Introduction to UNIX environment and command line basics

Stefano Ghignone
Institute for Sustainable Plant Protection – Turin Unit
CNR (Italy)

Why Unix?

The **Unix** operating system (OS) is popular in bioinformatics because of its powerful command-line tools that make scripting and performing automated analyses relatively easy.

- Unics = Uniplexed Information and Computing System
- Developed in the 1960's by MIT and AT&T Bell Laboratory
- Unix was widely adopted by the academic community in the 1970s

- The first computers all used command line interfaces (no GUIs)
- Unix is the operating system of choice for servers since it is highly stable and very secure
- Unix provided one of the first powerful computing system for clients to log into servers and run programs
- Unix has been developed into Linux, Solaris, Mac OSX and many other flavors

Why Unix?

The **Unix** operating system (OS) is popular in bioinformatics because of its powerful command-line tools that make scripting and performing automated analyses relatively easy.

- Unics = Uniplexed Information and Computing System
- Developed in the 1960's by MIT and AT&T Bell Laboratory

Most scientific computing is done in UNIX.

- The first computers all used command line interfaces (no GUIs)
- Unix is the operating system of choice for servers since it is highly stable and very secure
- Unix provided one of the first powerful computing system for clients to log into servers and run programs
- Unix has been developed into Linux, Solaris, Mac OSX and many other flavors

- Most academic universities have High-Performance Computing Clusters (HPCCs) that allow users to upload data and run programs, allowing **thousands of processors** to be used for a single program

- Parallel Computing = computationally intensive programs that normally take weeks to run can often be broken down into **thousands of small jobs** and run in parallel in a few hours

- HPCCs are often used for the alignment and variant detection of **Next-Generation Sequencing** data

  Next-generation sequencing (NGS), also known as high-throughput sequencing, is the catch-all term used to describe a number of different modern sequencing technologies including:

  - Illumina (Solexa) sequencing
  - Roche 454 sequencing
  - Ion torrent: Proton / PGM sequencing
  - SOLiD sequencing

  These recent technologies allow us to sequence DNA and RNA much more quickly and cheaply than the previously used Sanger sequencing, and as such have revolutionized the study of genomics and molecular biology.

UNIX

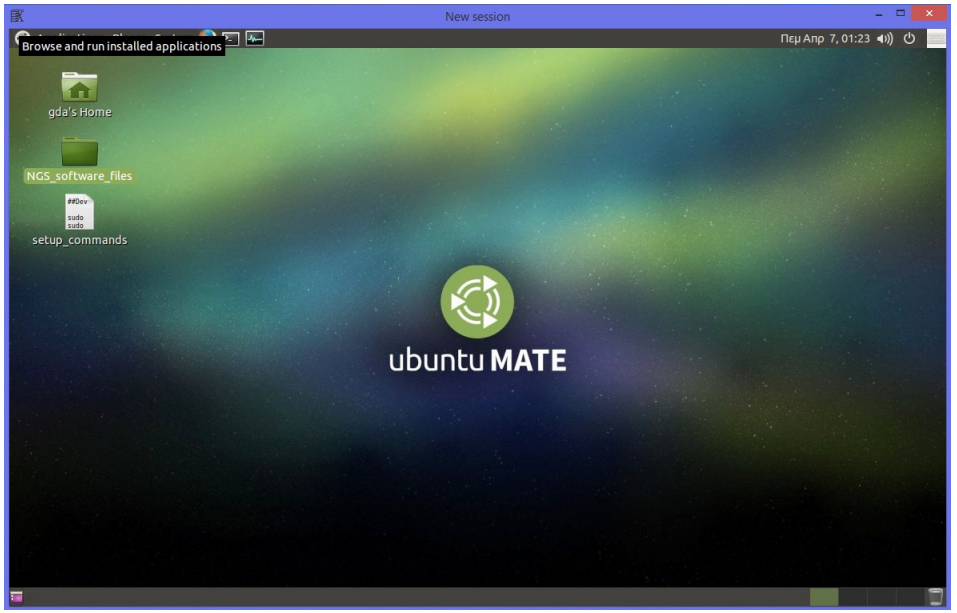
- Linux (Ubuntu, Debian, etc)
- Solaris
- OS X

All have a similar underlying system, and a similar set of command line tools.

