

Introduction to Linux

Adapted from Eric Bishop

27th April 2016

CIBIV mug shots



Introduction to the UNIX command line

- ▶ **Plan for today:**
 - ▶ What is Linux/UNIX?
 - ▶ How to use a shell
 - ▶ Navigating the file system
 - ▶ Working with files
 - ▶ Regular expressions
 - ▶ Running programs & file permissions



What is Linux?

- ▶ An operating system
- ▶ Open source
- ▶ Linux is a variant of Unix
 - ▶ So is Mac OS X, so much of this tutorial applies to Macs as well



Why Linux?

- ▶ Linux is **free**
- ▶ It's fully **customizable**
- ▶ It's **stable** (i.e. it almost never crashes)

- ▶ But: People are used to Windows/Mac! So why should they bother?

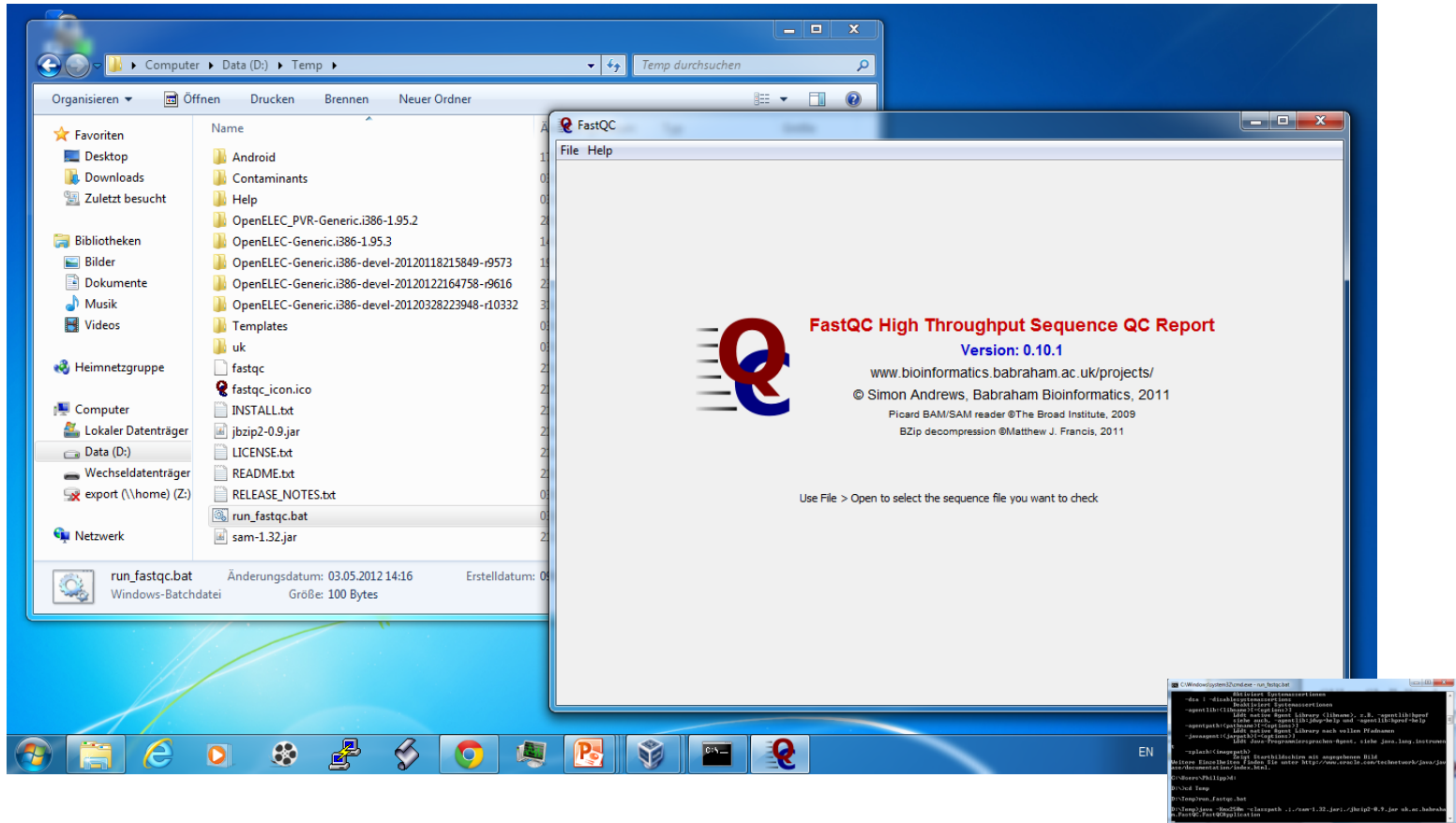


So why Linux?

- ▶ When working with sequencing data you don't really have another choice
 - ▶ It is possible to handle extremely **large files** without problems
 - ▶ Most of the **software** is developed/optimized for Linux
 - ▶ It is easy to run programs on **remote machines**



GUI vs. Shell: Windows



- ▶ Windows: focus on graphical user interface

GUI vs. Shell: Linux

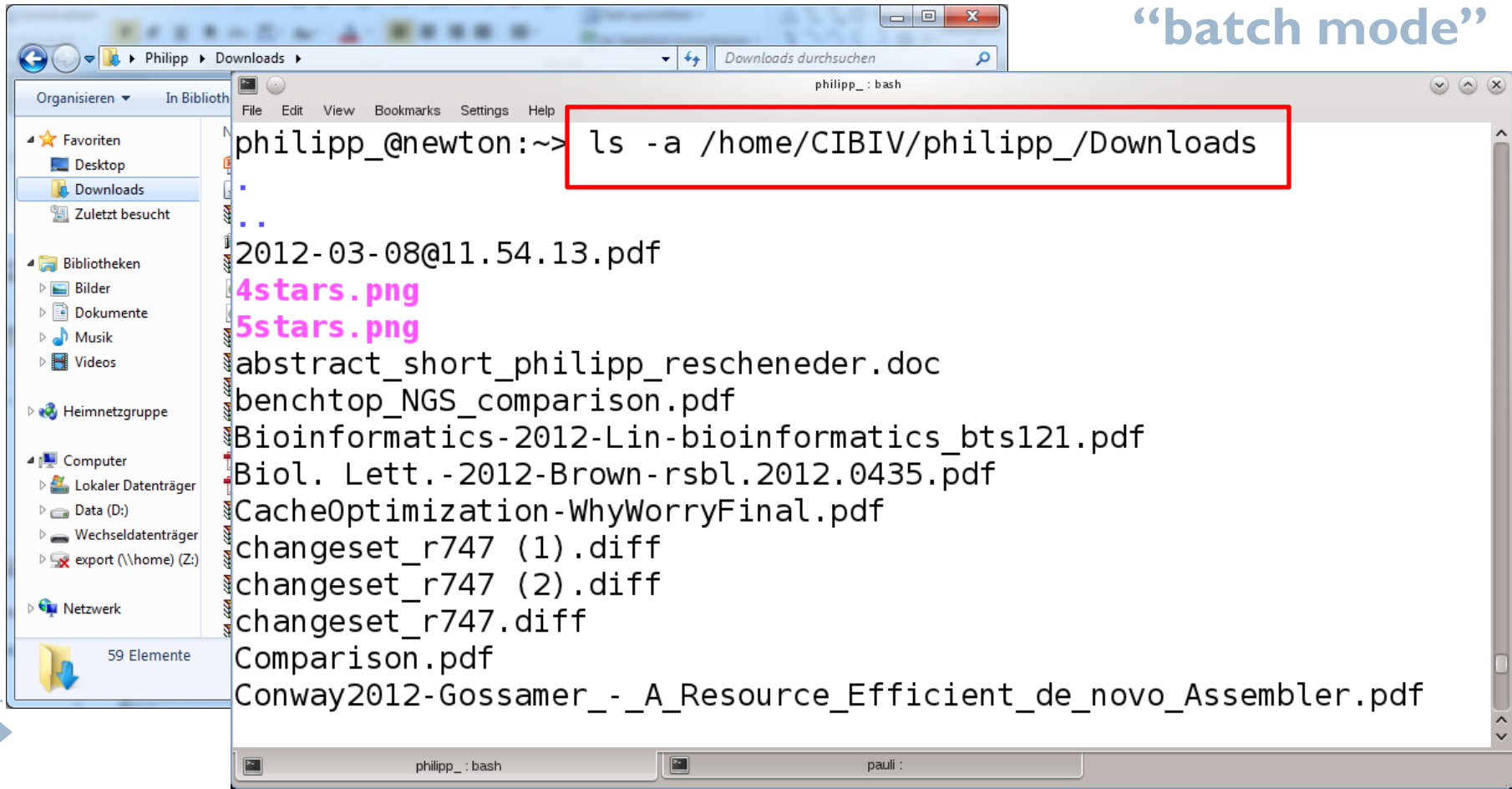
```
philipp@newton:/scratch/philipp> l
total 136
drwxr-xr-x 19 philipp_CIBIV 4.0K Aug 9 08:55 .
drwxrwxrwt 4 root root 34 Dec 21 2011 ..
drwxr-x--- 7 philipp_CIBIV 101 May 9 09:19 android-sdk-linux
-rw-r--r-- 1 philipp_CIBIV 29M Apr 9 20:00 android-sdk_r18-linux.tgz
drwxr-xr-x 4 philipp_CIBIV 31 Mar 5 10:29 backup
drwxr-xr-x 2 philipp_CIBIV 24 Aug 3 10:33 bin
drwxr-xr-x 7 philipp_CIBIV 77 Dec 21 2011 bin_backup
-rwxr-xr-x 1 philipp_CIBIV 3.9K Feb 28 17:06 callSNPs_test.sh
-rw-r--r-- 1 philipp_CIBIV 44K Aug 3 10:45 compmotives.tar.gz
drwxr-xr-x 9 philipp_CIBIV 4.0K Apr 2 08:37 eclipse
drwxr-xr-x 9 philipp_CIBIV 4.0K Apr 2 08:37 eclipse-cpp
drwxr-xr-x 9 philipp_CIBIV 4.0K Aug 8 08:22 eclipse-x64
drwxr-x--- 7 philipp_CIBIV 4.0K Feb 27 20:27 e-coli-k12-resequencing-1.3.0
-rw-r--r-- 1 philipp_CIBIV 768M Mar 2 13:26 e-coli-k12-resequencing-1.3.0.tgz.crdownload
-rw-r--r-- 1 philipp_CIBIV 4.8G Aug 3 12:31 ERR039480.fastq.gz
drwxr-xr-x 7 philipp_CIBIV 64 Mar 2 10:42 gdb
drwxr-xr-x 15 philipp_CIBIV 4.0K Mar 2 10:30 gdb-7.4
-rw-r--r-- 1 philipp_CIBIV 27M Jan 24 2012 gdb-7.4.tar.gz
drwxr-xr-x 2 philipp_CIBIV 66 Apr 27 09:56 lib64
drwxr-xr-x 5 philipp_CIBIV 74 Dec 21 2011 ngm-0.2.0
drwxr-xr-x 6 philipp_CIBIV 4.0K May 21 08:45 NGM_backup_20120521
drwxr-xr-x 3 philipp_CIBIV 17 Jul 10 08:54 ngm-core-backup
-rw-r--r-- 1 philipp_CIBIV 161K Dec 21 2011 ngm-src-0.2.0.tar.bz2
-rwxr-xr-x 1 philipp_CIBIV 74K Aug 10 2011 NGM_TestSuite
-rw-r--r-- 1 philipp_CIBIV 1.6G Aug 9 08:25 reads.fa
drwxr-xr-x 3 philipp_CIBIV 118 Aug 8 11:42 r-kurs
drwxr-xr-x 3 philipp_CIBIV 43 Oct 20 2011 simulated_data
-rw-r--r-- 1 philipp_CIBIV 1.3K Apr 2 09:19 test.fa
-rw-r--r-- 1 philipp_CIBIV 2.3G Aug 9 09:03 test.sam
drwxr-xr-x 5 philipp_CIBIV 4.0K Jun 28 11:34 tmp
philipp@newton:/scratch/philipp>
```

- ▶ Linux: focus on command line interface (GUI just an „Addon“)

GUI vs. Shell

- ▶ Show (hidden) files in Download folder

“Interactive”



“batch mode”

```
philipp_@newton:~> ls -a /home/CIBIV/philipp_/Downloads
.
..
2012-03-08@11.54.13.pdf
4stars.png
5stars.png
abstract_short_philipp_rescheneder.doc
benchtop_NGS_comparison.pdf
Bioinformatics-2012-Lin-bioinformatics_bts121.pdf
Biol. Lett.-2012-Brown-rsbl.2012.0435.pdf
CacheOptimization-WhyWorryFinal.pdf
changeset_r747 (1).diff
changeset_r747 (2).diff
changeset_r747.diff
Comparison.pdf
Conway2012-Gossamer_-_A_Resource_Efficient_de_novo_Assembler.pdf
```

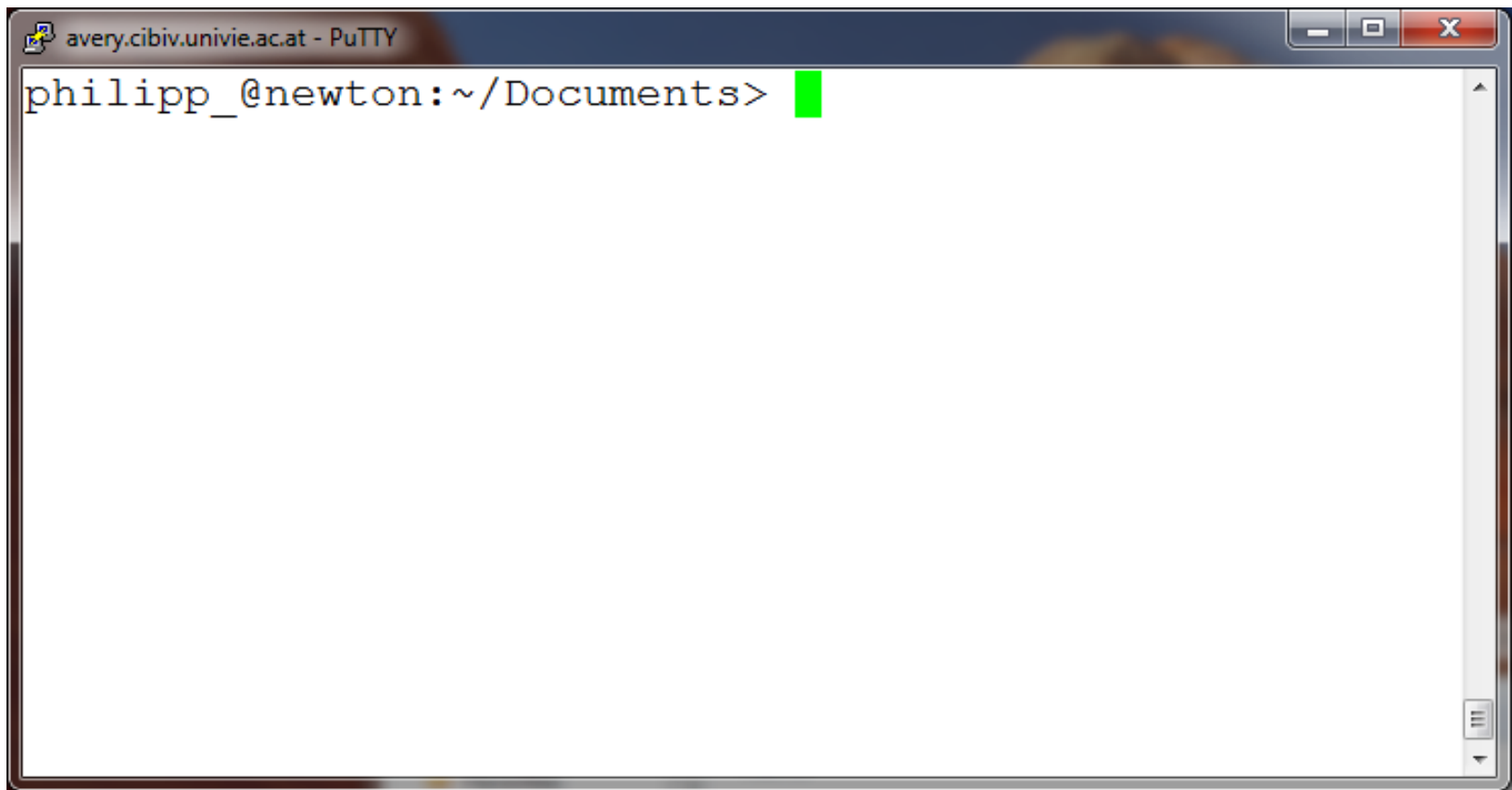
What exactly is a “shell”?

- ▶ The shell interprets commands the user types and manages their execution
 - ▶ The shell communicates with the internal part of the operating system called the **kernel**
 - ▶ The most popular shells are: tcsh, csh, korn, and **bash**
 - ▶ The differences are most times subtle
 - ▶ For this tutorial, we are using bash
- ▶ Shell commands are **CASE SENSITIVE!**



Connecting to a Unix/Linux system

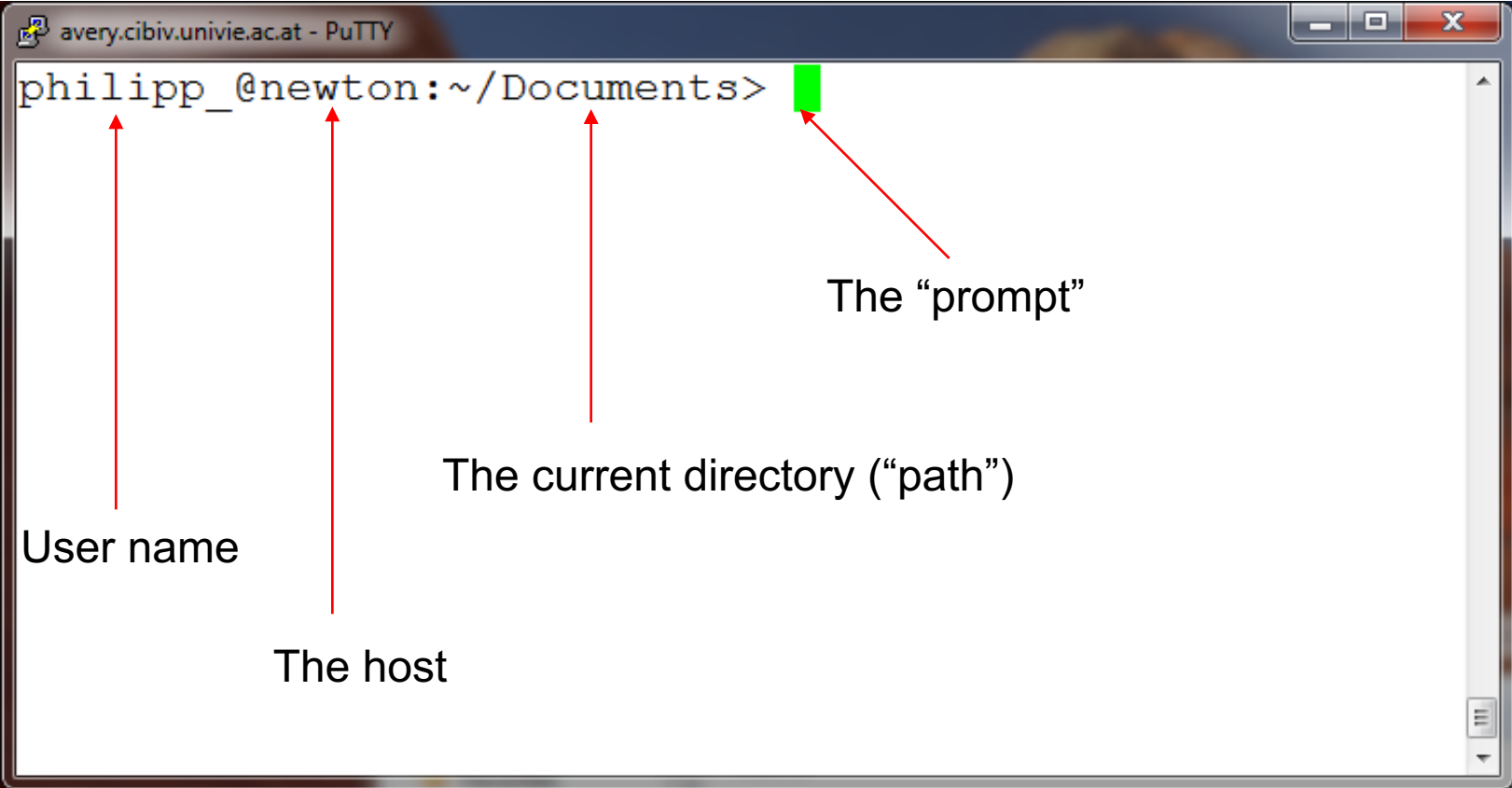
- ▶ Open up a terminal:



The image shows a screenshot of a PuTTY terminal window. The title bar at the top reads "avery.cibiv.univie.ac.at - PuTTY". The terminal content displays the shell prompt "philipp_@newton:~/Documents>" followed by a green cursor block. The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner and a scroll bar on the right side.

Connecting to a Unix/Linux system

- ▶ Open up a terminal:



The screenshot shows a PuTTY terminal window titled "avery.cibiv.univie.ac.at - PuTTY". The terminal displays the shell prompt "philipp_@newton:~/Documents>". A green cursor is positioned at the end of the prompt. Four red arrows point from text labels to parts of the prompt: "User name" points to "philipp_", "The host" points to "@newton", "The current directory ('path')" points to "~/Documents", and "The 'prompt'" points to ">".

```
philipp_@newton:~/Documents>
```

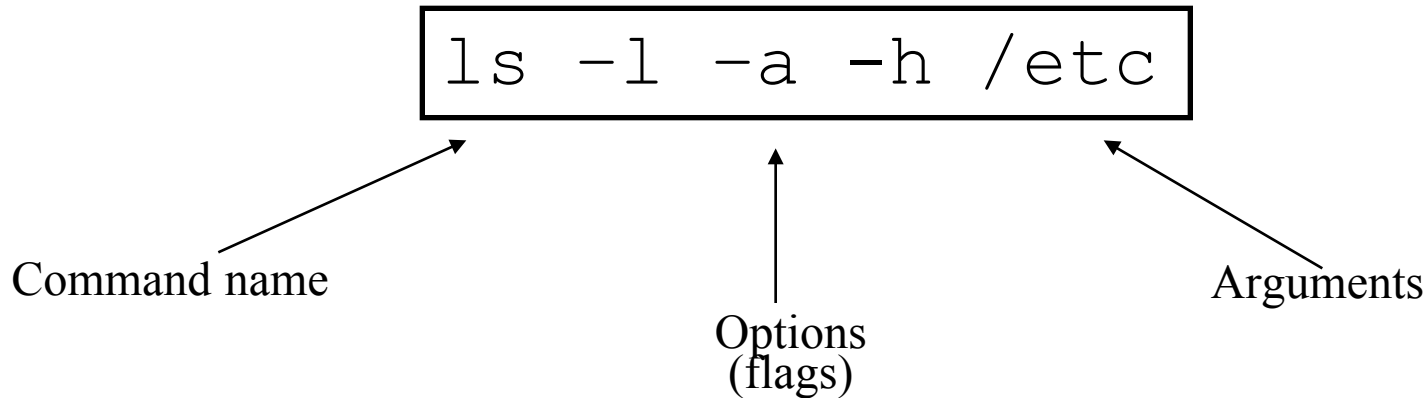
User name

The host

The current directory ("path")

The "prompt"

Executing a command



▶ Combine options:

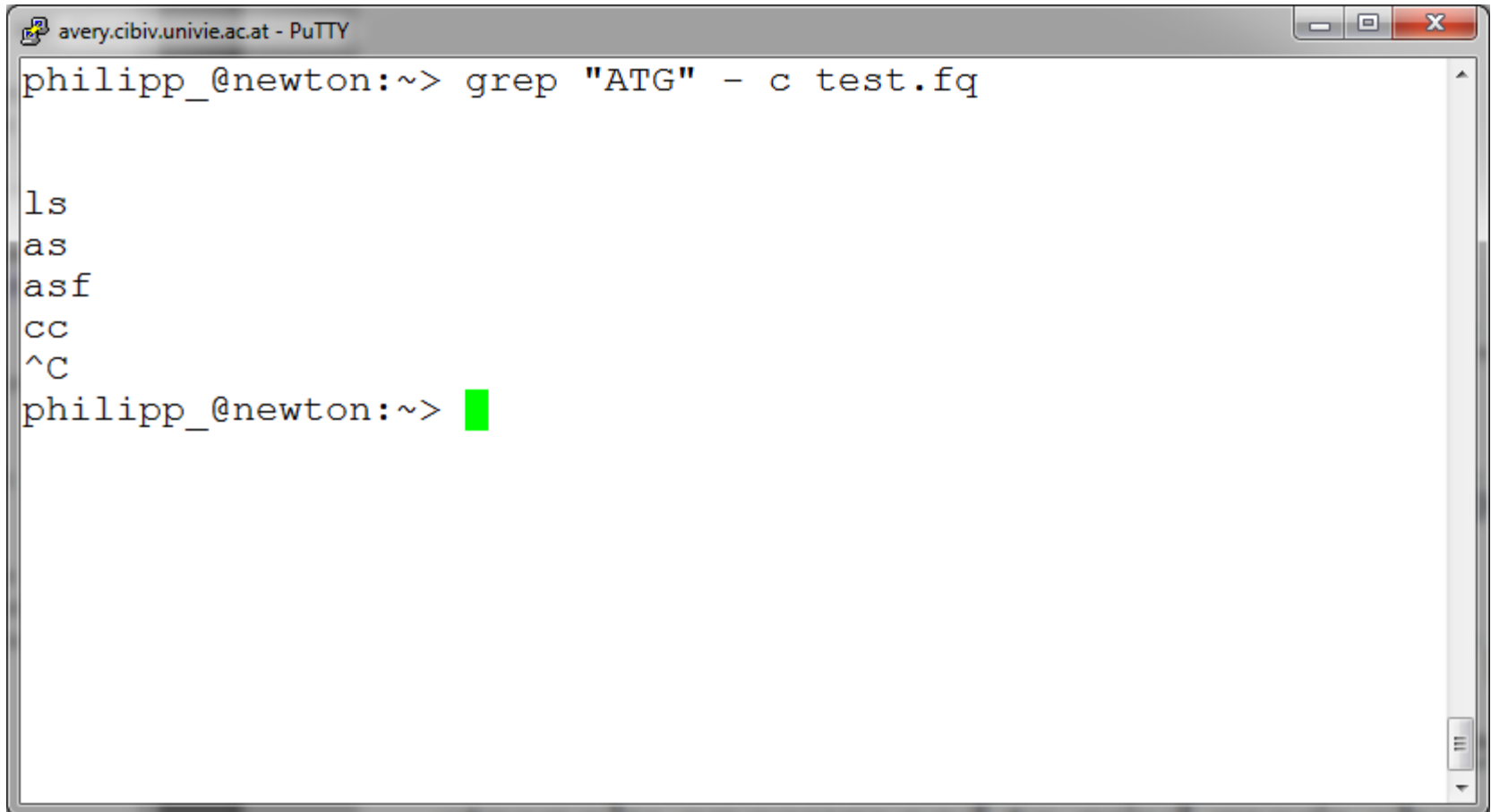
```
ls -lah /etc
```

▶ Long options:

```
ls -l --all -human-readable /etc
```



Terminating a running command



The screenshot shows a PuTTY terminal window titled "avery.cibiv.univie.ac.at - PuTTY". The prompt is "philipp_@newton:~>". The user has entered the command "grep 'ATG' -c test.fq". The terminal output shows the command being executed, followed by the output "ls", "as", "asf", "cc", and "^C". The prompt "philipp_@newton:~>" is followed by a green cursor block, indicating that the command has been terminated.

```
philipp_@newton:~> grep "ATG" -c test.fq

ls
as
asf
cc
^C
philipp_@newton:~> █
```

- ▶ Press Ctrl-C to terminate the command!



