CHARLES UNIVERSITY

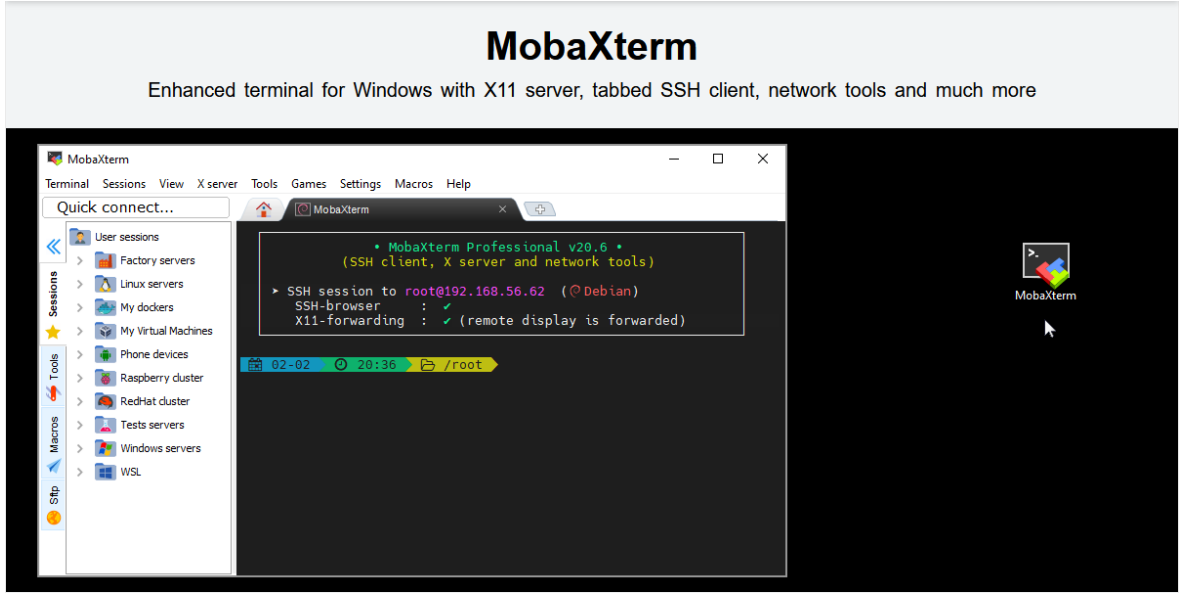
Bioinformatics and Microbiome Analysis MB140P94

Bioinformatic data and Linux

Tomáš Větrovský, Priscila Thiago Dobbler
Laboratory of Environmental Microbiology
Institute of Microbiology of the CAS

Access to the Linux server - WINDOWS

https://mobaxterm.mobatek.net/download-home-edition.html



Home Edition

Free

Full **X server** and **SSH** support

Remote desktop (RDP, VNC, Xdmcp)

Remote terminal (SSH, telnet, rlogin, Mosh)