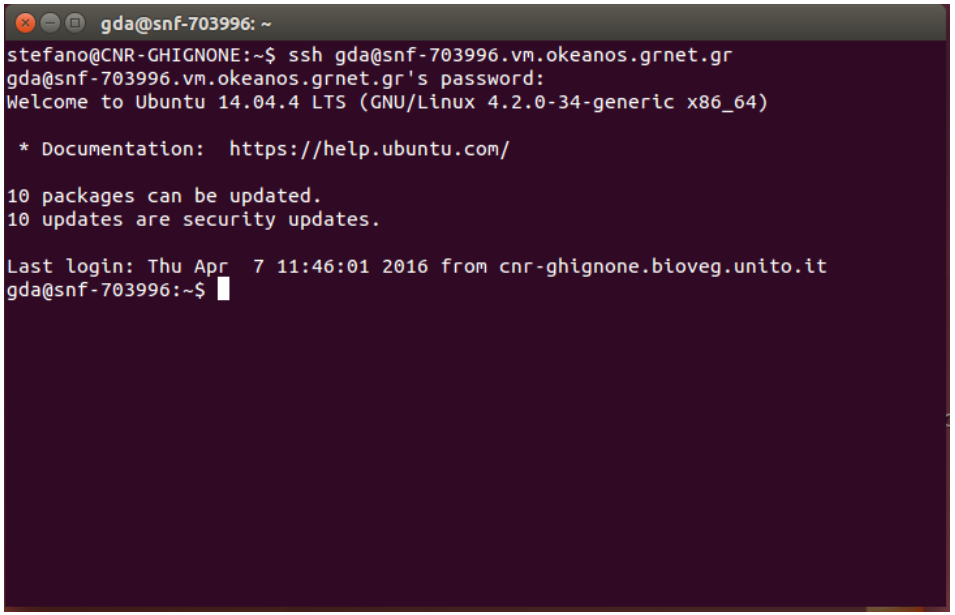
UNIX

- Linux (Ubuntu, Debian, etc)
- Solaris
- OS X

Desktop



Terminal



For PC download and install '**Putty**' SSH client
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

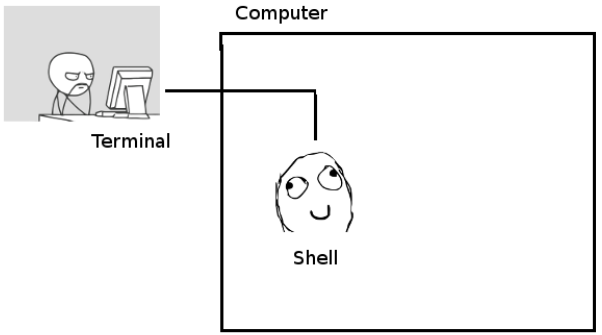
Originally a terminal was a machine like a typewriter.

Many terminals could be plugged into one computer, allowing many people to use it at once.

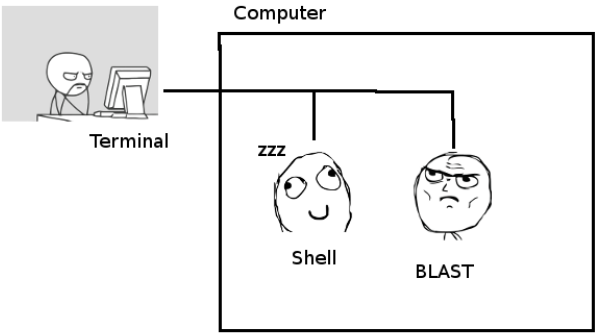
UNIX operating system allows multiple programs to run at once on the computer.

The terminal connects you to a "shell" program that “interprets” your commands.

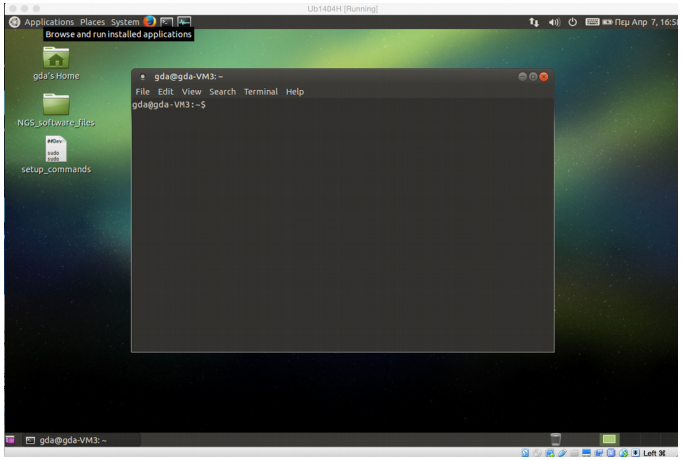
SHELL – is a terminal that allows users to enter commands that instruct the UNIX kernel to run programs. Usually the shell program is "bash" (Bourne Again Shell).



The terminal connects to the computer, and initially talks to a shell program.



When the shell starts up another program it connects it to your terminal as well. The shell goes to sleep until the program finishes.



Unix commands are written in the terminal

First example: the **date** command

By default the program output (current time and date) is directed to **standard output (stdout).**

By default the program errors are written to **standard error (stderr**)

Command outputs can be redirected to a not-existing file using **>** *(or overwrites !)* or appended to an existing file with **>> .**

Command syntax (i)

```
$ command [options] [files]
```

Basically, we type the name of the **command**, followed (if needed) by some **parameters/options** that modify the behavior of the command and finally (if needed), the names of the **files/directories** that the command needs to use/change/manipulate. Space is needed between them. Finally, we press ENTER.

One command may take more than one parameters/flags

Usually, command names are shortcuts for English verbs (list = ls, change directory = cd, Print working directory -> pwd, etc...)

Command syntax (ii)

In the same command line, more than one command can be executed, one after the other, using the **;** as separator (e.g.: **date ; pwd**)

The output of one command may be used as input in the next command, with the *pipe* symbol: **|** (e.g.: **Command1 | command2 > final_results_file**)

Use the **tab** key to *auto-complete* the name of a file/directory. Just type enough words of the file/directory, so that linux can understand what you are referring to.

The history command allows to see which commands has been typed before. (Type: **history**)

Pressing the **up arrow** key, the terminal will show the previous commands executed.

