H3ABioNet
Pan African Bioinformatics Network for H3Africa

Introduction to Unix

Sumir Panji / Amel Ghouila

Material adapted from: H3ABioNet Introduction to Bioinformatics Training and NGS Bioinformatics Course Africa 2021

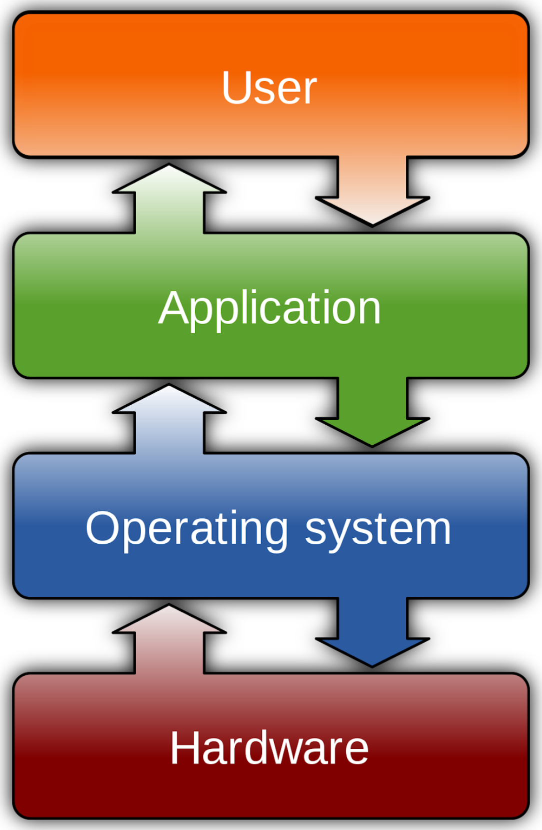
Overview

- Introduction to unix/Linux and uses
- Interacting with Linux
- Types of input and output
- Linux command line structure
- Navigating through the file system
- Some Linux navigation commands and examples
- Working with files in Linux
- Editing files and file permissions
- Some Linux commands for looking at file contents and examples
- Some Linux commands for searching for patterns and examples
- Some Linux commands for extracting contents from a file and examples
- Output redirection of Linux commands
- Combining multiple Linux commands
- Useful tips

Some definitions

- Operating system – a system comprising of software that manages resources such as hardware and software applications and bridges functions between the software and hardware

- Kernel – a computer program that is the main layer between the operating system and hardware and controls the systems functions



- https://en.wikipedia.org/wiki/Kernel_(operating_system)
- https://en.wikipedia.org/wiki/Operating_system

What is Unix / Linux

- Types of operating systems

- Unix – originally developed at Bell labs for AT&T

- A computer operating system that grew in popularity

- Variants of Unix include Linux and the MacOS

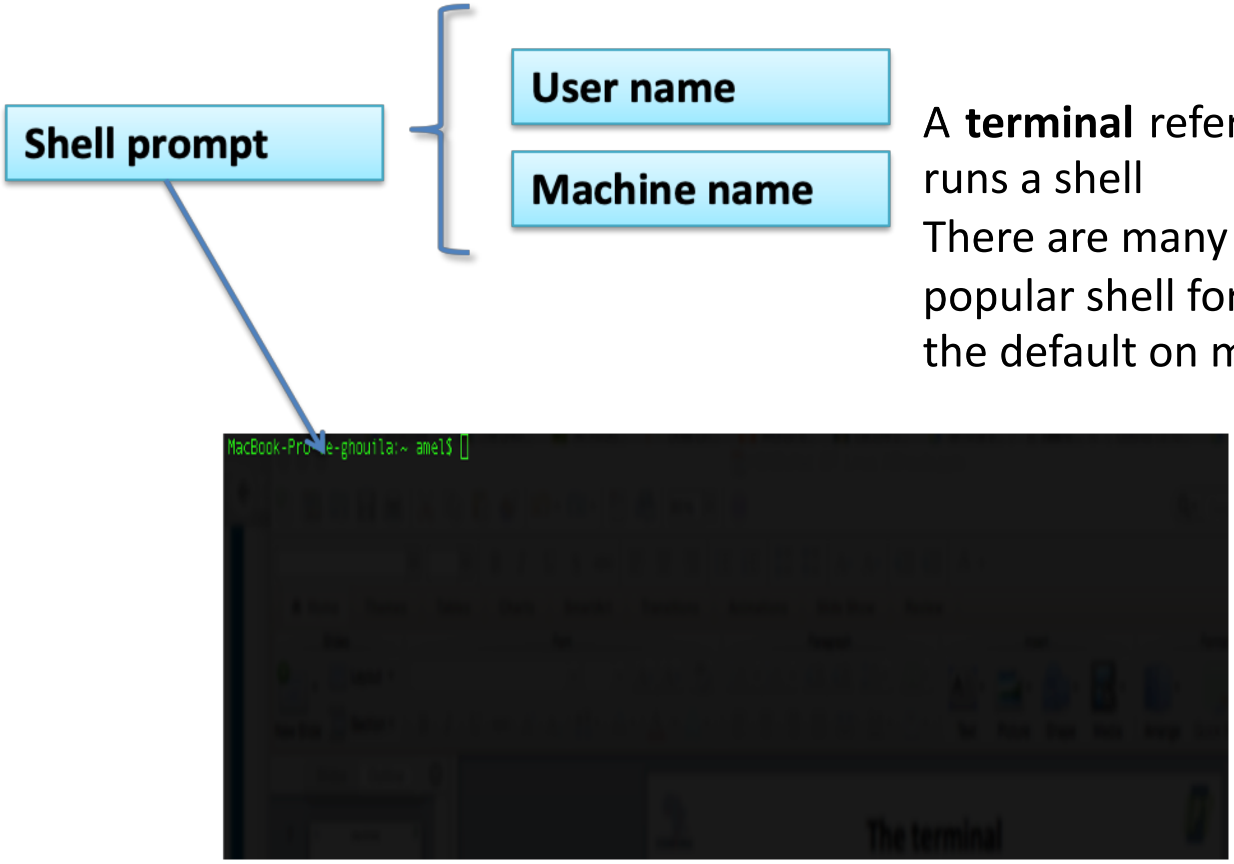
- Linux – types of open source Unix-like operating systems based on a linux kernel created by Linus Torvalds e.g. Ubuntu, Fedora, RedHat