Help!

- ▶ Endless number of commands/programs and parameters
- ▶ Whenever you need help with a command:
 - ▶ run the command with `-h` or `-help` as parameter
 - ▶ type “man” and the command name
 - ▶ ask Google!!
 - ▶ <http://stackoverflow.com/>



Stackoverflow

StackExchange

sign up log in tour help

Search Q&A



Questions Jobs Tags Users Badges Ask Question

Stack Overflow is a community of 4.7 million programmers, just like you, helping each other.

Join them; it only takes a minute:

Sign up

Join the Stack Overflow community to:



Ask programming questions



Answer and help your peers



Get recognized for your expertise

Top Questions

interesting 389 featured hot week month

5 votes

1 answer

16 views

Frama-c Magnesium : Unable to execute WP plugin on Windows

c windows frama-c

modified 35 secs ago user2888308 190

8 votes

3 answers

499 views

O(N) Identification of Permutations

c++ big-o permutation string-comparison standard-library

modified 38 secs ago Jonathan Mee 9,327

Jobs near you

Python Developer

Indoo.rs GmbH Wien, Austria

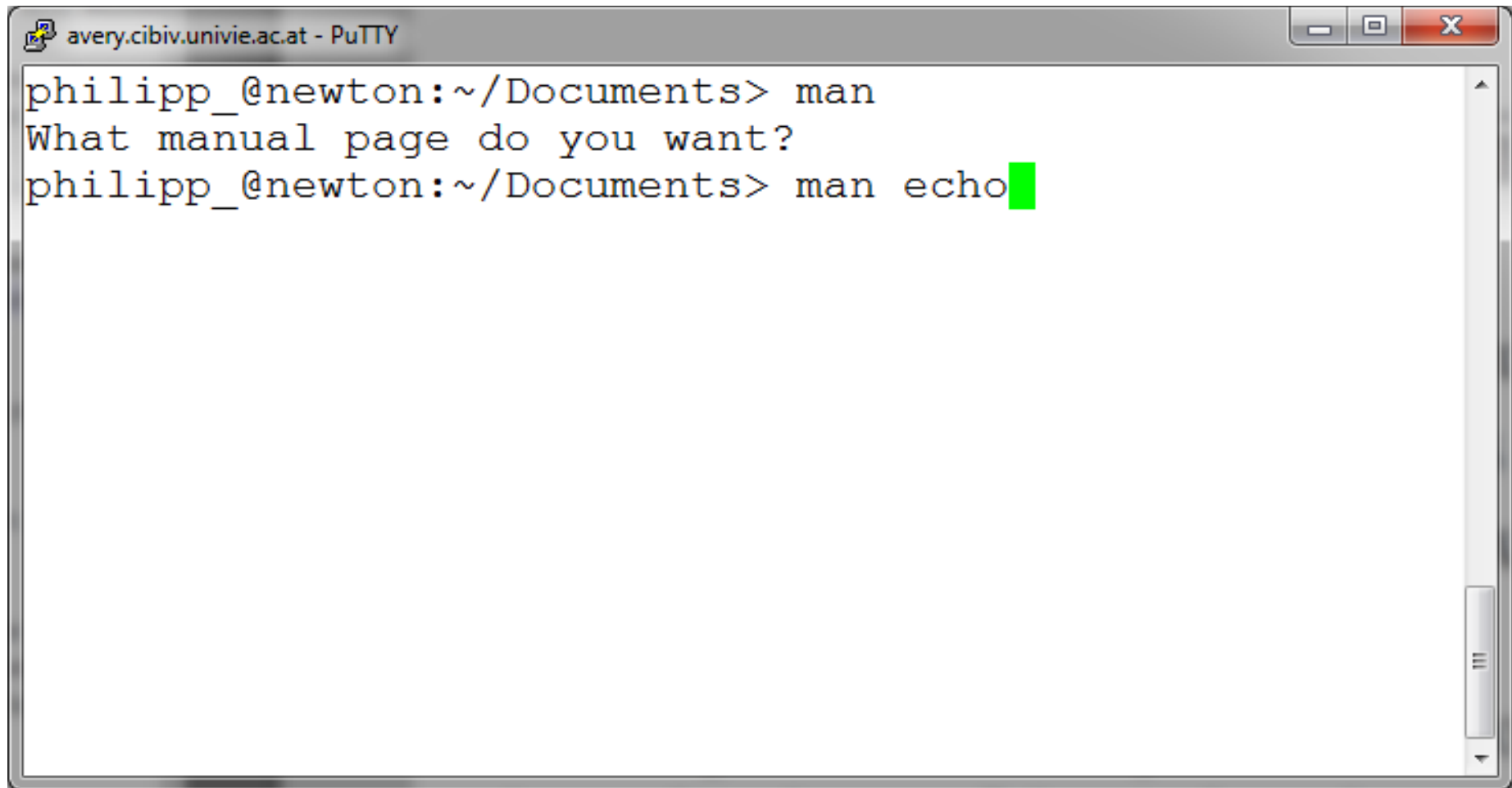
€38,000 - €45,000

python git

Web Application Entwickler (m/w)

IMS Nanofabrication AG Wien, Österreich

Help!



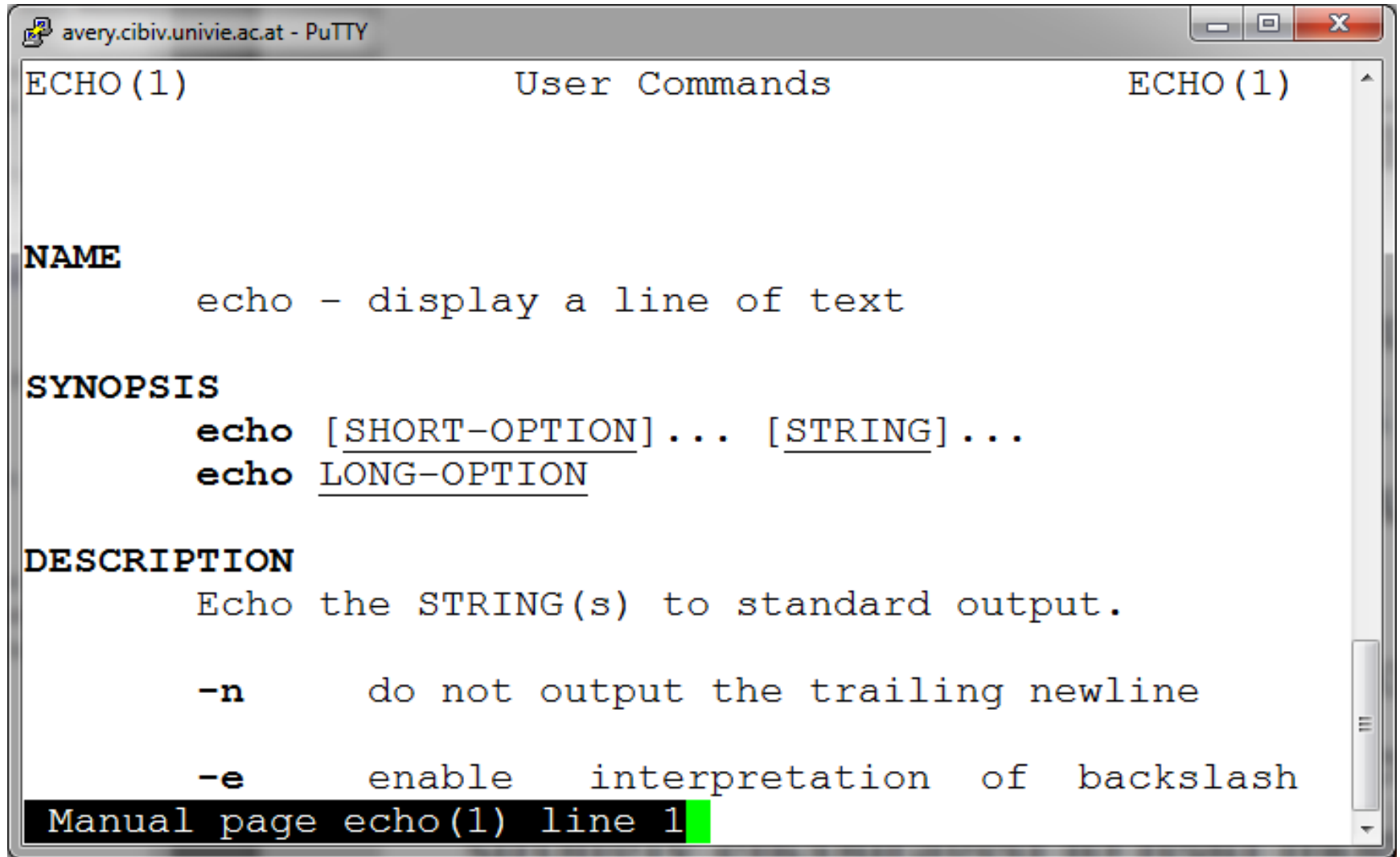
The image shows a PuTTY terminal window titled "avery.cibiv.univie.ac.at - PuTTY". The terminal content is as follows:

```
philipp_@newton:~/Documents> man
What manual page do you want?
philipp_@newton:~/Documents> man echo
```

The cursor is positioned at the end of the second line, indicated by a green block.



Help!



```
avery.cibiv.univie.ac.at - PuTTY
ECHO (1)                               User Commands                               ECHO (1)

NAME
    echo - display a line of text

SYNOPSIS
    echo [SHORT-OPTION]... [STRING]...
    echo LONG-OPTION

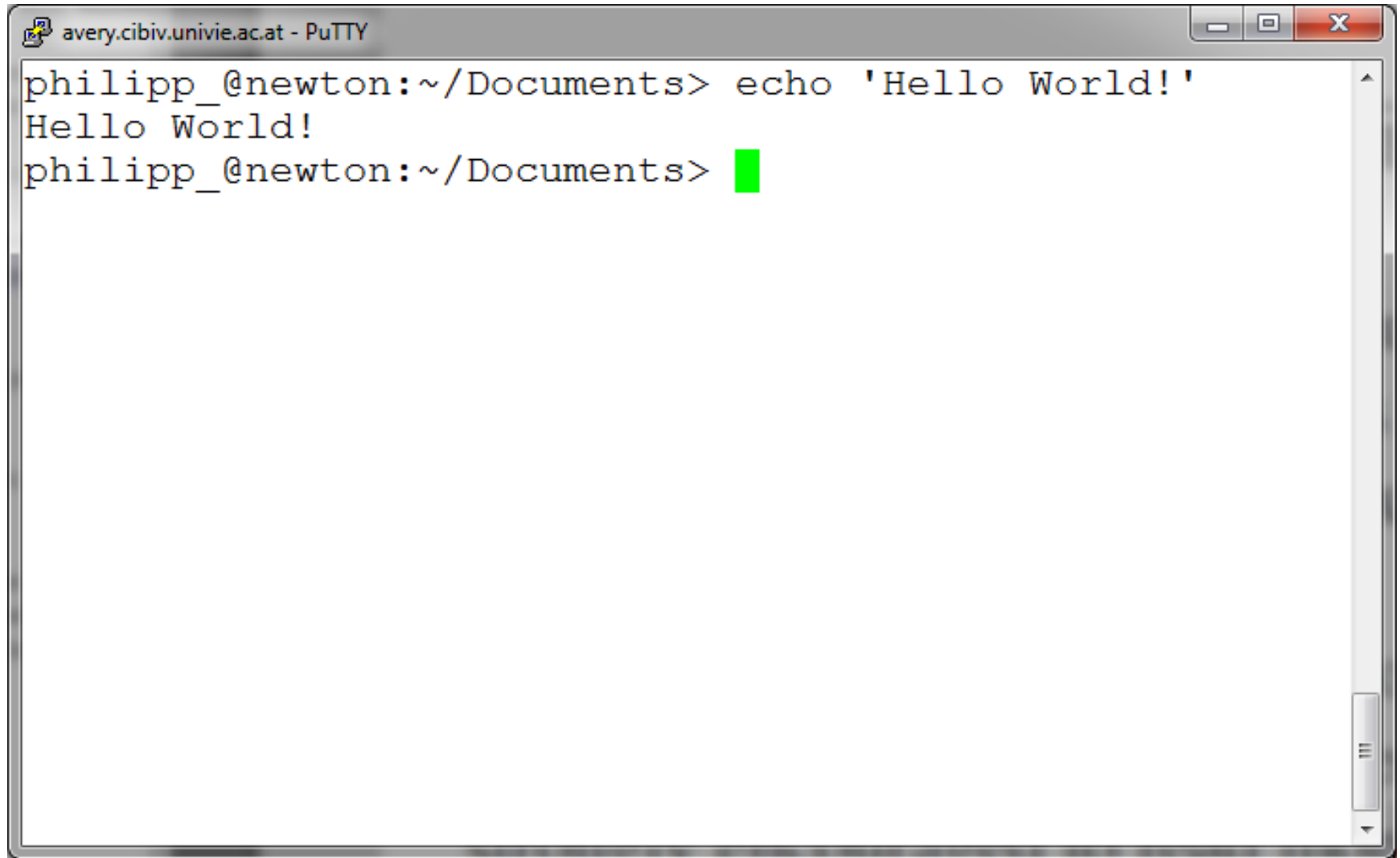
DESCRIPTION
    Echo the STRING(s) to standard output.

    -n      do not output the trailing newline

    -e      enable interpretation of backslash

Manual page echo(1) line 1
```

Help!



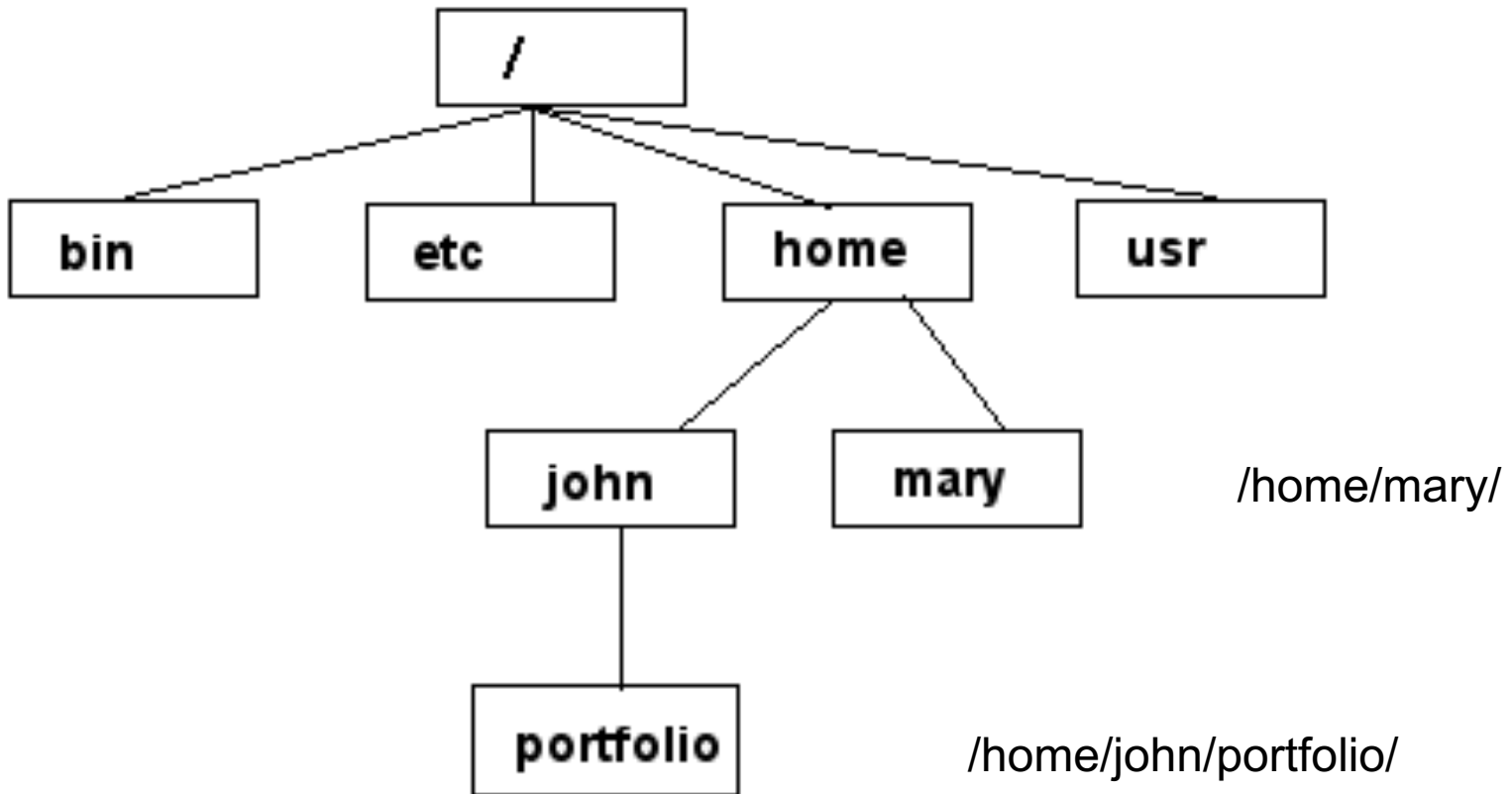
The image shows a PuTTY terminal window titled "avery.cibiv.univie.ac.at - PuTTY". The terminal displays the following text:

```
philipp_@newton:~/Documents> echo 'Hello World!'  
Hello World!  
philipp_@newton:~/Documents> █
```

The prompt "philipp_@newton:~/Documents>" is shown on two lines. The first line shows the command "echo 'Hello World!'" being entered. The second line shows the output "Hello World!". The third line shows the prompt again with a green cursor block (█) indicating the end of the input.

Unix/Linux File System

NOTE: Unix file names are **CASE SENSITIVE!**



The Path

File System

- ▶ **Absolute path**

- ▶ `ls /home/john/portfolio`

- ▶ **Relative path**

- ▶ `ls portfolio` or `ls ./portfolio`

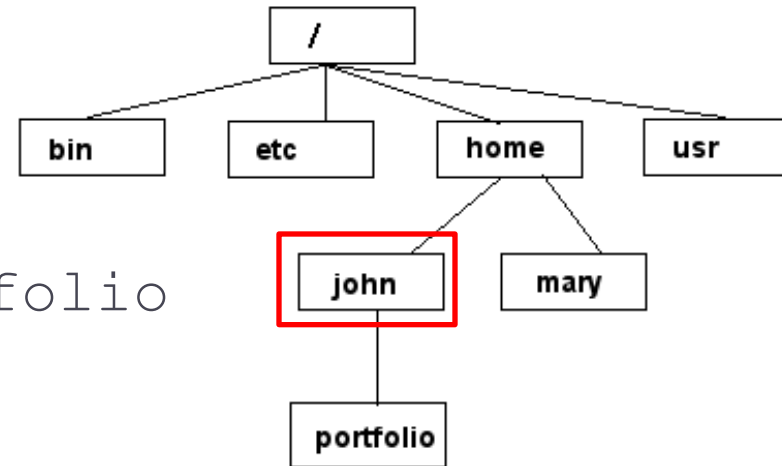
- ▶ **Shortcuts**

- ▶ Parent directory (`..`), home directory (`~`), last directory (`-`)

- ▶ **Wildcards**

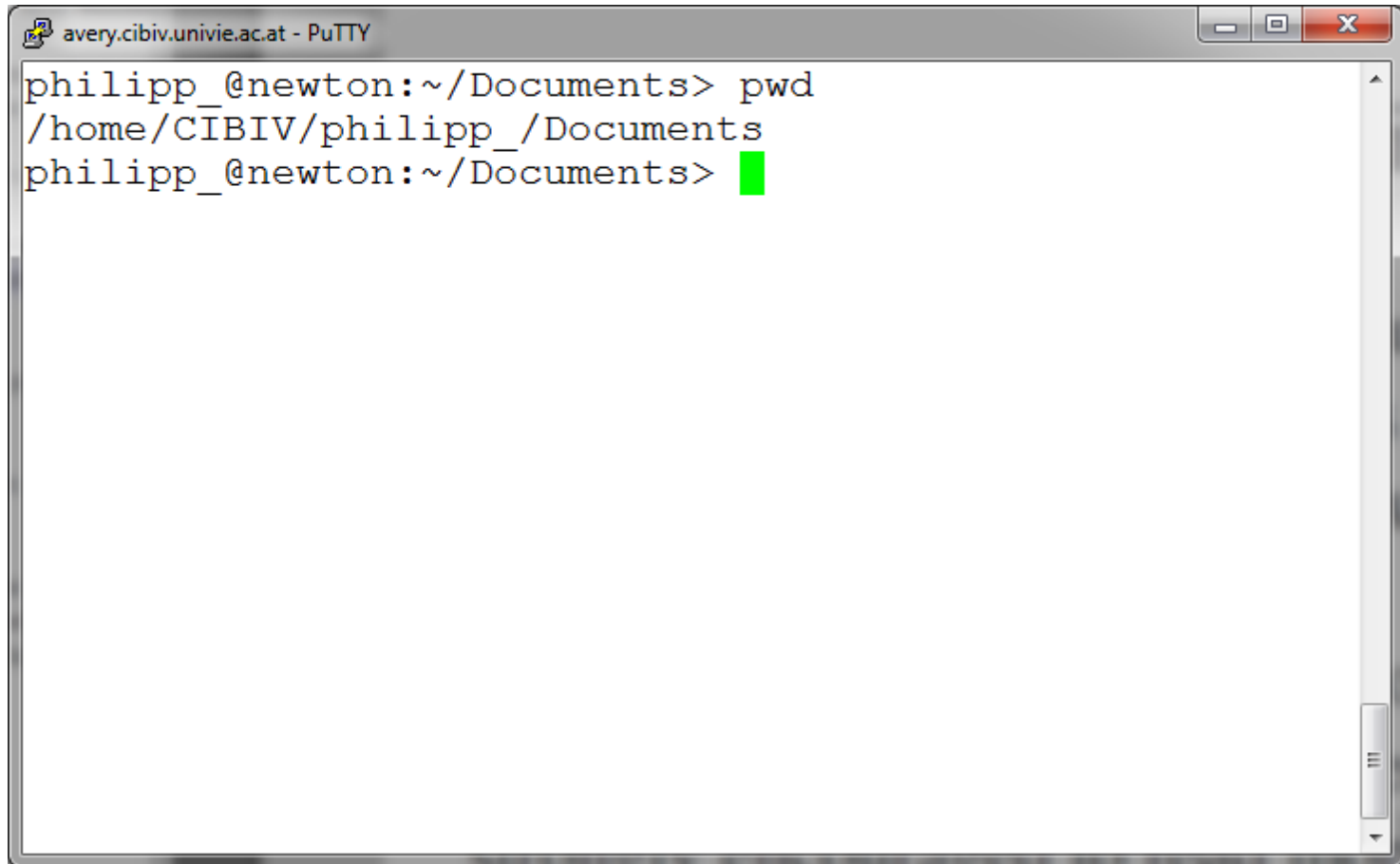
- ▶ Zero or more characters (`*`), exactly one character (`?`)