In UNIX there are two types of wildcards that can be used in filenames, to indicate/select multiple files (e.g.: **rm ***):

- The ***** wildcard will search for any number of characters before/after the star
- The **?** wildcard will only search for a single character

To find information about a command, use **man** (manual) (e.g. man ls)

Command syntax (iii)

...about redirection

combination of redirections

redirection of stderr to stdout

command 2>&1

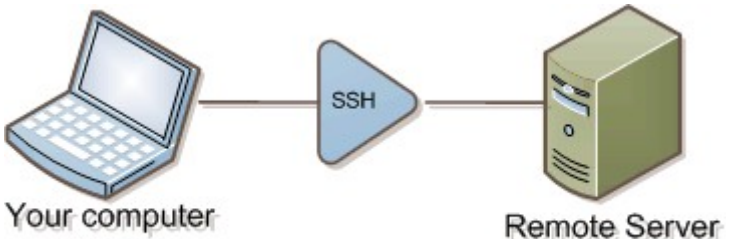
redirection of stdout and stderr to a file

command &> out_err.txt

Login and...what's happening?

Connection to a remote machine

Managed by the ssh (**Secure SHell**) program

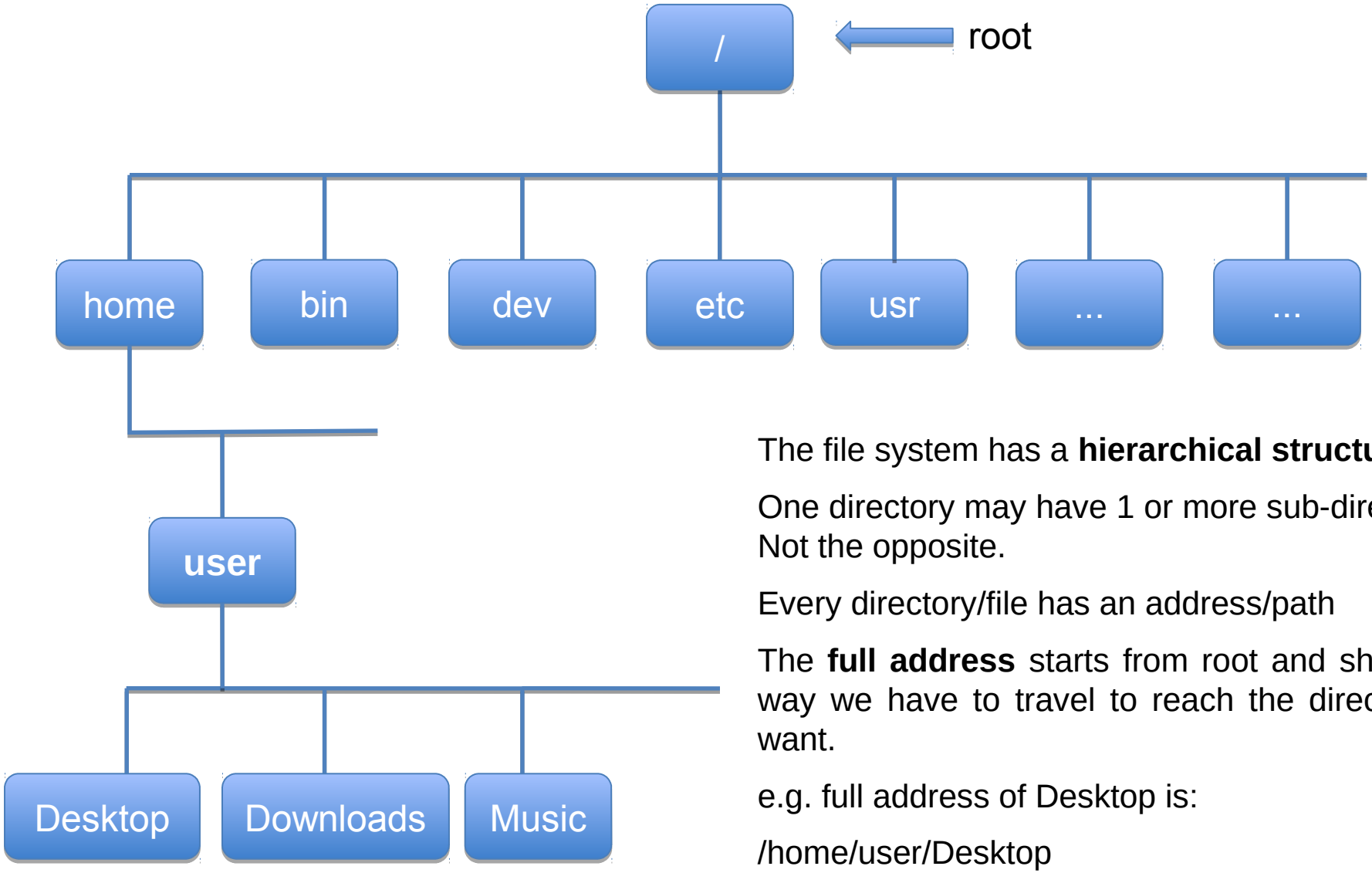


$ ssh username@hostname.unito.it

"Securely connect me to a shell on a remote machine"

```
stefano@GANDALF: ~
stefano@CNR-GHIGNONE:~$ ssh stefano@130.192.101.115
stefano@130.192.101.115's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.19.0-25-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Tue Jul  5 15:17:52 CEST 2016

  System load:    16.07             Users logged in:       2
  Usage of /home: 55.4% of 7.15TB   IP address for em1:    130.192.101.115
  Memory usage:   2%                IP address for em2:    192.168.1.2
  Swap usage:     0%                IP address for lxcbr0: 10.0.3.1
  Processes:      698

  Graph this data and manage this system at:
    https://landscape.canonical.com/

7 packages can be updated.
7 updates are security updates.

Last login: Tue Jul  5 15:17:52 2016 from gandalf.unito.it
stefano@GANDALF:~$
```

w/who	show who's connected
top/htop	show processes
df -h	report filesystem disk space usage
clear	clean the screen
exit	terminate the connection



The file system has a **hierarchical structure**.