X11-Forwarding

Automatic SFTP browser

Master password protection

Plugins support

Portable and installer versions

Full documentation

Max. **12** sessions

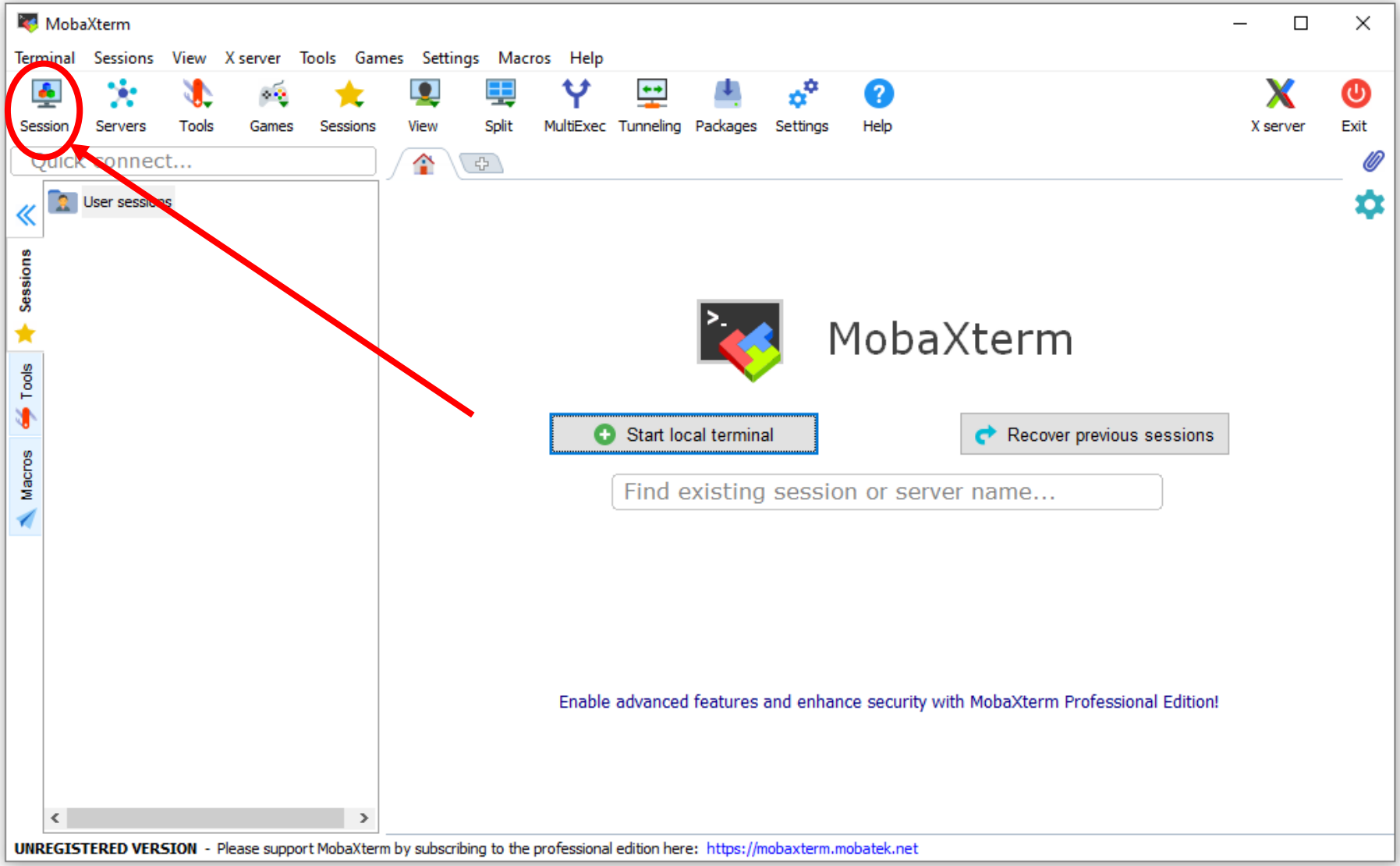
Max. **2** SSH tunnels

Max. **4** macros

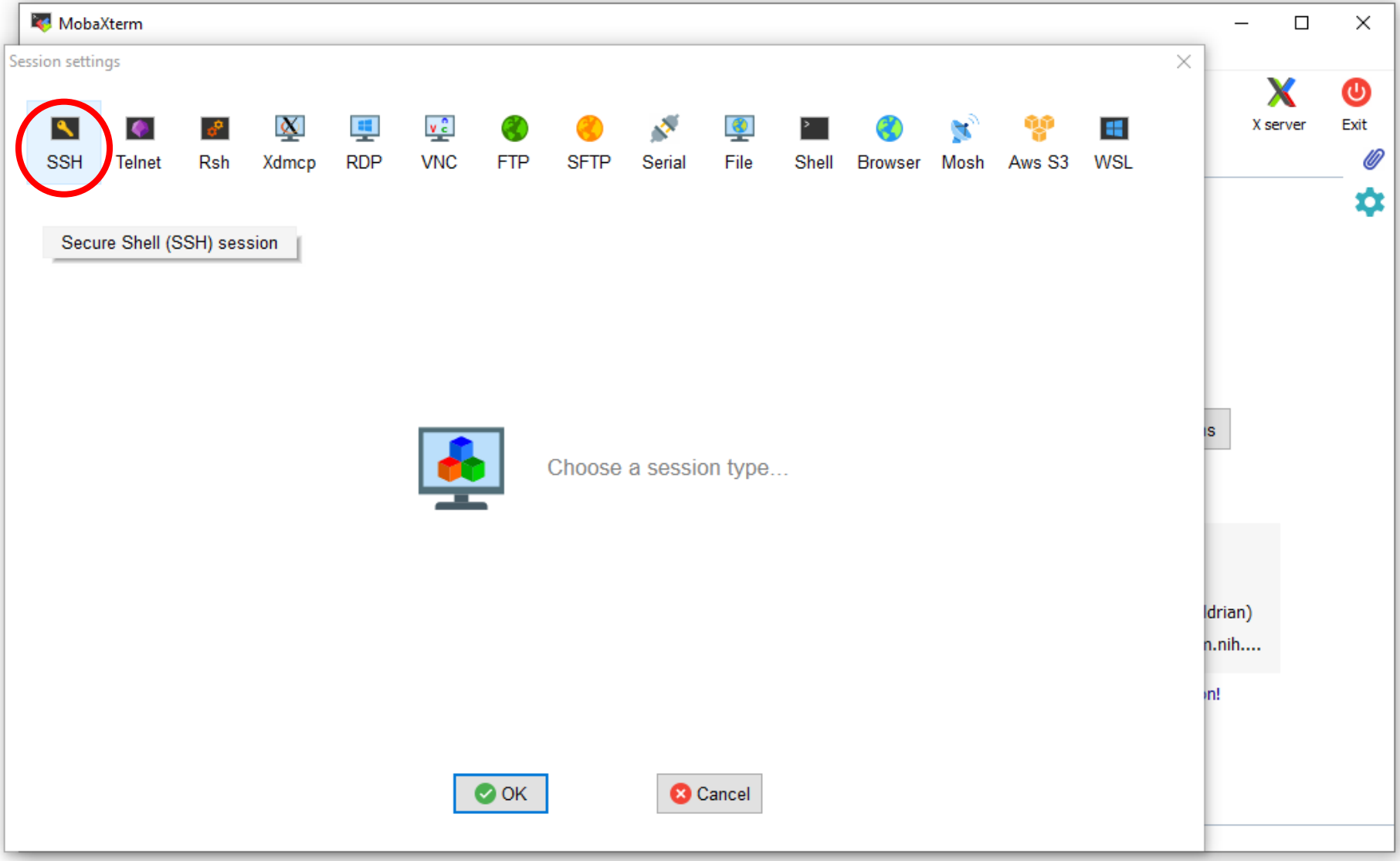
Max. **360** seconds for Tftp, Nfs and Cron

