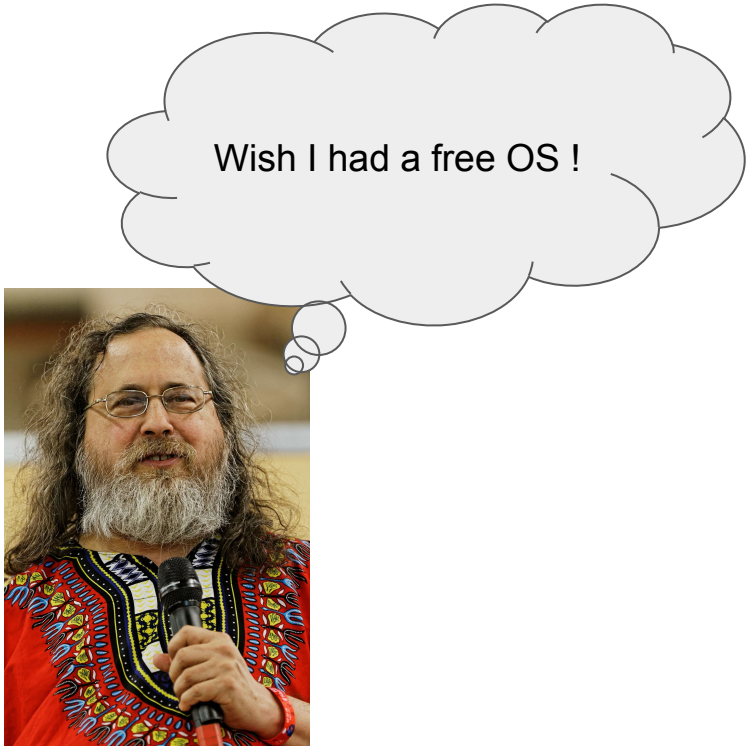
Linux crash course

Ing. Stanislav Smatana

In the beginning there was Richard Stallman



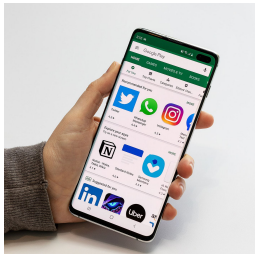
- In the 80s everything was proprietary
- R. Stallman founded free software foundation
- At the core of the project was unix-like operating system called GNU (GNU is not Unix)

But one piece was missing

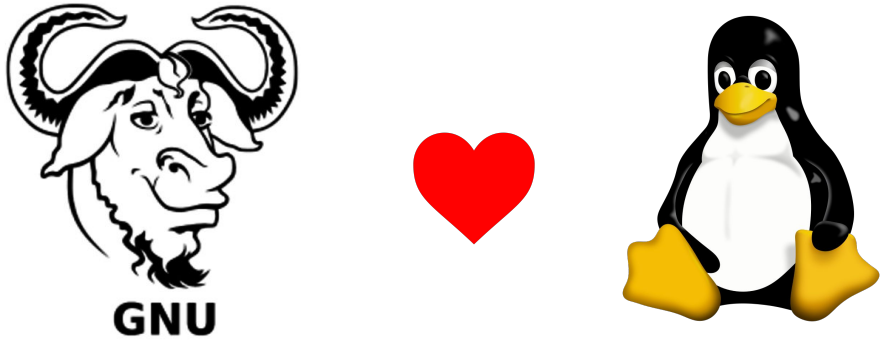
Engine ! (kernel)

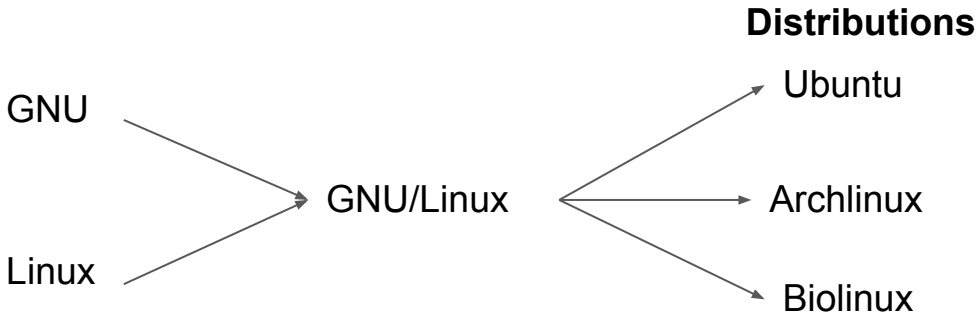
Meet Linus Torvalds

- Started his Unix-like kernel as a small hobby project 1991
- This kernel became known as Linux (guess why)
- Now he is the head of the Linux Foundation and BDFL of Linux
- His little project blew out of proportions ...

