TECNICHE DI RAPPRESENTAZIONE E MODELLIZZAZIONE DEI DATI

— Part 1 —

(2 CFU out of 6 total CFU)

Link moodle: https://moodle2.units.it/course/view.php?id=11703

Teams code: 0ftoqj8

Main goals of this part of the course:

- know the basics of Linux commands, bash and Python languages,
- become familiar with different working environments (i.e. shell, Git repository, Jupiter notebook),
- write simple scripts to analyse data

Intro

SEPTEMBER 2023

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

OCTOBER 2023

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Timeslots:

Thursdays: 11:15 $\longrightarrow$ 13:45
Fridays: 9:15 $\longrightarrow$ 10:30 break 10:45 $\longrightarrow$ 12:00

Office:

Astronomical Observatory of Trieste
via G.B. Tiepolo 11, 34143 - Trieste
Office: 1301 (third floor)
Phone: 040 3199243

Please, always send an email to
milena.valentini@units.it
to schedule a meeting

Intro

Text books, bibliography and useful resources

- Numerical Python in Astronomy and Astrophysics - A Practical Guide to Astrophysical Problem Solving *(Authors: W. Schmidt and M. Völschow*
- Think Python, 2nd Edition - How to Think Like a Computer Scientist *(Author: A. B. Downey)*
- How to Think Like a Computer Scientist (https://openbookproject.net/thinkcs/python/english3e/index.html)
- Python Scripting for Computational Science (*Author: H. P. Langtangen)*
- Parallel Programming with Python *(Author: J. Palach)*
- https://www.python.org/
- https://github.com/sarusso/ProgrammingLab
- https://moodle2.units.it/course/view.php?id=7455

Intro

Lecture 1: Introduction to operative systems and main Linux commands

Lecture 2: Linux commands, working environments, Anaconda and Jupyter Notebook

Lecture 3: Essentials of bash and Git

Lecture 4: Python - introduction, main concepts, errors, variables, scripts

Lecture 5: Python - operators, conditions, functions

Lecture 6: Python - strings, lists, tuples, dictionaries

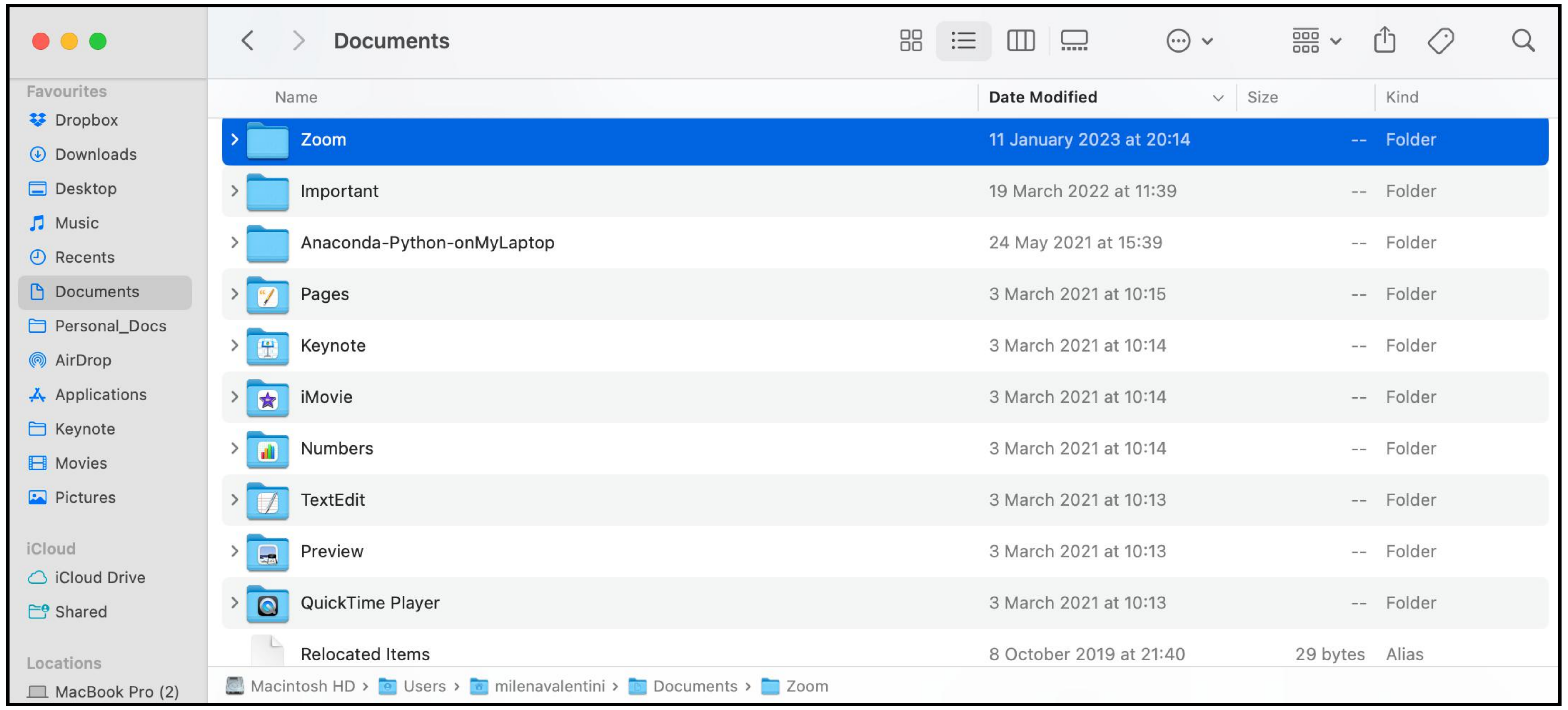
Lecture 7: Python - data structures, how to read/write from/a file, input/output, input from command line

Lecture 8: Python - classes, module import, libraries, arrays, objects, try/except, generators

Lecture 9: Python exercise and main ideas of parallelization with mpi4py

Useful working tools

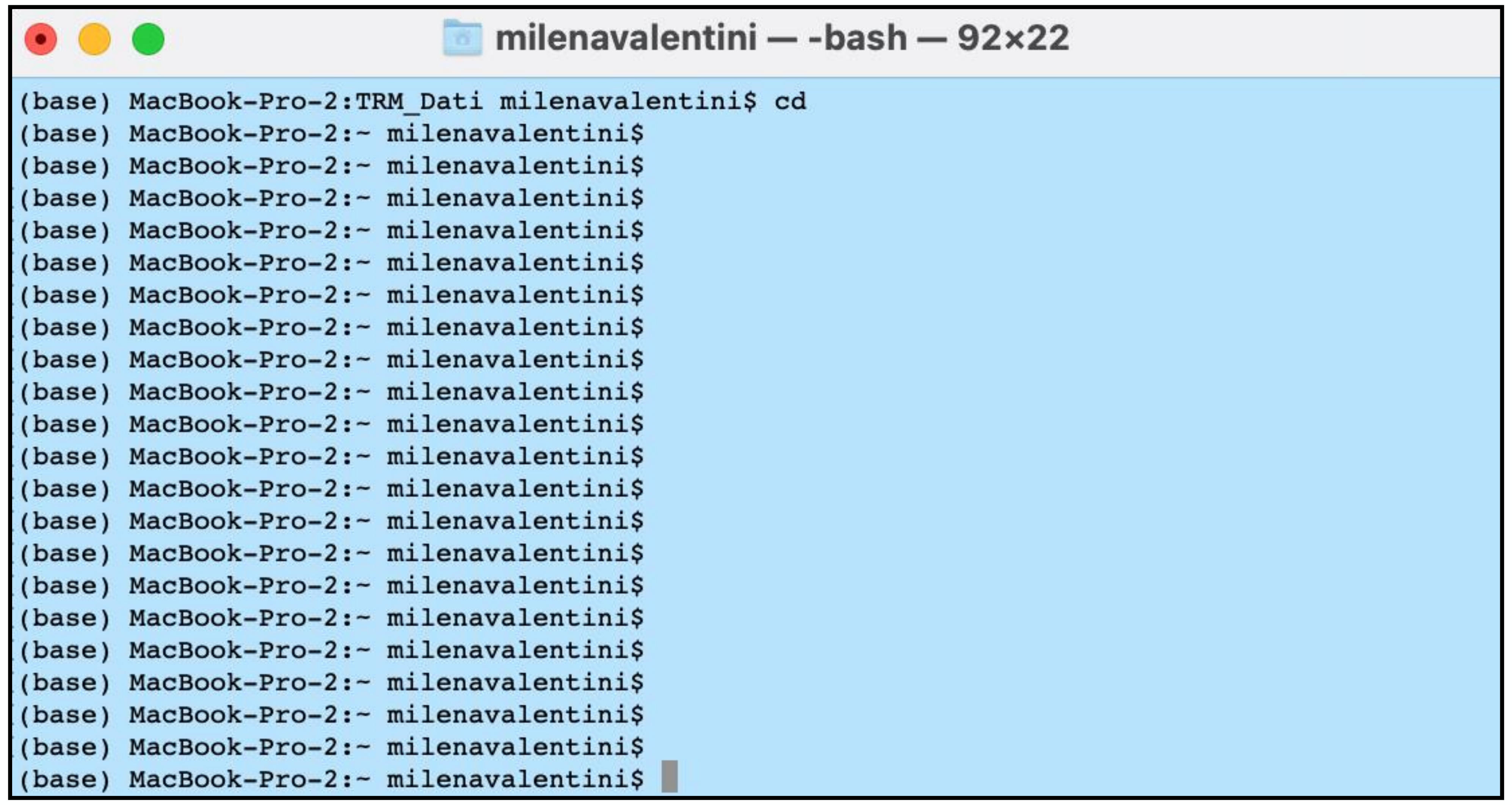
File browser or manager:
program of an Operative System (OS) which provides you with a user interface to manage folders and files