Download now

Access to the Linux server

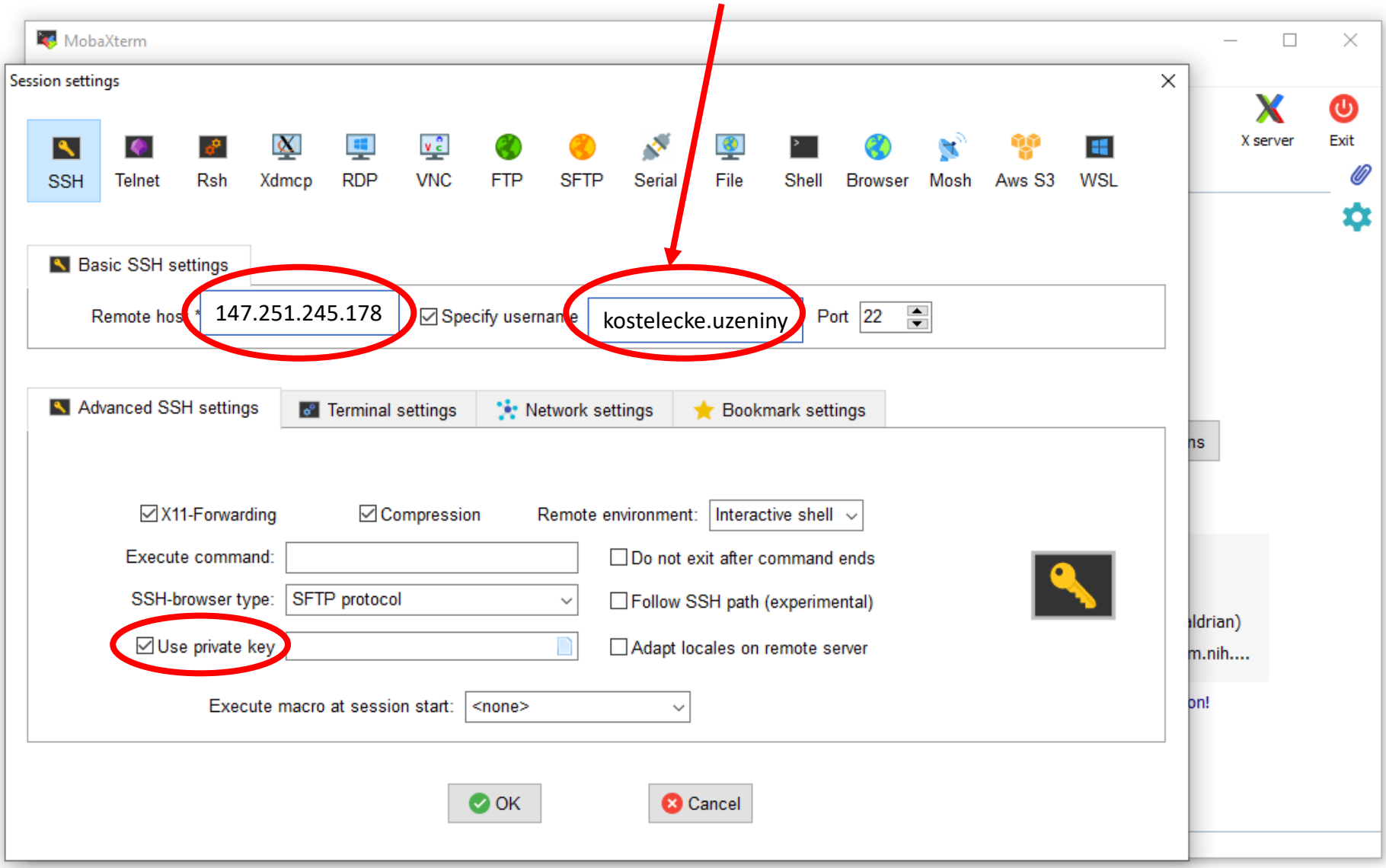


Access to the Linux server

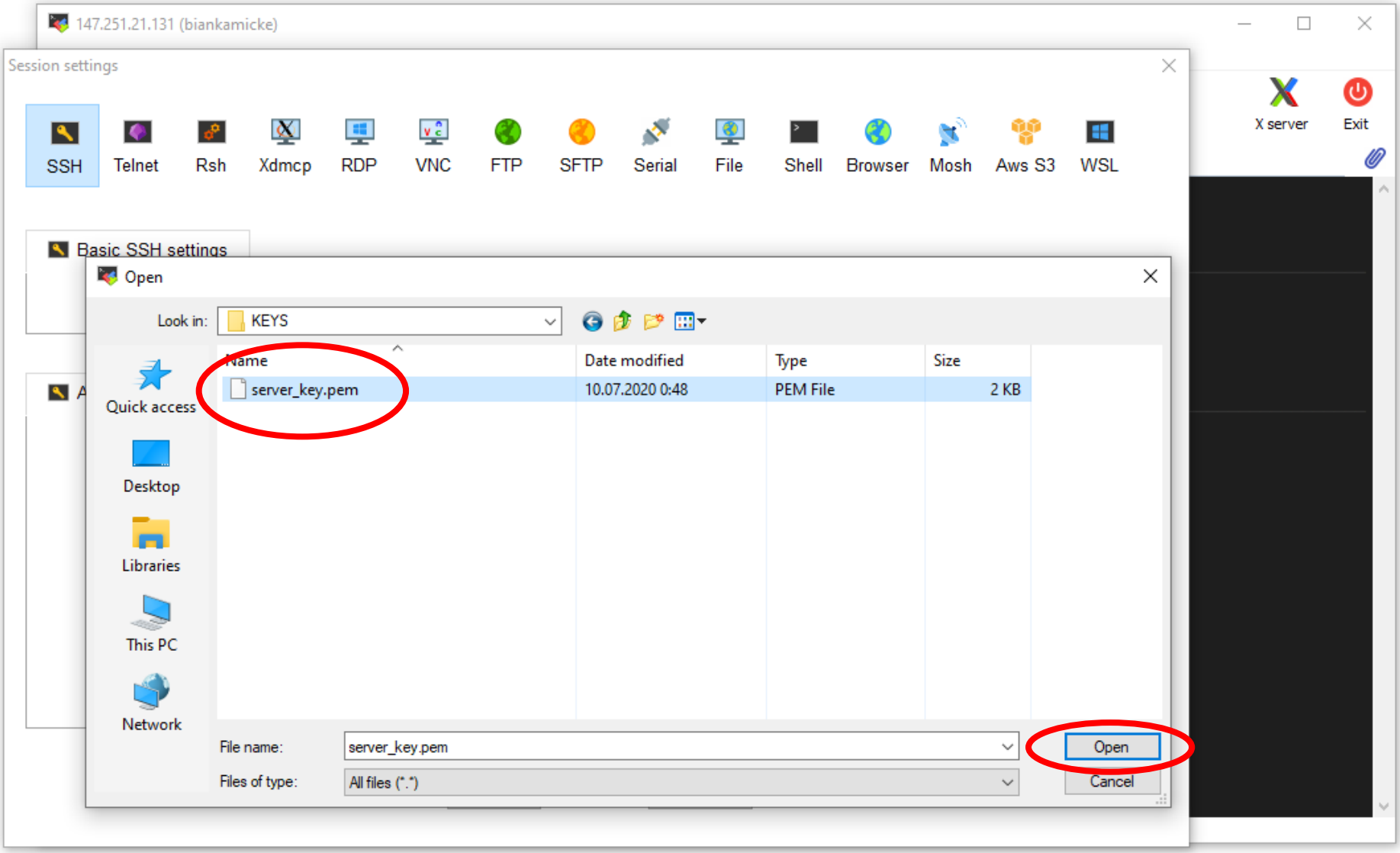


Access to the Linux server

The first part of email before „@“ – my example:
kostelecke.uzeniny@seznam.cz

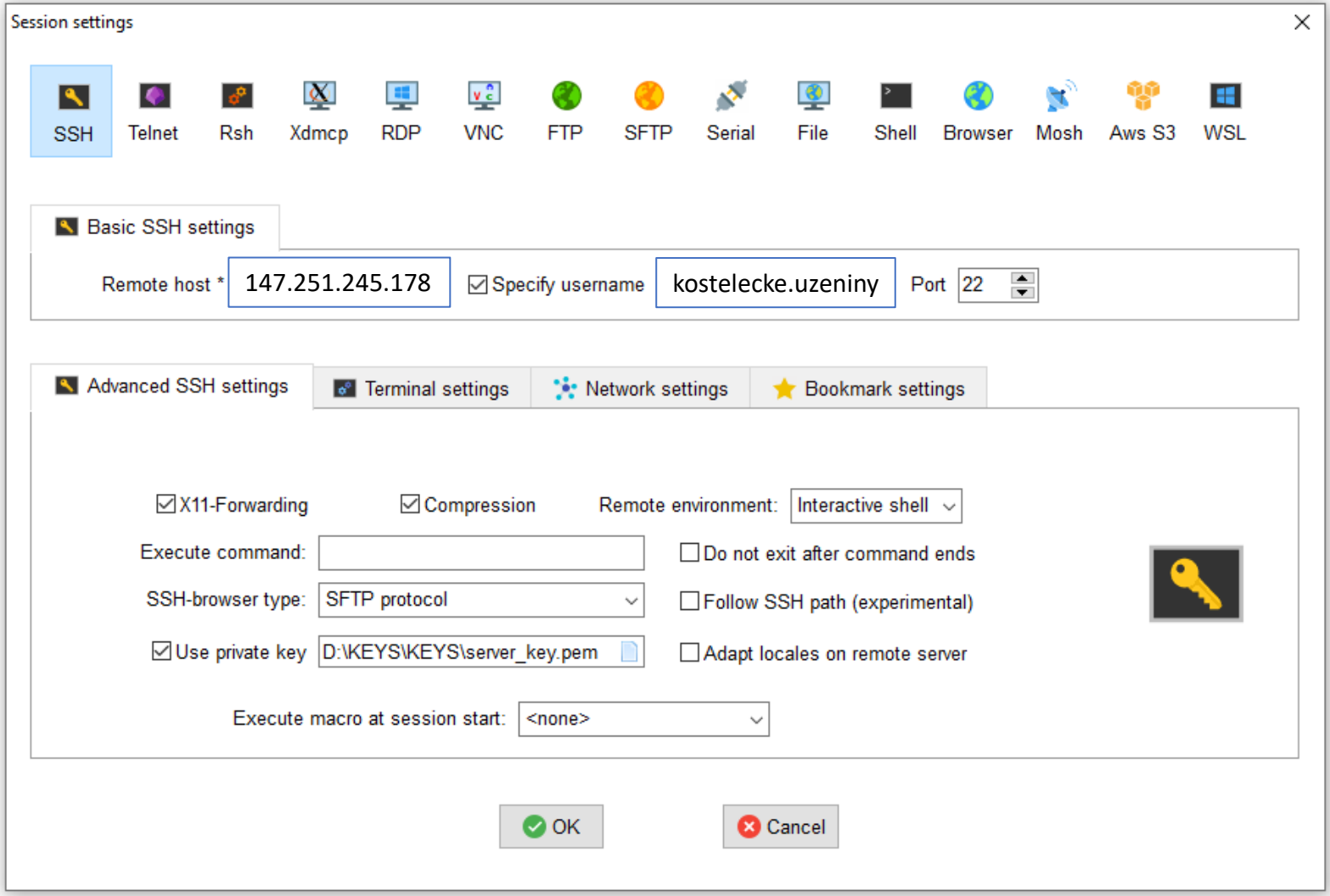


Access to the Linux server



https://drive.google.com/file/d/14dWJ64UZu4fwG_fccGqMFdf5tvoFFI-s/view?usp=share_link

Access to the Linux server



Access to the Linux server

6. PROCESSING

```
                    • MobaXterm 20.2 •
            (SSH client, X-server and networking tools)

 ➢ SSH session to [redacted]@147.251.21.131
   • SSH compression : ✔
   • SSH-browser     : ✔
   • X11-forwarding  : ✔  (remote display is forwarded through SSH)
   • DISPLAY         : ✔  (automatically set on remote server)

 ➢ For more info, ctrl+click on help or visit our website

Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-110-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Feb 23 16:26:15 UTC 2023

  System load:  0.0                Processes:             371
  Usage of /:   5.9% of 247.94GB   Users logged in:       0
  Memory usage: 0%                 IPv4 address for ens3: 192.168.0.38
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

53 updates can be applied immediately.
To see these additional updates run: apt list --upgradable