- https://en.wikipedia.org/wiki/Unix
- https://en.wikipedia.org/wiki/Linux

Why use Linux

- Most biological data is in text format or have tools that can access them e.g. fasta files, NGS data, RNA-Seq data, BAM, output of various biological software
- Linux is particularly suitable for working and manipulating big and numerous files without having to move out to other applications
- Powerful and flexible commands that can be used to process and analyse this data
- Linux commands can be combined in an almost unlimited fashion using pipes
- Linux / Unix is the standard operating system on most large computer systems in scientific research, in the same way that Microsoft Windows is the dominant operating system on desktop PCs
- Best multi-user and multi tasking OS, this is why it is the preferred operating system for large-scale scientific computing

Main interface to access Unix / Linux



```
MacBook-Pro-de-ghouila:~ amel$
```

A **terminal** refers to a wrapper program which runs a shell
There are many different Unix shells, the most popular shell for interactive use include Bash: the default on most Linux installations

Types of Input / Output

- Standard input (stdin)– user input to a command / system e.g. from a keyboard such as a command, or a keyboard and an input file e.g. cat file.fasta

- Standard output (stdout)– where the results of the input go to e.g. the computer screen or a file e.g. cat file.fasta > new_file.fasta

- Standard error (stderr) – any error messages that occur from executing the command

Linux command line structure

- Linux commands have the following structure:

Command name –[options] [arguments]
e.g. grep –c input_file.fasta

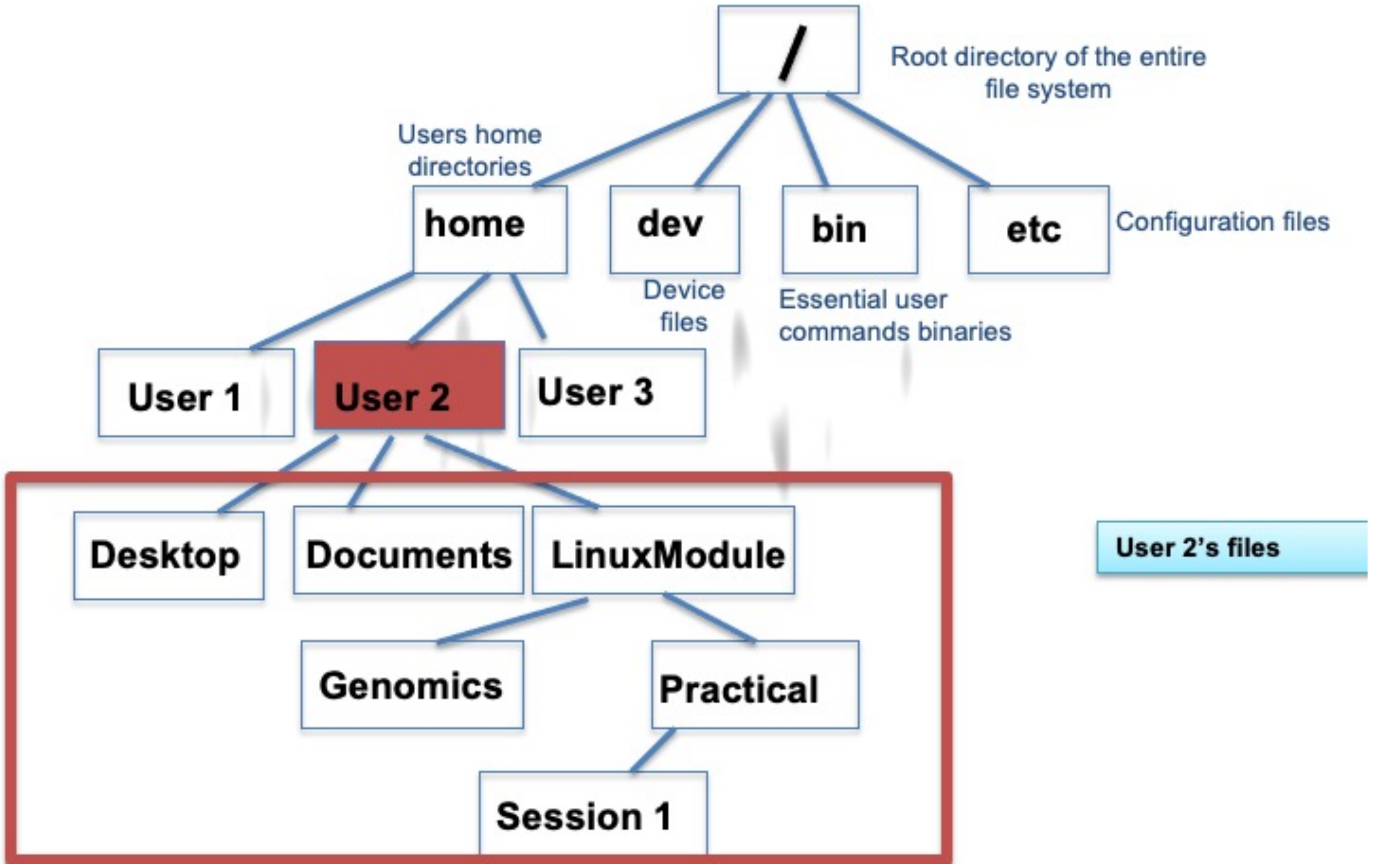
- Command line options can be found by typing man command name in the terminal