- ▶ **Use** tab-completion and **avoid** spaces in file names



Command: pwd (print working directory)

- ▶ To find your current path use “pwd”

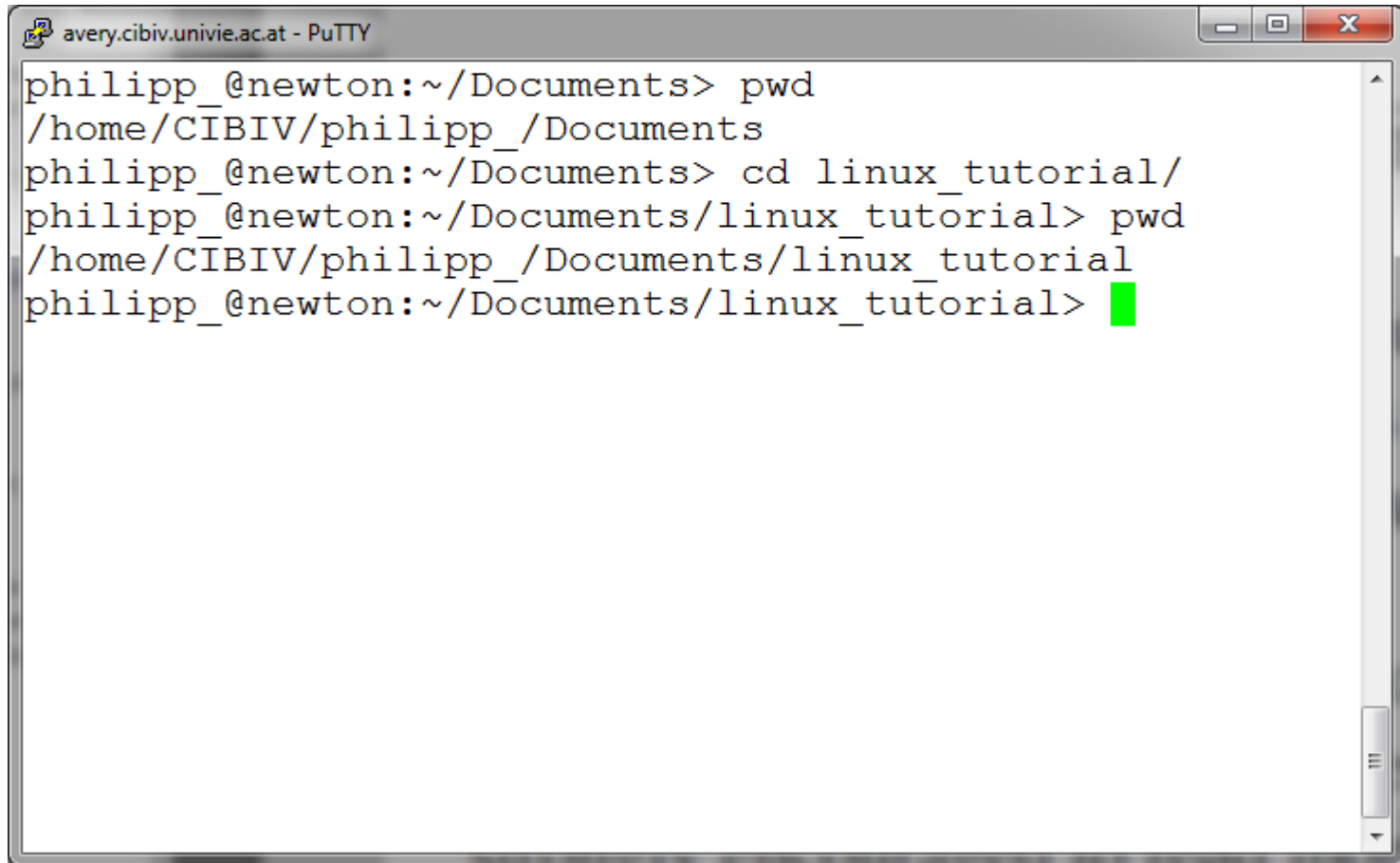


```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/Documents> pwd
/home/CIBIV/philipp_/Documents
philipp_@newton:~/Documents> █
```



Command: cd (change directory)

- ▶ To change to a specific directory use “cd”

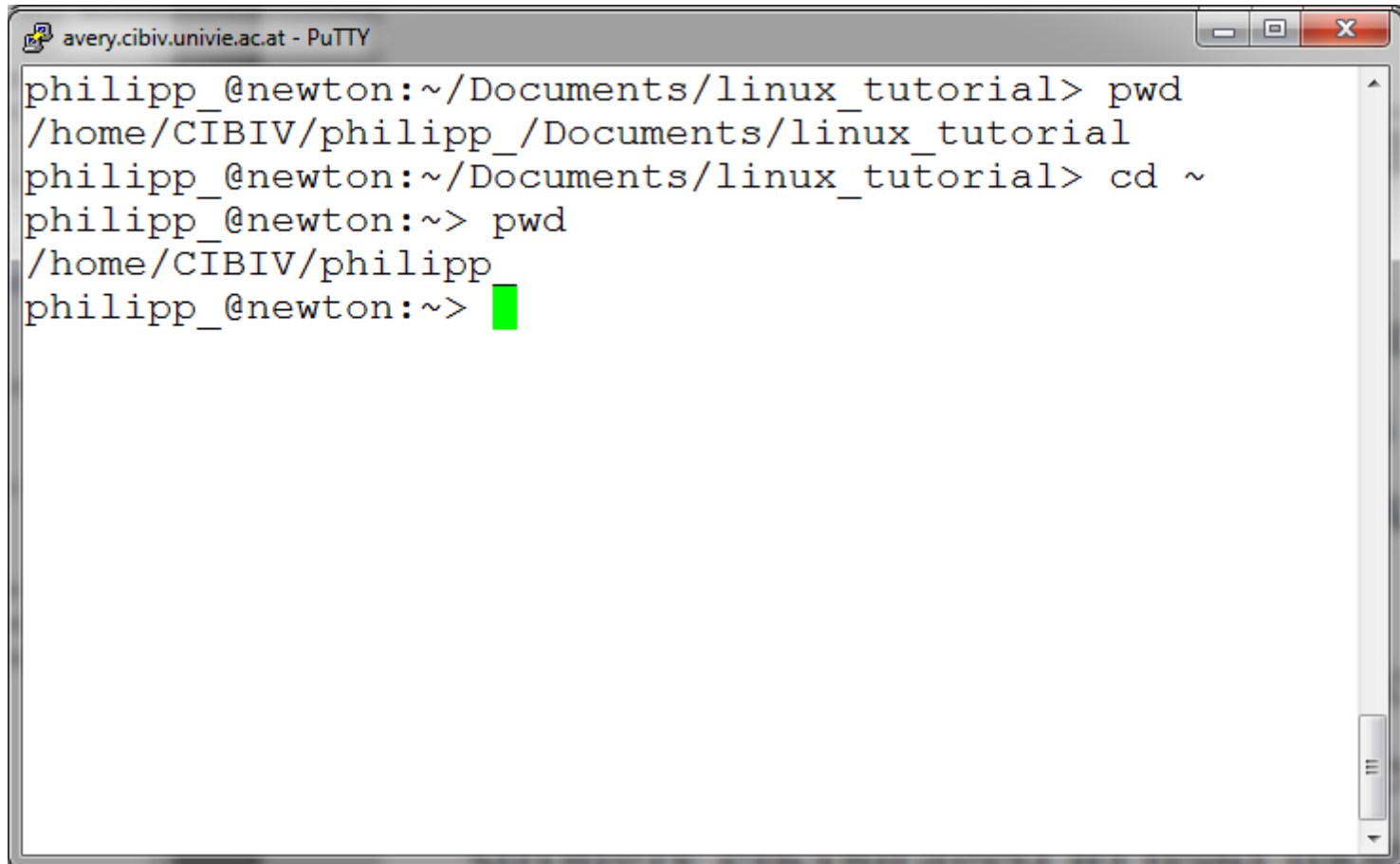
A screenshot of a PuTTY terminal window titled "avery.cibiv.univie.ac.at - PuTTY". The terminal shows a user named philipp_@newton navigating through directories. The user starts in the ~/Documents directory, runs 'pwd' to confirm the path, then runs 'cd linux_tutorial/' to move into a subdirectory. A second 'pwd' command confirms the new directory path. The prompt ends with a green cursor.

```
philipp_@newton:~/Documents> pwd
/home/CIBIV/philipp_/Documents
philipp_@newton:~/Documents> cd linux_tutorial/
philipp_@newton:~/Documents/linux_tutorial> pwd
/home/CIBIV/philipp_/Documents/linux_tutorial
philipp_@newton:~/Documents/linux_tutorial> █
```



Command: cd

- ▶ “~” is the location of your home directory

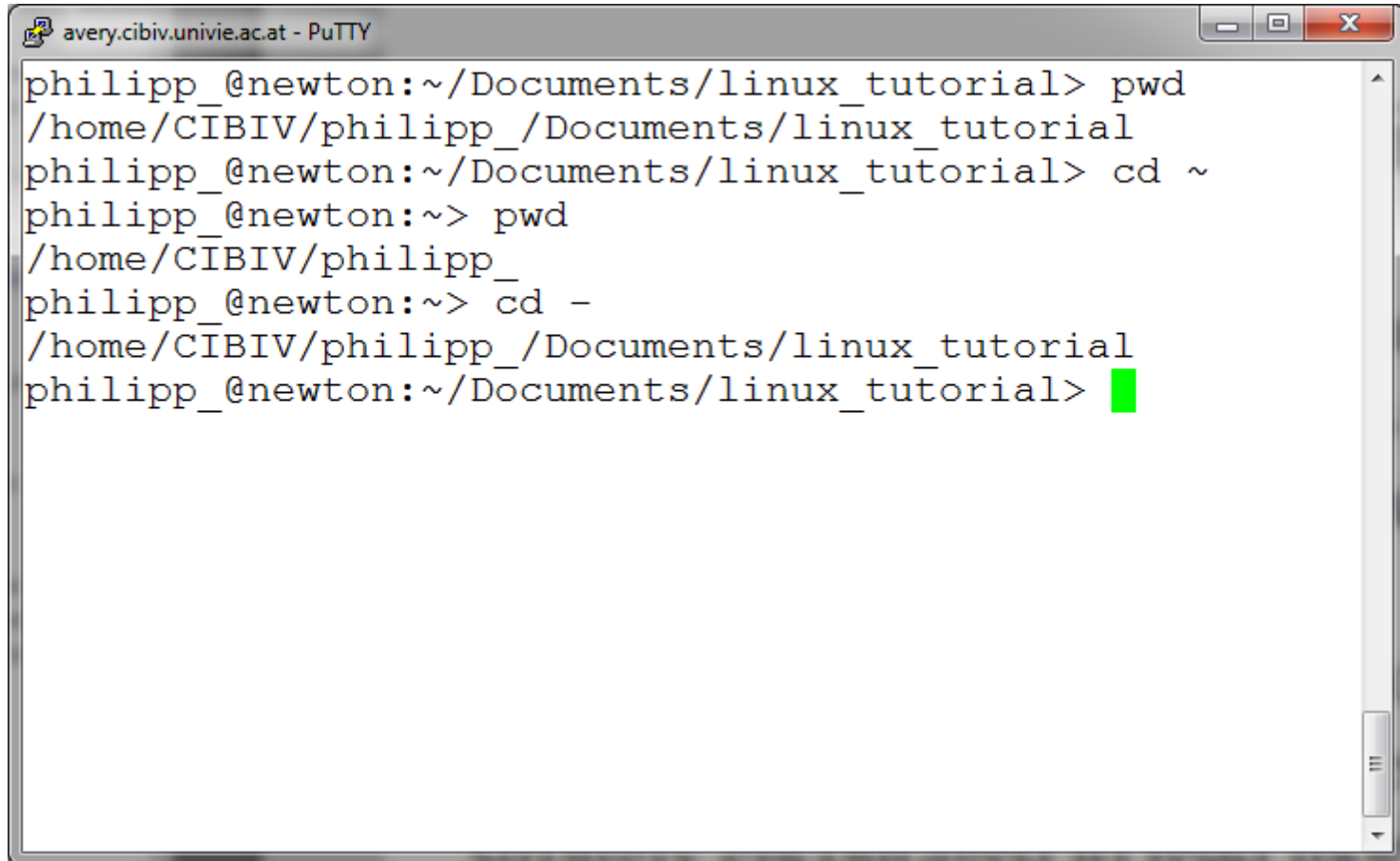


```
avery.cibiv.univie.ac.at - PuTTY
philipp_newton:~/Documents/linux_tutorial> pwd
/home/CIBIV/philipp/Documents/linux_tutorial
philipp_newton:~/Documents/linux_tutorial> cd ~
philipp_newton:~> pwd
/home/CIBIV/philipp
philipp_newton:~> █
```



Command: cd

- ▶ “-” is the location of the last directory

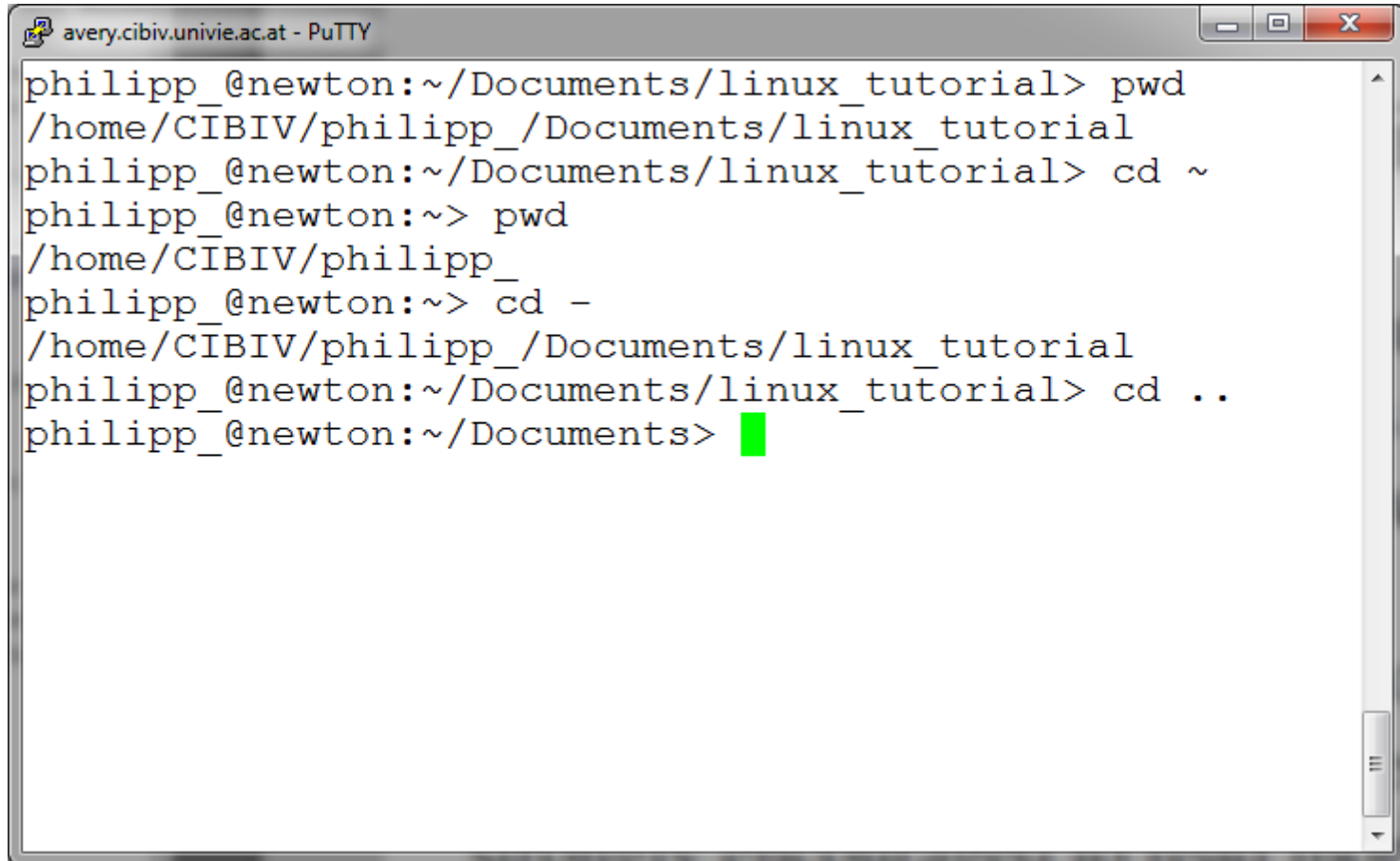


```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/Documents/linux_tutorial> pwd
/home/CIBIV/philipp_/Documents/linux_tutorial
philipp_@newton:~/Documents/linux_tutorial> cd ~
philipp_@newton:~> pwd
/home/CIBIV/philipp_
philipp_@newton:~> cd -
/home/CIBIV/philipp_/Documents/linux_tutorial
philipp_@newton:~/Documents/linux_tutorial> █
```



Command: cd

- ▶ “..” is the location of the parent directory

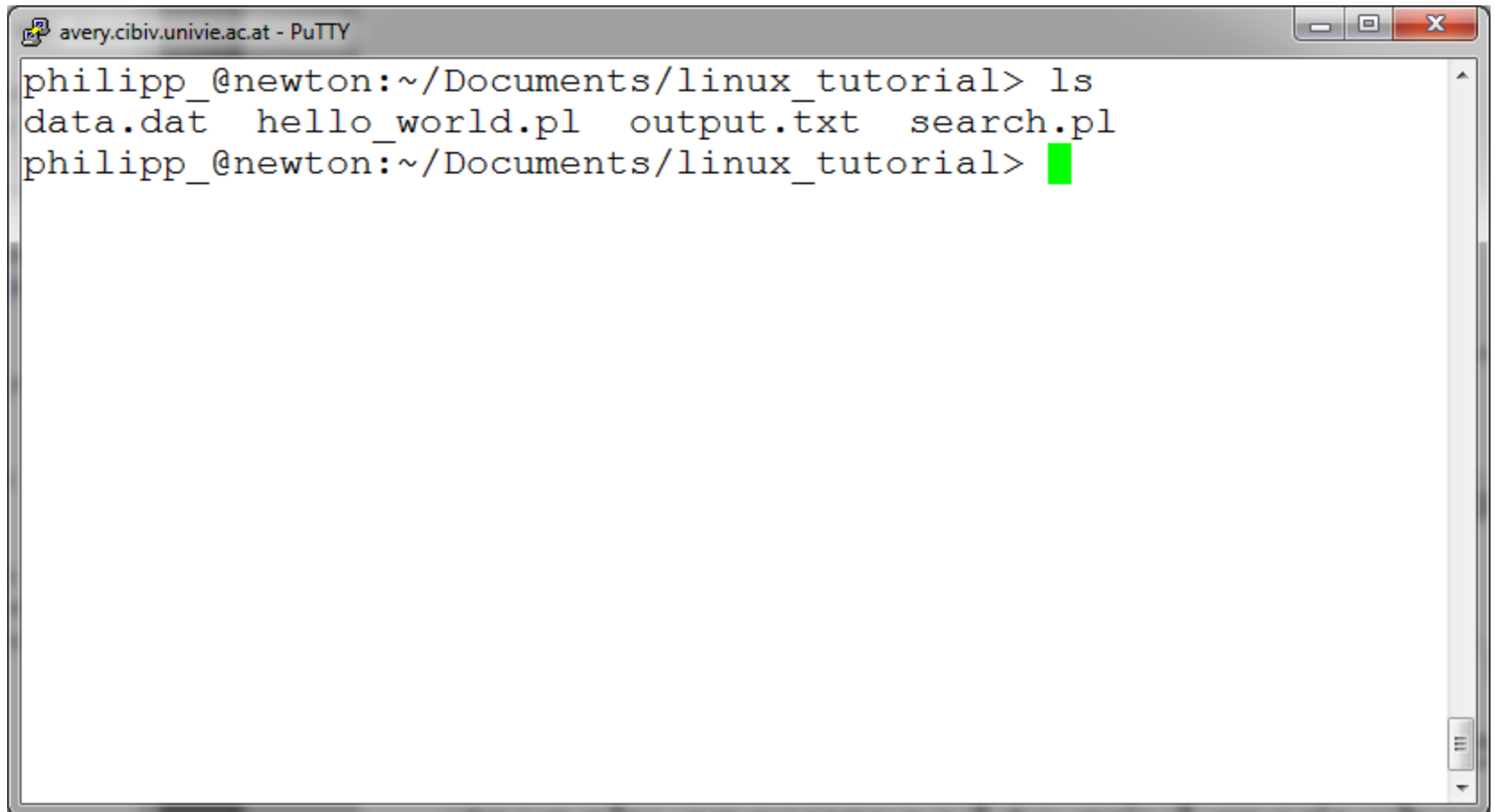


```
avery.cibiv.univie.ac.at - PuTTY
philipp_newton:~/Documents/linux_tutorial> pwd
/home/CIBIV/philipp_/Documents/linux_tutorial
philipp_newton:~/Documents/linux_tutorial> cd ~
philipp_newton:~> pwd
/home/CIBIV/philipp_
philipp_newton:~> cd -
/home/CIBIV/philipp_/Documents/linux_tutorial
philipp_newton:~/Documents/linux_tutorial> cd ..
philipp_newton:~/Documents>
```



Command: ls

- ▶ To list the files in the current directory use “ls”



The screenshot shows a PuTTY terminal window titled "avery.cibiv.univie.ac.at - PuTTY". The terminal displays the following text:

```
philipp_@newton:~/Documents/linux_tutorial> ls  
data.dat  hello_world.pl  output.txt  search.pl  
philipp_@newton:~/Documents/linux_tutorial> █
```

The terminal output shows the command 'ls' being executed, resulting in a list of files: 'data.dat', 'hello_world.pl', 'output.txt', and 'search.pl'. A green cursor is visible on the line following the command.