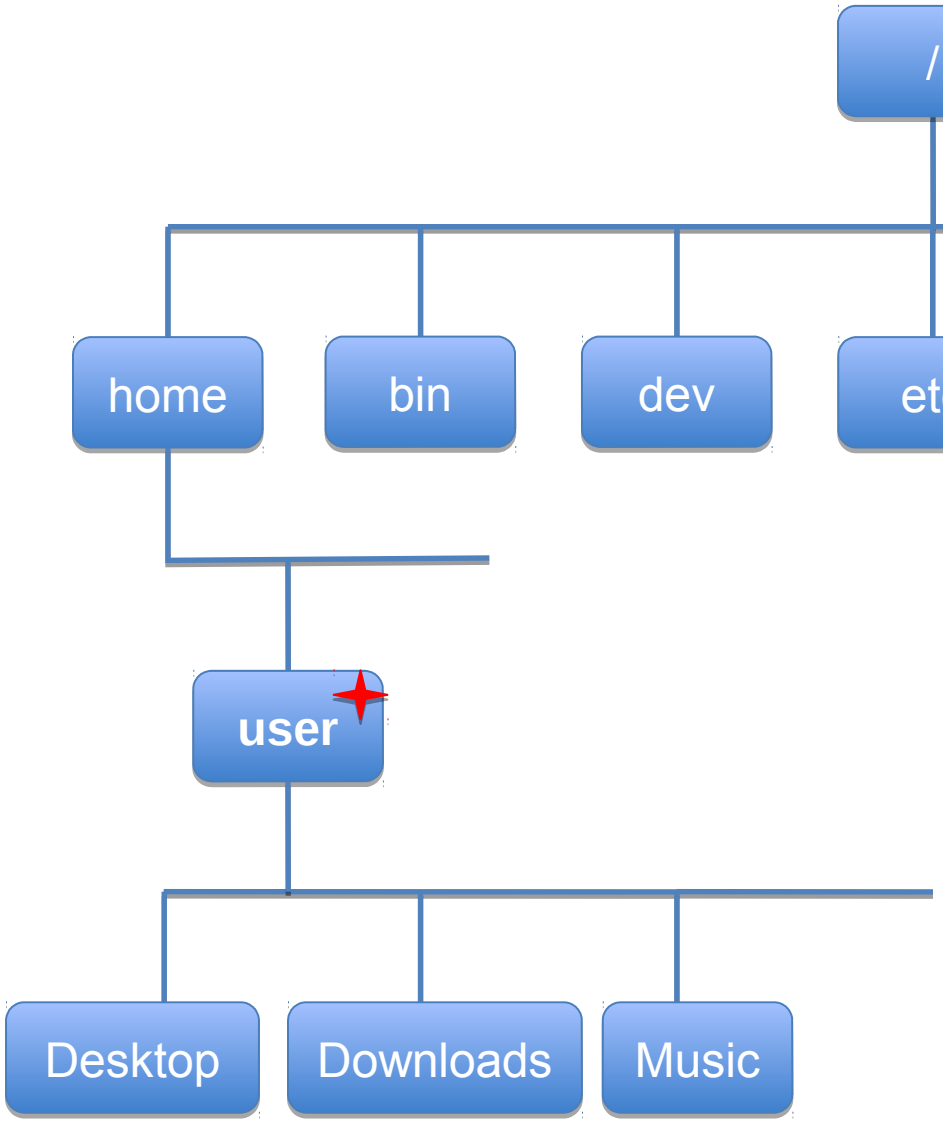
One directory may have 1 or more sub-directories. Not the opposite.

Every directory/file has an address/path

The **full address** starts from root and shows the way we have to travel to reach the directory we want.

e.g. full address of Desktop is:

/home/user/Desktop

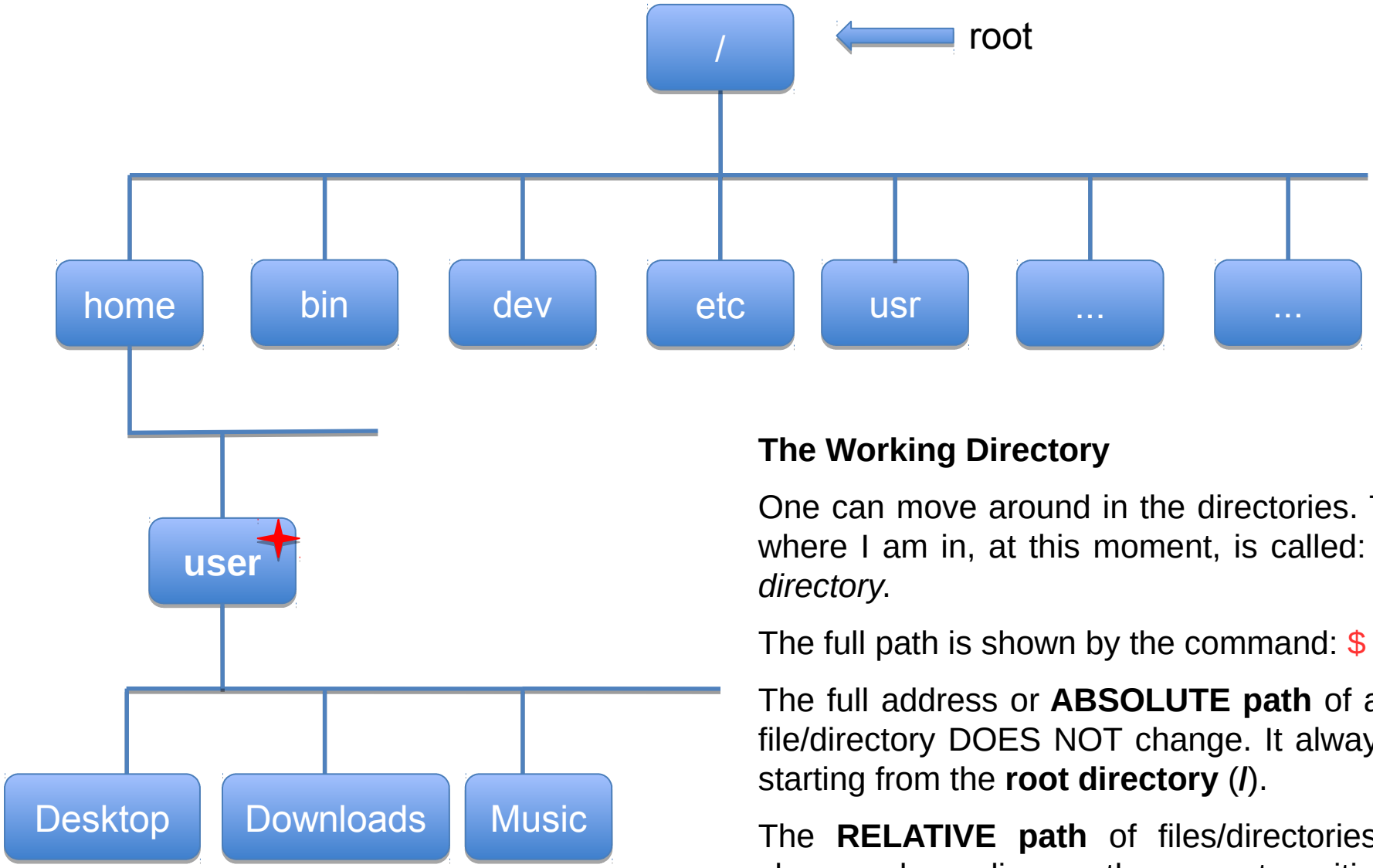


My home directory is where I start from, when I login:

/home/user

The home directory is symbolized with: ~ (“Tilde”)

This is different from the **directory “/home”**, that holds all the accounts/users.



The Working Directory

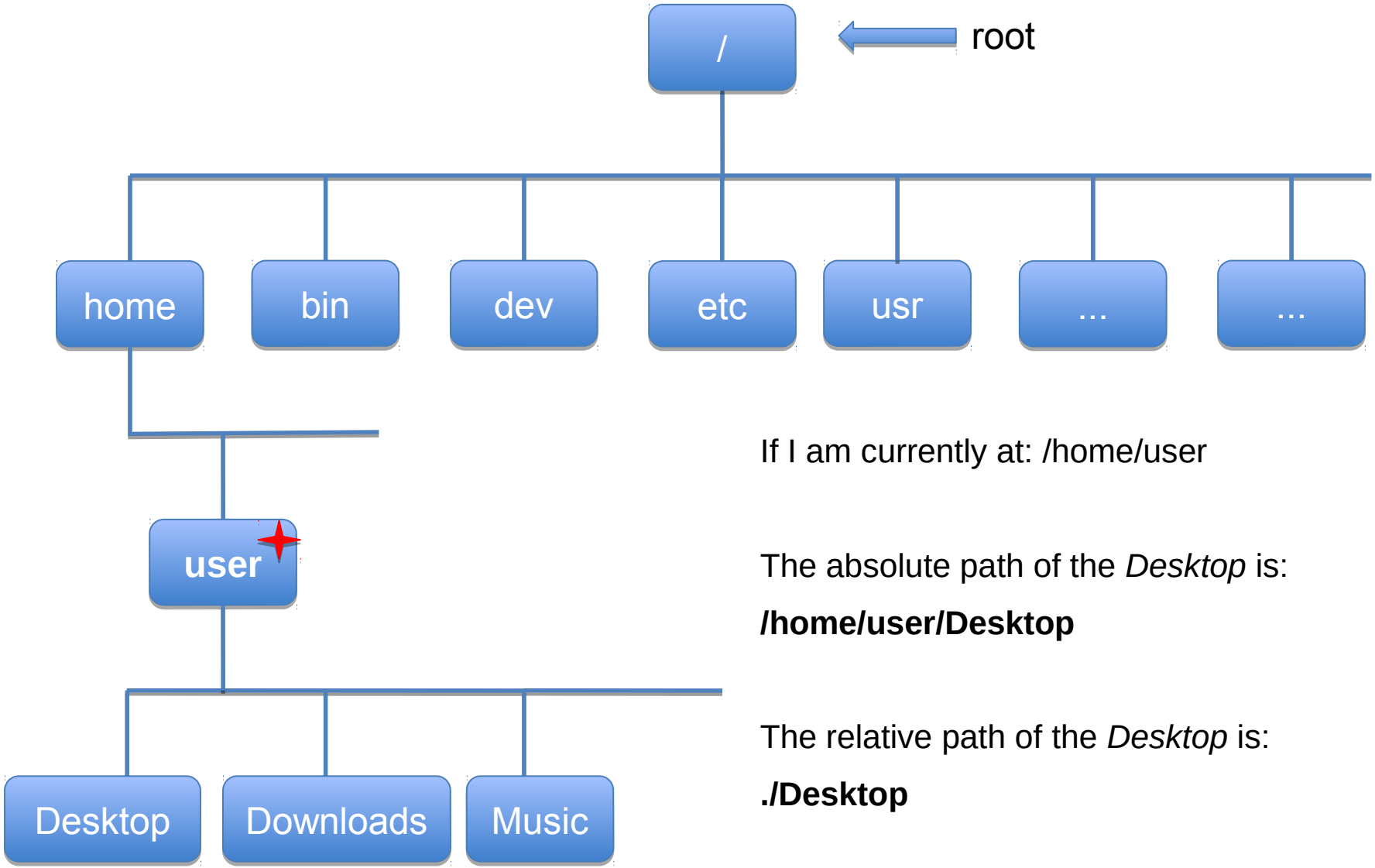
One can move around in the directories. The one where I am in, at this moment, is called: *working directory*.