*** System restart required ***
Last login: Thu Feb 23 14:26:57 2023 from [redacted]
[redacted]@intense-computing-machine:~$
```

Access to the Linux server - MAC

ssh -i server_key.pem <user_name>@147.251.245.178

- Open terminal and go the your **.ssh/** directory
- Create a file called **config**
- Add the following 1 line configuration in this file

```
Host * PubkeyAcceptedKeyTypes=+ssh-dss
```

p <path to your key> ~/.ssh/<name of your key>
Now you should be able to modify the permissions normally.

chmod 600 ~/.ssh/<your key's name>
Then ssh using WSL:

ssh -i ~/.ssh/<name of your key> <username>@<ip address>

LINUX - Let's start from the beginning

- **Linux is case sensitive**
- **Use [TAB] to finish the commands/file names/directory names**

```
pwd
mkdir myDir
cd myDir
cd ..
ls
```

Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-110-generic x86_64)

How do I install applications in Ubuntu?

you have to use the **sudo** command

Installing apps using apt is as easy as:

$ sudo apt install app_name

Uninstalling an app via apt is also super easy:

$ sudo apt remove app_name

To upgrade your installed apps, you'll first need to update the app repository:

$ sudo apt update

Once finished, you can update any apps that need updating with the following:

$ sudo apt upgrade

What if you want to update only a single app? No problem.