Command: ls

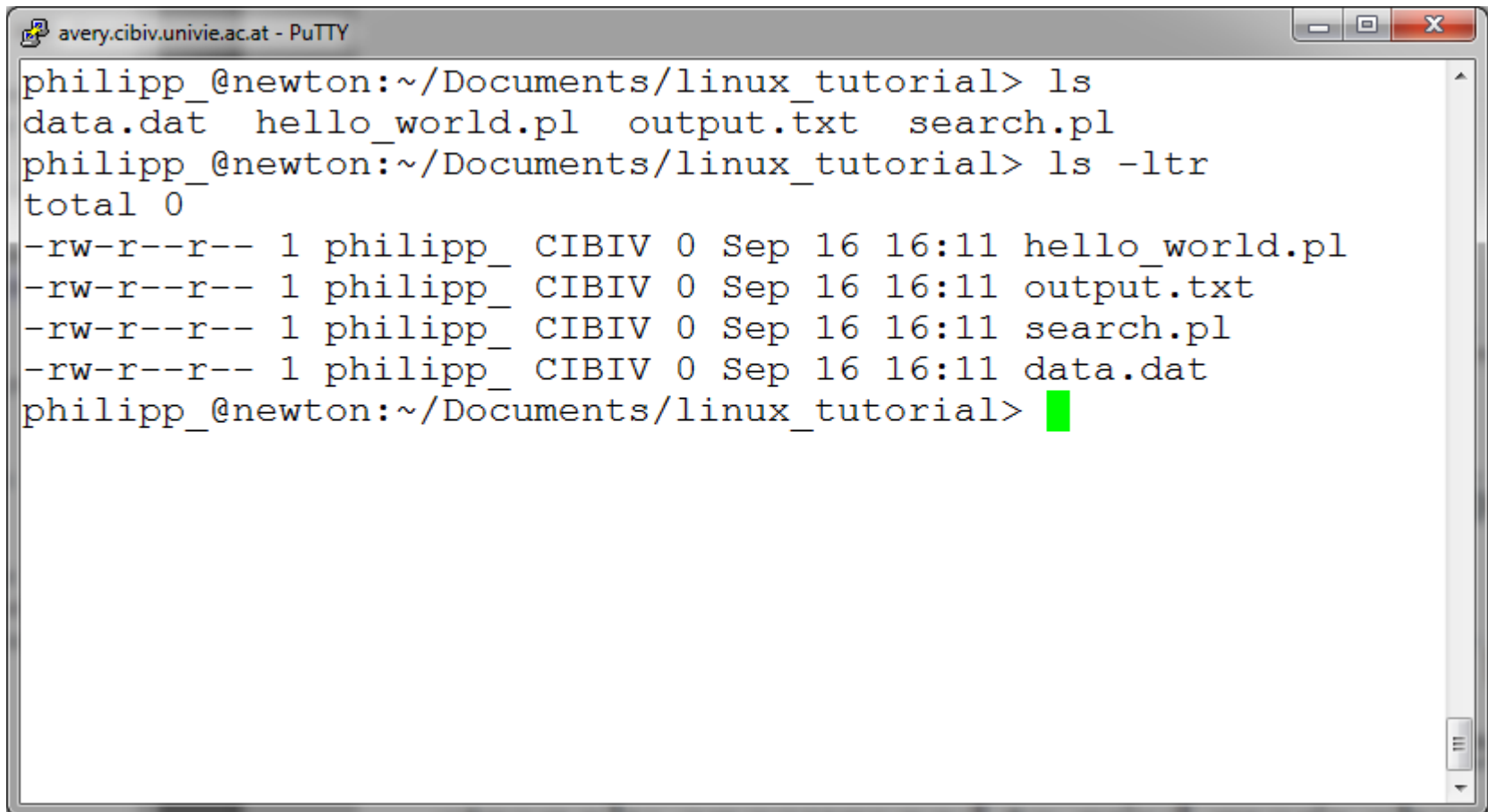
- ▶ **ls** has many options
 - ▶ `-l` long list (displays lots of info)
 - ▶ `-t` sort by modification time
 - ▶ `-S` sort by size
 - ▶ `-h` list file sizes in human readable format
 - ▶ `-r` reverse the order

- ▶ “`man ls`” for more options
- ▶ Options can be combined: “`ls -ltr`”



Command: ls -ltr

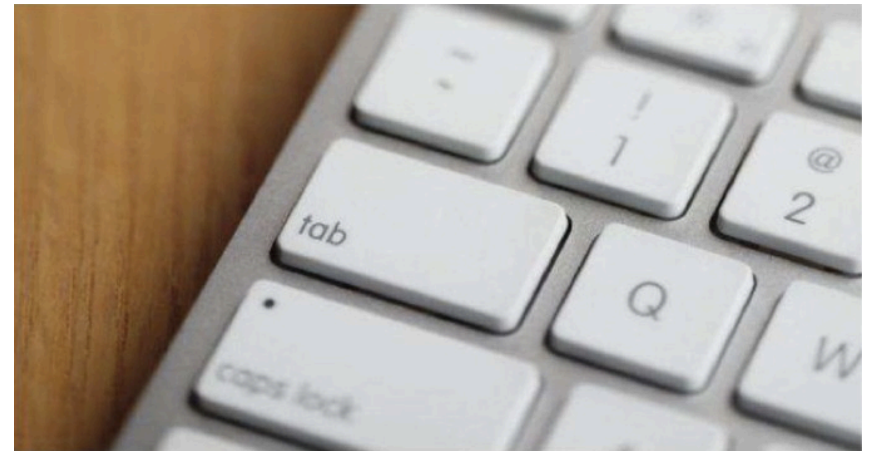
- ▶ List files by time in reverse order with long listing



```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/Documents/linux_tutorial> ls
data.dat  hello_world.pl  output.txt  search.pl
philipp_@newton:~/Documents/linux_tutorial> ls -ltr
total 0
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 hello_world.pl
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 output.txt
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 search.pl
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 data.dat
philipp_@newton:~/Documents/linux_tutorial> █
```

Tabulator key

The most important key when working with a shell

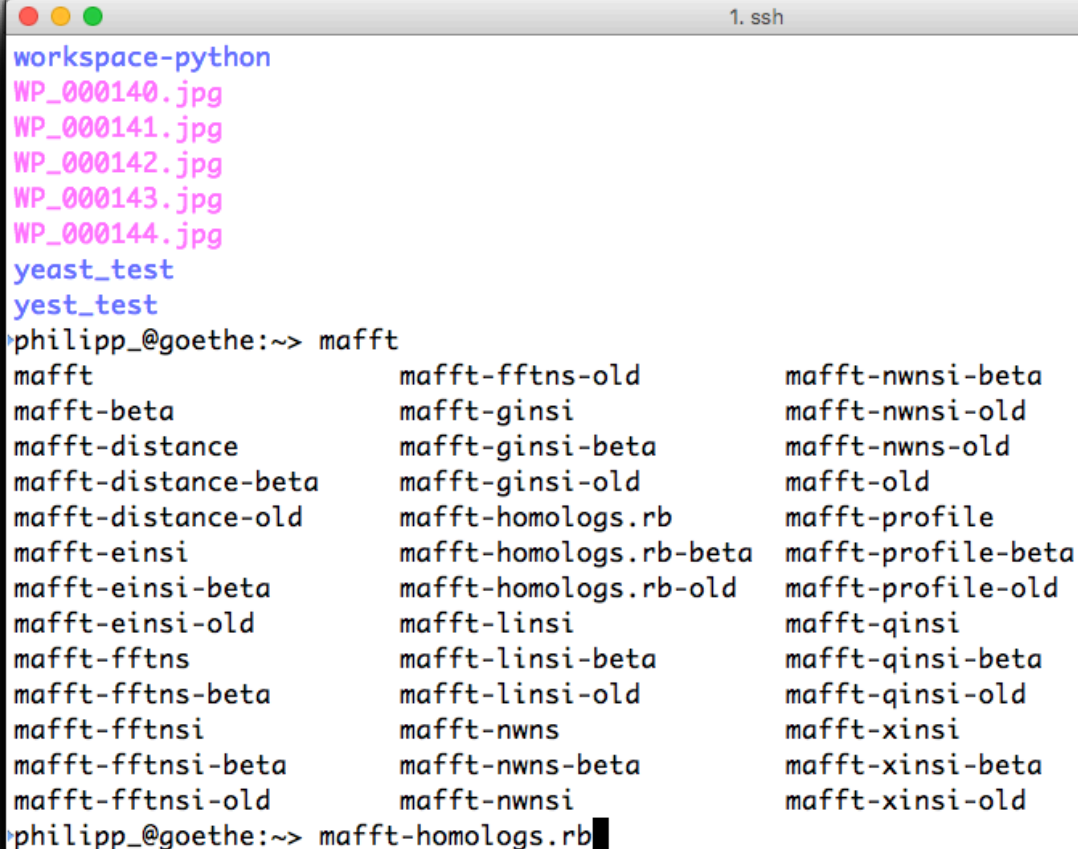


Tab Completion

- ▶ The **most important** thing when working the shell
- ▶ If you navigate to a folder just type the first 2-3 letters and then hit the tabulator key
- ▶ If there is just one file/folder that starts with those letters, bash will complete the name for you
- ▶ If there are more it will show you all options
- ▶ In that case type some more letters and try tab again
- ▶ Works with commands as well
- ▶ Avoids typing errors + you don't have to exactly know how each command/folder/file is called. You can just check as you type



Tab completion example

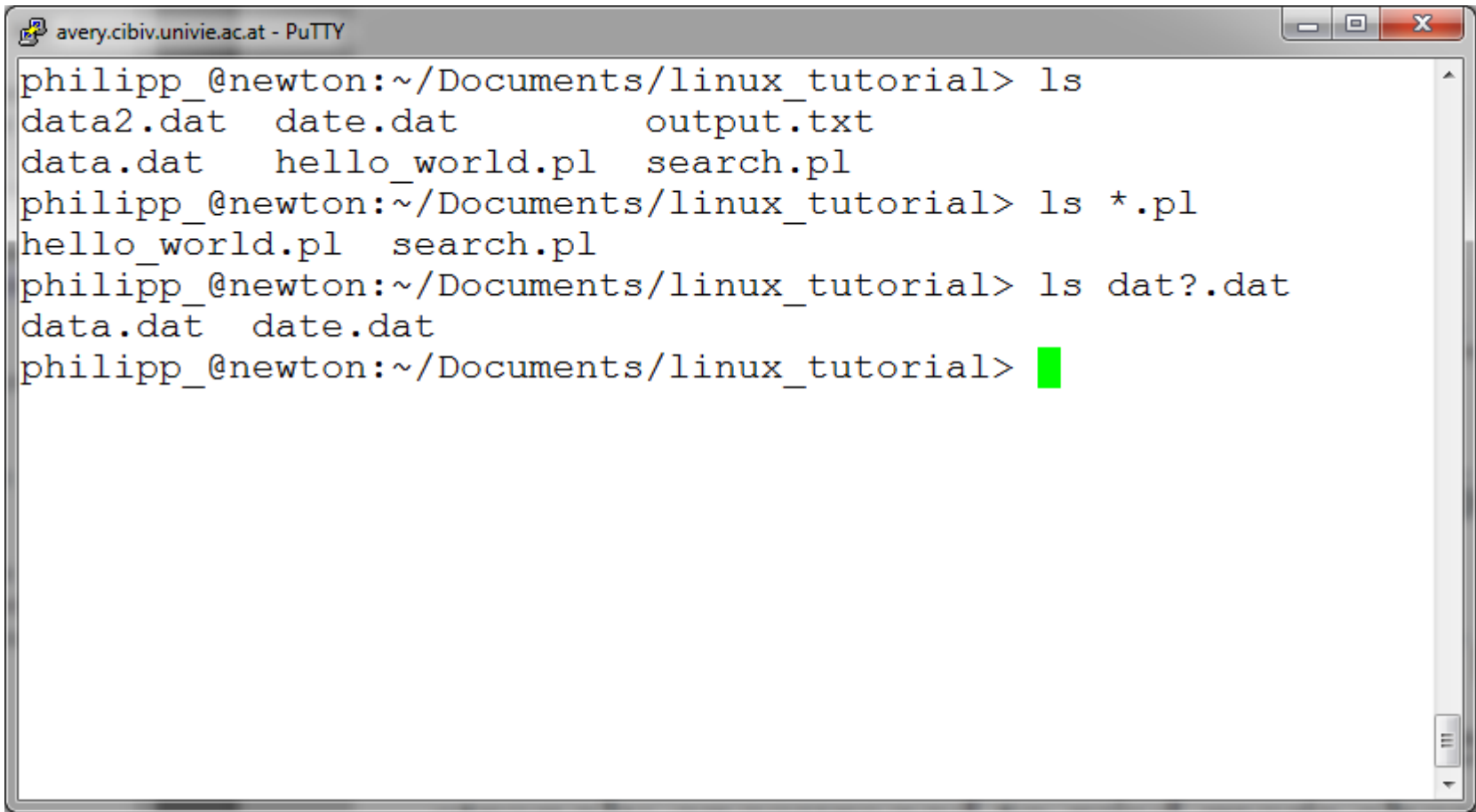


A terminal window titled "1. ssh" showing a list of files and directories. The user enters "mafft" and the terminal displays a list of suggestions. The user then enters "mafft-homologs.rb" and the terminal displays a list of suggestions for "mafft-homologs.rb".

```
workspace-python
WP_000140.jpg
WP_000141.jpg
WP_000142.jpg
WP_000143.jpg
WP_000144.jpg
yeast_test
yest_test
philipp_@goethe:~> mafft
mafft                                mafft-fftns-old          mafft-nwnsi-beta
mafft-beta                           mafft-ginsi              mafft-nwnsi-old
mafft-distance                       mafft-ginsi-beta        mafft-nwns-old
mafft-distance-beta                 mafft-ginsi-old         mafft-old
mafft-distance-old                 mafft-homologs.rb       mafft-profile
mafft-einsi                         mafft-homologs.rb-beta  mafft-profile-beta
mafft-einsi-beta                   mafft-homologs.rb-old   mafft-profile-old
mafft-einsi-old                    mafft-linsi              mafft-qinsi
mafft-fftns                         mafft-linsi-beta        mafft-qinsi-beta
mafft-fftns-beta                   mafft-linsi-old         mafft-qinsi-old
mafft-fftnsi                        mafft-nwns              mafft-xinsi
mafft-fftnsi-beta                  mafft-nwns-beta         mafft-xinsi-beta
mafft-fftnsi-old                   mafft-nwnsi             mafft-xinsi-old
philipp_@goethe:~> mafft-homologs.rb
```


Wildcards

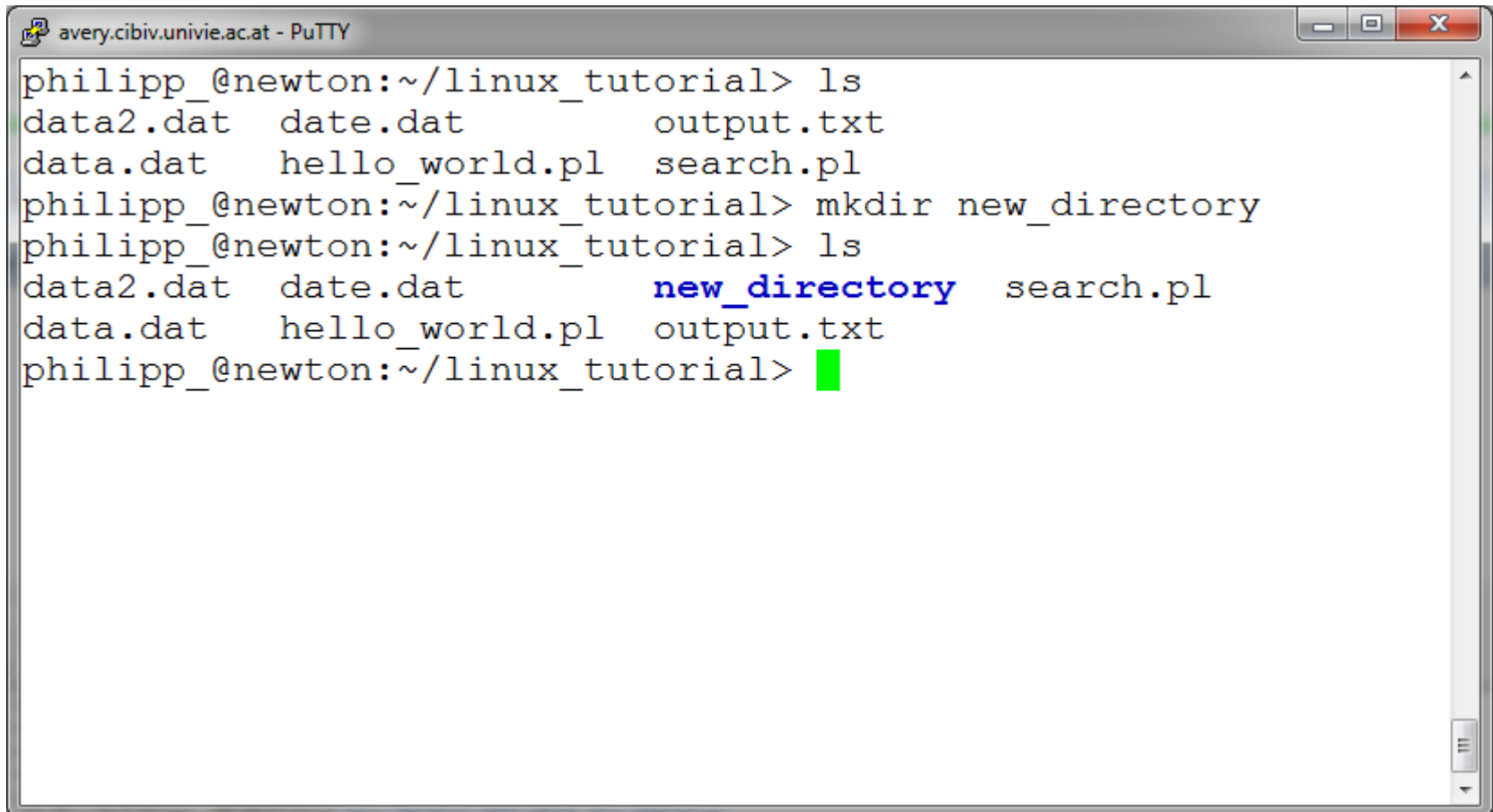
- ▶ “*” place holder for zero or more characters
- ▶ “?” place holder for exactly one character



```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/Documents/linux_tutorial> ls
data2.dat  date.dat      output.txt
data.dat   hello_world.pl  search.pl
philipp_@newton:~/Documents/linux_tutorial> ls *.pl
hello_world.pl  search.pl
philipp_@newton:~/Documents/linux_tutorial> ls dat?.dat
data.dat  date.dat
philipp_@newton:~/Documents/linux_tutorial> █
```

Command: mkdir

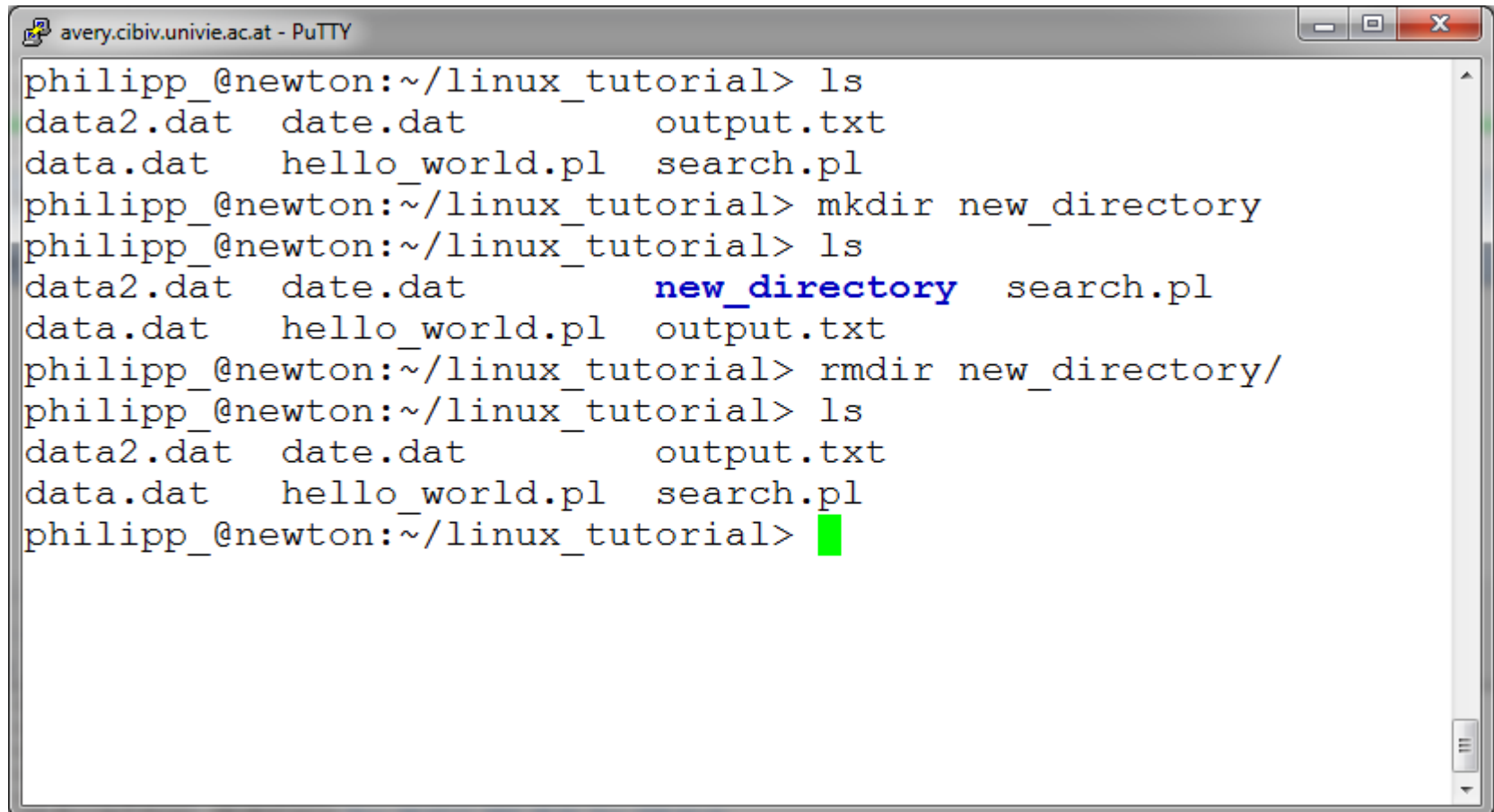
- ▶ To create a new directory use “mkdir”



```
avery.cibiv.univie.ac.at - PuTTY
philipp_newton:~/linux_tutorial> ls
data2.dat  date.dat      output.txt
data.dat   hello_world.pl search.pl
philipp_newton:~/linux_tutorial> mkdir new_directory
philipp_newton:~/linux_tutorial> ls
data2.dat  date.dat      new_directory  search.pl
data.dat   hello_world.pl output.txt
philipp_newton:~/linux_tutorial>
```

Command: rmdir

- ▶ To remove an **empty** directory use “rmdir”



```
avery.cibiv.univie.ac.at - PuTTY
philipp_newton:~/linux_tutorial> ls
data2.dat  date.dat      output.txt
data.dat   hello_world.pl search.pl
philipp_newton:~/linux_tutorial> mkdir new_directory
philipp_newton:~/linux_tutorial> ls
data2.dat  date.dat      new_directory  search.pl
data.dat   hello_world.pl output.txt
philipp_newton:~/linux_tutorial> rmdir new_directory/
philipp_newton:~/linux_tutorial> ls
data2.dat  date.dat      output.txt
data.dat   hello_world.pl search.pl
philipp_newton:~/linux_tutorial> █
```

Compressed data formats

We commonly need to download data or install a new tool. You may also need to share your data with others.

You will need to be very familiar with how to deal with compressed files (archives)



Compressed files / archives

Compressed formats → reduce the space requirements.

Same information but now optimized for size. Downside – needs to be decompressed to access the content.

1. **Compressed File** → a file reduced in size
2. **Compressed Archive** → multiple files combined then reduced in size

Important to remember:

Compression requires substantially more computational resources than decompression



Three major compression formats

- **ZIP** → **.zip** → zip/unzip
Used if you keep seeing Windows people
- **GZIP** → **.gz** → gzip/gunzip
The standard compression format, best tradeoff between speed vs compression
- **BZIP2** → **.bz/.bz2** → bzip2/bunzip2
Used by programming prima donnas: “Look how special I am, I’m even using a different compression format!”
- **Compress** → **.Z** → compress/uncompress
Used by people that don’t know what they are doing



Archives

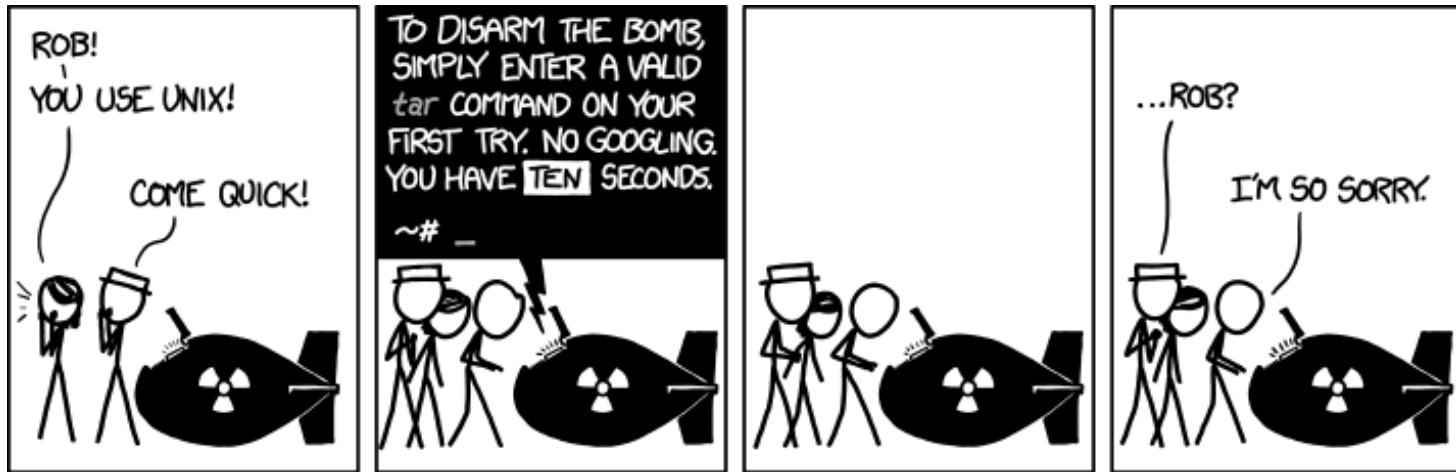
- **.zip** file → may contain one or multiple files or entire directory trees
- **.gz** → always is uncompressed to a single file

In the Unix world to compress multiple files you would need to create an archive then compress that archive.

