :hidden:
   :glob:
   :maxdepth: 2
   :titlesonly:
   :caption: Content

   linux/index
   mac/index
   windows/index
   tools/index
   coding/index
   writing/index
   multimedia/index
   security/index
   about/index
```

The section name should be the same as the folder name, but for Sphinx this is not required. Sphinx will take the name of the section from the title of the `section/index.rst` file.

17.4.2 Example Section

I want to document the new topic in SPL Knowhow repo, and want to create a section for it; let's call it `Section`

So I need to create a folder named `section/` (name is not important), and in it create a `section/index.rst` file like:

```
=============
Section Title
=============

.. figure:: img/logo.*
   :align: right
   :width: 150px

.. contents:: :local:

.. toctree::
   :glob:
   :maxdepth: 2
   :titlesonly:
   :caption: Content

   *
```

Note: The `.. toctree::` directive accept some parameters, in this case `:glob:` makes so you can use the * to include all the remaining files.

Note: The file path is relative to the index file, if you want to specify the absolute path, you need to prepend `/`

Now I can add additional ReST files like `section/intro.rst` and other files like `section/section_part_1.rst`, `ssection/ection_part_2.rst`, etc.

Section Images

Add an image folder in the section folder `section/img`, in case of additional documents ass a `section/docs` folder too.

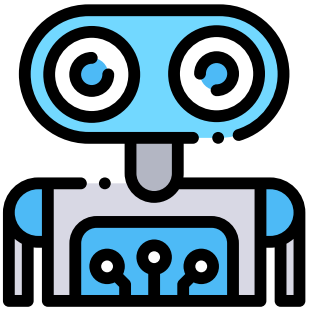
Write the contents

That's it, now you can add all you want in the new section `section` and all pages will show up in the documentation automatically.

17.5 License

Chapter 18

Welcome to Zawiki Knowledge Database



This Repo is a collection of markdown and ReStructuredText pages. Here you can find various informations about topics I've always forget. This pages let me help to remember less but know more.

18.1 Site purpose and structure

18.1.1 Getting started

Want to try it for yourself? Then jump to the *getting started* page and have fun, but first you need to learn *ReStructured Text* !!!

You can view the content as a:

- Webpage
- PDF
- Repo

18.1.2 Known Issues / TODOs

- Github CI not working for PDF creation
- Missing pages from original Zawiki
- missing links to config repo