

# Practical Bioinformatics

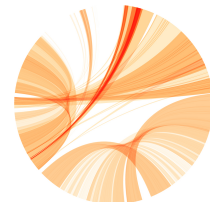
## Basic Linux

Stefan Wyder

stefan.wyder@uzh.ch



**Universität  
Zürich**<sup>UZH</sup>



**URPP  
Evolution  
in Action**

# Why Linux?

## **In Bioinformatics**

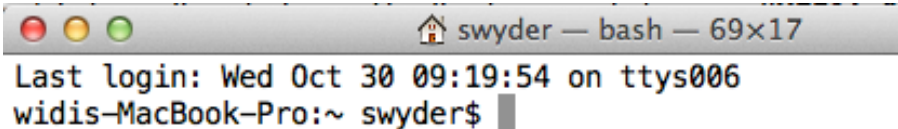
- Many bioinformatics and genomics tools are command-line only and are optimized to run on Linux.
- Very powerful with text files, even large (GBs)

## **In general**

- Experienced Linux users can do more with less effort and much faster
- Multi-user
- Lots of control and customization possibilities
- 0\$
- ....

# Why command line?

- to run bioinformatics software
- you can automate tasks more easily
- to interact with high-performance (high-memory) servers
- offers handy tools to work with text files

A screenshot of a macOS terminal window. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left, a home icon followed by the text 'swyder — bash — 69x17' in the center, and a close button on the right. The main content area of the terminal displays the text 'Last login: Wed Oct 30 09:19:54 on ttys006' on the first line, and 'widis-MacBook-Pro:~ swyder\$' on the second line, with a small black cursor block at the end of the second line.

```
swyder — bash — 69x17
Last login: Wed Oct 30 09:19:54 on ttys006
widis-MacBook-Pro:~ swyder$
```

- The shell is an interactive interpreter: it reads commands, finds the corresponding programs, runs them, and displays output
- Today we use the **Bash shell** (default shell of most Linux distributions and Mac OS X)

# The prompt

Prompt

```
eiger-uzh:~ swyder$
```

Hostname

User

Normal user

```
eiger-uzh:~ swyder$
```

Current path  
(abbreviated)



# Commands

- The tools philosophy was to have small programs to accomplish a particular task instead of trying to develop large monolithic programs to do a large number of tasks.
- "Designed to operate together"  
To accomplish more complex tasks, tools can simply be connected together.
- The shell comprises hundreds of commands, but if you know 25 you can achieve many things
- Commands are abbreviations to type less (ls:list, cp:copy, mv:move)
- Common structure:  
**Command** -**Option(s)** **Parameter(s)**  
ls -l /home/swyder/tmp

# Setting options

Command -Option(s) Parameter(s)

ls -l

ls -l -r -h

ls -l -r -h

ls -rlh

ls --reverse --human-readable      long options !only GNU tools!

The order of options does not matter  
unless they override each other (e.g. sorting)

# Options, grep as an example

more fruitlist.txt

```
apple
banana
pineapple
pear
peach
```

- `$ grep apple fruitlist.txt`  
apple  
pineapple
- `$ grep -w apple fruitlist.txt` (or `grep -x`)  
apple
- `$ grep -v apple fruitlist.txt`  
banana  
pear  
peach
- `$ grep apple *.txt`  
fruitlist.txt:apple  
fruitlist.txt:pineapple  
recipeFruitSalad.txt:1 pineapple  
recipePinaColada.txt:2oz fresh pineapple juice
- `$ grep apple fruitlist.txt recipeFruitSalad.txt`

Also options to color, to show context, search with compl patterns

# Getting help

**man** <command>

man cp

CP(1) BSD General Commands Manual CP(1)

NAME

cp -- copy files

SYNOPSIS

cp [-R [-H | -L | -P]] [-fi | -n] [-apvX] source\_file target\_file  
cp [-R [-H | -L | -P]] [-fi | -n] [-apvX] source\_file ... target\_directory

DESCRIPTION

In the first synopsis form, the cp utility copies the contents of the source\_file to the target\_file. If more than one source\_file is specified, the contents of each named source\_file is copied to the destination target\_directory. The name of the destination target\_directory must be a directory. If cp detects an attempt to copy a file to itself, the copy will fail.

The following options are available:

- a Same as -pPR options. Preserves structure and attributes of files but not directory structure.
- f If the destination file cannot be opened, remove it and create a new file, without preserving permissions. (The -f option overrides any previous -n option.)