e.g. man grep

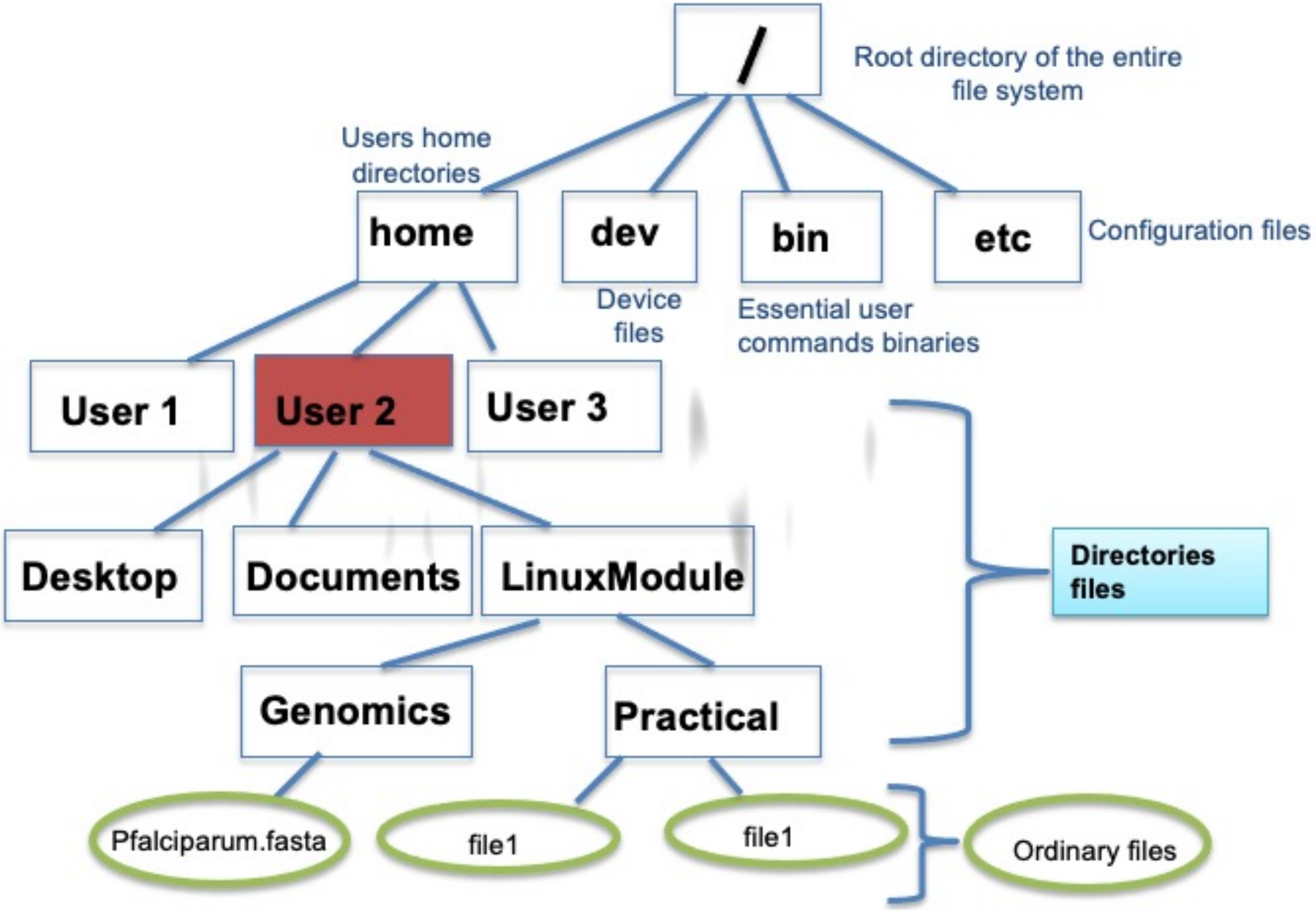
- Tip – if you enter a Linux command without any options, or need to exit the command press the control key and the c key on the keyboard: ctrl c

- Tip – if learning Linux, look at examples of usage for the command you are interested in on online forums to get better ideas of how they can be used

What does one deal with – Directories / files



What does one deal with – Directories / files



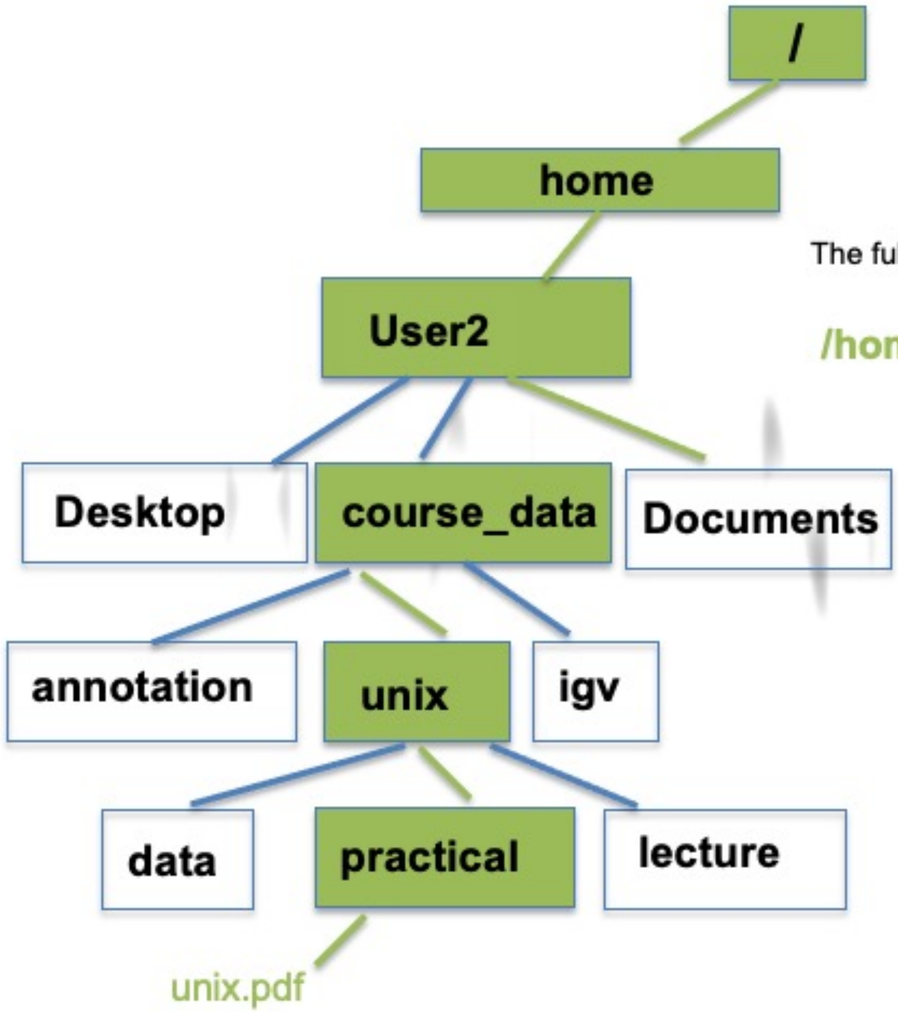
Basic Navigation

- When you first log in on a UNIX system, the working directory is your home directory.
- While working you will be associated to one directory called the working directory or the current directory
- An abbreviation of the working directory is displayed is displayed as part of the prompt on your terminal
- The command pwd gives the absolute path of the working directory e.g /home/User 2/Genomics/