$ sudo apt update app_name

Linux Screen

- Screen or GNU Screen is a terminal multiplexer
- Open any number of "windows " (virtual terminals) inside the session
- **Processes running in Screen will continue to run when their window is not visible even if you get disconnected!**

Starting named session

screen -S [session_name]
 # create a screen "examples1"
screen -S examples1

 # detach from Screen session
Ctrl+a d

 # reattach to a Screen session
screen -r

Linux Screen

- Screen or GNU Screen is a terminal multiplexer
- Open any number of "windows " (virtual terminals) inside the session
- **Processes running in Screen will continue to run when their window is not visible even if you get disconnected!**

 # detach from Screen session if you are in any
Ctrl+a d

<u>Multiple named sessions</u>

 # create a screen "examples2"
screen -S examples2

 # detach from screen Session
Ctrl+a d

 # list the current running screen sessions
screen -ls

 # reattach to a Screen session
screen -r

Linux Screen

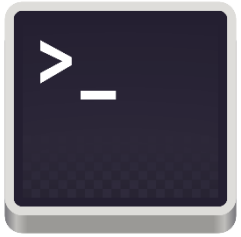
- Screen or GNU Screen is a terminal multiplexer
- Open any number of "windows " (virtual terminals) inside the session
- **Processes running in Screen will continue to run when their window is not visible even if you get disconnected!**

 # force reattach to a (detached) Screen session
screen -r -d examples2

<u>Kill detached screen session</u>

screen -X -S [session # you want to kill] quit
screen -X -S examples1 quit

Basic Linux Screen Usage



Below are the most basic steps for getting started with screen:

1. On the command prompt, type screen -r [name]
2. Run the desired program
3. Use the key sequence **Ctrl-a + Ctrl-d** to detach from the screen session
4. Reattach to the screen session by typing screen –r [name]

Standard output (STDOUT)

Standard output is a stream to which a program **writes its output data**. The program requests data transfer with the *write* operation. Not all programs generate output. For example, the *file rename* command (variously called *mv*, *move*, or *ren*) is silent on success.

 # testing STDOUT
echo "This is normal output"

 # store STDOUT
echo "This is a header." > header.txt

 # append STDOUT
echo "…this is an addition." >> header.txt

 # check the result (cat, less, head)
cat header.txt

 # store manual info stdout to a file
man cat > body.txt

CTRL+c - interupt

Task 1: Create a file "fulltext.txt" combining the header and body.

Copy and rename

 # copy a file
cp fulltext.txt fulltext.info

 # rename a file
mv header.txt header.head

 # remove a file
rm fulltext.txt

Word count

 # copy a file
wc -l fulltext.info > number.out

Task: What does the return value of "**wc -L fulltext.info**" mean?

Text editors – create and edit text files

vi editor
vim editor
nano editor <- this is my favorite

<u>Nano create a text file</u>

 # create an empty file
nano [file name]

1. Write a text…
2. Save it by "Ctrl+o"
3. Leave editor "Ctrl+x"

Text editors – create and edit text files

 # create empty file
nano test.txt

1. Write a text…
   This is line 1.
   This is line 2.
   This is line 3.
2. Save it by "Ctrl+o"
3. Leave editor "Ctrl+x"

 # convert text files with Unix line breaks to DOS or Mac line breaks
unix2dos test.txt

 # show non-printing characters
cat -v test.txt

 # convert text files with DOS or Mac line breaks to Unix line breaks
dos2unix test.txt

grep – search in text files

 # store list of files to a file
ls > files.txt

*# get all lines contains word "te" and "*xt*"*
grep 'te' files.txt
grep 'xt' files.txt

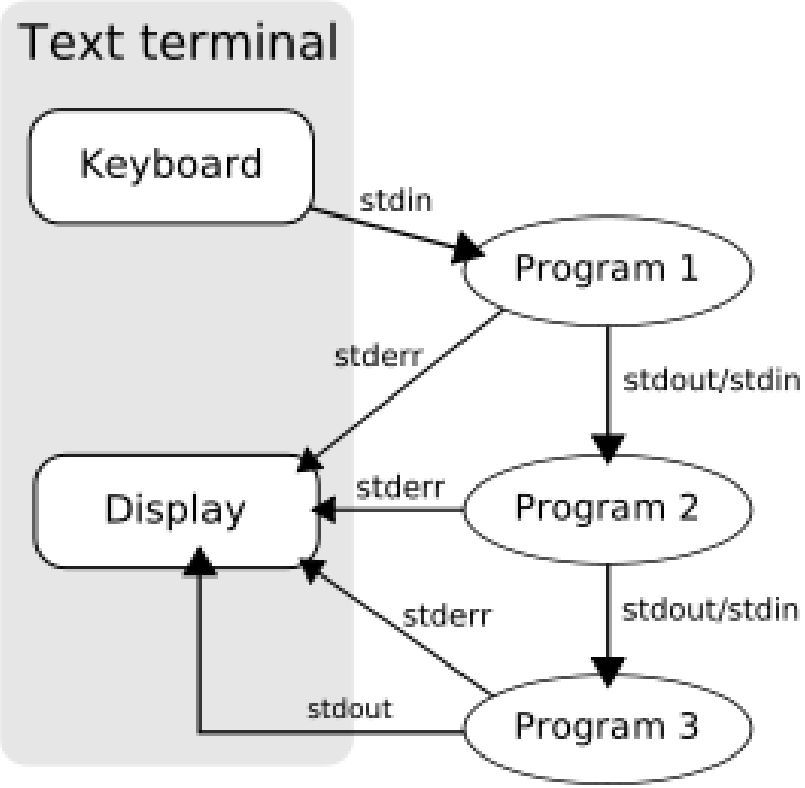
 # get all lines starting by word “te"
grep '^te' files.txt

*# get all lines ending by word "*xt*"*
grep 'xt$' files.txt

Task: What will the commands "**grep -A 1**" and "**grep -B 1**" do?