...

space: scroll down a page  
b: scroll up  
q: quit man

<command> **--help**

cp --help

a less verbose help  
(not working on Mac OS)

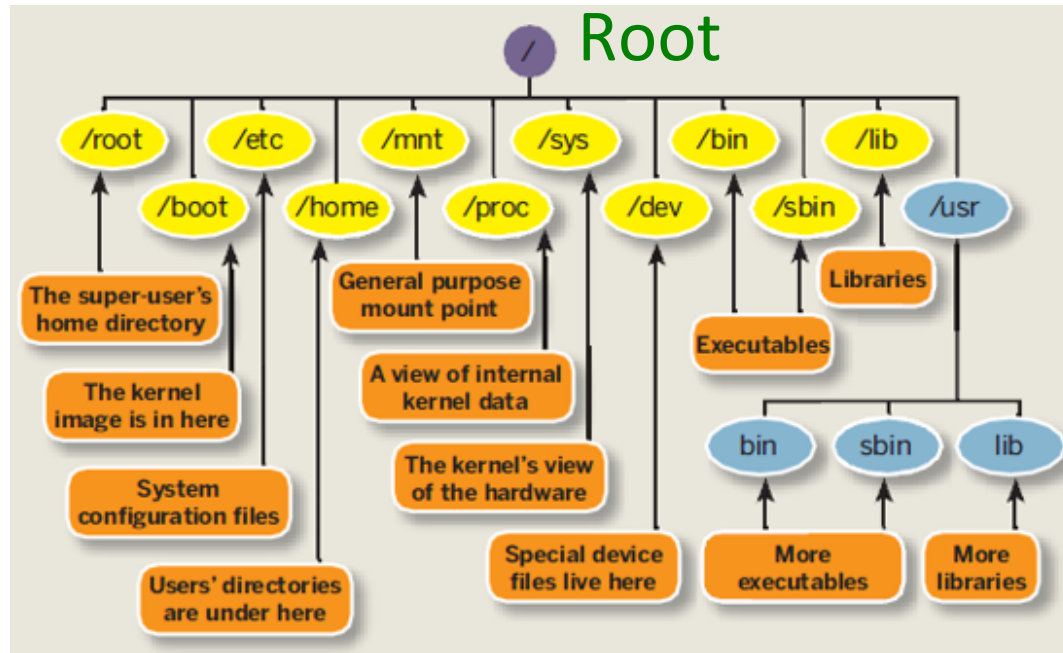
**The web**

<http://linuxconfig.org/linux-commands>



# **Working with files and directories**

# Directory structure



<http://www.tuxradar.com/>

- Everything the system uses is located somewhere under root '/'.
- Every user has his home directory, e.g. /home/swyder
- Do your work in your home directory
- The system folders can only be modified by the administrator

# File and Directory names

- Upper- and lower-case matter
- up to 256 characters long
- Every character except / is allowed. But by convention special characters like \$äéÜ are not used.
- Don't use white spaces. Use underscores (\_), hashes (-) and dots (.) to separate words  
Oct2013\_RNAseq  
~~Oct2013\RNAseq~~

# Main commands

Command	Meaning
ls	Content of current directory
cd <i>dir_name</i>	Change to directory
cd	Change to home directory
cd ~	Change to home directory
mkdir	Make a directory
cp	Copy a file/directory
rm	Delete a file/directory

# **Working with text files**

# Main text commands

UNIX has an extensive toolkit for text extraction, reporting and manipulation

Task	Commands
Show	<b>less</b> , more, head, tail, cat
Search/Extract	<b>grep</b> , <b>cut</b> , awk, uniq
Manipulate	<b>sort</b> , tr, sed, join, paste
Replace	<b>tr</b> , sed
Count	<b>wc</b> , <b>uniq -c</b>
Compare	comm, diff



# Piping

# Piping

## Philosophy

"filters": simple programs which only do 1 thing  
the output of a filter is the input of the next

```
grep "mRNA" test.gff | less
```

```
grep -w "gene" test.gff | cut -f 1 | sort | uniq -c
```



# Redirection

> Writing the output to a file

>> Appending the output to a file

```
ls > output.txt
```

```
grep -w "gene" test.gff | cut -f 1 | sort | uniq -c > output.txt
```

```
ls >> output.txt
```

< Reading from file

```
wc < hello.txt > hello_counts.txt
```

# What we learned in Part 1

- **Command** -**Option(s)** **Parameter(s)**  
`ls -rlh ~/data`
- Working with files / directories  
ls, cd, mkdir, cp, mv, rm
- Directory structure  
everything is under the root: /
- Working with text files  
head, less, grep, cut, sort, tr, wc, uniq
- Tools can be connected by "|"
- The Mac OS X Shell differs from the typical Linux shell

# Sources & Links

## Acknowledgements

- Some exercises are from von Mering group (IMLS, UZH)

## Material

- SIB course <http://edu.isb-sib.ch/course/view.php?id=41>
- O'reilly Books <http://oreilly.com/linux/>
- Video tutorials (~100 min) <http://software-carpentry.org/v4/shell/index.html>
- Cheatsheet <http://www.embnet.org/sites/default/files/quickguides/guideUNIX.pdf>

# Additional topics

- Connecting with Unix/Linux servers
- File/Dir Compression&Extraction
- Installing and running software
- Permissions
- Differences in the shell MA

# Server commands

Command	Task
<code>ssh -X user@hostname</code>	Connect to server
<code>scp &lt;what&gt; &lt;to where&gt;</code>	Transfer file from/to server
<code>sftp user@hostname</code>	Transfer file from/to server (interactive)

# Connect to remote computer

```
$ ssh username@mnf-44.uzh.ch
```

```
RSA key fingerprint is 71:ed:af:1f:d6:0a:43:05:8d:11:34:68:  
2c:2d:79:01.
```

```
Are you sure you want to continue connecting (yes/no)?
```

Type "yes" and press ENTER.