Basic Navigation

- A path locates a given file in the system hierarchy
- An absolute path in the file system hierarchy for a given file or folder describes the parents all the way up to the root e.g. /home/User 2/Genomics/Pfalciparum.fasta
- A relative path describes the path to the file starting from the current working directory e.g if in User 2, then Genomics/Pfalciparum.fasta

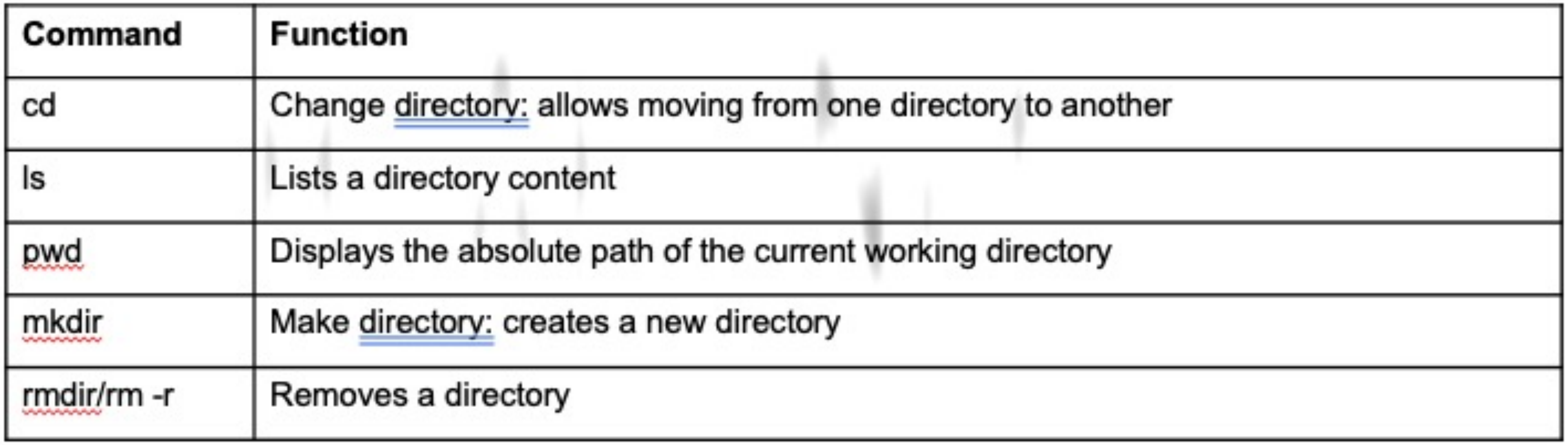
Basic Navigation



The full path to unix.pdf is:

/home/User2/course_data/unix/practical/unix.pdf

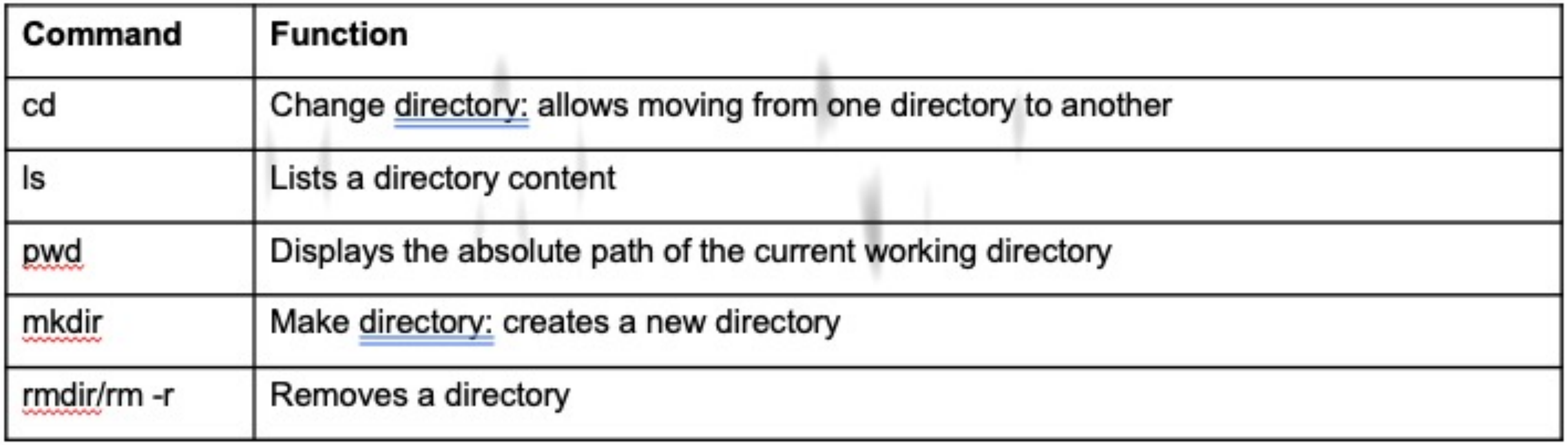
Basic Navigation – useful commands



Command	Function
cd	Change directory: allows moving from one directory to another
ls	Lists a directory content
pwd	Displays the absolute path of the current working directory
mkdir	Make directory: creates a new directory
rmdir/rm -r	Removes a directory

- cd ../
- cd ../../../
- cd /home/User2/
- cd ~ specifies your home directory and paths starting from your home directory e.g.~/course_data