Pipeline

- is a mechanism for inter-process communication using message passing
- is a set of processes chained together by their standard streams, so that the output text of each process (stdout) is passed directly as input (stdin) to the next one
- the first process is not completed before the second is started, but they are executed concurrently




```
#chain of three processes...
process1 | process2 | process3
```

For example, to list files in the current directory (ls), retain only the lines of ls output containing the string "txt" (grep), and count the lines (wc), a user types the following into the command line of a terminal:

```
#prepare list of commands...
ls -l | grep "txt" | wc -l
```

https://en.wikipedia.org/wiki/Pipeline_(Unix)

Download a sequence file from the internet

 # wget

wget http://www.biomed.cas.cz/mbu/lbwrf/example1.fq.gz

 # gunzip

gunzip example1.fq.gz

Fastq file structure

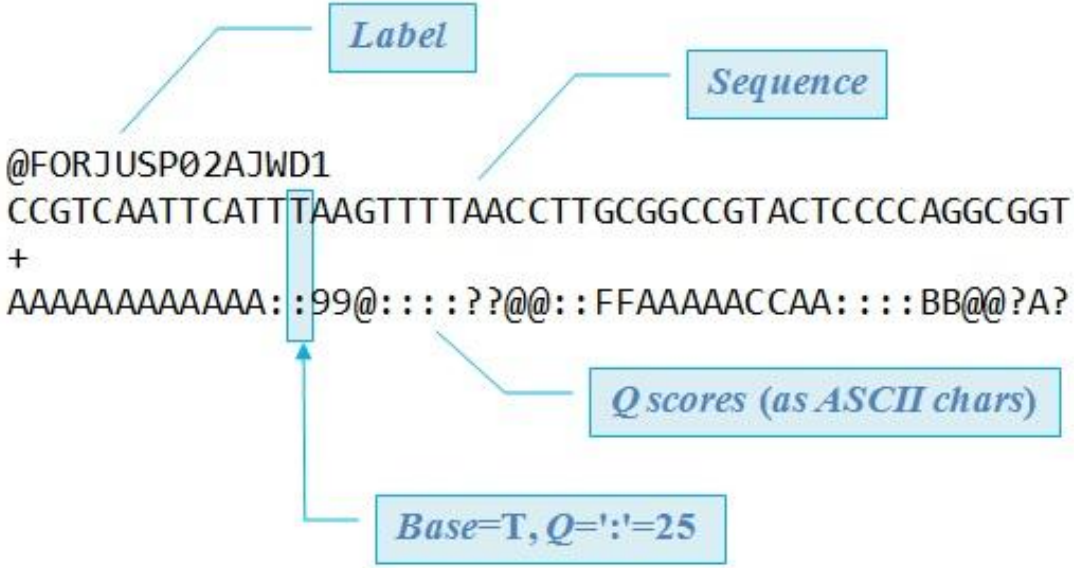


Table 2. Phred quality scores are logarithmically linked to error probabilities (http://en.wikipedia.org/wiki/Phred_quality_score)

Phred quality score	Probability of incorrect base call	Base call accuracy
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1000	99.90%
40	1 in 10 000	99.99%
50	1 in 100 000	99.999%
60	1 in 1 000 000	99.9999%

FASTQ - Quality filtering - fastq-mcf

Usage: fastq-mcf [options] <adapters.fa> <reads.fq> [mates1.fq ...]

 # simple quality filtering by quality mean threshold
fastq-mcf --qual-mean 35 n/a example1.fq > filtered.fq

Task 1: Use **grep** command to count number of sequence s in original file and filtered file.

Task 2: Why just **grep '@'** is not the correct way?

FASTQ to FASTA

sed can be used to selectively print the desired lines from a file, so if you print the first and 2rd line of every 4 lines, you get the sequence header and sequence needed for fasta format.

sed -n '1~4s/^@/>/p;2~4p' filtered.fq > final1.fasta

Or you can use the program "**seqret**":

seqret -sequence filtered.fq -outseq final2.fasta

Task: What is the difference between final1.fasta and final2.fasta

Fasta file

>M02149:53:000000000-AANLH:1:1101:14924:1701 1:N:0:0
TACGGAGGGTGCAAGCGTTAATCGGAATCACTGGGCGTAAAGCGCAC
GTAGGCTGTCTGGTAAGTCAGGGGTGAAATCCCGCGGCTCACCCGCG
GAATTGCCCTTGATACTGCTGGACTTGAGTTCGGGAGAGGGTGGCGG
AATTCCAGGTGTAGGAGTGAAAGGCGTAGATAGCAGGAGGAACATCA
GGGGCGAAGGCGGCCACCTGGACCGATACTGACGCTGAGGTGCGAA
AGCGTGGGGAGGAAACAGG

 # linearize a FASTA sequence

```
awk '/^>/{print s? s"\n"$0:$0;s="";next}{s=s sprintf("%s",$0)}END{if(s)print s}' final2.fasta
```

Pairwise sequence alignment

is used to identify regions of similarity that may indicate functional, structural and/or evolutionary relationships between two biological sequences (protein or nucleic acid).

Needleman–Wunsch algorithm

match = 1 mismatch = -1 gap = -1

		G	C	A	T	G	C	G
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

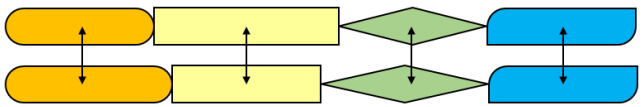
It is also sometimes referred to as the optimal matching algorithm and the global alignment technique.

Initialization
First row and first column are subject to gap penalty

Scoring
Score can be negative

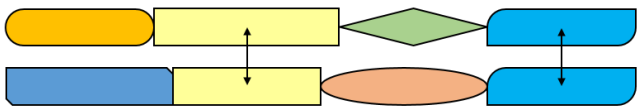
Traceback
Begin with the cell at the lower right of the matrix, end at top left cell