Name	Date Modified	Size	Kind
Zoom	11 January 2023 at 20:14	--	Folder
Important	19 March 2022 at 11:39	--	Folder
Anaconda-Python-onMyLaptop	24 May 2021 at 15:39	--	Folder
Pages	3 March 2021 at 10:15	--	Folder
Keynote	3 March 2021 at 10:14	--	Folder
iMovie	3 March 2021 at 10:14	--	Folder
Numbers	3 March 2021 at 10:14	--	Folder
TextEdit	3 March 2021 at 10:13	--	Folder
Preview	3 March 2021 at 10:13	--	Folder
QuickTime Player	3 March 2021 at 10:13	--	Folder
Relocated Items	8 October 2019 at 21:40	29 bytes	Alias

Macintosh HD › Users › milenavalentini › Documents › Zoom

Shell/terminal/console/command prompt:
interface to interact with the computer
via command line
without relying on graphical unit interfaces

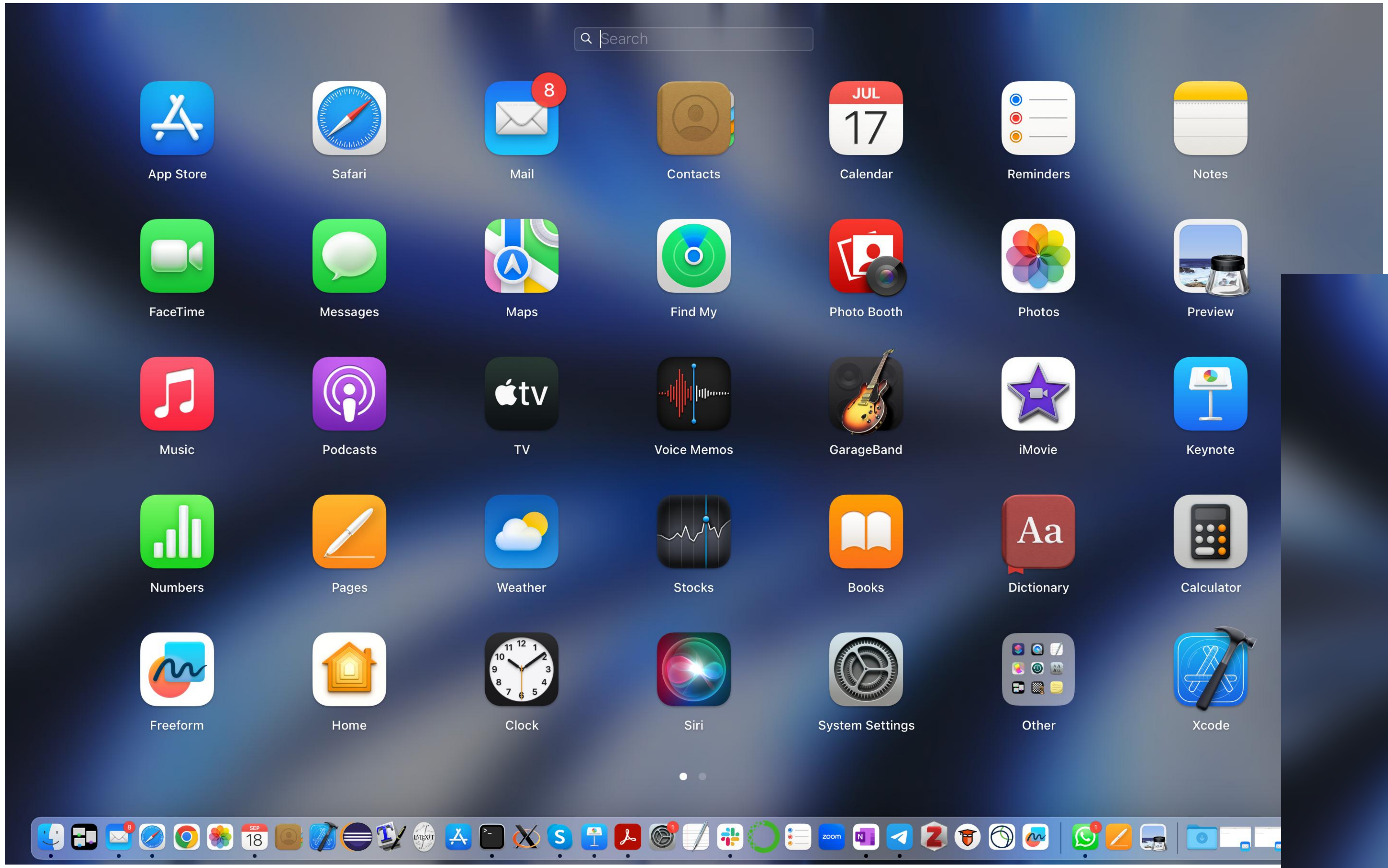


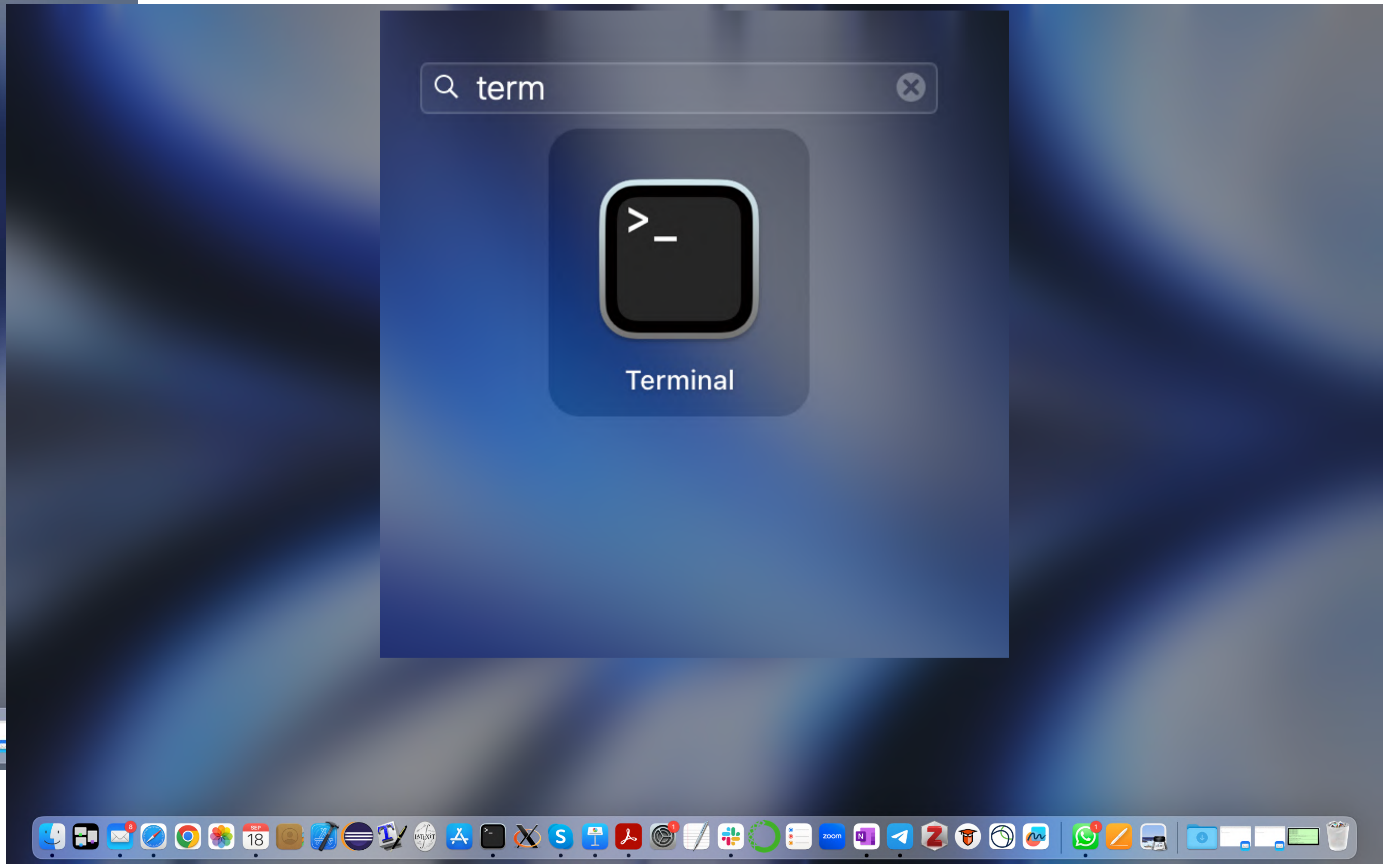
```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ cd
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$
```

The command line tool

OS: Mac

Look for the Terminal among the applications





The command line tool

OS: Linux/Unix

Look for the terminal among the applications or use the shortcut Ctrl+Alt+T

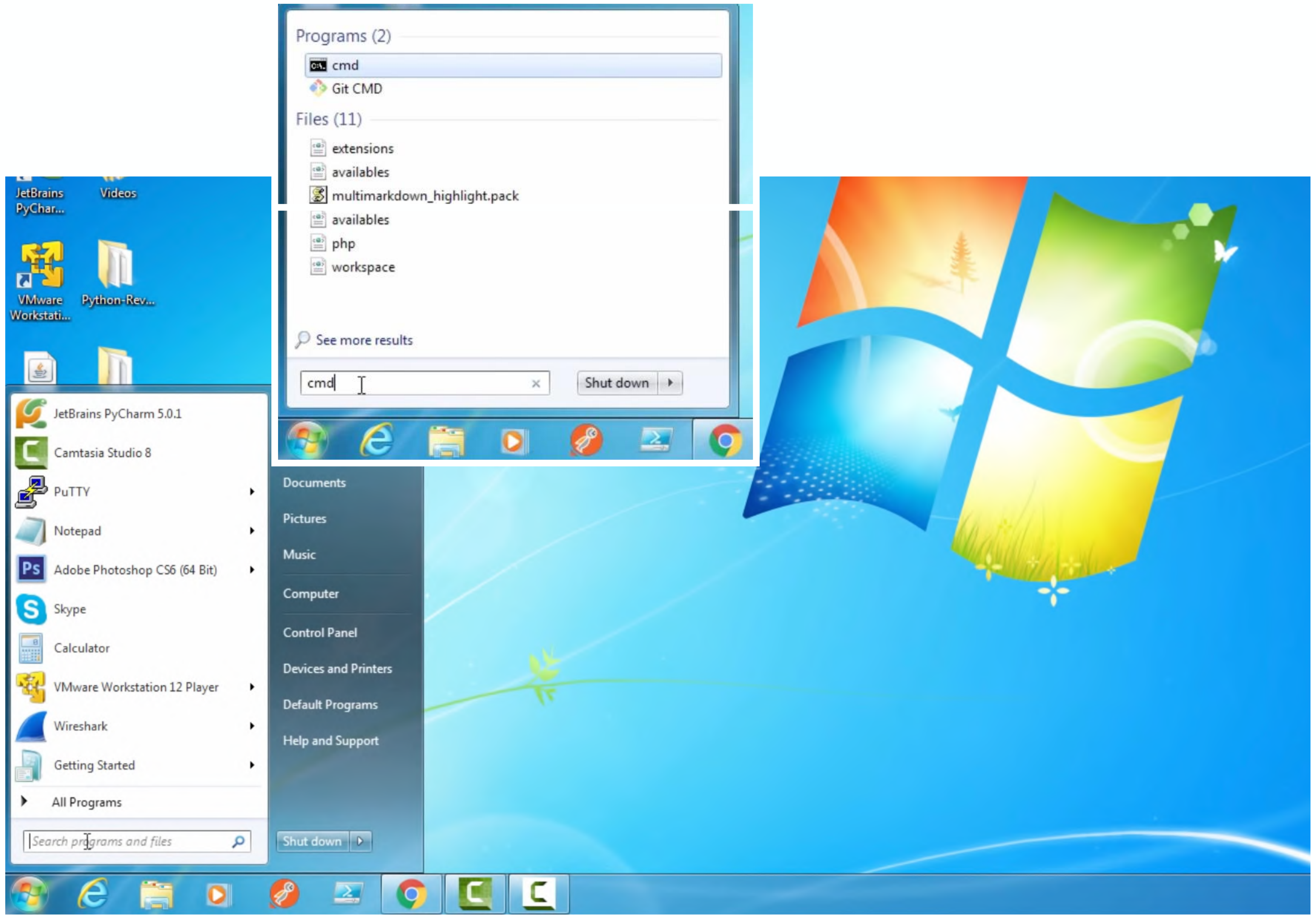
Dash home
vmware-tools-distrib
ter
Applications
Terminal
UXTerm
XTerm
[redacted]@ubuntu: ~