Then you will be asked for your password.

```
$ bash                # we will use the bash shell
```

```
$ whoami              # what is my username on this host
```

```
$ uname -a            # show basic info of the host OS
```

```
$ df -h               # displays free disk space
```

# File/Dir compression

		Command	Meaning
1 file	{	<b>gzip</b> <i>filename</i>	compress file with gzip (adds .gz extension)
		<b>gunzip</b> <i>filename.gz</i>	decompress file with gzip (removes .gz extension)
Many files/ Directories	{	<b>tar xzf</b> <i>filename.tar.gz</i>	extract/decompress files from tar.gz archive
		<b>tar zcvf</b> <i>archive.tar.gz</i> <i>folder_to_compress</i>	creates archive.tar.gz
		<b>unzip</b> <i>filename.zip</i>	unzip archive

zmore, zless, zgrep, ...

# Running programs

- Programs are **executable** files (permissions!)
- Run a bash script

```
#make file executable  
chmod +x script.sh  
#Run it  
./script.sh
```

or

```
#no need for chmod  
bash script.sh
```

- Its the same for any script (python, perl,...)

```
./script.py
```

or

```
python script.py
```

- Run a binary

```
chmod +x bowtie  
./bowtie
```



# Installing Software (binaries)

- Packages

Using a [package manager](#) - takes also care of dependencies

- Linux:

- Ubuntu: via .deb files (e.g. aptitude or apt-get)

- Fedora/SUSE: via .rpm files

- Mac OS X: homebrew (my favourite), MacPorts, fink

- Compiling from source

Typically open-source software is written in C/C++ -> GCC compiler

- Linux: install gcc using the package manager (apt-get search gcc, then apt-get install gcc-XXXX)

- Mac OS X: install gcc using homebrew (brew search gcc, ...) or via XCode

```
./configure
```

```
make
```

```
make install      # optional
```

```
make clean        # optional
```

# To install an executable

Bash only looks at certain directories for commands/software/programs  
\$PATH is a variable

```
echo $PATH
```

```
/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/opt/X11/bin:/usr/texbin
```

1. You copy it into one of the folders in \$PATH
2. You add its directory to \$PATH
3. You create a symbolic link to it into a folder contained in \$PATH

## Details

2. `export PATH=$PATH:directory`
3. `sudo ln -s executable directory`

# Summary: executables

- Come in two flavours: Scripts / Binaries
- Execute permissions must be set:

```
chmod +x programname
```

- Scripts mostly start with the shebang line, telling the shell which interpreter to use. E.g.

```
#!/usr/bin/perl
```

- Executing of executables

```
./prg
```

```
./prg.sh      (bash prg.sh)
```

```
python prg.py
```

```
perl prg.pl
```

# Permissions

```
$ ls -l
```

```
-rw-r--r--  1 swyder  staff    6677 30 Okt 22:02 At.gff
-rw-r--r--@  1 swyder  staff  3723486 28 Okt 17:30 CheatSheetguideUNIX.pdf
drwxr-xr-x 11 swyder  staff    374 11 Nov 16:13 FASTAS
-rw-r--r--  1 swyder  staff     21 30 Okt 22:04 indA.txt
```

Permissions

Owner

Group


Last modification

File size  
(bytes)


File type  
d : directory  
- : regular file

# Changing Permissions

r: read  
w: write  
x: execute



```
$ chmod [ugo][+-][rwx] file
```



u: user  
g: group  
o: other/world

Make a file executable (for you)

```
$ chmod +x file  
chmod ug+rx my_file
```

Setting exact permission (only reading, for you)

```
$ chmod =r file
```

Removing permissions (for you)

```
$ chmod -wx file
```

# Permission code

Owner	Group	Other
<b>r w x</b>	<b>r w x</b>	<b>r - x</b>
4+2+1	4+2+1	4+0+1



7



7



5

r: read  
w: write  
x: execute

`chmod 775 file`

# **Differences in the Shell**

## **Linux - Mac OS**

# Differences Linux - Mac OS X

- Mac line breaks are '\r\n' (and Windows: '\r') instead of the standard Linux '\n'  
When working in the command line make sure the files have the right format  
`perl -pi -e 's/\r\n?/\n/g' <filename>`  
which you could alias and put in the .bashrc
- Mac has no GNU tools by default  
less options – less powerful  
for installation install homebrew  
then: `brew install coreutils`  
All GNU commands are then installed with the prefix 'g': `gls`, `gcp`, `gsed`,...
- Mac has non-standard folder structure  
e.g. home (~) is `/Users/swyder`