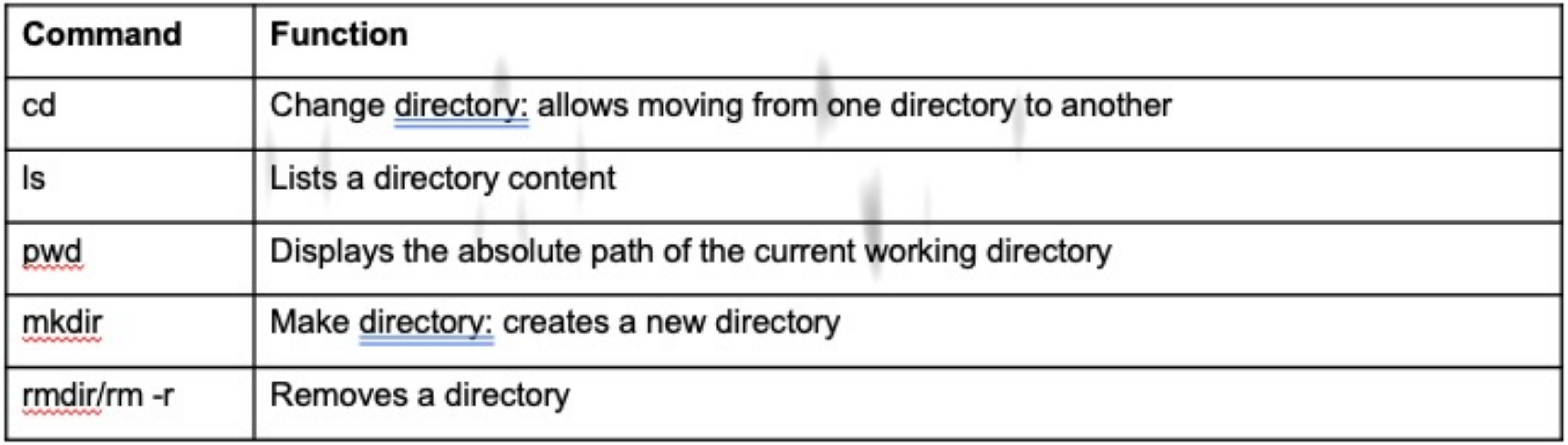
Basic Navigation – useful commands



Command	Function
cd	Change directory: allows moving from one directory to another
ls	Lists a directory content
pwd	Displays the absolute path of the current working directory
mkdir	Make directory: creates a new directory
rmdir/rm -r	Removes a directory

- ls lists the content of the current directory by default
- Command structure ls [OPTION] [dirname]
- Some useful options:
    - -l: shows sizes, modified date and time, file or folder name and owner of file and permissions
    - - a: List all files including hidden file starting with '.'
    - -lh: shows file sizes in human readable format
    - -R: recursively lists sub-directories
    - -lS: sorting by file sizes
    - ls ../../../ - list the files in three directories above my current working directory

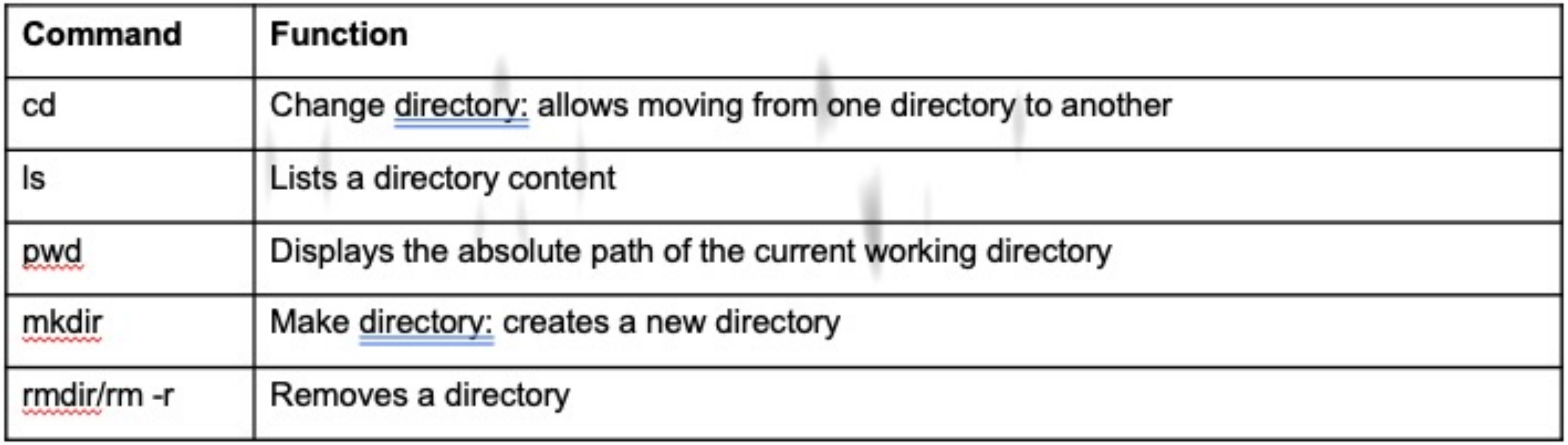
Basic Navigation – useful commands



Command	Function
cd	Change directory: allows moving from one directory to another
ls	Lists a directory content
pwd	Displays the absolute path of the current working directory
mkdir	Make directory: creates a new directory
rmdir/rm -r	Removes a directory

- mkdir: makes a directory
- Command structure: mkdir dirname [path]
- mkdir dirname: would create a directory with the specified dirname
- The new created directory will be created in your current working directory
- If you want to create it elsewhere, you have to specify the path: mkdir dirname path

Basic Navigation – useful commands



Command	Function
cd	Change directory: allows moving from one directory to another
ls	Lists a directory content
pwd	Displays the absolute path of the current working directory
mkdir	Make directory: creates a new directory
rmdir/rm -r	Removes a directory

- rmdir: removes a directory
- Command structure: rmdir dirname [path]
- It would remove the directory called dirname
- The directory should be in your current working directory
- If you want to remove it from elsewhere, you have to specify the path: rmdir dirname path
- rmdir works if there is no contents in the directory – if not an error message will appear: “Directory not empty”

Working with files in Linux

Copy, move and remove

- cp: copy files and directories

```
cp <pathfrom> <path to>
cp /home/User2/practical/unix.pdf .
cp /home/User2/practical/unix.pdf /home/User3/practical/
```

- mv: move or rename files and directories

```
mv /home/User2/practical1/unix.pdf /home/User2/pratical2/
```

- rm: remove files and directories

```
 rm pathname
rm /home/User2/practical/unix.pdf
```

Working with files in Linux - cp

- Simplest form: cp file1 file2

  ➔ Copy the contents of file1 into file2. If file2 does not exist, it is created. Otherwise, file2 is silently overwritten with the contents of file1.