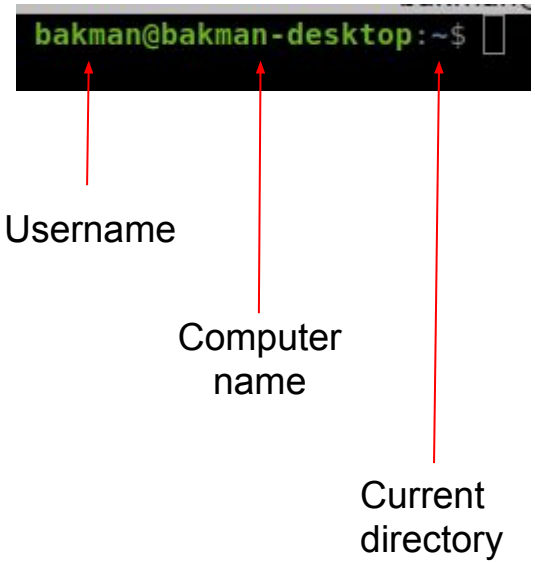


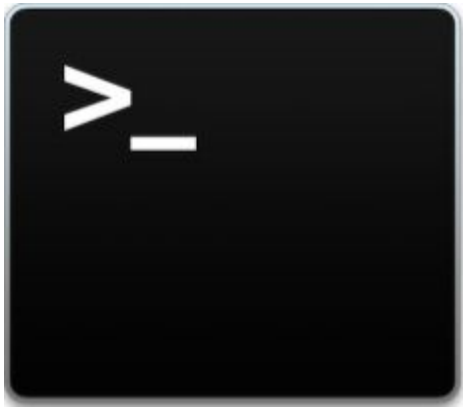
And so GNU/Linux was born





Let’s jump into the terminal





Like in a file browser, terminal is always opened in some directory, which is called the **current working directory**.

Let’s try a bunch of commands

Type in command and hit enter !

Commands

- **ls** - list current directory
- **pwd** - print current directory
- **mkdir** <name> - create directory “name”
- **cd** <name> - set current directory to “name”
- **echo** “text” - prints “text”
- **cat** <name> - print contents of file “name”

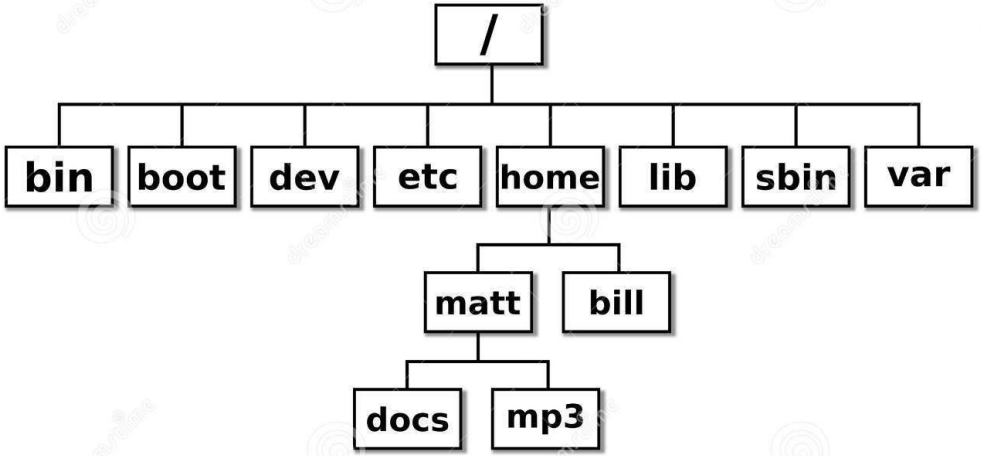
Redirection

- command **>** file - stores output of command in file

Keys

- ↑↓ - command history
- **tab** - command completion
- **Ctr + c** - kills current command (does not copy !)
- **Ctrl + r** - search in history

The file system



Absolute path and relative path

Absolute path

/home/matt/mp3/supersong.mp3

slash

succession of directories

Relative path (to the CWD !!)

CWD = /home/matt

mp3/supersong.mp3

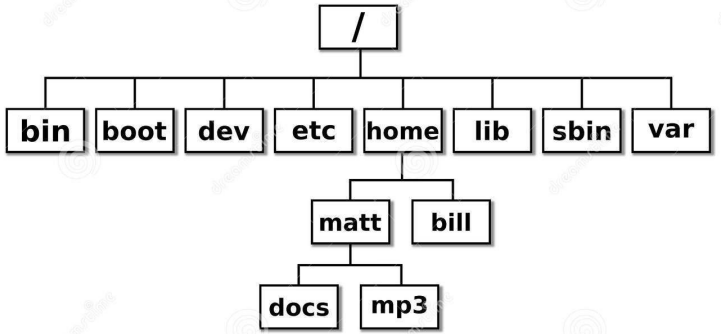
No slash !!