The full path is shown by the command: $ **pwd**

The full address or **ABSOLUTE path** of a certain file/directory DOES NOT change. It always refers starting from the **root directory** (*/*).

The **RELATIVE path** of files/directories DOES change, depending on the current position in the filesystem.

Moving around the Filesystem



If I am currently at: /home/user

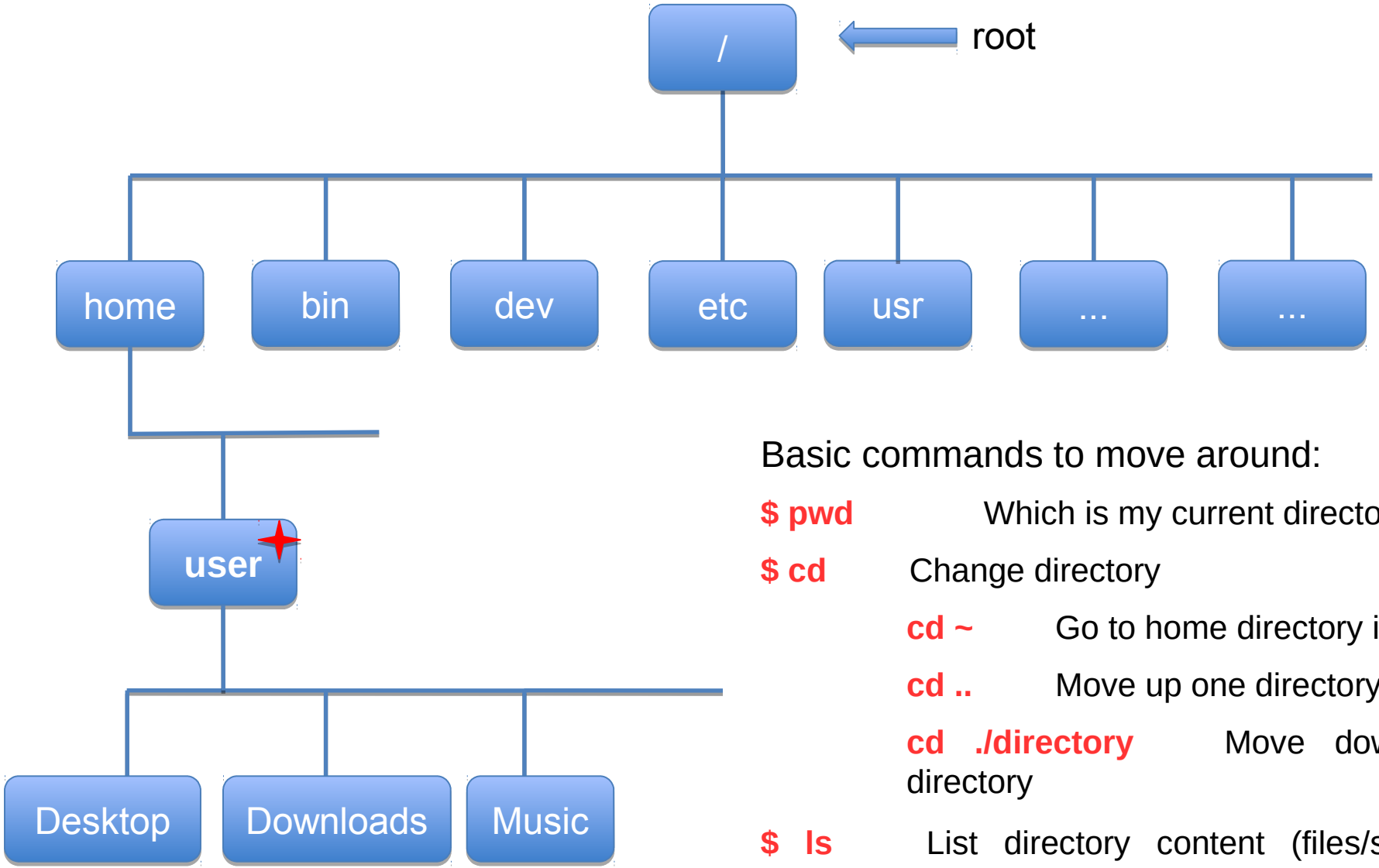
The absolute path of the *Desktop* is:

/home/user/Desktop

The relative path of the *Desktop* is:

./Desktop

Moving around the Filesystem



Basic commands to move around:

$ pwd Which is my current directory?