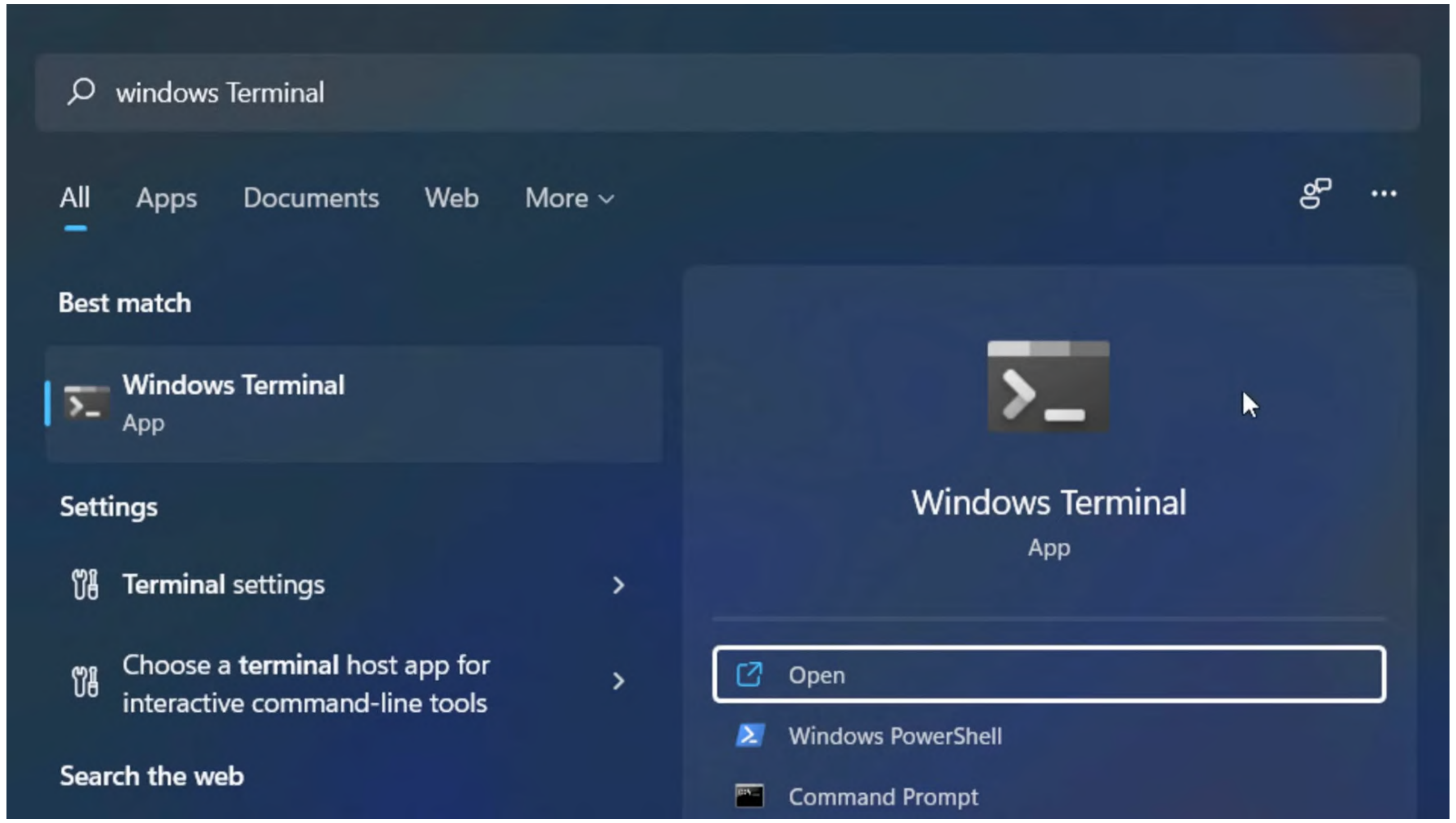
```
[redacted]@ubuntu:~$
```

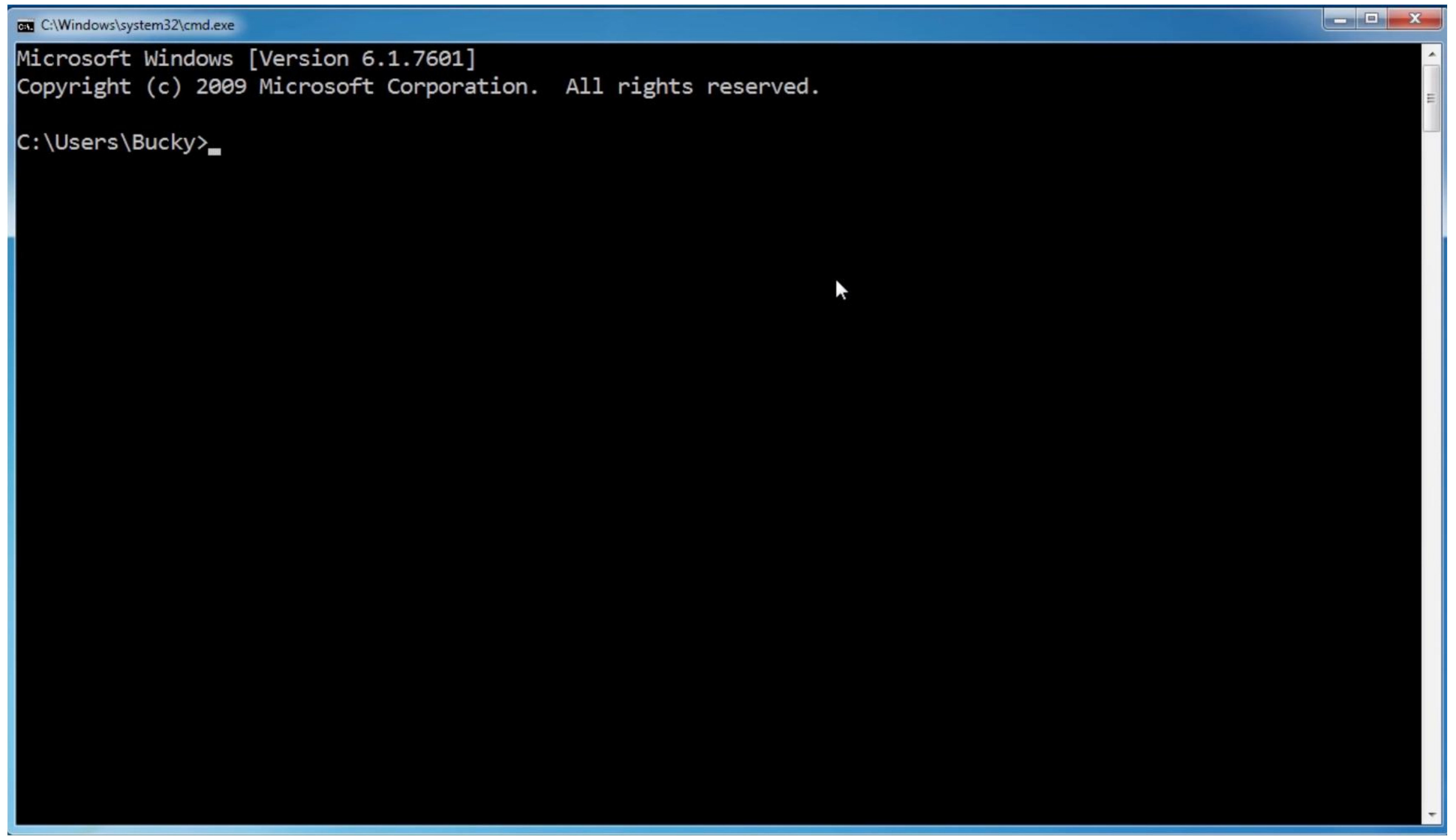
The command line tool

OS: Windows

Look for the command prompt or for the terminal among available programs and applications







```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Bucky>_
```

About Unix

Vocabulary

UNIX usually refers to a specific OS developed at the end of 1960s

Unix is commonly used to refer to a class of OSes derived/developed from UNIX

Unix-like systems are OSes which behave close to Unix OSes (not fully compliant with UNIX specifics)

Linux is a family of Unix OSes.

Ubuntu is one among the several Linux distributions.

Why learning Unix pays off

- open-source
- several infrastructures are Unix/Linux-based
- supercomputers and machine for HPC are Unix-based
- allows you to better understand how an OS really works
- several available programming tools

The shell

A shell is a key software component of an OS that allows the user to interact with it via command line.

Unix provides several shells, which differ one another for their complexity,
specifics of language scripting and peculiar features. Among available shells:
Bourne shell (sh), Bourne Again shell (bash), C shell (csh), Korn shell (ksh), Z shell (Zsh)

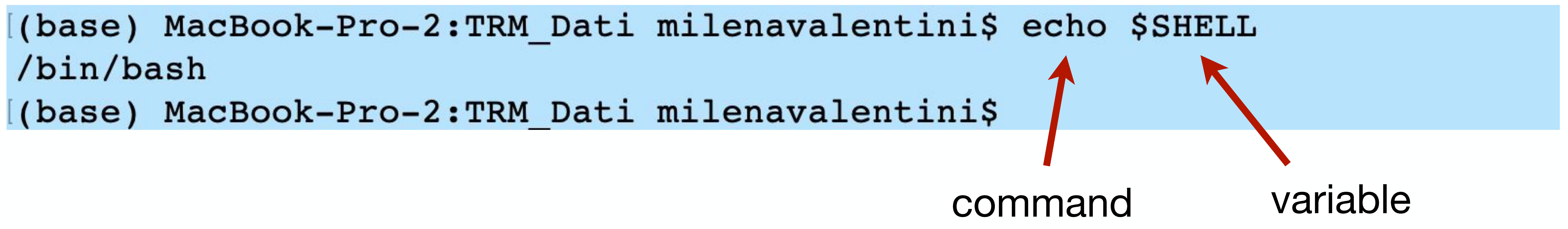
The shell

A shell is a key software component of an OS that allows the user to interact with it via command line.