Extensions: **.tar.gz** or **.tgz**



Using tar commands



<http://www.xkcd.com/1168/>

Creating archives with tar (tape archive)

The **tar** command can collect multiple files/directories into one file.

```
tar <commands> output-file <input files>
```

Commands: **c**reate, **e**xtract, **g**zip, **f**ile, **l**ist, **v**erbose

```
tar cvf myfile.tar sample1.fq sample2.fq
```

Major annoyance: accidentally listing the file name as the archive name will destroy the file that you are trying to archive!



Tar can handle entire directories

- Goes through and collects everything and packs it into one file, then compresses the file
- Always check the file extensions, it will tell you what it is
- Sign of a inexperienced software developer
tar-bomb → a tarball whose contents “explode” over your directory (see **tarbomb.tar.gz** on the website).

Defensive measures:

- List the content of the archive and make decision
- Create a new directory, expand the file there (this is what Mac/Windows does by default regardless what is in the archive)



Exercises 1-6



Displaying a file

- ▶ Various ways to display a file in Unix
 - ▶ cat
 - ▶ less
 - ▶ head
 - ▶ tail



Command: less

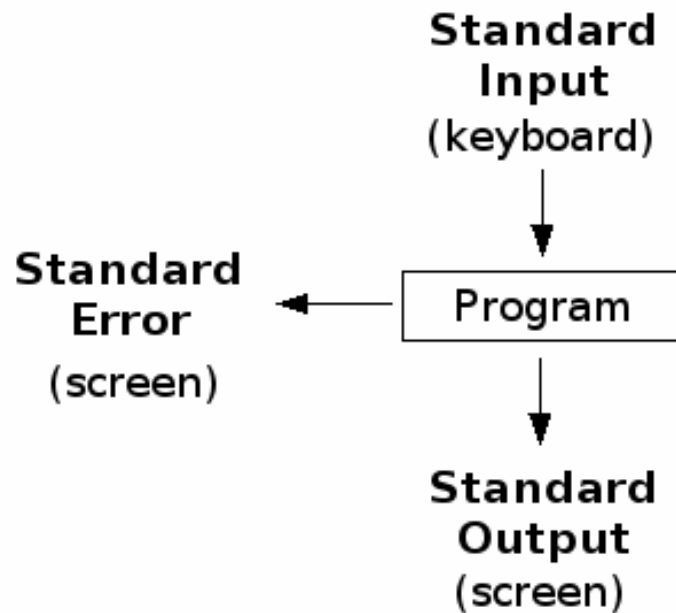
- ▶ “less” displays a file, allowing forward/backward movement within it
 - ▶ return scrolls forward one line, space one page
 - ▶ y scrolls back one line, b one page
 - ▶ Navigate with cursor keys (up/down/left/right)
- ▶ Use “/” to search for a string
- ▶ Press q to quit
- ▶ **man pages** use less

- ▶ **Example: Check a SAM file**

```
less -S mapped_reads.sam
```



Program input and output



```
oort:~ philipp$ ./programA
That's a normal message
Error: This is an error message!
oort:~ philipp$
```

A screenshot of a terminal window showing the execution of a program. The prompt is "oort:~ philipp\$". The user enters the command "./programA". The program outputs two lines: "That's a normal message" and "Error: This is an error message!". The prompt returns to "oort:~ philipp\$".

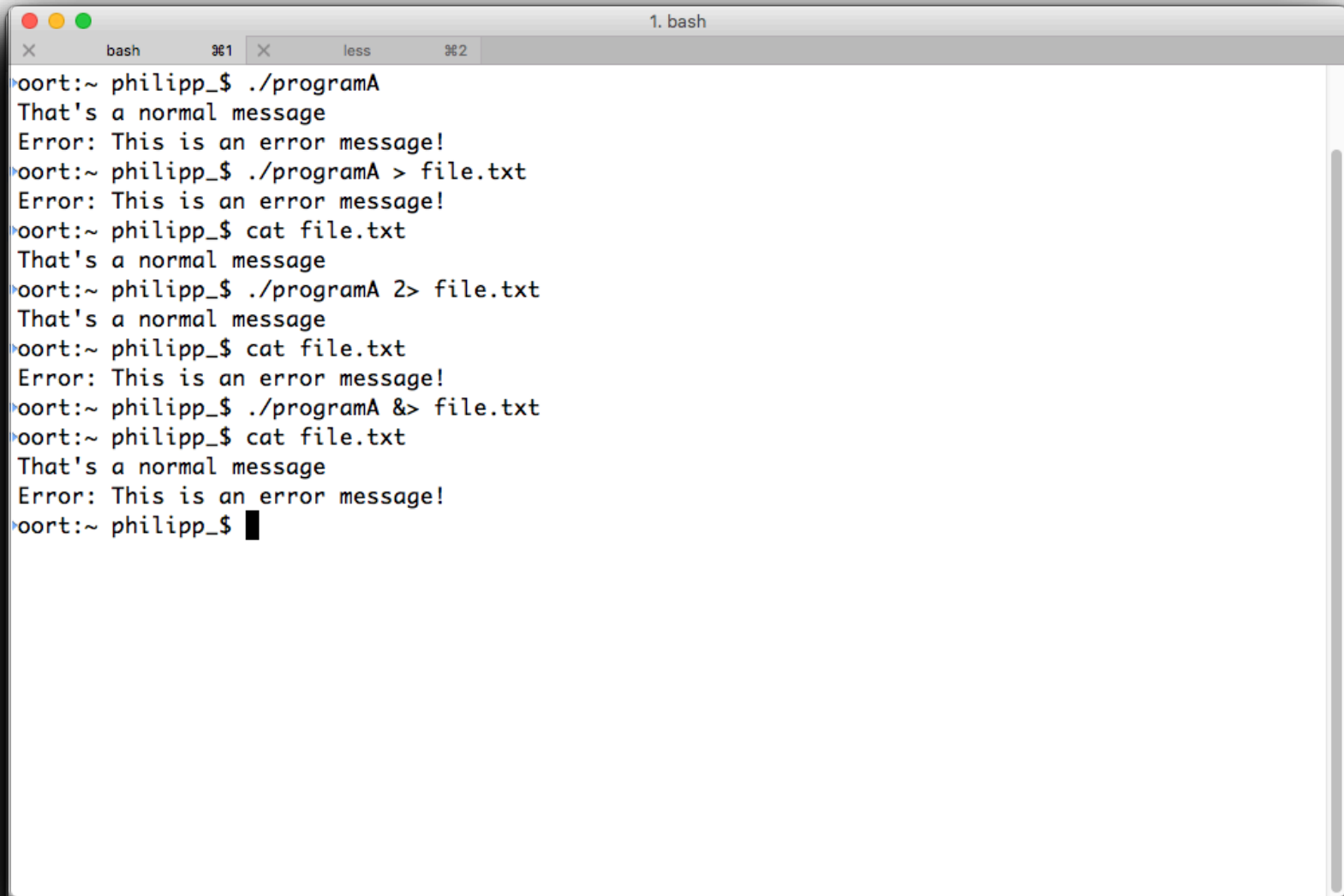


Redirect program output

- ▶ Output of programs can be redirected to a file:
- ▶ `program_a > file.txt`
 - ▶ `program_a`'s output is written to the file called "file.txt"
 - ▶ If file exists, it will be **overwritten**
- ▶ `program_b >> file.txt`
 - ▶ `program_b`'s output is appended to the file called "file.txt"
 - ▶ If "file.txt" doesn't exist, it will be created
- ▶ "`>`" and "`>>`" redirect standard output
- ▶ "`2>`" and "`2>>`" redirect standard error
- ▶ "`&>`" and "`&>>`" redirects both



Input/Output Redirection



```
1. bash
bash %1 less %2
root:~ philipp_$ ./programA
That's a normal message
Error: This is an error message!
root:~ philipp_$ ./programA > file.txt
Error: This is an error message!
root:~ philipp_$ cat file.txt
That's a normal message
root:~ philipp_$ ./programA 2> file.txt
That's a normal message
root:~ philipp_$ cat file.txt
Error: This is an error message!
root:~ philipp_$ ./programA &> file.txt
root:~ philipp_$ cat file.txt
That's a normal message
Error: This is an error message!
root:~ philipp_$ █
```


Input/Output Redirection



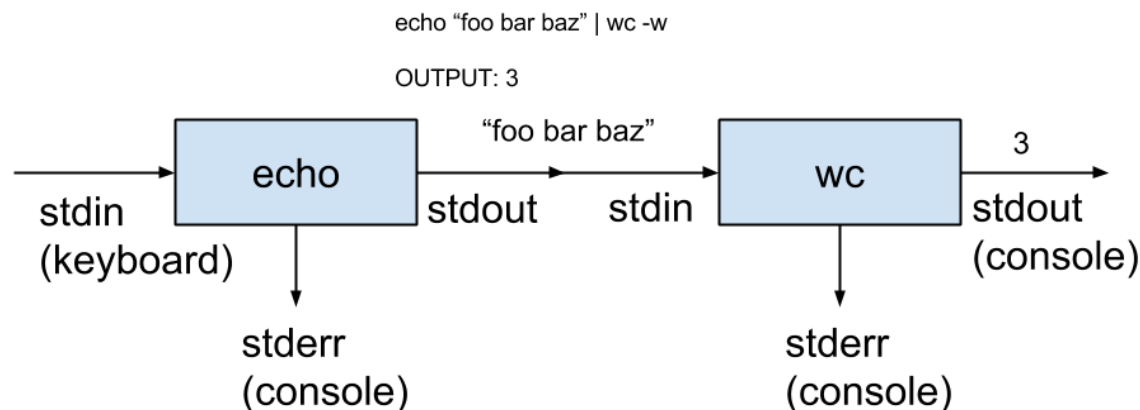
The image shows a PuTTY terminal window with the title bar "avery.cibiv.univie.ac.at - PuTTY". The terminal displays a list of files: data.dat, date.dat, hello.txt, hello_world.pl, lines.txt, list.txt, names2.txt, names.txt, output.txt, and search.pl. At the bottom, a status bar shows "lines 1-10/10 (END)" with a green cursor.

```
data.dat  
date.dat  
hello.txt  
hello_world.pl  
lines.txt  
list.txt  
names2.txt  
names.txt  
output.txt  
search.pl  
lines 1-10/10 (END)
```



Piping

- ▶ Programs can take their input directly from another program
- ▶ Example:
 - ▶ `program_a | program_b`
- ▶ Allows to combine an arbitrary number of tools!



Command: cat

- ▶ Dumps an entire file to standard output
- ▶ Good for displaying short, simple files and for concatenating files

- ▶ Example: Concatenate three FASTQ files

```
cat SRA0001.fq SRA0002.fq SRA0003.fq > SRA000_all.fq
```



Command: head

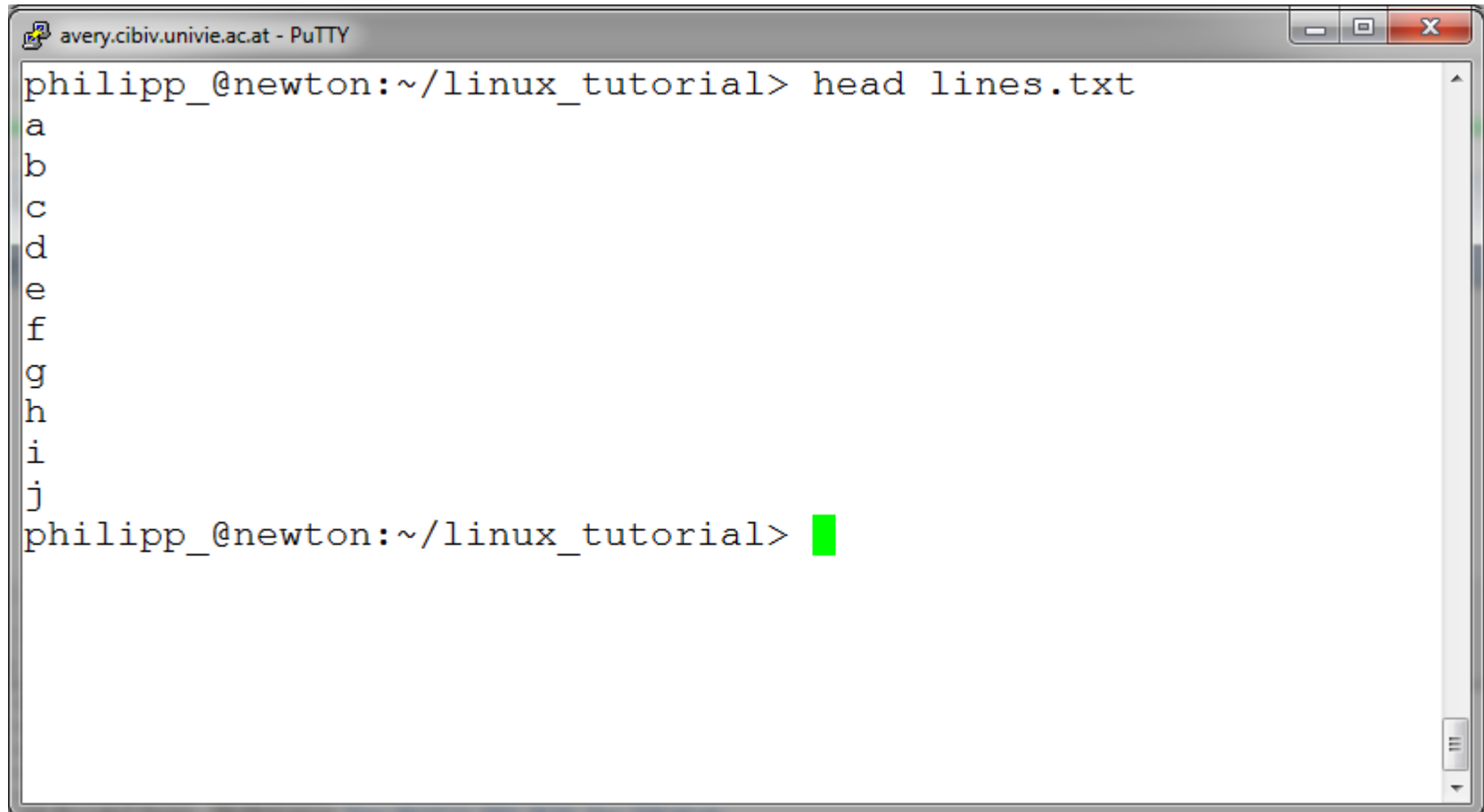
- ▶ “head” displays the top part of a file
 - ▶ By default it shows the first 10 lines
 - ▶ -n option allows you to change that
-
- ▶ **Example: Display the first 10 sequences of the dataset**

```
head -n 40 SR012310.fq
```



Command: head

- ▶ Here's an example of using "head":

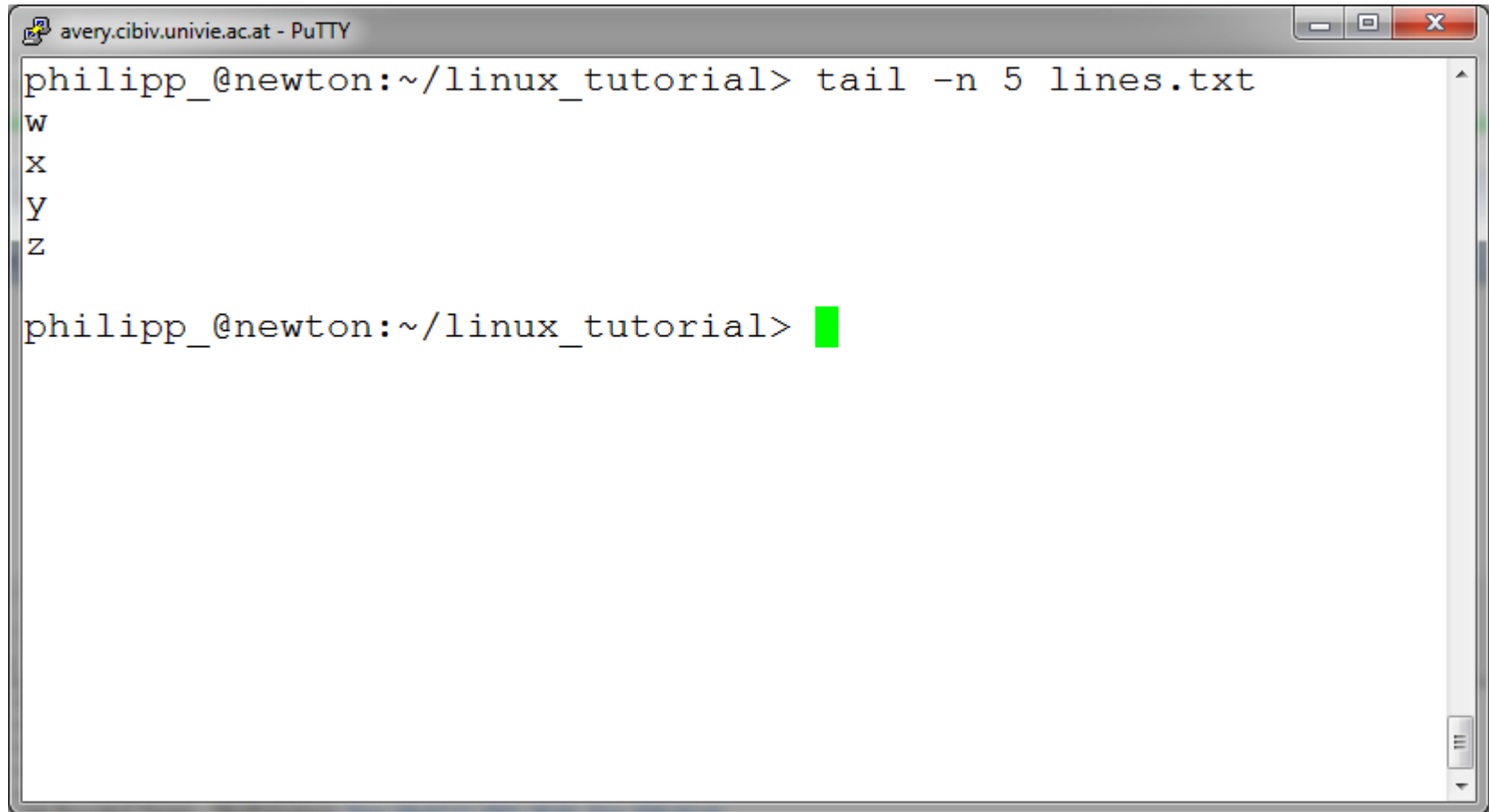


```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/linux_tutorial> head lines.txt
a
b
c
d
e
f
g
h
i
j
philipp_@newton:~/linux_tutorial> █
```



Command: tail

- ▶ Same as head, but shows the last lines



```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/linux_tutorial> tail -n 5 lines.txt
w
x
y
z

philipp_@newton:~/linux_tutorial> █
```

The screenshot shows a PuTTY terminal window with the title 'avery.cibiv.univie.ac.at - PuTTY'. The user 'philipp_@newton' is in the directory '~/linux_tutorial'. They have executed the command 'tail -n 5 lines.txt', which outputs the last five lines of the file 'lines.txt', which are 'w', 'x', 'y', and 'z'. The prompt 'philipp_@newton:~/linux_tutorial>' is followed by a green cursor block.



Creating and editing files in Unix/Linux

- ▶ Requires the use of an Editor
- ▶ Various Editors:
 - 1) nano / pico
 - 2) vi
 - 3) emacs



Editing a file using pico or nano

- ▶ Type “pico” or “nano” at the prompt

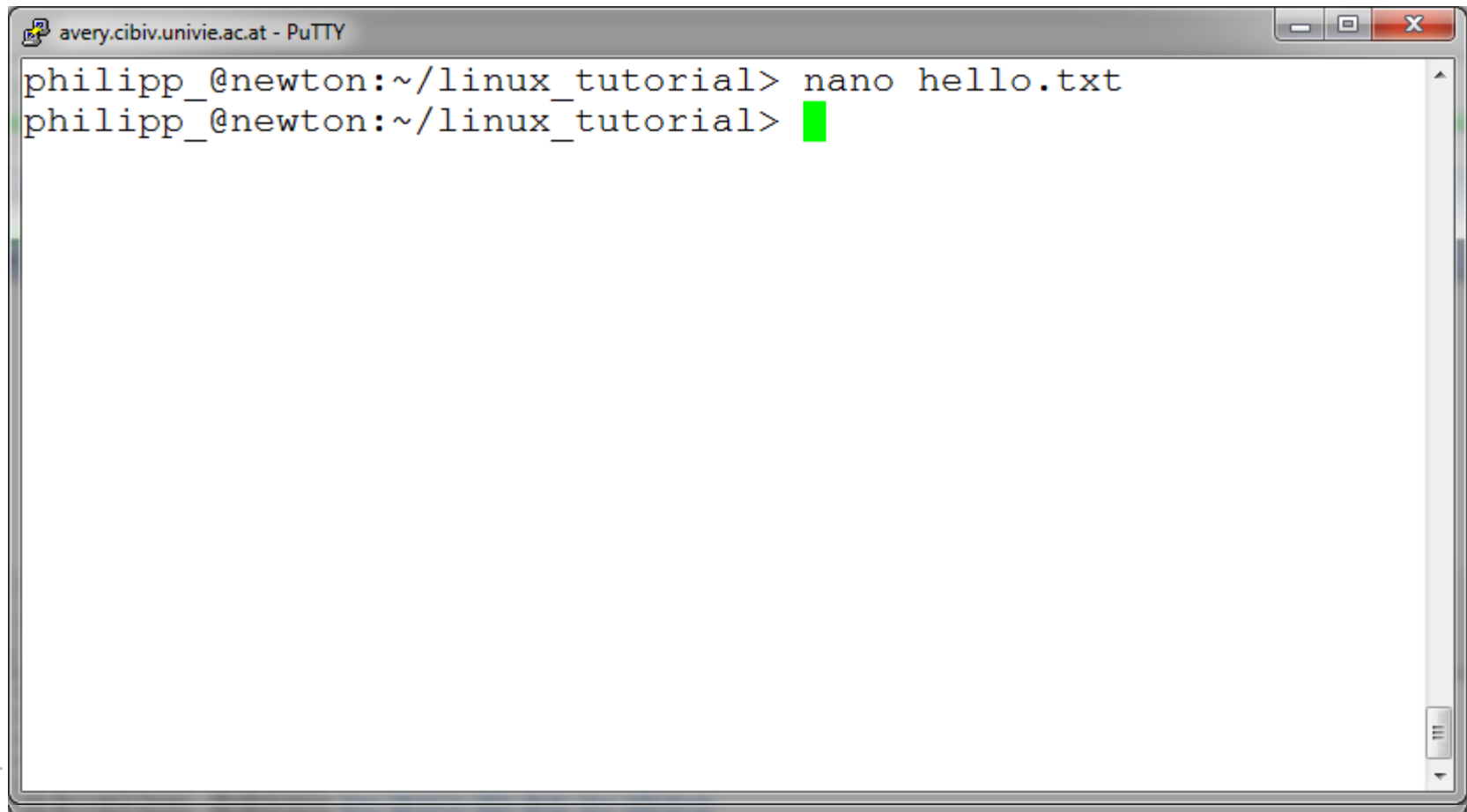


```
avery.cibiv.univie.ac.at - PuTTY
GNU nano 2.2.6      File: hello.txt      Modified
Hello World!█

[ New File ]
^G Get Hel ^O WriteOu ^R Read Fi ^Y Prev Pa ^K Cut Tex ^C Cur Pos
^X Exit    ^J Justify ^W Where I ^V Next Pa ^U UnCut T ^T To Spell
```


Editing a file using pico

- ▶ To save use “ctrl-o”
- ▶ To exit use “ctrl-x”



```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/linux_tutorial> nano hello.txt
philipp_@newton:~/linux_tutorial> █
```