- cp filename dirpath

  ➔ Make a copy of the file (or directory) into the specified destination directory

Working with files in Linux - cp

- To prevent over writing file contest, can use the interactive mode with the cp command with the option -i

- cp -i file1.fasta file2.fasta

  ➔Same as the previous one. However, if file2 exists, the user is notified before overwriting file2 with the content of file1

- cp –R pathdir1 pathdir2

  ➔ Copy the contents of the directory dir1. If directory dir2 does not exist, it is created. Otherwise, it creates a directory named dir1 within directory dir2

Working with files in Linux - mv

The mv command moves or renames files and directories depending on how it is used

- **To rename a file:**

```
mv filename1 filename2
```

If file2 exists, its contents are silently replaced with the contents of file1. To avoid overwriting, use the interactive mode:

```
mv -i  filename1 filename2
```

- **To move a file (or a directory) to another directory**:

```
mv file  dirpath
```

- **To move different files (or a directory) to another directory**:

```
mv file1 file2 file3  dirpath
```

- **To move directory to another directory**:

```
mv dir1  dir2
```

If dir2 does not exist, then dir1 is renamed dir2. If dir2 exists, the directory dir1 is moved within directory dir2

Working with files in Linux - rm

The rm command deletes files and directories

- **To remove a file:**

```
rm filename
```

- **To remove many files:**

```
rm filename1 filename2
```

- **Add the interactive mode to prompt user before deleting with –i**

```
rm -i filename1 filename2
```

- **Delete directories with all their contents**

```
rm -r dir1 dir2
```

Working with files in Linux – be careful of rm

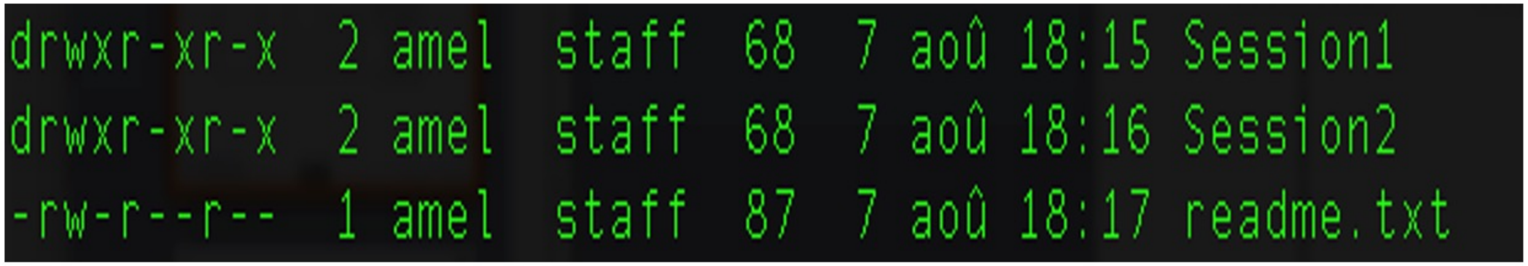
- Linux does not have an undelete command or a recycle bin
- You can inflict terrific damage on your system with rm if you are not careful, particularly with wildcards e.g. delete all your data files
- Once you delete something with rm, it's gone (unless your systems administrator can bring it from backups)
- Try this trick before using rm: construct your command using ls instead first

File names in Linux

- No real distinction between the names of ordinary files and the names of directory files
- No two files in the same directory can have the same name
- Files in different directories can have the same name
- Linux is case-sensitive: course_data, course_Data and Course_data are different and would represent three distinct files
- In most cases, file extensions are optional (.txt, .exe, etc.)
- Tip – hit the tab key to after typing the first few letters of the filename

Working with files - permissions

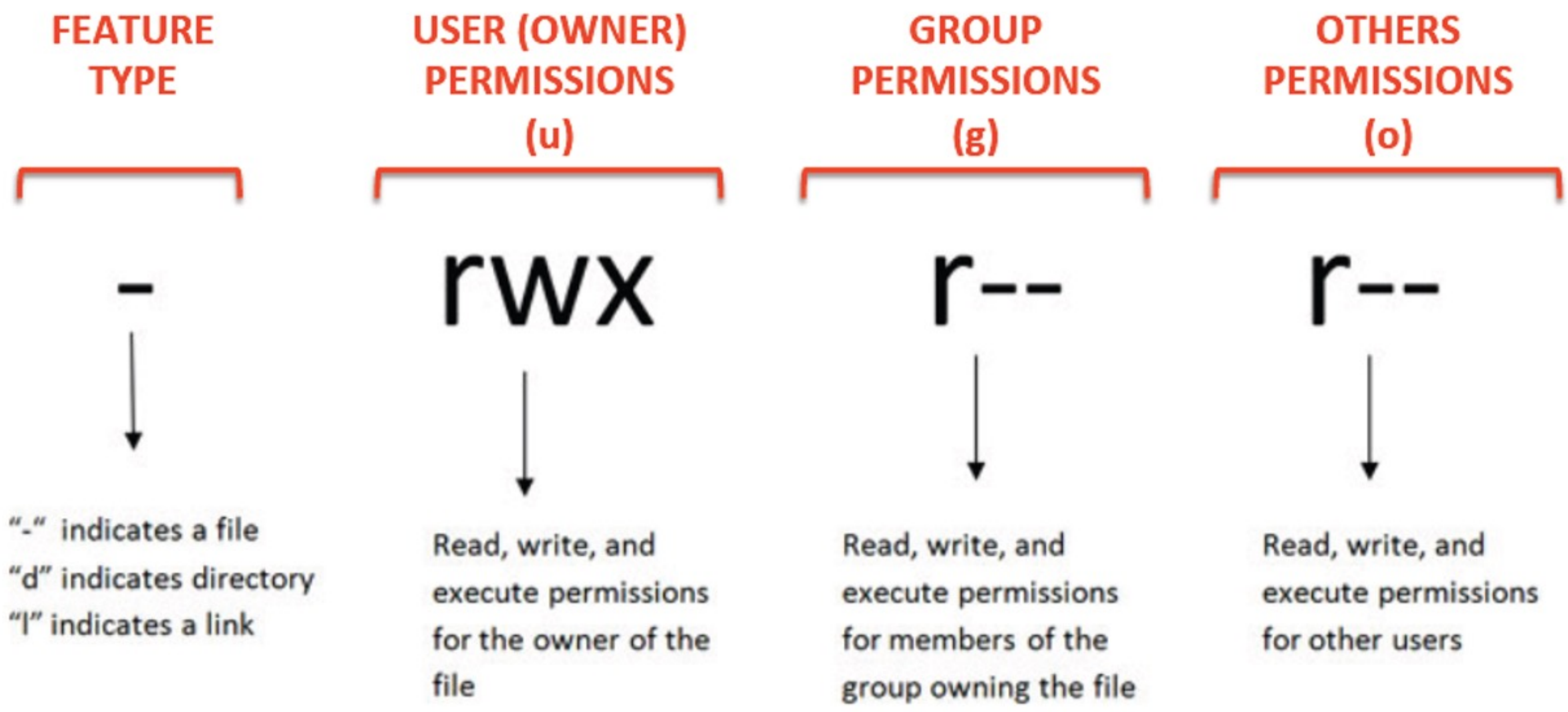
ls –l



```
drwxr-xr-x  2 amel  staff  68  7 aoû 18:15 Session1
drwxr-xr-x  2 amel  staff  68  7 aoû 18:16 Session2
-rw-r--r--  1 amel  staff  87  7 aoû 18:17 readme.txt
```

```
----------     -------  -------  --------  -----------  -------------
    |             |        |        |          |              |
    |             |        |        |          |          File Name
    |             |        |        |          |
    |             |        |        |          +---  Modification Time
    |             |        |        |
    |             |        |        +-------------   Size (in bytes)
    |             |        |
    |             |        +----------------------        Group
    |             |
    |             +-------------------------------        Owner
    |
    +---------------------------------------------   File Permissions
```