Unix provides several shells, which differ one another for their complexity, specifics of language scripting and peculiar features. Among available shells: Bourne shell (sh), Bourne Again shell (bash), C shell (csh), Korn shell (ksh), Z shell (Zsh)

The bash shell is common, quite flexible and provided as default in the majority of Linux distributions.

To verify that you're using bash



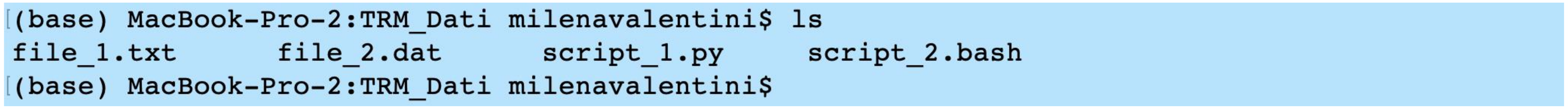
```
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$ echo $SHELL
/bin/bash
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

Command line

Command, general form:

command [flags] [argument1] [argument2] …

example:
  ls -l -a (or: ls -la)



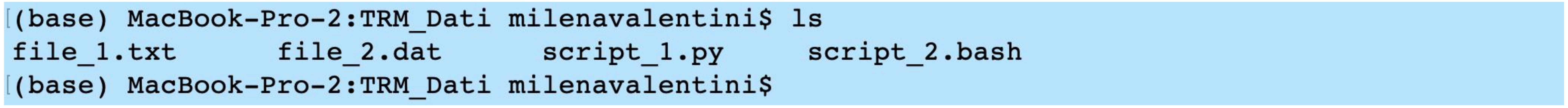
```
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls
file_1.txt      file_2.dat      script_1.py     script_2.bash
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

Command line

Command, general form:

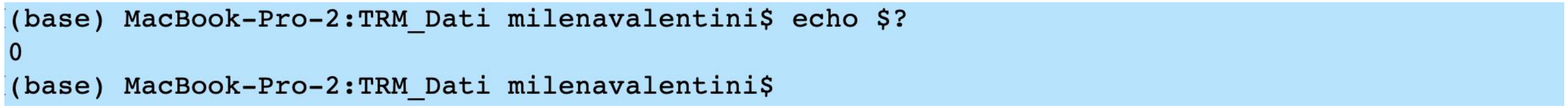
command [flags] [argument1] [argument2] …

example:
ls -l -a (or: ls -la)



```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls
file_1.txt      file_2.dat      script_1.py     script_2.bash
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

All commands have a return code (0 if the command line execution has been completed successfully)
To access the content of the return code: echo $?



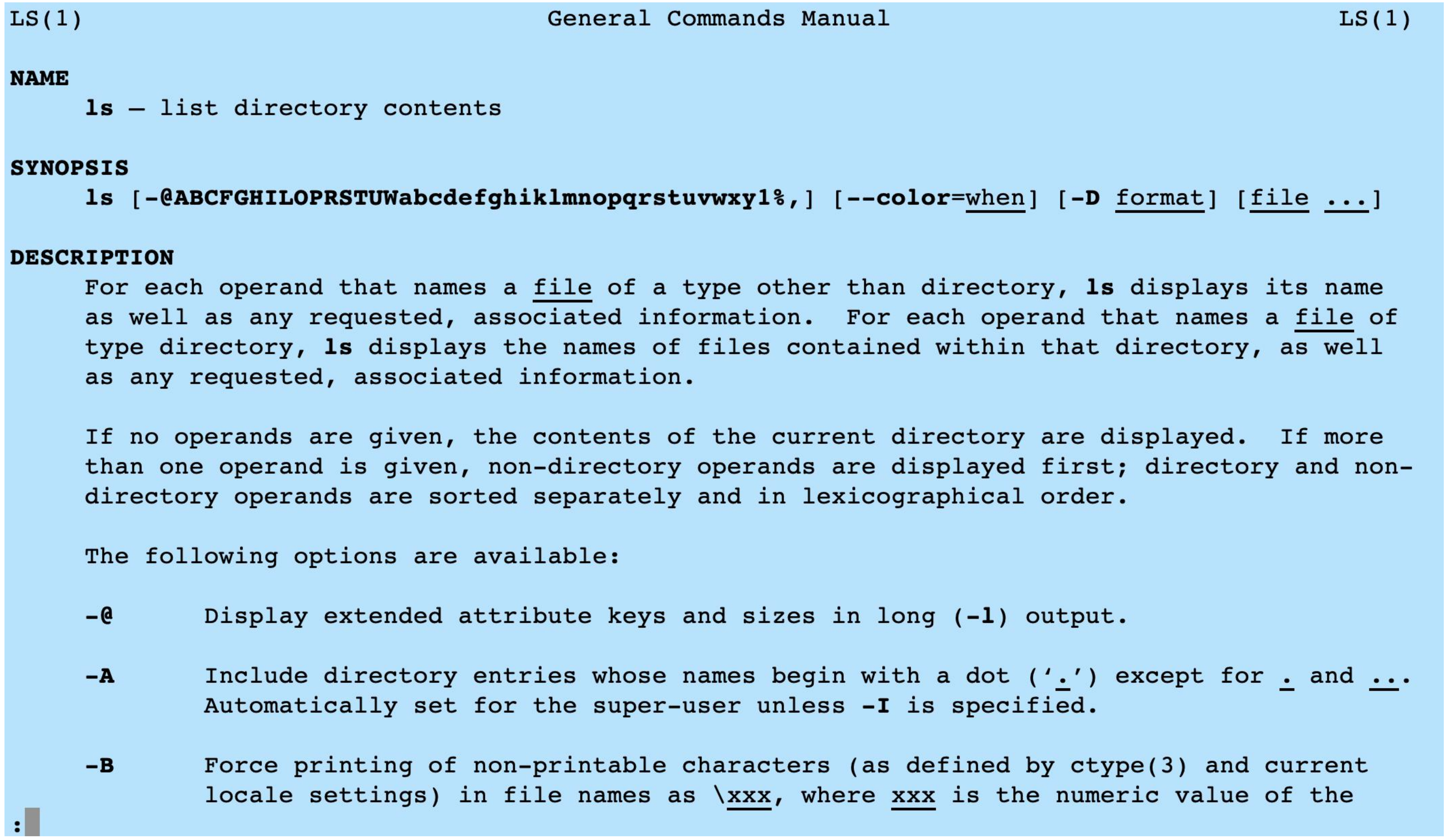
```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ echo $?
0
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

Command line

Command, general form:

command [flags] [argument1] [argument2] …

All UNIX commands
have a help documentation
man [command]



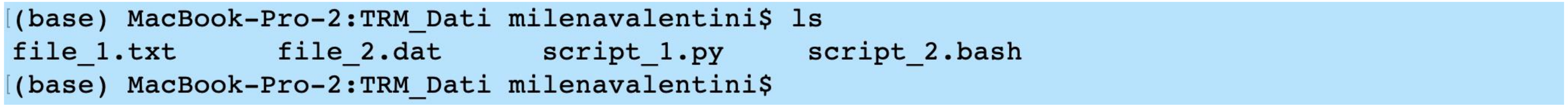
```
LS(1)                        General Commands Manual                        LS(1)

NAME
     ls – list directory contents

SYNOPSIS
     ls [-@ABCFGHILOPRSTUWabcdefghiklmnopqrstuvwxy1%,] [--color=when] [-D format] [file ...]

DESCRIPTION
     For each operand that names a file of a type other than directory, ls displays its name
     as well as any requested, associated information.  For each operand that names a file of
     type directory, ls displays the names of files contained within that directory, as well
     as any requested, associated information.

     If no operands are given, the contents of the current directory are displayed.  If more
     than one operand is given, non-directory operands are displayed first; directory and non-
     directory operands are sorted separately and in lexicographical order.

     The following options are available:

     -@      Display extended attribute keys and sizes in long (-l) output.

     -A      Include directory entries whose names begin with a dot (‘.’) except for . and ...
             Automatically set for the super-user unless -I is specified.

     -B      Force printing of non-printable characters (as defined by ctype(3) and current
             locale settings) in file names as \xxx, where xxx is the numeric value of the
:
```