Pairwise sequence alignment



Global Alignment

Needleman–Wunsch algorithm



Local Alignment

Smith–Waterman algorithm

Initialization
First row and first column are set to 0

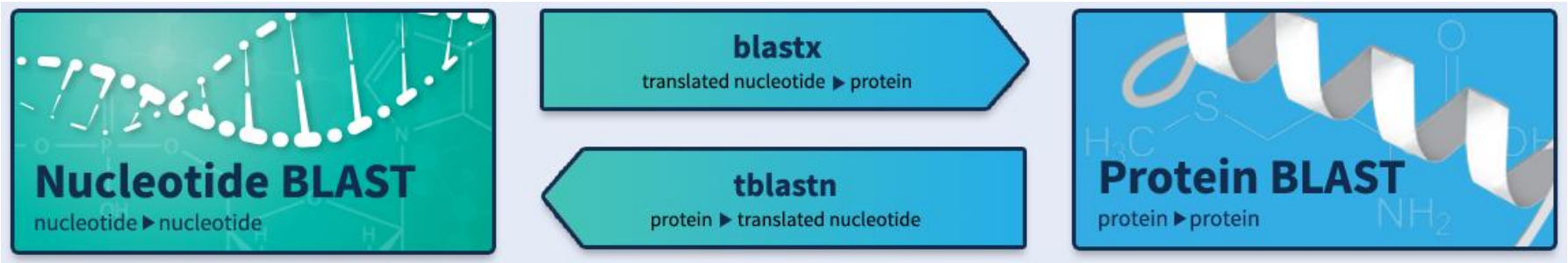
Scoring
Negative score is set to 0

Traceback
Begin with the highest score, end when 0 is encountered

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

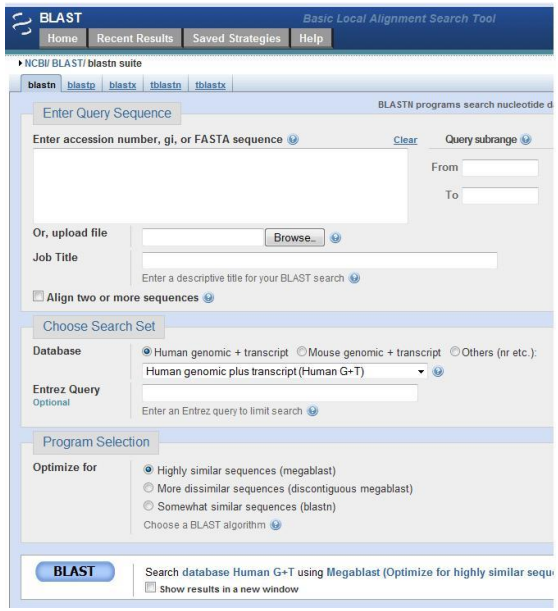
3	6	9	7	10	13
G	T	T	-	A	C
I	I	I		I	I
G	T	T	G	A	C

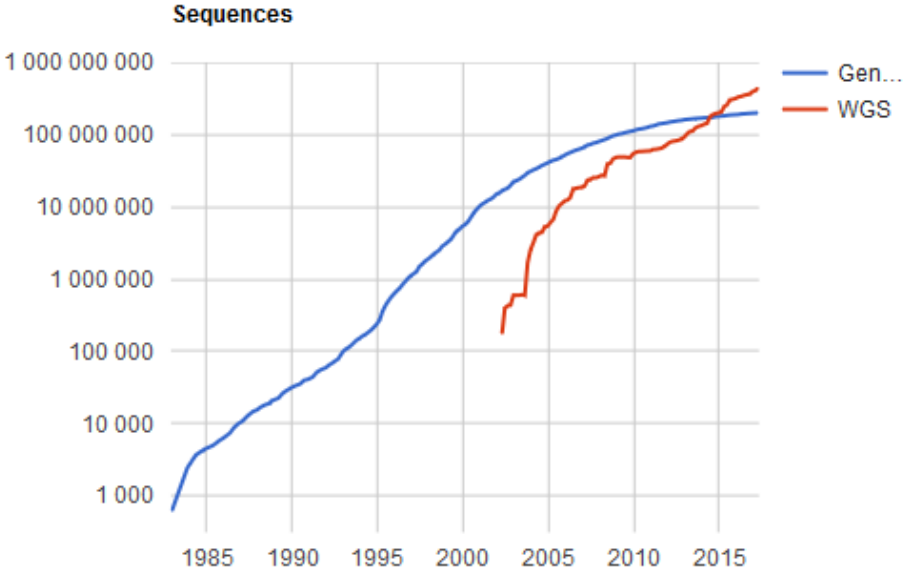
Basic Local Alignment Search Tool



BLAST Online

https://blast.ncbi.nlm.nih.gov/



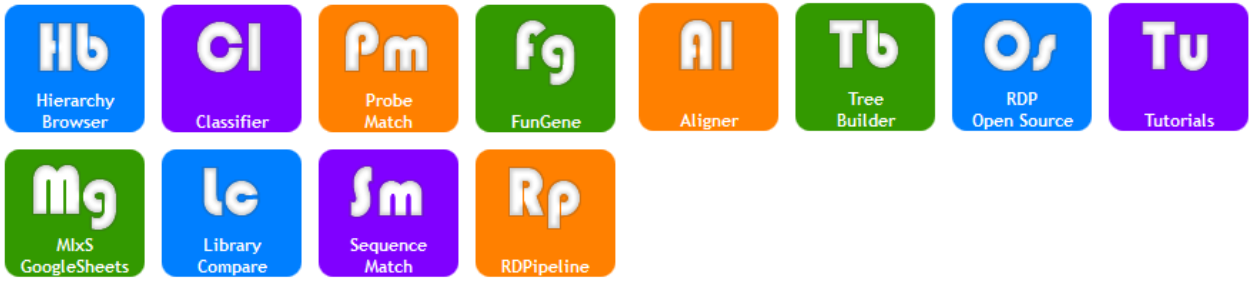


- **largest ☺**
- **too large ☹**
- **many errors in annotation ☹**

Curated databases

http://rdp.cme.msu.edu/

RDP – *Ribosomal Database Project*
provides quality-controlled, aligned and annotated Bacterial and Archaeal 16S rRNA sequences, and Fungal 28S rRNA sequences, and a suite of analysis tools to the scientific community



https://www.arb-silva.de/

SILVA
provides comprehensive, quality checked and regularly updated datasets of aligned small (16S/18S, SSU) and large subunit (23S/28S, LSU) ribosomal RNA (rRNA) sequences for all three domains of life (Bacteria, Archaea and Eukarya).

BLAST - local reference database

 # get reference fasta
wget http://www.biomed.cas.cz/mbu/lbwrf/16S_RDP_Release11.zip

 # unzip
unzip 16S_RDP_Release11.zip

 # make a BLAST database
makeblastdb -in 16S_RDP_Release11.fas -dbtype 'nucl' -out 16S_database

 # run BLAST
blastn -query final1.fasta -db 16S_database -out final1_16S_blastn.txt -evalue 1E-5 -outfmt 6 -num_threads 2

 # get the best hits
sort -t$'\t' -k1,1 -k12,12gr -k11,11g -k3,3gr final1_16S_blastn.txt | sort -u -k1,1 --merge > final1_16S_blastn_best.txt

Task: Compare number of records in final1_16S_blastn_best.txt with number of sequences in final1.fasta.