$ cd Change directory

cd ~ Go to home directory in /user

cd .. Move up one directory

cd ./directory Move down one directory

$ ls List directory content (files/subdirs)

$ ls -lh

```
drwxrwr-x    2    root root      4096    apr  1    16:51    dir
-rwxrwr-x    1    root root      1092    apr  1    16:53    file
1     2          3      4      5        6                7          8
```

1. Type: directory (d), file (-) or link (l)
2. Permissions: read (r), write (w) or execute (x); defined in blocks of 3 characters: first block – owner's rights; second block – group's rights; last block – rights of any user. The dash character (-) means that the respective permission is not set.
3. Number of links: directory (2), file (1). Dir links increase in case of presence of subdirs.
4. Owner
5. Group
6. Size in Byte. If the option -h (human) is used, then file size is rounded and K (Kilobyte), M (Megabyte) or G (Gigabyte) are used.
7. Modification Date and Time
8. Name

Special files: . (*dot*) and .. (*dotdot*); they point to the current directory and to the parent directory, respectively.

Exercises

(1) Open the terminal and print your current ‘position’ in the filesystem

(2) Move to the Desktop and print the Directory content

(3) Print the content of the Download directory

(4) From Desktop, move to the Download directory

(5) Move back to your home directory

(6) List the content of the /usr/lib/perl directory using absolute and relative paths

Exercises

(1) Open the terminal and print your current 'position' in the filesystem

(1) $ pwd

(2) Move to the Desktop and print the Directory content

(1) $ cd ./Desktop

(2) $ ls -lh

(3) Or $ cd ./Desktop ; ls -lh

(3) Print the content of the Download directory

(1) $ ls -lh ../Download (relative)

(2) $ ls -lh ~/Download

(4) From Desktop, move to the Download directory

(1) $ cd ../Download

(5) Move back to your home directory

(1) $ cd

(6) List the content of the /usr/lib/perl directory using absolute and relative paths

(1) $ ls -lh /usr/lib/perl

(2) $ ls -lh ../../usr/lib/perl

Viewing and Editing Files

more/less — browse or page through a text file.
Usage: press the **space bar** to go forward a page, type **b** to go back a page and type **q** to quit.
$ less file1

cat — display and concatenate arguments to standard output
$ cat file1 file2 > file3

head — Show the first 10 lines of a file
Basic Options: **-#** show first # lines of the specified file
$ head -20 file

tail — show last 10 lines of a file
Basic Options: **-f** keep the file open and whenever lines are appended to the file, print them.
$ tail -f file

wc — display the number of lines, words and characters in a file
$ wc file

grep — search for a character string in a file
grep PATTERN FILE
Basic Options: **-c** print only a count of matching lines
$ grep -c “>” file.fasta or **grep -c “@HWI-ST365” file.fastq**

nano/pico
vi/vim
emacs

text editors. *vi* is the standard UNIX editor. Try to use them, and choose your preferred one.

awk

pattern-matching language. Useful to process text files, and extract data, also by means of regular-expression.
#Print the second and the fourth word on each line of a file
$ awk '{print $2, $4}' file

sed

pattern-matching engine, that can perform basic text transformations (manipulation on lines of text).

#print on terminal the file with all occurrences of the string 'red' changed to 'hat'
$ sed 's/red/hat/g' file

#change all occurrences of the string 'red' to 'hat' in the file
$ sed -i 's/red/hat/g' file

Manipulating Files and Directories

cp — copy files
$ cp fromfile tofile

mv — move or rename files
$ mv fromfile tofile

ln — create a link between source and destination files
Basic Options: **-s** creates symbolic link
$ ln -s source dest

mkdir — create a new subdirectory in the current directory
$ mkdir subdir

touch — update the time stamp on existing files, create new empty files
$ touch file

wget — hits a URL and download the information to a file
$ wget remotefile

rm — remove (delete) files (*rmdir* removes empty directories)
Basic Options: -r recursive, descend into subdirs removing files
$ rm file
$ rm -r NotEmptyDirectory

File Packaging and Compression

gzip/gunzip

compress and uncompress files in GNU Zip format. Compressed files have suffix *.gz*.

#Compress *file* to create *file.gz*. Original *file* is deleted

$ gzip file

#uncompress *file.gz* to create *file*. Original *file.gz* is deleted

$ gunzip file.gz

compress/ uncompress

compress and uncompress files in standard Unix compression format. Compressed files have suffix *.Z*

#Compress *file* to create *file.Z*. Original *file* is deleted

$ compress file

#uncompress *file.Z* to create *file*. Original *file.Z* is deleted

$ uncompress file.gz

tar

compact a directory and all its content (files and subdirs) into a single file with name of the packed directory, and with *.tar* extension.