Command line

Command, general form:

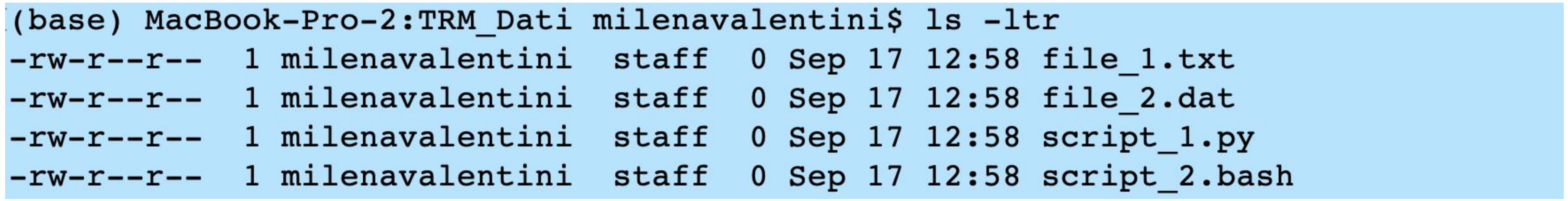
command [flags] [argument1] [argument2] …

example:
ls -l -a (or: ls -la)

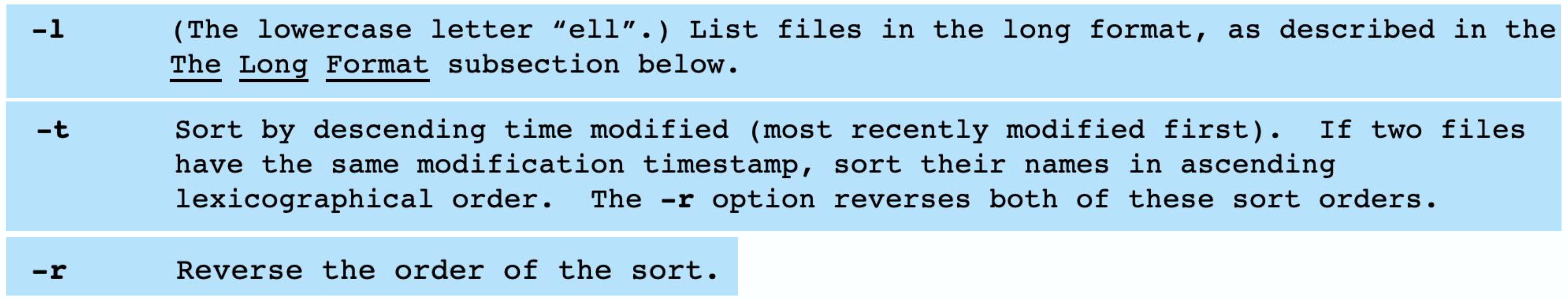


```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls
file_1.txt      file_2.dat      script_1.py     script_2.bash
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

example:



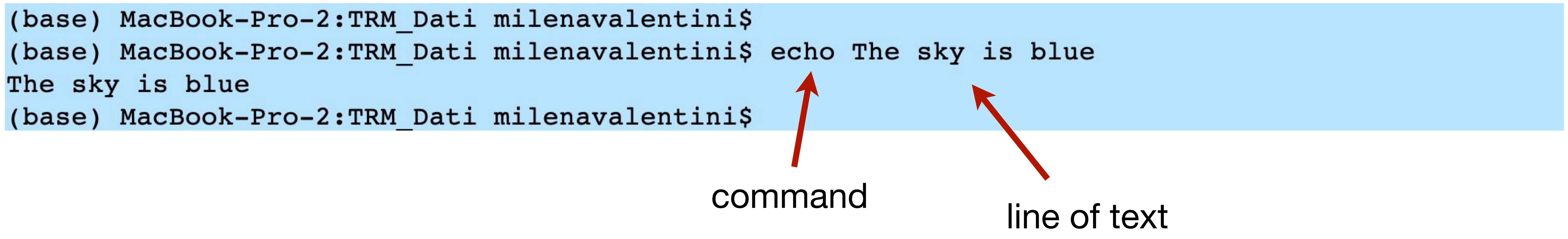
```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -ltr
-rw-r--r--  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt
-rw-r--r--  1 milenavalentini  staff  0 Sep 17 12:58 file_2.dat
-rw-r--r--  1 milenavalentini  staff  0 Sep 17 12:58 script_1.py
-rw-r--r--  1 milenavalentini  staff  0 Sep 17 12:58 script_2.bash
```



```
-l      (The lowercase letter "ell".) List files in the long format, as described in the
        The Long Format subsection below.

-t      Sort by descending time modified (most recently modified first).  If two files
        have the same modification timestamp, sort their names in ascending
        lexicographical order.  The -r option reverses both of these sort orders.

-r      Reverse the order of the sort.
```

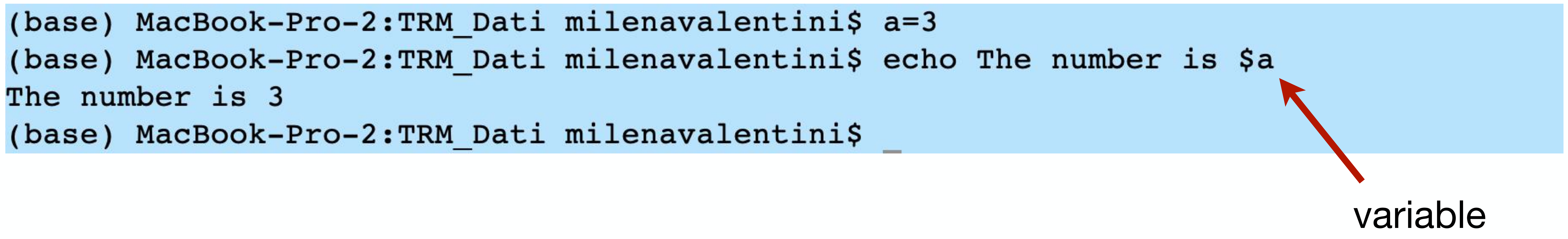
Command line

The **echo** command displays on the screen (i.e. standard output) the strings provided as arguments



```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ echo The sky is blue
The sky is blue
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

echo can also output the value of a specific variable, it this is preceded by the $ character:



```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ a=3
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ echo The number is $a
The number is 3
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ _
```

Command line

A few useful commands:

The **passwd** command allows you to change the user's password

The **who** command prints information about users currently logged in

whoami outputs the username associated to the actual user ID

The **export** command exports environment variables

If called with no arguments, **export** prints all the variables in the shell environment

The command **unset** frees the variables

Command line

The **export** command exports environment variables

Environment variables are variables which set and define the behaviour of the environment.
They are typically accessed through the shell.

To view all exported variables on your shell:

```
milenavalentini$ export -p
```

They are set when you open a new shell session. If you change any of the variable values at anytime, the export command allows you to update the current shell session about the change.

Some among the commonly used environment variables:
$USER, $PATH, $PWD, $LANG, $UID, $SHELL, $HOME

The $PATH variable contains search path for commands and executables.

The **pwd** command returns the actual path/current working directory

Command line

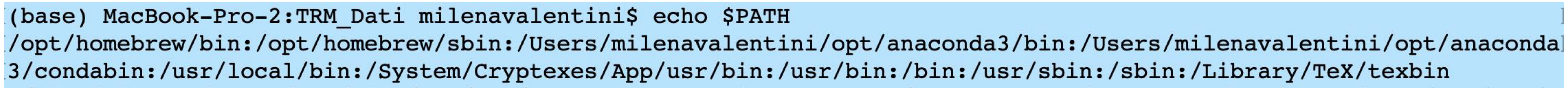
give the command $\longrightarrow$ locate the command $\longrightarrow$ execute the command

To locate the command $\longrightarrow$ search path

$\downarrow$

list of directories
to locate commands

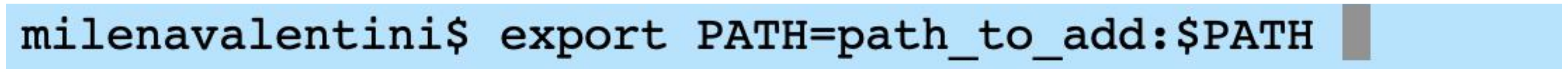
Modify the search path $\longrightarrow$ $PATH variable



```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ echo $PATH
/opt/homebrew/bin:/opt/homebrew/sbin:/Users/milenavalentini/opt/anaconda3/bin:/Users/milenavalentini/opt/anaconda
3/condabin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Library/TeX/texbin
```

The system checks these directories from left to right when running a program.