Relative path special strings

CWD = /home/bill

../matt/mp3/supersong.mp3

. - current directory
.. - parent directory



File-related commands

- **pwd** - print current working directory
- **ls** - list contents of current directory
- **cd** <path> - change current directory to path
- **mkdir** <path> - create directory
- **rm** <path> - remove file
- **rmdir** <path> - remove directory (only if it is empty)
- **cp** <path1> <path2> - copy file in path1 into path2
- **mv** <path1> <path2> - move file in path1 into path2
- **cat** <path> - print contents of file in path
- **head** <path> - print n first lines of a file in a path
- **tail** <path> - print last n lines of a file in a path

wget https://raw.githubusercontent.com/dwyl/english-words/master/words.txt

Complex commands

$ **command** [options] [arguments]

Options

- Modify behaviour of a command
- Either short (-l) or (--long)
- Can be given in any order
- E.g. ls -l

Arguments

- Typically paths
- Their order is important !
- E.g. cat my_file.txt

Specific for every command ! Help can be usually found by running command **-h**, command **--help** or **man** command.

The mighty pipe - counting english words with ing

command1 | command2

1. Download list of words

   wget https://raw.githubusercontent.com/dwyl/english-words/master/words.txt

2. Inspect

   head words.txt

3. Count all words

   cat words.txt | **wc** -l

4. Count only words containing “ing”

   cat words.txt | **grep** “ing” | **wc** -l

Automation - creating a script

```
#! /bin/bash

#This is comment and it is ignored
wget https://raw.githubusercontent.com/dwyl/english-words/master/words.txt

cat words.txt | wc -l
cat words.txt | grep "ing" | wc -l
```

Save as script.sh and run using command **bash script.sh**

Improving our little script - variables

Variables allow you to **save values** under a given name.

```
NAME=”Peter”

echo “$NAME”

echo “Hello $NAME”
```

Outputs of commands can be saved as variables

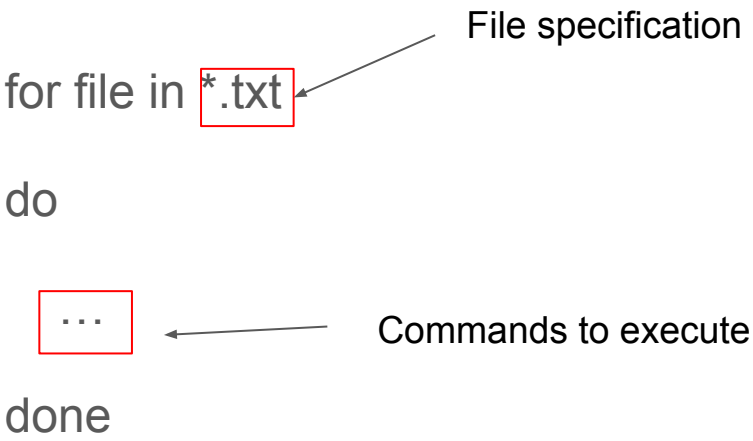
```
NAME=$(whoami)

echo “$NAME”

echo “Hello $NAME”
```

Improved script

```
#! /bin/bash

wget https://raw.githubusercontent.com/dwyl/english-words/master/words.txt 2> /dev/null

NUM_WORDS=$(cat words.txt | wc -l)
NUM_WORDS_WITH_ING=$(cat words.txt | grep "ing" | wc -l )

echo "There are $NUM_WORDS english words and out of them, $NUM_WORDS_WITH_ING end with ing !"
```

Looping

For loop allows you to perform a set of operations on all specified files.



Counting reads in sequence files

1. Download tar.gz archive with sequence files from the link on the bottom of this slide (do this through browser, not through terminal)
2. Run “tar xvfz sequences.tar.gz” in terminal to unpack files
3. You should see files a.fasta and b.fasta

https://filesender.cesnet.cz/?s=download&token=dfbab33a-05a8-c4b6-70f1-05721c0576e4

Counting sequences in files - script

```
#! /bin/bash

for f in *.fasta
do
    N_SEQUENCES=$(cat $f | grep ">" | wc -l)
    echo "$f $N_SEQUENCES"
done
```

Other useful utilities

- top / htop - show currently running processes
- mc - file manager
- nano - text editor
- ssh user@server - connect terminal to remote server
- less - make long output scrollable
- sort
- sed
- awk
- tar
- gunzip
- chmod
- chown

IF control structure

```
If condition
then
   …
fi
```

```
If [ $file -d ]
then
        echo “It’s a directory !”
fi
```

Enables program to decide, what path of computation will be taken based on previous computational result.

Other types of loops

- **while**

```
i=0
while [ $i -lt 4 ]
do
        echo $i
        i=$[$i+1]
```

- **until**

```
until [ $i -eq 4 ]
do
        echo $i
        i=$[$i+1]
done
```

Some advice

- Write steps of your bioinformatic analyses into scripts, otherwise you will forget what have you done with your data.
- Comment scripts. You will be surprised how quickly you forget what your code means !
- Name your files consistently.

Other resources

- https://stackoverflow.com/questions/68372/what-is-your-single-most-favorite-command-line-trick-using-bash
- http://www.proccli.com/2012/01/useful-bash-tricks
- https://www.thegeekstuff.com/2010/08/bash-shell-builtin-commands/