$ tar cvf myarchive.tar mydir *#create*

$ tar xvf myarchive.tar #extract

#gzipped tar files

$ tar czvf myarchive.tar.gz mydir *#create*

$ tar xzvf myarchive.tar.gz #extract

zip/unzip

compress and uncompress files in Windows Zip format.
Compressed files have suffix *.zip*. Does not delete original file(s)

```
$ zip myfiles.zip file1 file2 file3....      #pack
$ zip -r myfiles.zip directory               #pack recursively
$ unzip myfile.zip                           #unpack
```

Exercises

(1) Retrieve a fasta (example.fasta) and a fastq (Cont-2-9_R1.subset.fastq) files from github

wget -c https://github.com/PhD-Toolbox-course/2018_PhD_Toolbox_course/tree/master/Resources/Day1'file'

(2) Inspect the files (cat/less)

(3) View the beginning (head) and the end of the files (tail)

(4) Calculate how many sequences they contain
$ grep -c ">" or grep "…"

(5) Create a subset of sequences. Which strategy would you use?

(6) Count how many occurrences of the pattern "ATTCAAAT" appear in the .fastq file

(7) Retrieve the sequences (with all the fastq data) containing "ATTCAAAT"

(1) Retrieve a fasta (example.fasta) and a fastq (Cont-2-9_R1.subset.fastq) files from github

wget -c https://github.com/PhD-Toolbox-course/2018_PhD_Toolbox_course/tree/master/Resources/Day1'file'

(2) Inspect the files (cat/less)

(3) View the beginning (head) and the end of the files (tail)

(4) Calculate how many sequences they contain
$ grep -c ">" or grep "…"

(5) Create a subset of sequences. Which strategy would you use?

```
awk 'BEGIN { RS = "\n>"; FS = "\n"; OFS = "" };
{
if (NR == 1) {
        print $1
} else
if (NR > 1) {
        print ">"$1
}
$1=""
print
}' infile.fasta
```

Exercises

(1) Retrieve a fasta (example.fasta) and a fastq (Cont-2-9_R1.subset.fastq) files from github

wget -c https://github.com/PhD-Toolbox-course/2018_PhD_Toolbox_course/tree/master/Resources/Day1'file'

(2) Inspect the files (cat/less)

(3) View the beginning (head) and the end of the files (tail)

(4) Calculate how many sequences they contain
$ grep -c ">" or grep "…"

(5) Create a subset of sequences. Which strategy would you use?

(6) Count how many occurrences of the pattern "ATTCAAAT" appear in the .fastq file

(7) Retrieve the sequences (with all the fastq data) containing "ATTCAAAT"

blastn

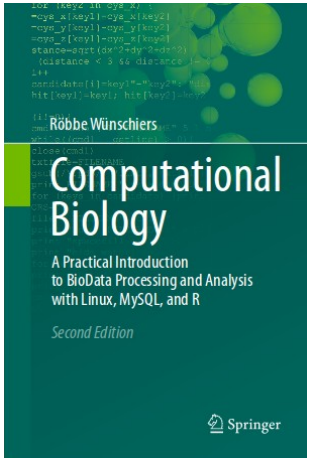
blastn -query fasta.file -db nr -outfmt 6 -num_alignments 1 -num_descriptions 1 -out haktan.txt -dust no -task blastn

Only with blast 2.6.0

Mkdir Programs
wget ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/ncbi-blast-2.6.0+-x64-linux.tar
tar xzf ncbi-blast-2.6.0+-x64-linux.tar.gz
cd ncbi-blast-2.6.0+/bin/
Pwd
export PATH=/home/stefano/Programs/ncbi-blast-2.6.0+/bin:$PATH

blastn -query example.2.fasta -db nt *-remote*

Useful references

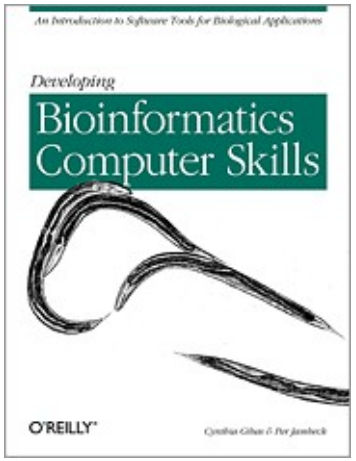


Computational Biology

A Practical Introduction to BioData Processing and Analysis with Linux, MySQL, and R

Authors: Röbbe Wünschiers

ISBN: 978-3-642-43097-8 (Springer)

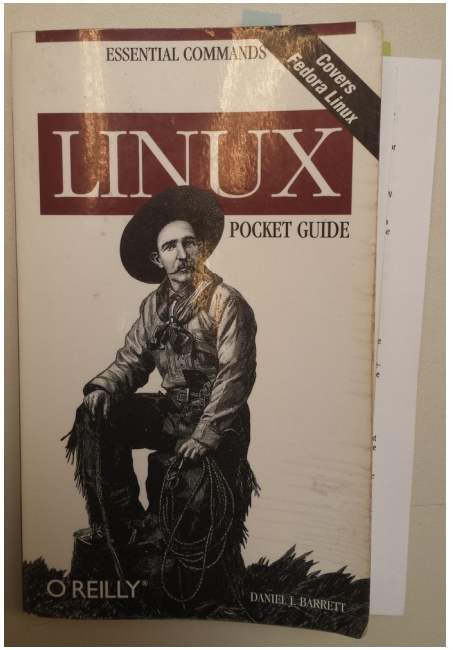


Developing Bioinformatics Computer Skills

An Introduction to software Tools for Biological Applications

Authors: Cynthia Gibas, Per Jambeck

ISBN: 978-1-56592-664-6 (O'Reilly Media)



Linux Pocket Guide

Essential Commands

Authors: Daniel J. Barrett

ISBN: 978-0-596-00628-0 (O'Reilly Media)

Overused real sample