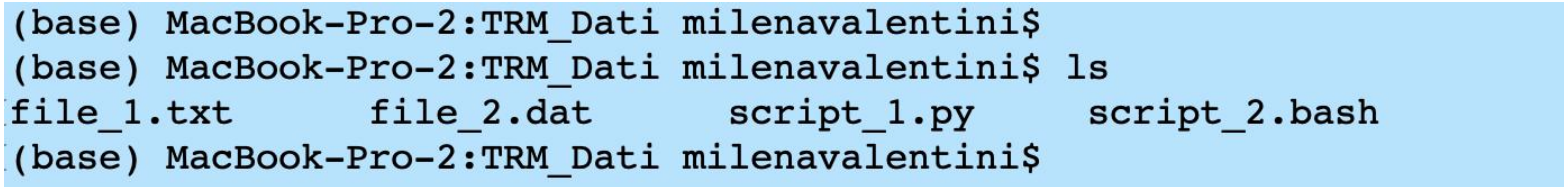
How to add a directory to the $PATH environment variable:



```
milenavalentini$ export PATH=path_to_add:$PATH
```

Command line

The **ls** command is used to inspect properties of files and directories.



```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls
file_1.txt      file_2.dat      script_1.py     script_2.bash
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

ls takes the names of one or more filenames or directories as arguments.

The file and directory names are optional: if not provided, UNIX processes the current directory.
By default, the list of files within a directory is sorted by filename (the sort order can be modified using relevant flags).

Note that ls does not process files starting with . (hidden files, mainly used to store user preferences) unless you use the -a flag (same for those starting with ..)

To use the **ls** command on a directory and its files, read permissions on those directory and files are needed

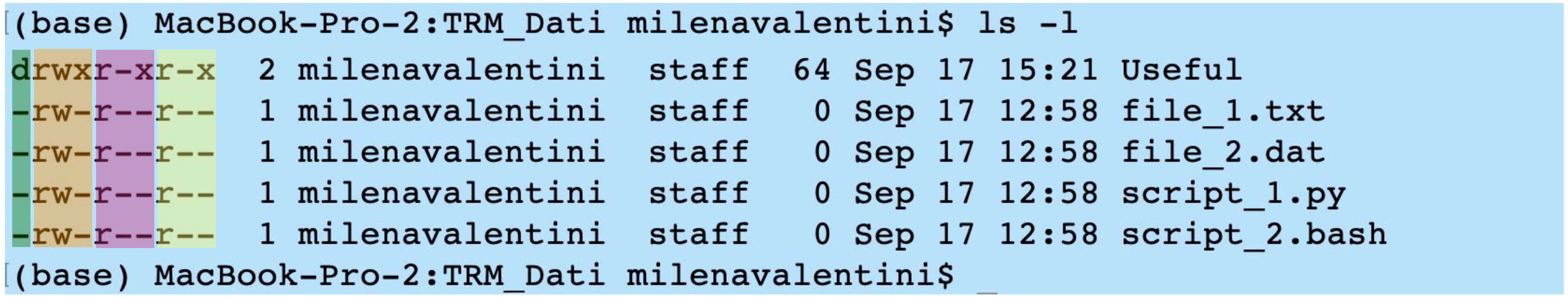
Command line

To manipulate files and manage directories and their content, the user needs permissions

Main rights of a user:

r	read
w	write
x	execute

Representation:



```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l
drwxr-xr-x  2 milenavalentini  staff  64 Sep 17 15:21 Useful
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 file_1.txt
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 file_2.dat
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_1.py
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_2.bash
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ _
```

first digit: d identifies a folder, - a file

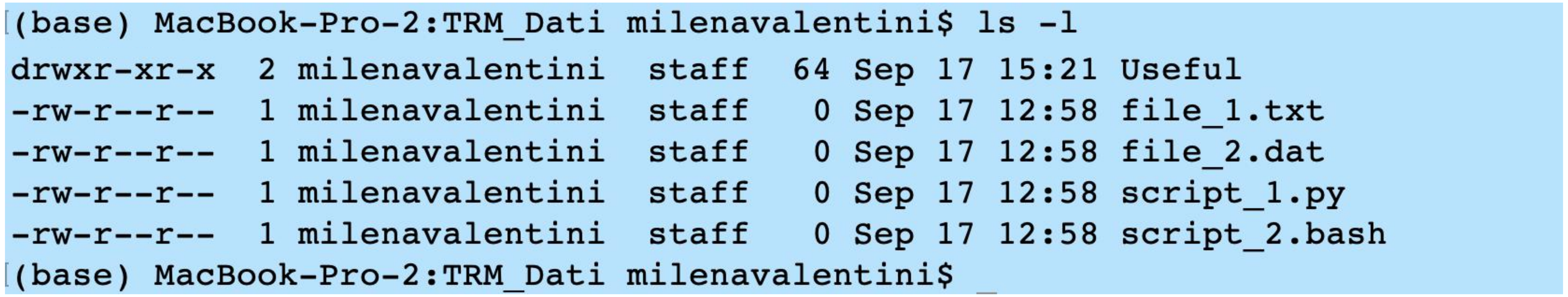
2nd, 3rd, 4th digits: readable, writable, executable by the owner (u, i.e. user)

5th, 6th, 7th digits: readable, writable, executable by the group (g, i.e. group)

8th, 9th, 10th digits: readable, writable, executable by everybody (a or o, i.e. all/others)

Command line

To manipulate files and manage directories and their content, the user needs permissions

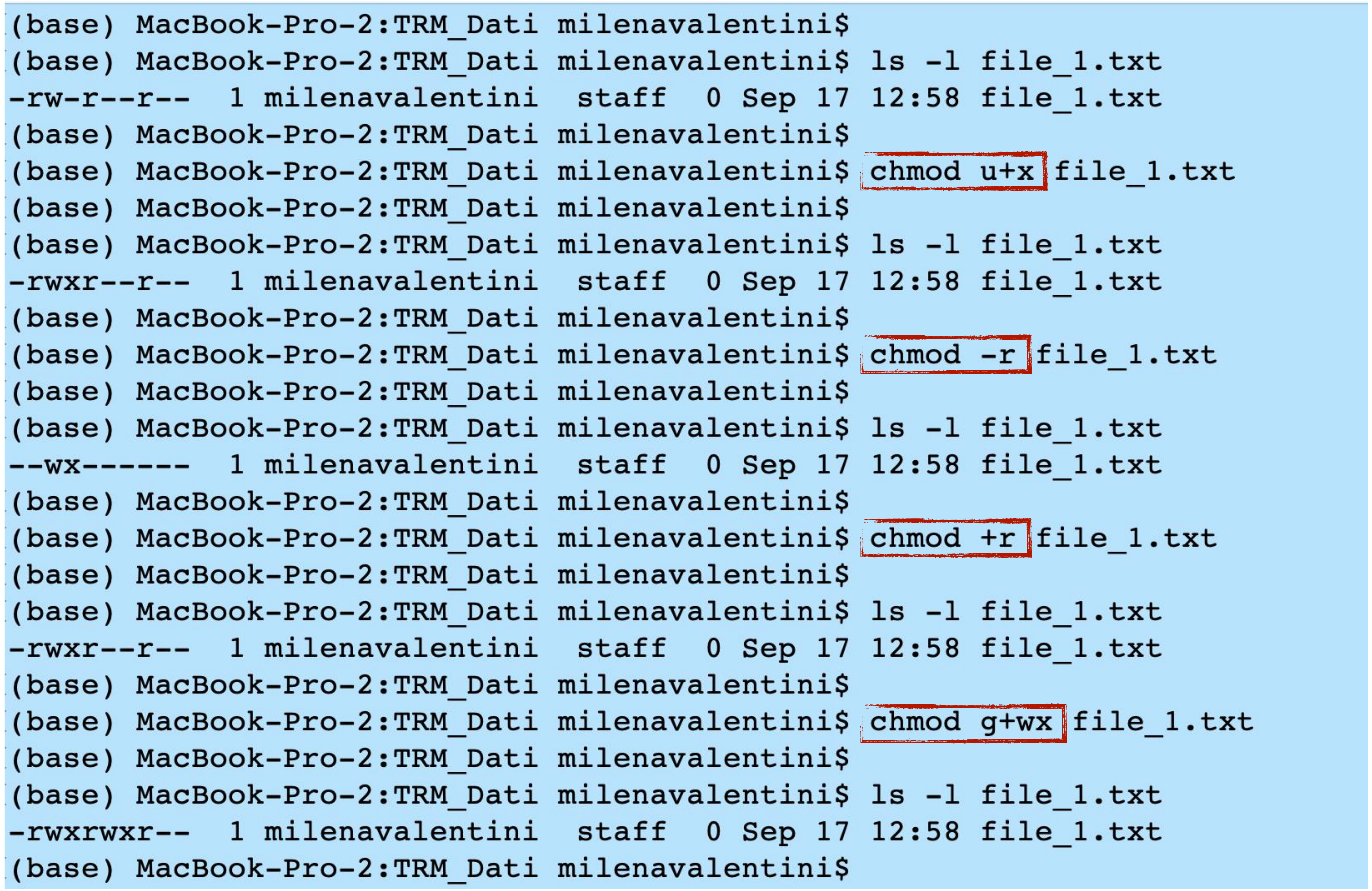