BLASTn output format 6

1. `qseqid` query or source (e.g., gene) sequence id
2. `sseqid` subject or target (e.g., reference genome) sequence id
3. `pident` percentage of identical matches
4. `length` alignment length (sequence overlap)
5. `mismatch` number of mismatches
6. `gapopen` number of gap openings
7. `qstart` start of alignment in query
8. `qend` end of alignment in query
9. `sstart` start of alignment in subject
10. `send` end of alignment in subject
11. `evalue` expect value
12. `bitscore` bit score

BLASTn output format 6

1. **qseqid** M03794:8:000000000-AJCUU:1:2114:9990:17907
2. **sseqid** Flavobacterium_sp._SRS18|…
3. **pident** 94.178
4. **length** 292
5. **mismatch** 17
6. **gapopen** 0
7. **qstart** 8
8. **qend** 299
9. **sstart** 434
10. **send** 143
11. **evalue** 7.35e-124
12. **bitscore** 446

E-value & Bit-score

E-value (*The smaller the E-value, the better the match*)
E-value is the number of expected hits of similar score that could be found just by chance. E-value of 10 means that up to 10 hits can be expected to be found just by chance, given the same size of a random database.
E-value can be used as a first quality filter for the BLAST search result, to obtain only results equal to or better than the number given by the -evalue option. Blast results are sorted by E-value by default (best hit in first line).

The E-value depends on the size of the used sequence database. Since large databases increase the chance of false positive hits, the E-value corrects for the higher chance. It's a correction for multiple comparisons. This means that a sequence hit would get a better E-value when present in a smaller database.

$$E = m \times n / 2^{\text{bit-score}}$$

m - query sequence length
n - total database length (sum of all sequences)

Bit-score (*The higher the bit-score, the better the sequence similarity*)
The bit-score is the requires size of a sequence database in which the current match could be found just by chance. The bit-score is a $\log_2$ scaled and normalized raw-score. Each increase by one doubles the required database size ($2^{\text{bit-score}}$).
Bit-score does not depend on database size. The bit-score gives the same value for hits in databases of different sizes and hence can be used for searching in an constantly increasing

awk

show hit (second column) information
awk -F'\t' '{print $2}' final1_16S_blastn_best.txt

show second part of the second column (two separators)
awk -F'\t' '{print $2}' final1_16S_blastn_best.txt | awk -F '[|;]' '{print $2}'

see a frequency of similarity scores (sort, uniq)
awk -F '\t' '{print $3}' final1_16S_blastn_best.txt | sort | uniq -c | sort –nr > sim.txt

Task: Create a file containing a list of frequencies of each class.

For Loop in bash shell

A ‘for loop’ is a bash programming language statement which allows code to be repeatedly executed. A for loop is classified as an iteration statement i.e. it is the repetition of a process within a bash script. For example, you can run UNIX command or task 5 times or read and process list of files using a for loop. A for loop can be used at a shell prompt or within a shell script itself.

```bash
#!/bin/bash
for i in 1 2 3 4 5
do
   echo "Welcome $i times"
done
```

```bash
#!/bin/bash
for i in {1..5}
do
   echo "Welcome $i times"
done
```

```bash
#!/bin/bash
for (( c=1; c<=5; c++ ))
do
   echo "Welcome $c times"
done
```

OUTPUT:

```
Welcome 1 times
Welcome 2 times
Welcome 3 times
Welcome 4 times
Welcome 5 times
```

```bash
for file in *.*
do
  echo "${file}"
done
```

Print all file names in current folder

```bash
for file in *.fastq
do
  echo "${file}"
done
```

Print names of all fastq file in current folder

https://www.cyberciti.biz/faq/bash-for-loop/

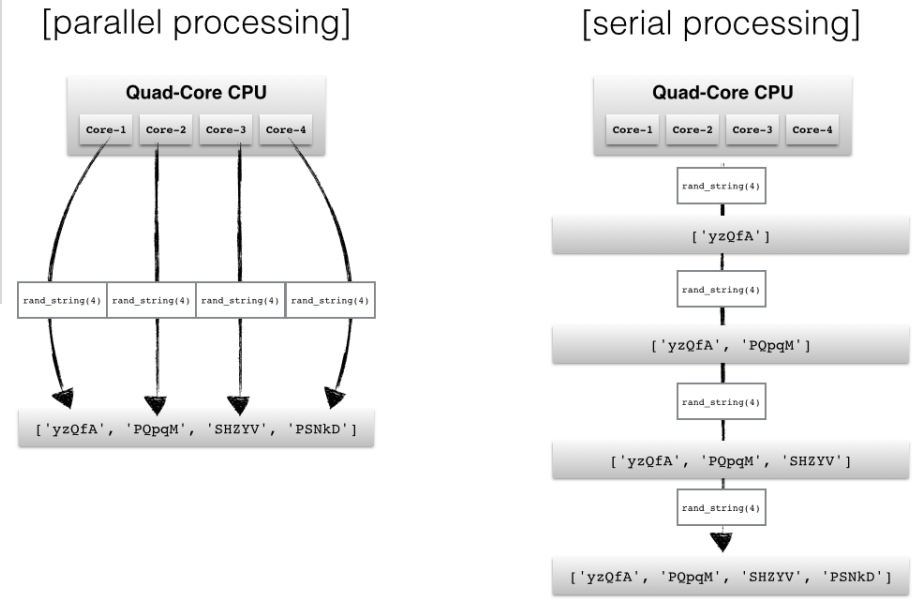
Run programs in parallel (parallel)

GNU parallel is a shell tool for executing jobs in parallel using one or more computers. A job can be a single command or a small script that has to be run for each of the lines in the input. The typical input is a list of files, a list of hosts, a list of users, a list of URLs, or a list of tables. A job can also be a command that reads from a pipe. GNU parallel can then split the input and pipe it into commands in parallel.

```
#prepare list of commands...
for i in {1..5}
do
 echo "echo 'This is some text ${i}' > file${i}.txt"
done > create_text_files.sh


#check commands list file...
less create_text_files.sh


#run it in parallel...
cat create_text_files.sh | parallel
```



BLUE – input variables/files
GREEN – output variables/files

https://www.gnu.org/software/parallel/
http://sebastianraschka.com/Articles/2014_multiprocessing.html