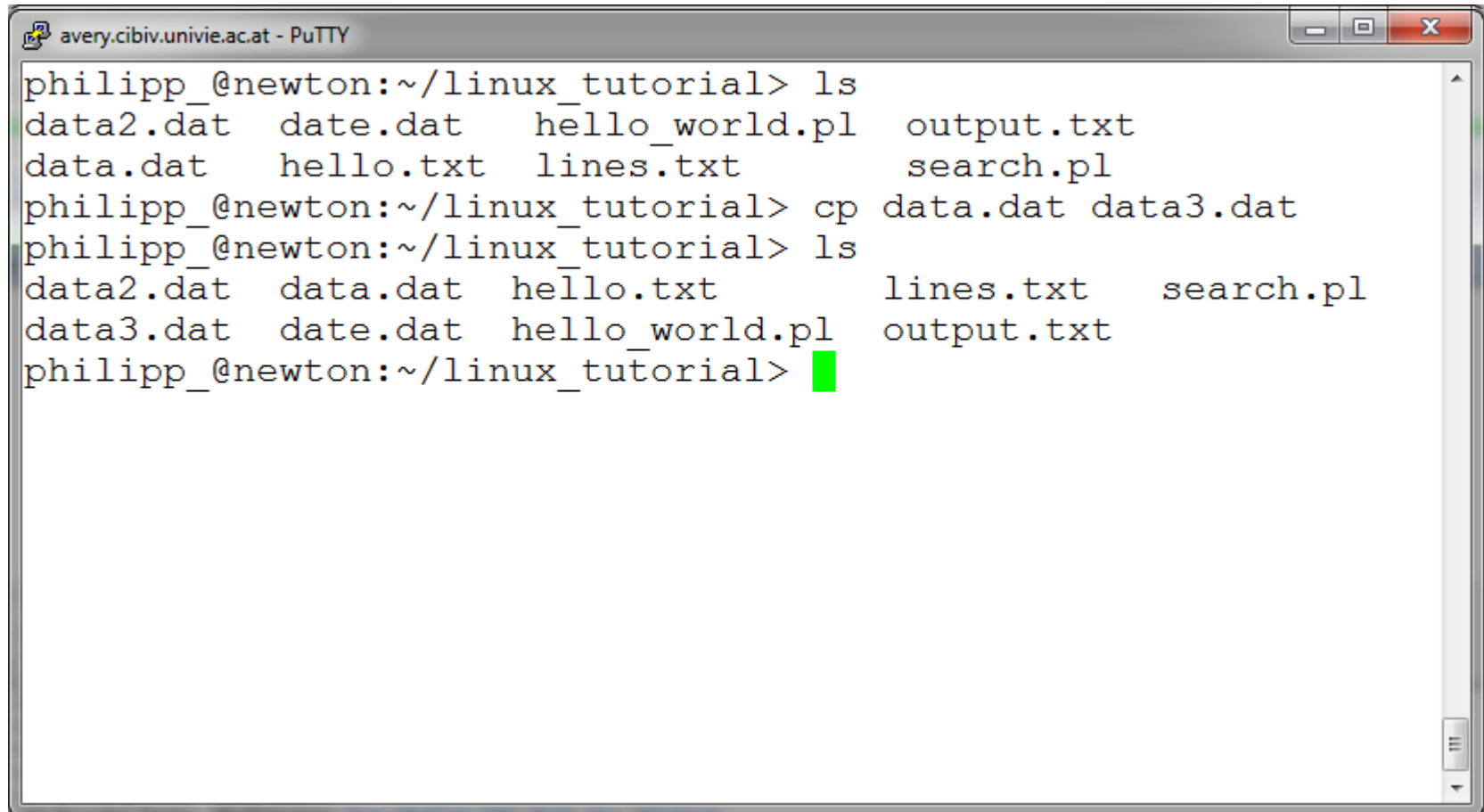
File Commands

- ▶ Copying a file: `cp`
- ▶ Move or rename a file: `mv`
- ▶ Remove a file: `rm`



Command: cp

- ▶ To copy a file use “cp”

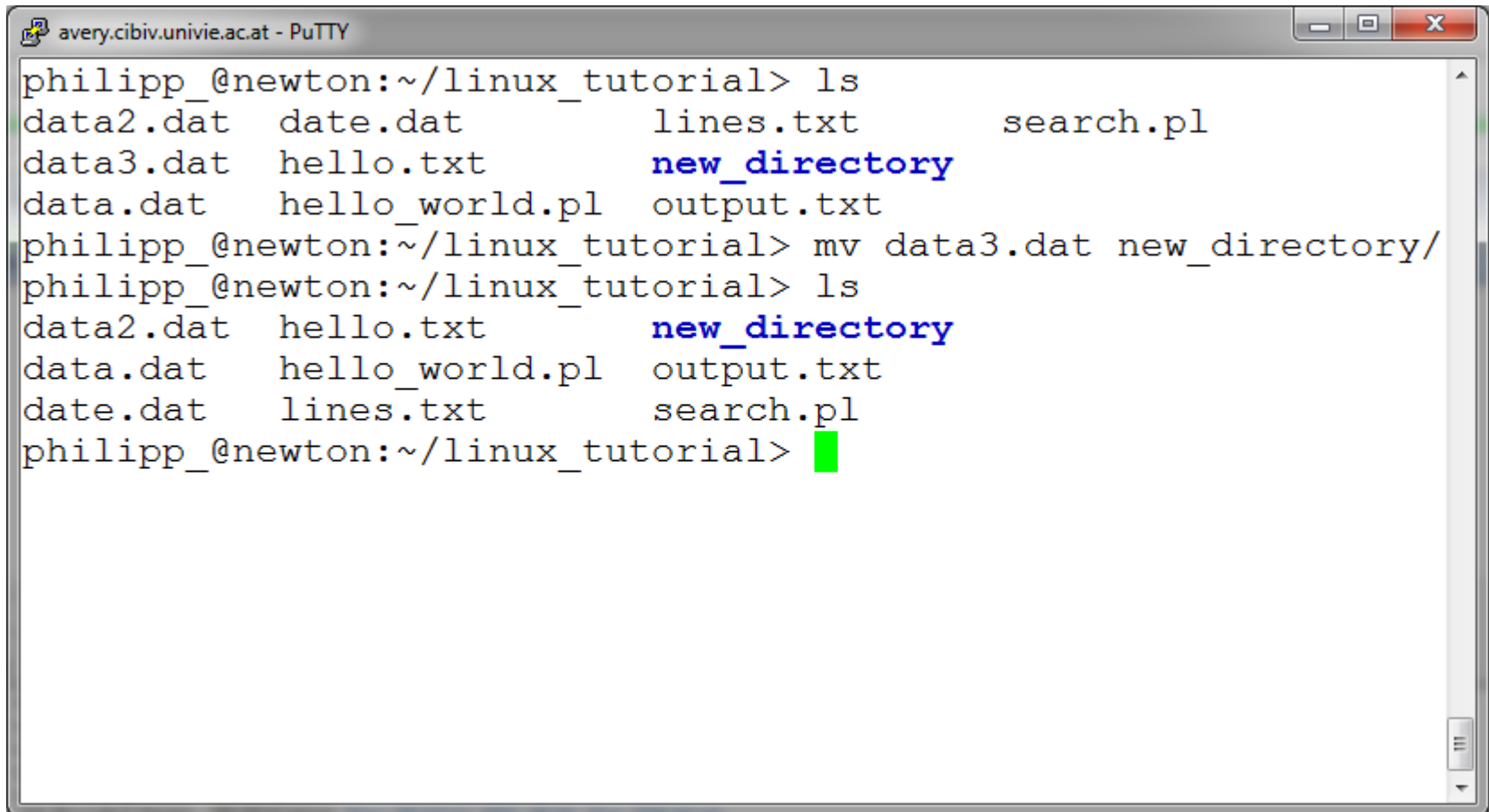


```
avery.cibiv.univie.ac.at - PuTTY
philipp_newton:~/linux_tutorial> ls
data2.dat  date.dat  hello_world.pl  output.txt
data.dat   hello.txt  lines.txt       search.pl
philipp_newton:~/linux_tutorial> cp data.dat data3.dat
philipp_newton:~/linux_tutorial> ls
data2.dat  data.dat  hello.txt       lines.txt  search.pl
data3.dat  date.dat  hello_world.pl  output.txt
philipp_newton:~/linux_tutorial> █
```



Command: mv

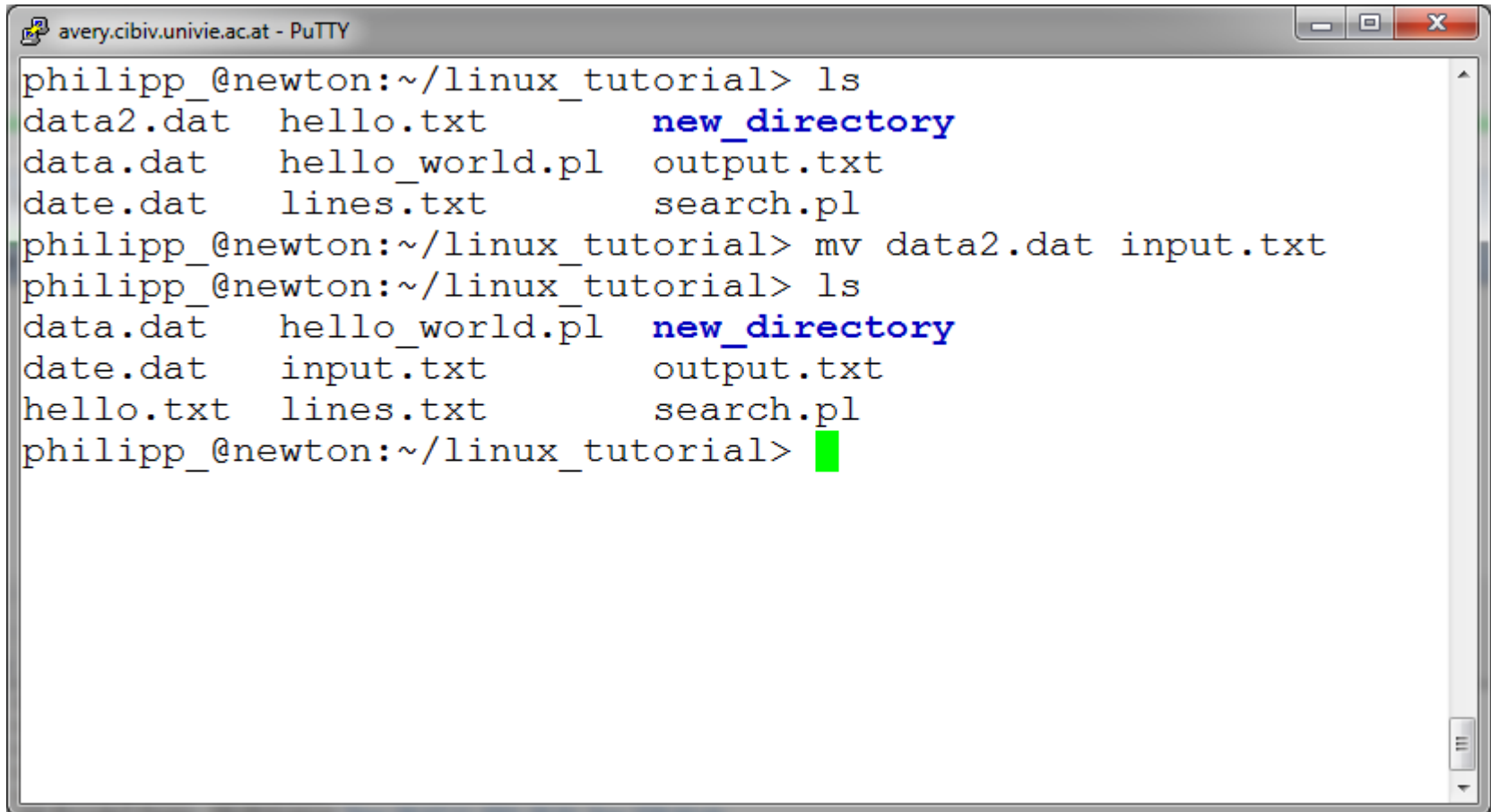
- ▶ To move a file to a different location use “mv”



```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/linux_tutorial> ls
data2.dat  date.dat      lines.txt     search.pl
data3.dat  hello.txt     new_directory
data.dat   hello_world.pl output.txt
philipp_@newton:~/linux_tutorial> mv data3.dat new_directory/
philipp_@newton:~/linux_tutorial> ls
data2.dat  hello.txt     new_directory
data.dat   hello_world.pl output.txt
date.dat   lines.txt     search.pl
philipp_@newton:~/linux_tutorial> █
```

Command: mv

- ▶ mv can also be used to rename a file

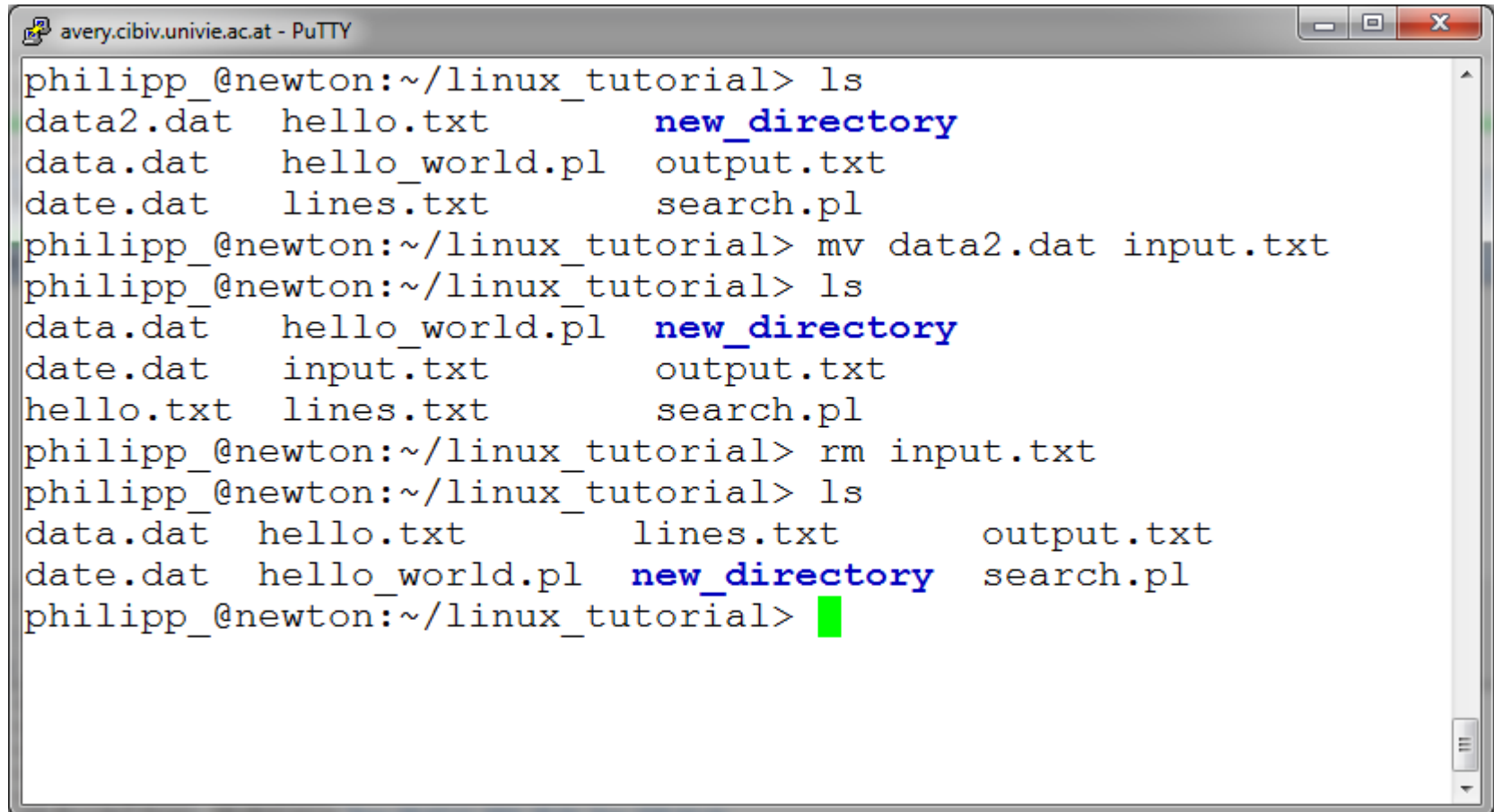


```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/linux_tutorial> ls
data2.dat  hello.txt          new_directory
data.dat   hello_world.pl    output.txt
date.dat   lines.txt         search.pl
philipp_@newton:~/linux_tutorial> mv data2.dat input.txt
philipp_@newton:~/linux_tutorial> ls
data.dat   hello_world.pl    new_directory
date.dat   input.txt         output.txt
hello.txt  lines.txt        search.pl
philipp_@newton:~/linux_tutorial> █
```



Command: rm

- ▶ To remove a file use “rm”

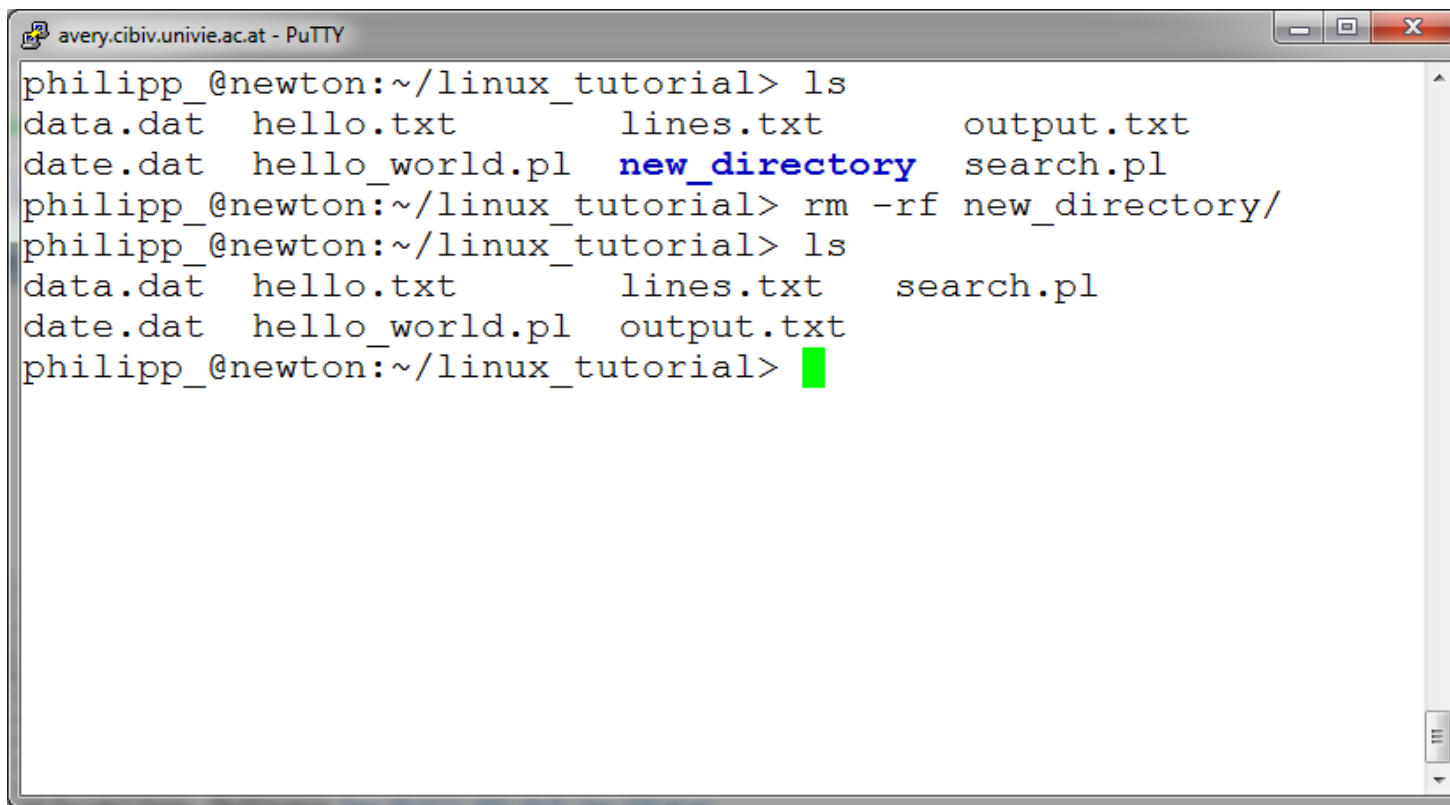


```
avery.cibiv.univie.ac.at - PuTTY
philipp_newton:~/linux_tutorial> ls
data2.dat  hello.txt          new_directory
data.dat   hello_world.pl    output.txt
date.dat   lines.txt         search.pl
philipp_newton:~/linux_tutorial> mv data2.dat input.txt
philipp_newton:~/linux_tutorial> ls
data.dat   hello_world.pl    new_directory
date.dat   input.txt         output.txt
hello.txt  lines.txt         search.pl
philipp_newton:~/linux_tutorial> rm input.txt
philipp_newton:~/linux_tutorial> ls
data.dat   hello.txt         lines.txt       output.txt
date.dat   hello_world.pl   new_directory   search.pl
philipp_newton:~/linux_tutorial> █
```



Command: rm

- ▶ To remove a file “recursively”: `rm -r`
- ▶ Used to remove all files and directories



```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/linux_tutorial> ls
data.dat  hello.txt      lines.txt      output.txt
date.dat  hello_world.pl new_directory  search.pl
philipp_@newton:~/linux_tutorial> rm -rf new_directory/
philipp_@newton:~/linux_tutorial> ls
data.dat  hello.txt      lines.txt      search.pl
date.dat  hello_world.pl output.txt
philipp_@newton:~/linux_tutorial> █
```

- ▶ Be very careful, deletions are **permanent** in Unix/Linux

Exercises 7-18



Regular expression

A **regular expression** (regex) is a special text string for describing a search pattern.

A **regular expression** provides concise and flexible means to "match" (specify and recognize) strings of text, such as particular characters, words, or patterns of characters.

▶ **Examples:**

- ▶ ATG matches ATG
- ▶ A[TU]G matches ATG and AUG
- ▶ A.G matches AAG, ABG, ACG, ADG, ...



Regex: Metacharacters

- ? The ? (question mark) matches the preceding character 0 or 1 times only, for example, `colou?r` will find both `color` (0 times) and `colour` (1 time).
- * The * (asterisk or star) matches the preceding character 0 or more times, for example, `tre*` will find `tree` (2 times) and `tread` (1 time) and `trough` (0 times).
- + The + (plus) matches the previous character 1 or more times, for example, `tre+` will find `tree` (2 times) and `tread` (1 time) but NOT `trough` (0 times).
- {n} Matches the preceding character, or character range, n times exactly.
- {n,m} Matches the preceding character at least n times but not more than m times, for example, `'ba{2,3}b'` will find `'baab'` and `'baaab'` but NOT `'bab'` or `'baaaab'`. Values are enclosed in braces (curly brackets).



Regex: Ranges

- [] Match anything inside the square brackets for ONE character position once and only once, for example, [12] means match the target to 1 and if that does not match then match the target to 2 while [0123456789] means match to any character in the range 0 to 9.
 - The - (dash) **inside square brackets** is the 'range separator' and allows us to define a range, in our example above of [0123456789] we could rewrite it as [0-9].
You can define more than one range inside a list, for example, [0-9A-C] means check for 0 to 9 and A to C (but not a to c).
 - ^ The ^ (circumflex or caret) **inside square brackets** negates the expression (we will see an alternate use for the circumflex/caret **outside** square brackets later), for example, [^Ff] means anything except upper or lower case F and [^a-z] means everything except lower case a to z.



Regex: Metacharacters

- ^ The ^ (circumflex or caret) **outside square brackets** means look only at the beginning of the target string, for example, **^Win** will not find Windows in **STRINGI** but **^Moz** will find **Mozilla**.
- \$ The \$ (dollar) means look only at the end of the target string, for example, **fox\$** will find a match in 'silver **fox**' since it appears at the end of the string but not in 'the fox jumped over the moon'.
- .
- The . (period) means any character(s) in this position, for example, **ton.** will find **tons, tone** and **tonneau** but not **wanton** because it has no following character.
- () The ((open parenthesis) and) (close parenthesis) may be used to group (or bind) parts of our search expression together.
- | The | (vertical bar or pipe) is called **alternation** in techspeak and means find the left hand OR right values, for example, **gr(a|e)y** will find 'gray' or 'grey' and has the sense that if the first test is not valid the second will be tried, if the first is valid the second will not be tried.