Working with files - permissions



Permissions are broken into 4 sections

FEATURE TYPE

USER (OWNER) PERMISSIONS (u)

GROUP PERMISSIONS (g)

OTHERS PERMISSIONS (o)

```
-    rwx    r--    r--
```

"-" indicates a file
"d" indicates directory
"l" indicates a link

Read, write, and execute permissions for the owner of the file

Read, write, and execute permissions for members of the group owning the file

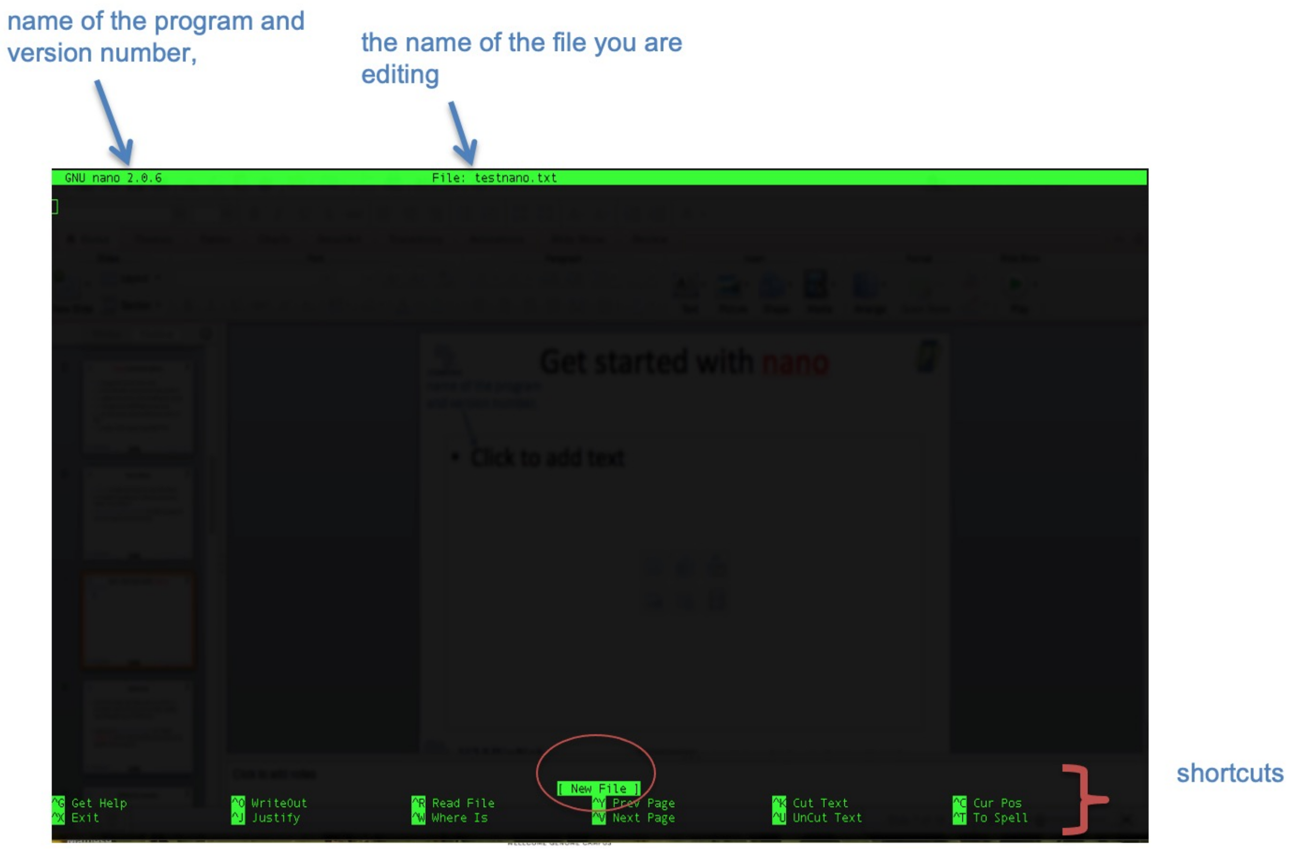
Read, write, and execute permissions for other users

Source: www.pluralsight.com

Working with files - permissions

- The chmod command is used to change the permissions of a file or a directory
- Syntax: chmod [options] permissions filename.fasta
- Only the owner of the file can use chmod to change the permissions
- Permissions define permissions for the owner, the group of users and anyone else (others)
- There are two ways to specify the permissions:
    - ✓Symbols: alphanumeric characters
    - ✓Octals: digits (0 to 7)
    - ✓e.g. chmod 755 filename.fasta