Regex: Character classes

<code>[:digit:]</code>	Only the digits 0 to 9
<code>[:alnum:]</code>	Any alphanumeric character 0 to 9 OR A to Z or a to z.
<code>[:alpha:]</code>	Any alpha character A to Z or a to z.
<code>[:blank:]</code>	Space and TAB characters only.
<code>[:xdigit:]</code>	Hexadecimal notation 0-9,A-F, a-f.
<code>[:punct:]</code>	Punctuation symbols . , " ' ? ! ; : # \$ % & () * + - / < > = @ [] \ ^ _ { } ~
<code>[:print:]</code>	Any printable character.
<code>[:space:]</code>	Any whitespace characters (space, tab, NL, FF, VT, CR). Many system abbreviate as \s.
<code>[:graph:]</code>	Exclude whitespace (SPACE, TAB). Many system abbreviate as \W.
<code>[:upper:]</code>	Any alpha character A to Z.
<code>[:lower:]</code>	Any alpha character a to z.



Regex: Extensions

Character Class Abbreviations

- `\d` Match any character in the range 0 - 9 (equivalent of POSIX `[:digit:]`)
- `\D` Match any character NOT in the range 0 - 9 (equivalent of POSIX `[^[:digit:]]`)
- `\s` Match any whitespace characters (space, tab etc.). (equivalent of POSIX `[:space:]` EXCEPT VT is not recognized)
- `\S` Match any character NOT whitespace (space, tab). (equivalent of POSIX `[^[:space:]]`)
- `\w` Match any character in the range 0 - 9, A - Z and a - z (equivalent of POSIX `[:alnum:]`)
- `\W` Match any character NOT the range 0 - 9, A - Z and a - z (equivalent of POSIX `[^[:alnum:]]`)

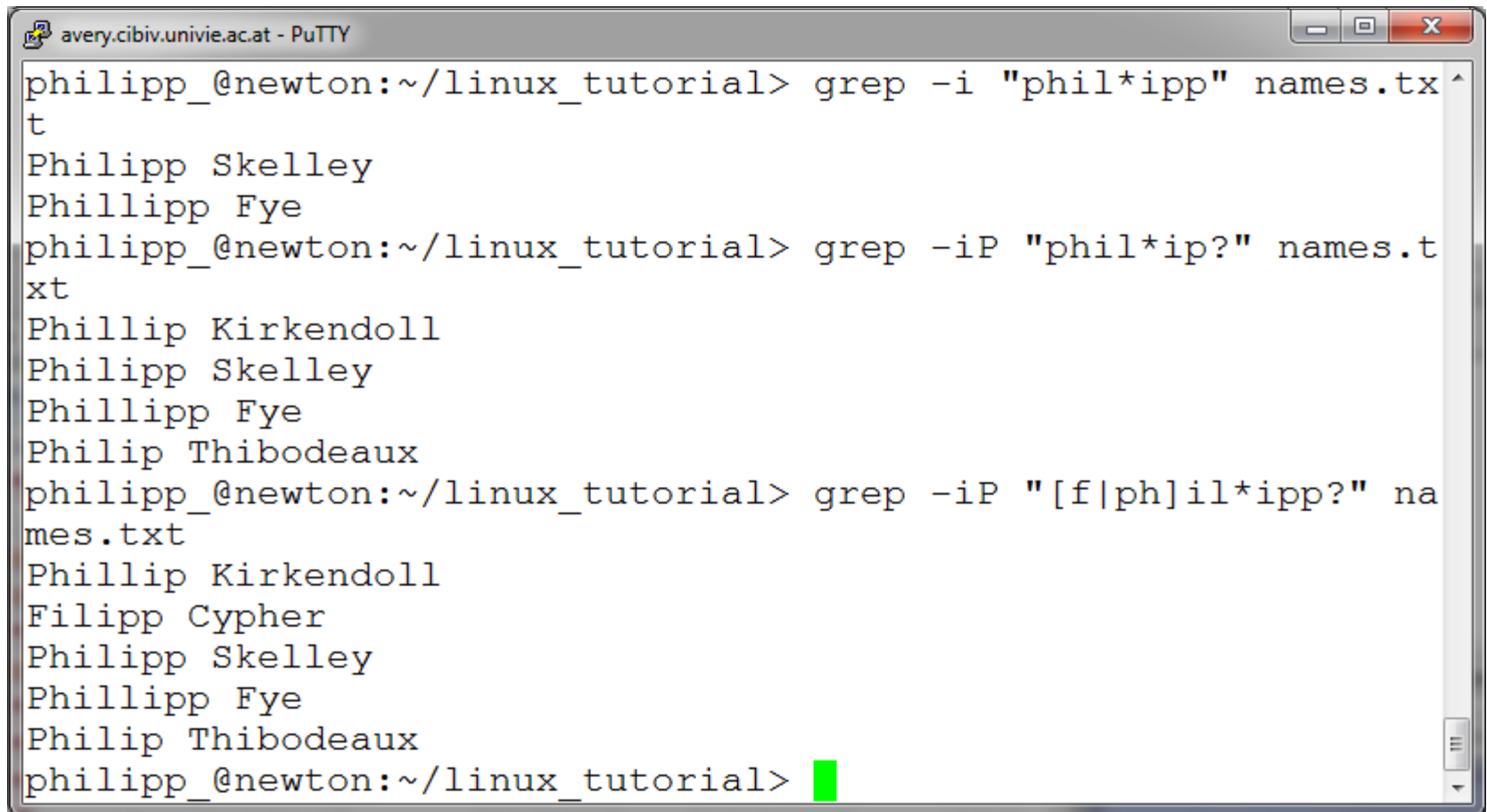
Positional Abbreviations

- `\b` Word boundary. Match any character(s) at the beginning (`\bxx`) and/or end (`xx\b`) of a word, thus `\bton\b` will find ton but not tons, but `\bton` will find tons.
- `\B` Not word boundary. Match any character(s) NOT at the beginning (`\Bxx`) and/or end (`xx\B`) of a word, thus `\Bton\B` will find wantons but not tons, but `ton\B` will find both wantons and tons.



Command: grep

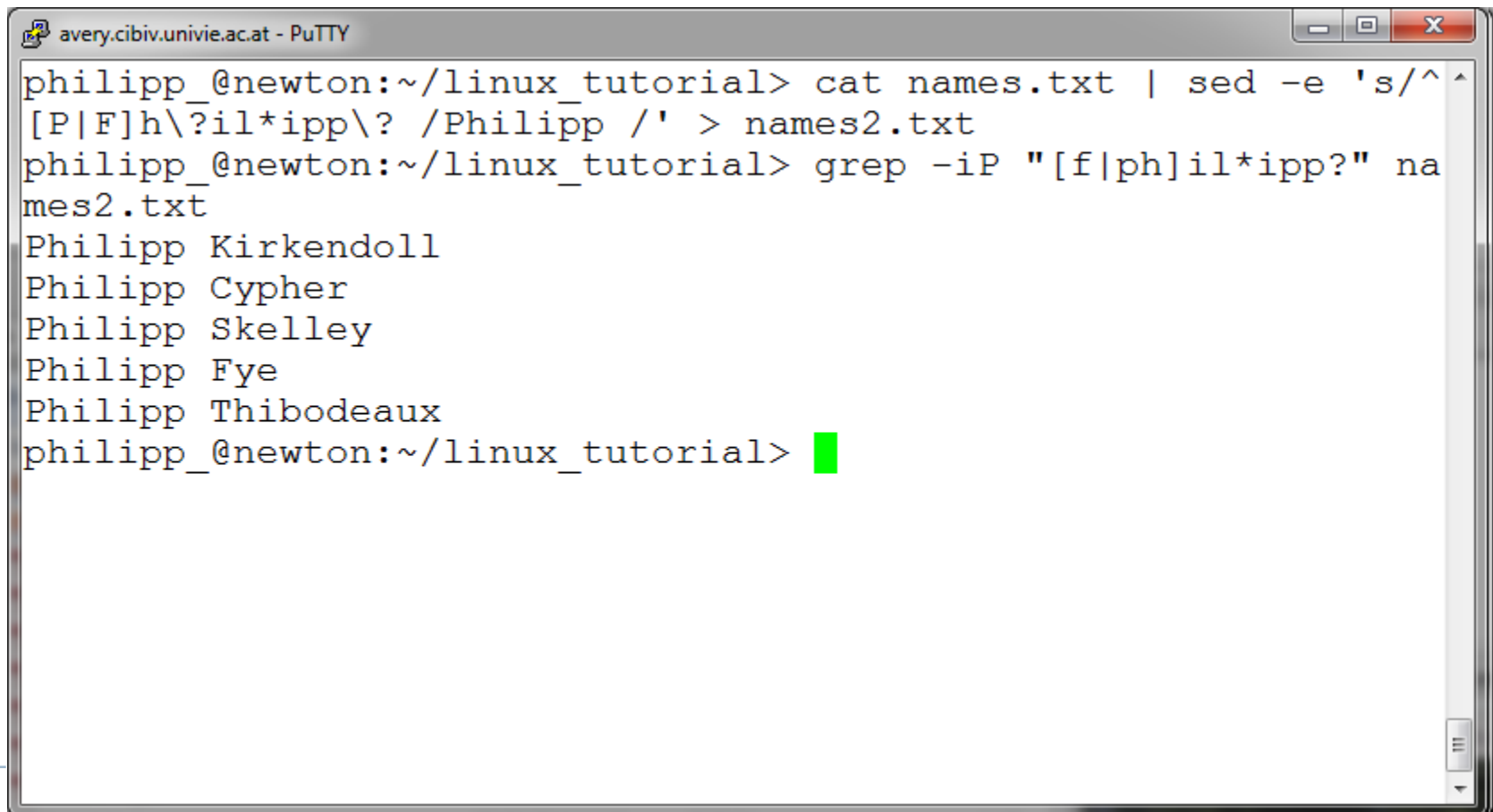
- ▶ To search files for a specific string use “grep”



```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/linux_tutorial> grep -i "phil*ipp" names.txt
Philipp Skelley
Phillipp Fye
philipp_@newton:~/linux_tutorial> grep -iP "phil*ip?" names.txt
Phillip Kirkendoll
Philipp Skelley
Phillipp Fye
Philip Thibodeaux
philipp_@newton:~/linux_tutorial> grep -iP "[f|ph]il*ipp?" names.txt
Phillip Kirkendoll
Filipp Cypher
Philipp Skelley
Phillipp Fye
Philip Thibodeaux
philipp_@newton:~/linux_tutorial>
```

Command: sed

- ▶ sed is a stream editor for filtering and transforming text
- ▶ Powerful but complicated => useful for substitution



```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/linux_tutorial> cat names.txt | sed -e 's/^[^
[P|F]h\?il*ipp\? /Philipp /' > names2.txt
philipp_@newton:~/linux_tutorial> grep -iP "[f|ph]il*ipp?" na
mes2.txt
Philipp Kirkendoll
Philipp Cypher
Philipp Skelley
Philipp Fye
Philipp Thibodeaux
philipp_@newton:~/linux_tutorial>
```


File permissions

- ▶ Each file in Unix/Linux has an associated permission level
- ▶ This allows the user to prevent others from reading/writing/executing their files or directories
- ▶ Use “`ls -l filename`” to find the permission level of that file



Permission levels

- ▶ “r” means “read only” permission
- ▶ “w” means “write” permission
- ▶ “x” means “execute” permission
 - ▶ In case of directory, “x” grants permission to list directory contents

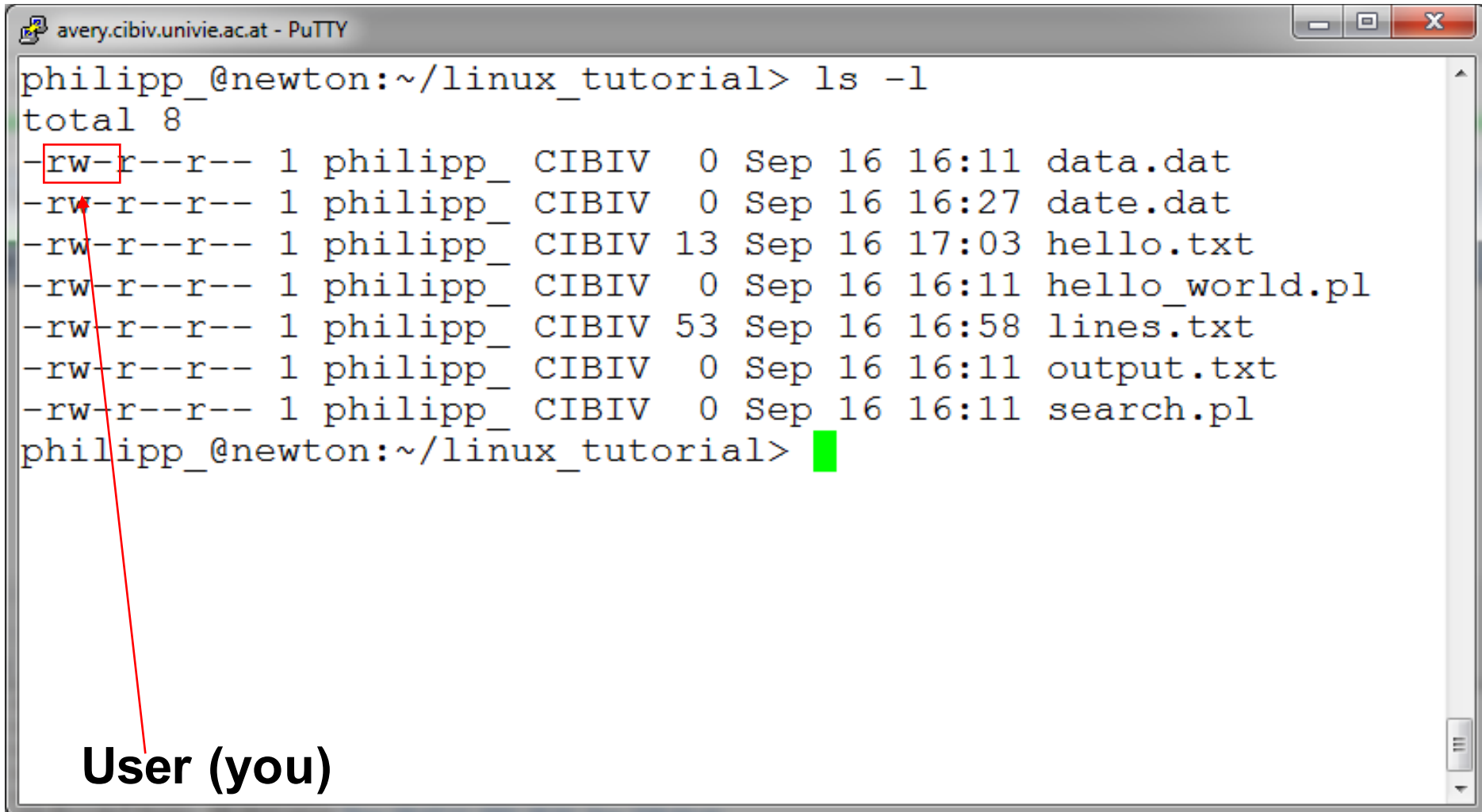


File Permissions

```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/linux_tutorial> ls -l
total 8
-rw-r--r-- 1 philipp CIBIV 0 Sep 16 16:11 data.dat
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:27 date.dat
-rw-r--r-- 1 philipp_ CIBIV 13 Sep 16 17:03 hello.txt
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 hello_world.pl
-rw-r--r-- 1 philipp_ CIBIV 53 Sep 16 16:58 lines.txt
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 output.txt
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 search.pl
philipp_@newton:~/linux_tutorial>
```

User (you) **Group**

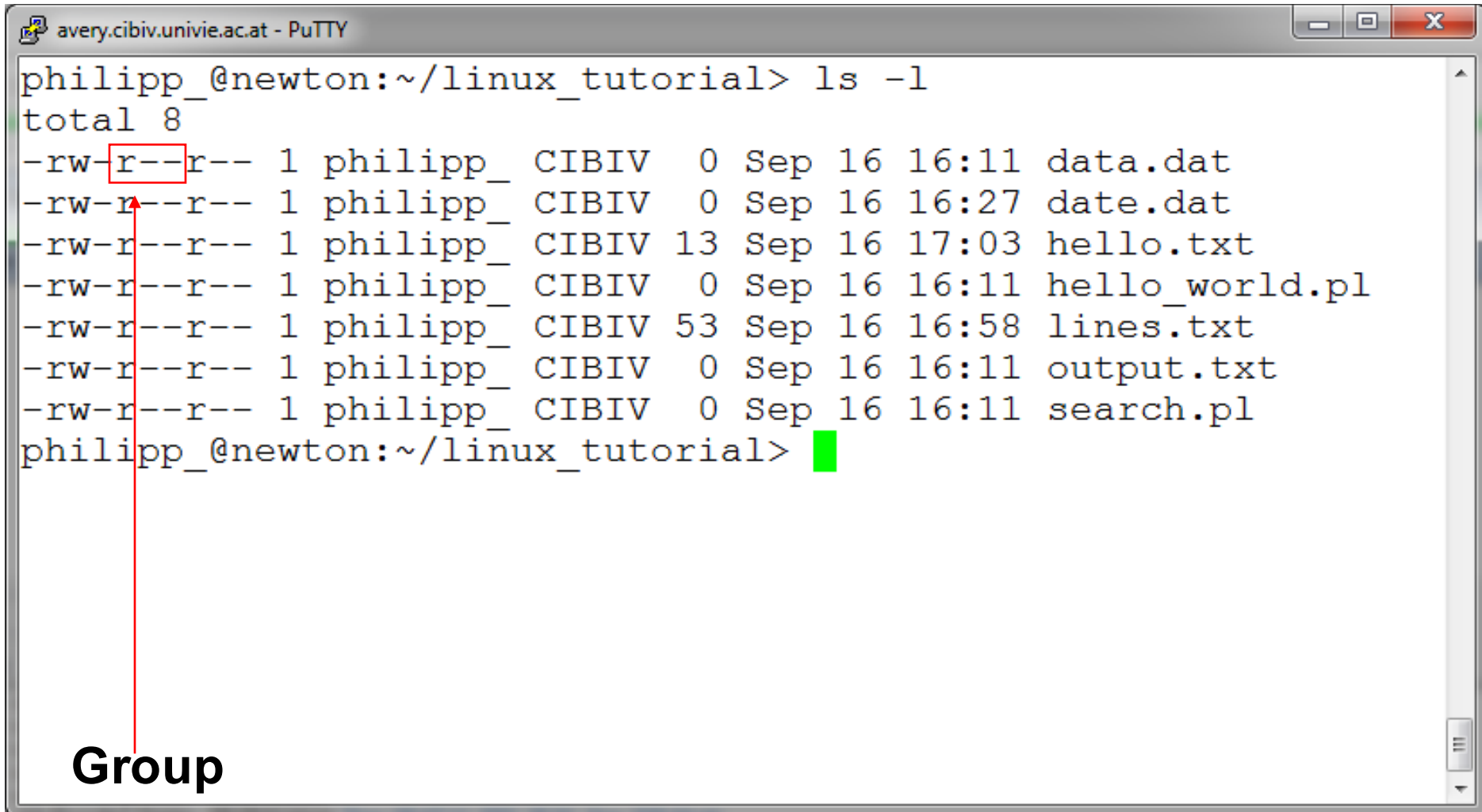
File Permissions



```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/linux_tutorial> ls -l
total 8
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 data.dat
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:27 date.dat
-rw-r--r-- 1 philipp_ CIBIV 13 Sep 16 17:03 hello.txt
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 hello_world.pl
-rw-r--r-- 1 philipp_ CIBIV 53 Sep 16 16:58 lines.txt
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 output.txt
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 search.pl
philipp_@newton:~/linux_tutorial>
```

User (you)

File Permissions



```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/linux_tutorial> ls -l
total 8
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 data.dat
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:27 date.dat
-rw-r--r-- 1 philipp_ CIBIV 13 Sep 16 17:03 hello.txt
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 hello_world.pl
-rw-r--r-- 1 philipp_ CIBIV 53 Sep 16 16:58 lines.txt
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 output.txt
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 search.pl
philipp_@newton:~/linux_tutorial>
```

Group



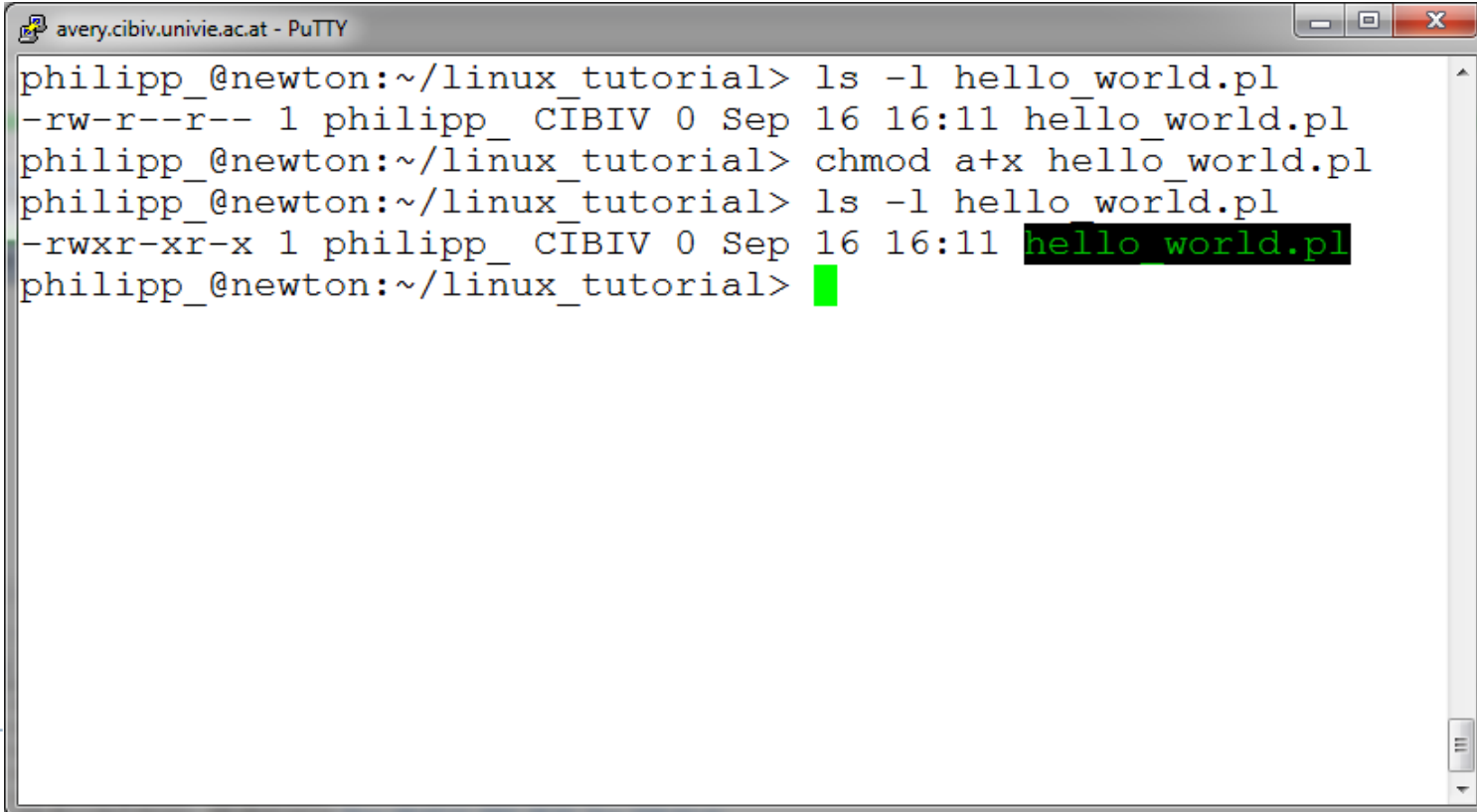
File Permissions

```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/linux_tutorial> ls -l
total 8
-rw-r--r-- 1 philipp_ CIBIV  0 Sep 16 16:11 data.dat
-rw-r--r-- 1 philipp_ CIBIV  0 Sep 16 16:27 date.dat
-rw-r--r-- 1 philipp_ CIBIV 13 Sep 16 17:03 hello.txt
-rw-r--r-- 1 philipp_ CIBIV  0 Sep 16 16:11 hello_world.pl
-rw-r--r-- 1 philipp_ CIBIV 53 Sep 16 16:58 lines.txt
-rw-r--r-- 1 philipp_ CIBIV  0 Sep 16 16:11 output.txt
-rw-r--r-- 1 philipp_ CIBIV  0 Sep 16 16:11 search.pl
philipp_@newton:~/linux_tutorial>
```

“The World”

Command: chmod

- ▶ If you own the file, you can change its permissions with “chmod”
 - ▶ Syntax: `chmod [user/group/others/all]+[permission] [file(s)]`
 - ▶ Below we grant execute permission to all:



```
avery.cibiv.univie.ac.at - PuTTY
philipp_@newton:~/linux_tutorial> ls -l hello_world.pl
-rw-r--r-- 1 philipp_ CIBIV 0 Sep 16 16:11 hello_world.pl
philipp_@newton:~/linux_tutorial> chmod a+x hello_world.pl
philipp_@newton:~/linux_tutorial> ls -l hello_world.pl
-rwxr-xr-x 1 philipp_ CIBIV 0 Sep 16 16:11 hello_world.pl
philipp_@newton:~/linux_tutorial>
```

Running a program (a.k.a. a job)

- ▶ Make sure the program has executable permissions
- ▶ Use “./” to run the program

- ▶ **Example:**

```
wget http://www.cibiv.at/~philipp/files/bwa.tar.gz  
tar xvfz bwa.tar.gz  
chmod u+x bwa  
./bwa
```



ssh, scp

- ▶ ssh is used to securely log in to remote systems, successor to telnet

- ▶ ssh [username]@[hostname]

- ▶ Try:

ssh yourusername@localhost

Type “exit” to log out of session

- ▶ Scp is used to copy files to/from remote systems, syntax is similar to cp:

- ▶ scp [local path] [username]@[hostname]:[remote file path]

- ▶ Try:

- ▶ **scp hello.txt yourusername@localhost:scp-test.txt**

Running a program: an example

- ▶ Running the sample perl script “hello_world.pl”



The image shows a PuTTY terminal window titled "avery.cibiv.univie.ac.at - PuTTY". The terminal displays the following text:

```
philipp_@newton:~/linux_tutorial> ./hello_world.pl  
Hello World!  
philipp_@newton:~/linux_tutorial> █
```

The terminal output shows the command `./hello_world.pl` being executed, resulting in the output `Hello World!`. A green cursor is visible on the line following the prompt.



Summary: most important tools

- ▶ ls, cd, mkdir, rm, mv, cp
- ▶ less, head/tail, cat
- ▶ grep
- ▶ tar/g(u)zip/unzip
- ▶ ssh/scp
- ▶ cut
- ▶ sort
- ▶ wc
- ▶ du
- ▶ awk
- ▶ Additional commands:
 - ▶ top
 - ▶ clear
 - ▶ find
 - ▶ history
 - ▶ time



Exercises 19-28



Unix Web Resources

- ▶ <http://www.ee.surrey.ac.uk/Teaching/Unix/>
- ▶ <http://www.ugu.com/sui/ugu/show?help.beginners>
- ▶ <http://en.wikipedia.org/wiki/Unix>



Command: wc

- ▶ To count the characters, words, and lines in a file use “wc”
- ▶ The first column in the output is lines, the second is words, and the last is characters



Command: top

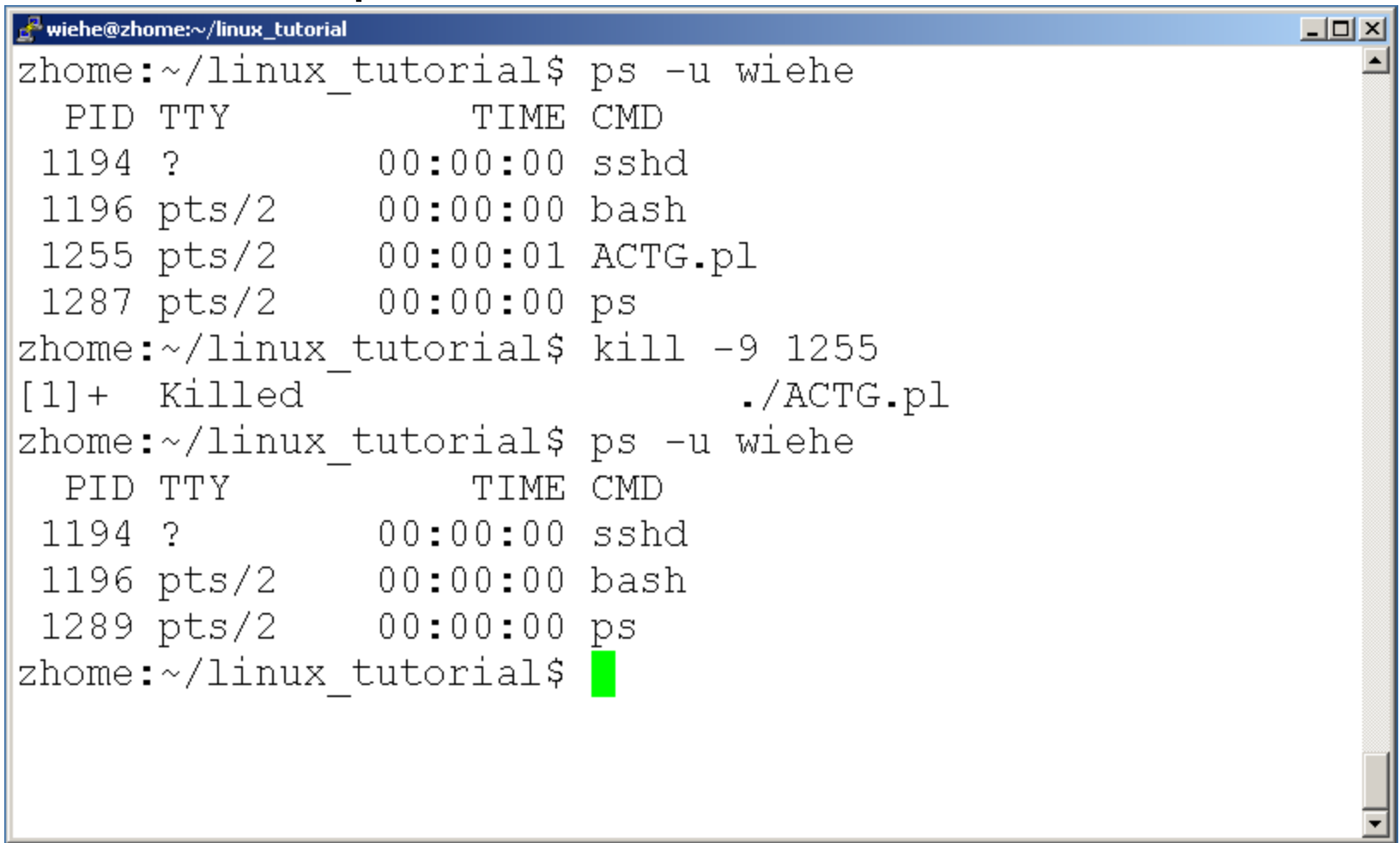
- ▶ To view the CPU usage of all processes:

```
wiehe@zhome:~/linux_tutorial
top - 13:46:33 up 50 days,  4:26,  2 users,  load avera
Tasks:  total,      running,      sleeping,      stoppe
Cpu(s):    us,      sy,      ni,      id,      w
Mem:      total,      used,      free,
Swap:     total,      used,      free,
█
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM
3403	root	15	0	0	0	0	S	0.7	0.0
1	root	16	0	1604	324	292	S	0.0	0.0
2	root	RT	0	0	0	0	S	0.0	0.0
3	root	34	19	0	0	0	S	0.0	0.0
4	root	RT	0	0	0	0	S	0.0	0.0
5	root	34	19	0	0	0	S	0.0	0.0
6	root	RT	0	0	0	0	S	0.0	0.0
7	root	34	19	0	0	0	S	0.0	0.0
8	root	RT	0	0	0	0	S	0.0	0.0
9	root	34	19	0	0	0	S	0.0	0.0

Command: kill

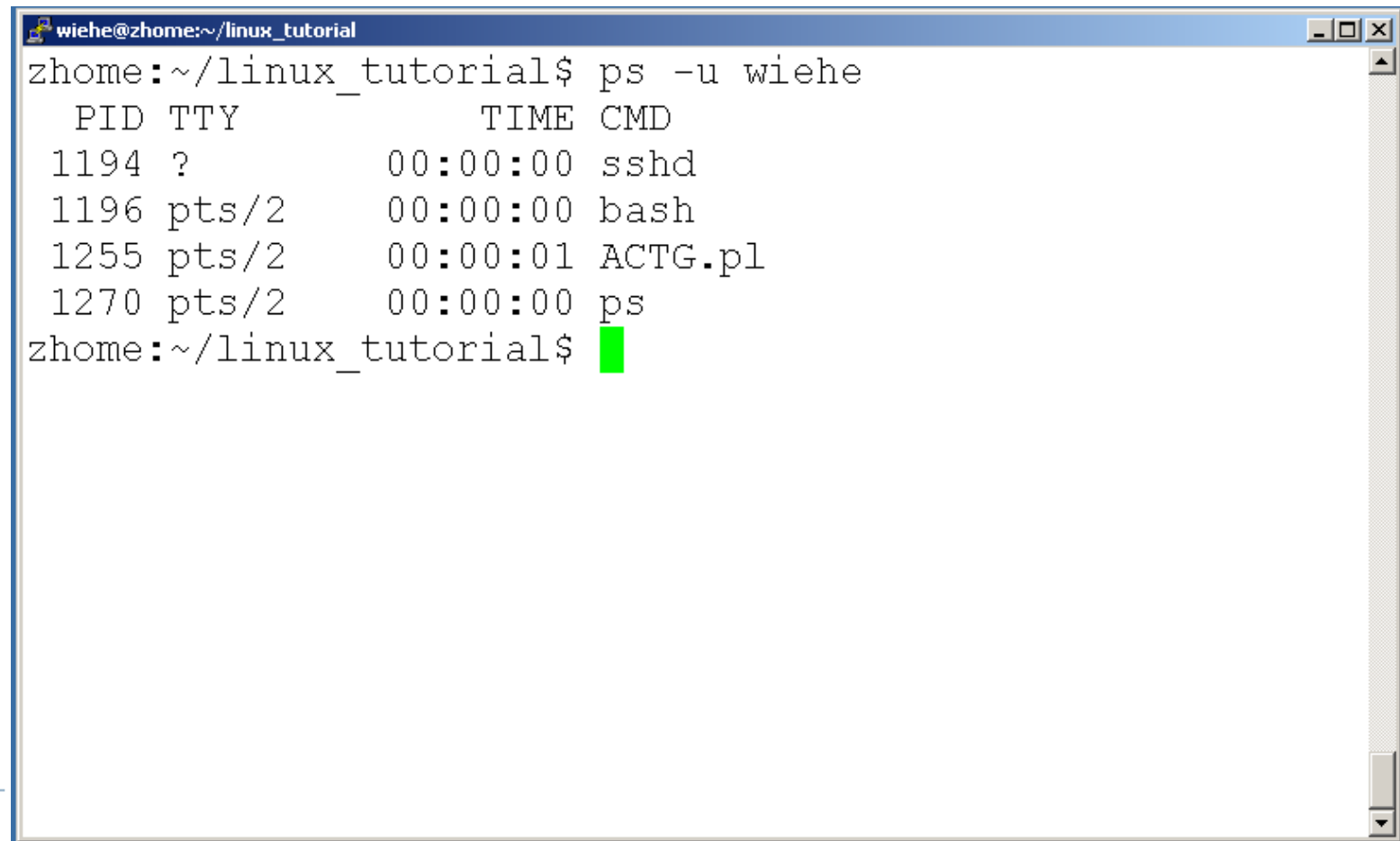
- ▶ To terminate a process use “kill”



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ps -u wiehe
  PID TTY          TIME CMD
 1194 ?            00:00:00 sshd
 1196 pts/2        00:00:00 bash
 1255 pts/2        00:00:01 ACTG.pl
 1287 pts/2        00:00:00 ps
zhome:~/linux_tutorial$ kill -9 1255
[1]+  Killed                  ./ACTG.pl
zhome:~/linux_tutorial$ ps -u wiehe
  PID TTY          TIME CMD
 1194 ?            00:00:00 sshd
 1196 pts/2        00:00:00 bash
 1289 pts/2        00:00:00 ps
zhome:~/linux_tutorial$
```


Command: ps

- ▶ To view the processes that you're running:



```
wiehe@zhome:~/linux_tutorial
zhome:~/linux_tutorial$ ps -u wiehe
  PID TTY          TIME CMD
 1194 ?            00:00:00 sshd
 1196 pts/2        00:00:00 bash
 1255 pts/2        00:00:01 ACTG.pl
 1270 pts/2        00:00:00 ps
zhome:~/linux_tutorial$
```

Command: diff

- ▶ To compare to files for differences use “diff”
 - ▶ Try: `diff /dev/null hello.txt`
 - ▶ `/dev/null` is a special address -- it is always empty, and anything moved there is deleted



AWK

```
awk ' pattern { action } '
```

Line oriented

1. Tries to match the pattern to the line
2. If it is a match execute the action
3. Automatically split the data by whitespace
4. No pattern means match everything.



Special variables

Awk **automatically splits** the input by whitespace (spaces and tabs) and assigns names to them:

\$0 the entire line

\$1 first field

\$2 second field

...

NF the number of fields

NR the number of the current line



Operators

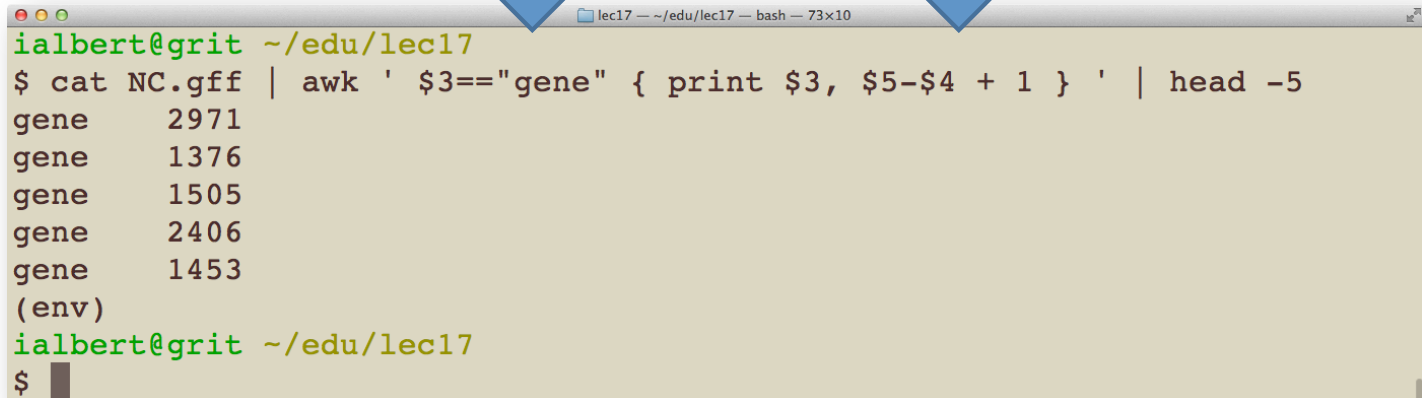
- `+ - * /` for numerical context
- `> <` for comparison
- `%` modulo division (remainder of division)
- `<space>` string concatenation
- `==, !=` equal, not equal
- `~, !~` match, no match (regular expressions)



AWK program in action

matching

action



```
ialbert@grit ~/edu/lec17
$ cat NC.gff | awk ' $3=="gene" { print $3, $5-$4 + 1 } ' | head -5
gene    2971
gene    1376
gene    1505
gene    2406
gene    1453
(env)
ialbert@grit ~/edu/lec17
$
```



List the gene names and sizes

```
split.awk (~/.edu/lec17)
1
2 $3 == "gene" {
3     split($9, x, ";")
4
5     split(x[1], y, " ")
6
7     print $3, y[2]
8 }
```

Write one step at a time

```
lec17 -- ~/.edu/lec17 -- bash -- 43x10
ialbert@grit ~/.edu/lec17
$ cat NC.gff | awk -f split.awk | head -5
gene      "NP"
gene      "VP35"
gene      "VP40"
gene      "GP"
gene      "VP30"
(env)
ialbert@grit ~/.edu/lec17
$
```



Full program

```
split.awk (~/.edu/lec17)
1
2 $3 == "gene" {
3
4     # Split the 9th column by ;
5     split($9, x, ";")
6
7     # The gene name is in the first resulting element.
8     # Split that by space. The gene name is the second
9     # element.
10    split(x[1], y, " ")
11
12    # Remove the double quotes around the gene name
13    name = y[2]
14
15    # Global substitution of " with empty space.
16    # Since " is also a special character we have to
17    # write it as \"
18    gsub("\\\"", "", name)
19
20    # Print the type of the feature, the name
21    print $3, name, $5 - $4 + 1
22 }
```

```
lec17 -- ~/.edu/lec17 -- bash -- 45x10
ialbert@grit ~/.edu/lec17
$ cat NC.gff | awk -f split.awk | head -5
gene      NP        2971
gene      VP35     1376
gene      VP40     1505
gene      GP       2406
gene      VP30     1453
(env)
ialbert@grit ~/.edu/lec17
$
```


The whitespace curse: spaces and tabs

- Many tools will auto-split by whitespace → this was thought to be convenient but is also the source of extremely subtle errors → leads to a column shift in a tab file if a field contains spaces
- **Always specify the character to be split by!**
- This refers to programming languages as well!

Do not use the **split()** methods with their default behavior (Python , Perl etc) unless you perfectly understand what they do)

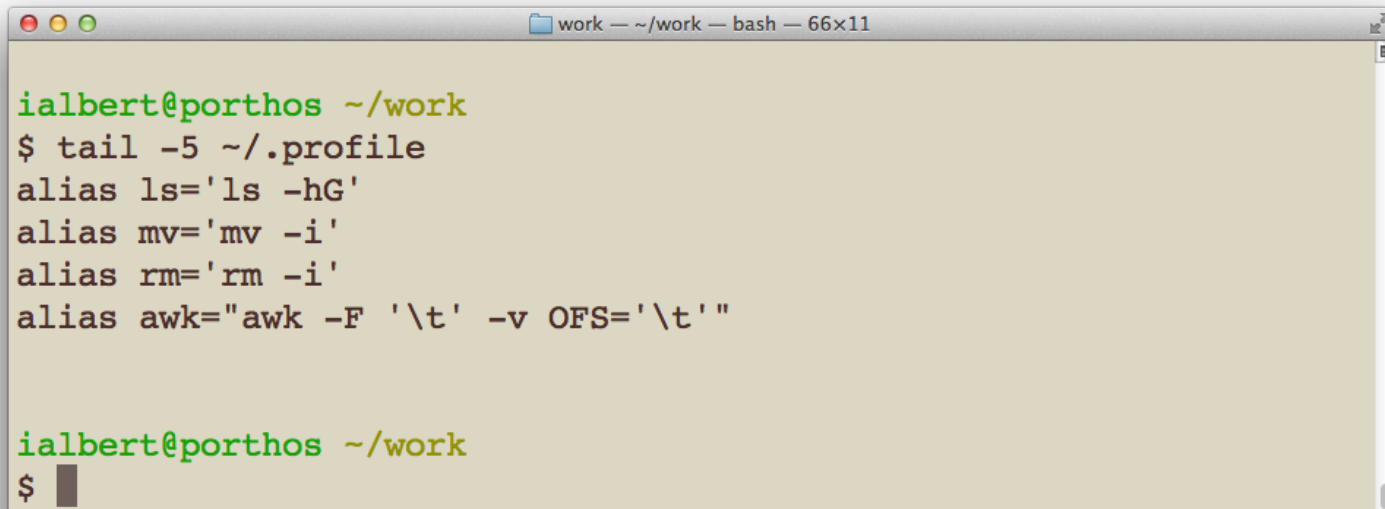


Use tabs as input and output separator

```
alias awk="awk -F '\t' -v OFS='\t'"
```

Tip: you can add this to the **.profile** or **.bashrc** file in your root folder so that it is activate all the time

Note: file names that start with a dot **.** are only listed if you do a **ls -a**



```
work -- ~/work -- bash -- 66x11

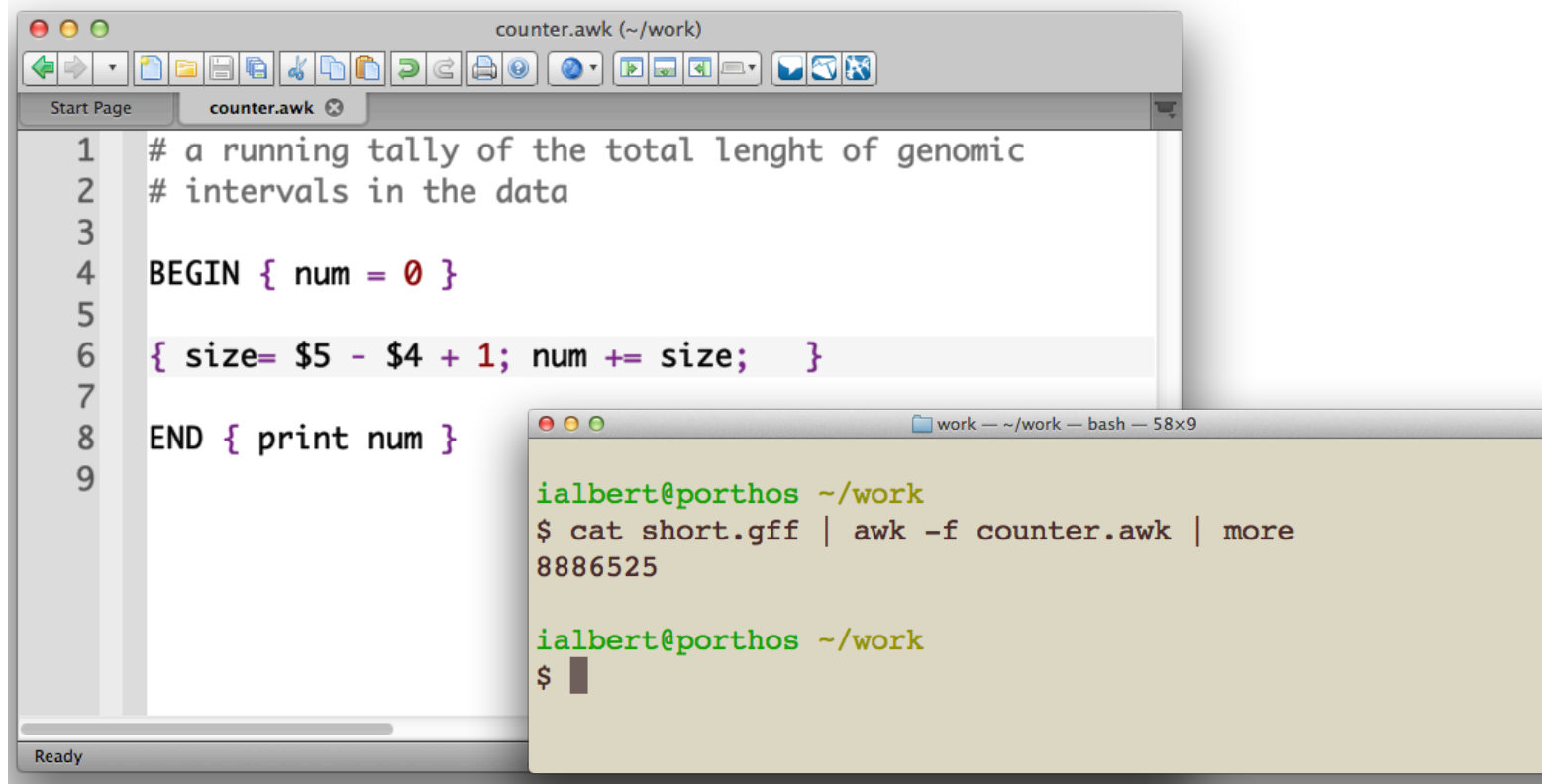
ialbert@porthos ~/work
$ tail -5 ~/.profile
alias ls='ls -hG'
alias mv='mv -i'
alias rm='rm -i'
alias awk="awk -F '\t' -v OFS='\t'"

ialbert@porthos ~/work
$
```



Special patterns

- BEGIN → before the stream starts
- END → after the stream ends



The image shows a code editor window titled 'counter.awk (~/.work)' and a terminal window. The code editor contains the following text:

```
1 # a running tally of the total length of genomic
2 # intervals in the data
3
4 BEGIN { num = 0 }
5
6 { size= $5 - $4 + 1; num += size; }
7
8 END { print num }
9
```

The terminal window shows the execution of the script:

```
ialbert@porthos ~/.work
$ cat short.gff | awk -f counter.awk | more
8886525

ialbert@porthos ~/.work
$
```

Advanced AWK

- conditionals : **if**
- loops: **for, while**
- **break, continue**
- **associative data structures (hash, dictionary)**

You can do all that though at that point
it is probably better to learn Python

But you **can do a lot with just basic awk!**

**Awk's power comes from its simplicity – more complex programming
is done with specialized programming languages**