Editing file contents – Text editors


name of the program and version number,
the name of the file you are editing
shortcuts

```
GNU nano 2.0.6                          File: testnano.txt

                                          [ New File ]
^G Get Help      ^O WriteOut      ^R Read File     ^Y Prev Page     ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is      ^V Next Page     ^U UnCut Text    ^T To Spell
```

- nano: a simple and easy-to-use text editor - installed by default in many other Linux distributions
- gedit is also very easy to use
- vim, emacs, Geany: excellent programs, but do require some learning

Looking at file contents

- **cat**: view the content of a short file (prints the whole file contents to stdout)

```
cat filename.fasta
```

- **more**: view the content of a long file and navigate through it

```
more filename.fasta
```

- **less**: view the content of a long file, by portions

```
less filename.fasta
```

- **head**: view the first lines of a long file

```
head –n 100 filename.fasta
```

- **tail**: view the last lines of a long file

```
tail –n 100 filename.fasta
```

Looking at file contents - less

- The less command displays a text file content, one page at a time – differs from more as allows backward and forward scrolling

e.g. man less

less filename.fasta

- Move a page down: either use the page down key or space
- Move a page up: use the page up key
- To exit less, type q
- To go to the end of the text file, type g
- To find a pattern such as name in the file while using less, type /pattern e.g. /name

Looking at file contents – head / tail

- **head** command displays a text file content, by default: 10 first lines at a time

```
head [options] filename.fasta
```

- **tail** command displays a text file content, by default: 10 last lines at a time

```
tail [options] filename.fasta
```

- use –n to change the number of lines you want to display e.g. tail –n 100 filename.fasta

```
head –n 100 filename.fasta
```

Getting basic counts – wc

- wc prints newline, word, and byte counts for each file

```
wc –l filename.fasta
```

Some useful options:

- **-c**: print the byte counts
- **-m:** print the character counts
- **-l:** print the newline counts
- For more info about the different commands, remember to use man commandname and look at example usage on forums like stack exchange

Searching within files – grep

- **grep** ("**g**lobal **r**egular **e**xpression **p**rofile") is used to search for the occurrence of a specific pattern (regular expression…) in a file
- grep outputs the whole line containing that pattern

Examples:

Extract lines containing the term sequence from a file:

```
grep sequence filename.fasta
```

Extract lines that do not contain a certain pattern from a file like sequence:

```
grep –v  sequence  filename.fasta
```

Searching within files – grep

- **grep**: to search for the occurrence of a specific pattern (regular expression using the wildcards…) in a file

  grep **<pattern>** <filename>

  e.g. grep '>' filename.fasta

  e.g. grep '>' -c filename.fasta

  e.g. ls –l | grep *.fasta

- Prints out the lines that match the > character (fasta headings)
- Provides the count of the number of fasta headers within a file
- Prints out the file names from the ls command output that is piped to the grep command which end with .fasta

Basic file operation commands

- **sort**: reorder the content of a file “alphabetically”

syntax: sort [options] filename.fasta

- **uniq**: removes duplicated lines

syntax: uniq [options] filename.fasta

- **join**: compare the contents of 2 files, outputs the common entries

syntax: join [options] filename1.fasta filename2.fasta

- **diff**: compare the contents of 2 files, outputs the differences

syntax: diff [options] filename1.fasta filename2.fasta

Basic file operation commands – sort

- **sort** outputs a sorted order of the file content based on a specified sort key (default: takes entire input)

Syntax: sort [options] filename.fasta

- Sorted files are used as an input for several other commands so sort is often used in combination to other commands such as the uniq one

e.g. sort genelist_file.txt | uniq to get the unique genes in the file

- For [options] see man and forums for ideas on usage

Basic file operation commands – uniq

- **uniq** outputs a file with no duplicated lines i.e. a single entry per a line
- Uniq requires a sorted file as an input

  Syntax: uniq [options] sorted_filename.fasta
- Useful option is **-c** to output each line with its number of repeats
- e.g. sort genelist_file.txt | uniq –c to get a count of the number of times a gene name appears in the file
- For [options] see man and forums for ideas on usage

Comparing files – diff

- **diff** is used to compare 2 input files and displays the different entries

- Can be used to highlight differences between 2 versions of the same file

- Default output: common lines not showed, only different lines are indicated and shows what has been added (**a**), deleted (**d**) or changed (**c**)

```
diff [options] filename1.fasta filename2.fasta
```

- For [options] see man and forums for ideas on usage

Command redirection

- By default all results of the command will outputted to the stdout i.e. your screen
- Can redirect the output of the commands to a new file, or append to existing one
- Useful if wanting to pull out accession headers, or parts of a file for use in another application or analysis

Command redirection

- Syntax: command options filename.fasta.in > filename.fasta.out
- The second file will be created if it does not exist
- Careful – if the output file exists, it will be **overwritten** by the new output
- e.g. grep ’>’ filename.fasta > accessions.txt
- e.g. grep ‘>’ filename.fasta2 > accession.txt (will overwrite existing file)
- e.g. grep ‘>’ filename.fasta2 >>accession.txt (will append to an existing file)

Combining Linux commands – pipe operator

- As one becomes more familiar with unix / linux commands, one wants to perform multiple operations to get to a certain point

- Inefficient (painful) to run each command individually, generate and output file and input for the next command

- To take the output of one command and redirect it as the input for another command – use the | (pipe) character

- e.g. grep '>' | sort | uniq to get the unique fasta headers in a file
- e.g. grep '>' | sort | uniq –c > output_file.txt to get a count of the number of times a fasta header appears in a file stored in a file created called output_file.txt

H3ABioNet
Pan African Bioinformatics Network for H3Africa

Some useful tips

- Use tab completion - it will save you time and prevent errors!
- Build commands slowly
- man command_name often gives you help
- Always have a quick look at files with less or head to double check their format
- Watch out for data in headers and that you don’t accidentally grep some if you don’t want them
- If you did something smart but can’t remember what it was, try typing history
- Google is normally better at giving usage examples (prioritise stackoverflow.com results, they’re normally good)

H3ABioNet

H3ABioNet

Thank you!!!

Acknowledgements: Amel Ghouila and course material adapted from H3ABioNet IBT and NGS Bioinformatics Course Africa 2021

H3ABioNet
Pan African Bioinformatics Network for H3Africa