```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l
drwxr-xr-x  2 milenavalentini  staff  64 Sep 17 15:21 Useful
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 file_1.txt
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 file_2.dat
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_1.py
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_2.bash
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ _
```

Command line

To manipulate files and manage directories and their content, the user needs permissions

How to change permissions
by exploiting the chmod command:

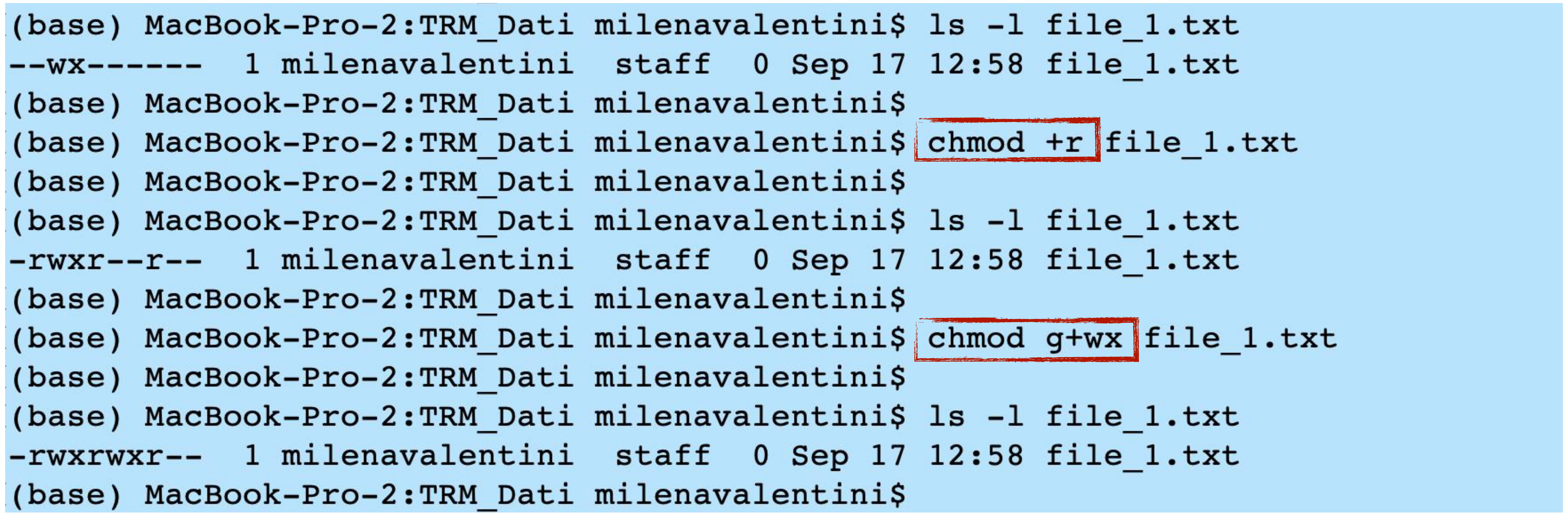


```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt
-rw-r--r--  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod u+x file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt
-rwxr--r--  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod -r file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt
--wx------  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod +r file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt
-rwxr--r--  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod g+wx file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt
-rwxrwxr--  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

Command line

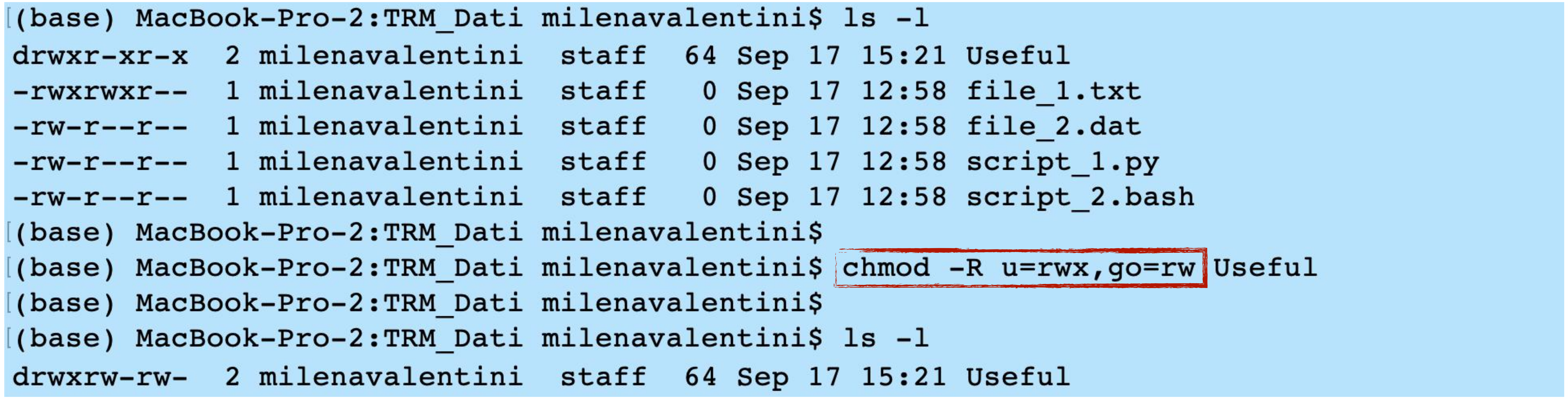
To manipulate files and manage directories and their content, the user needs permissions

How to change permissions
by exploiting the chmod command:



```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt
--wx------  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod +r file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt
-rwxr--r--  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod g+wx file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l file_1.txt
-rwxrwxr--  1 milenavalentini  staff  0 Sep 17 12:58 file_1.txt
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

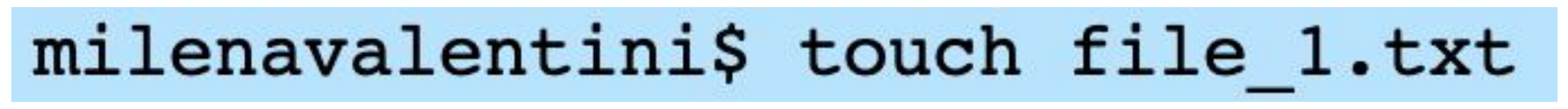
How to deal with
directories’ permissions:



```
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l
drwxr-xr-x  2 milenavalentini  staff  64 Sep 17 15:21 Useful
-rwxrwxr--  1 milenavalentini  staff   0 Sep 17 12:58 file_1.txt
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 file_2.dat
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_1.py
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_2.bash
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$ chmod -R u=rwx,go=rw Useful
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$
[(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l
drwxrw-rw-  2 milenavalentini  staff  64 Sep 17 15:21 Useful
```

Command line

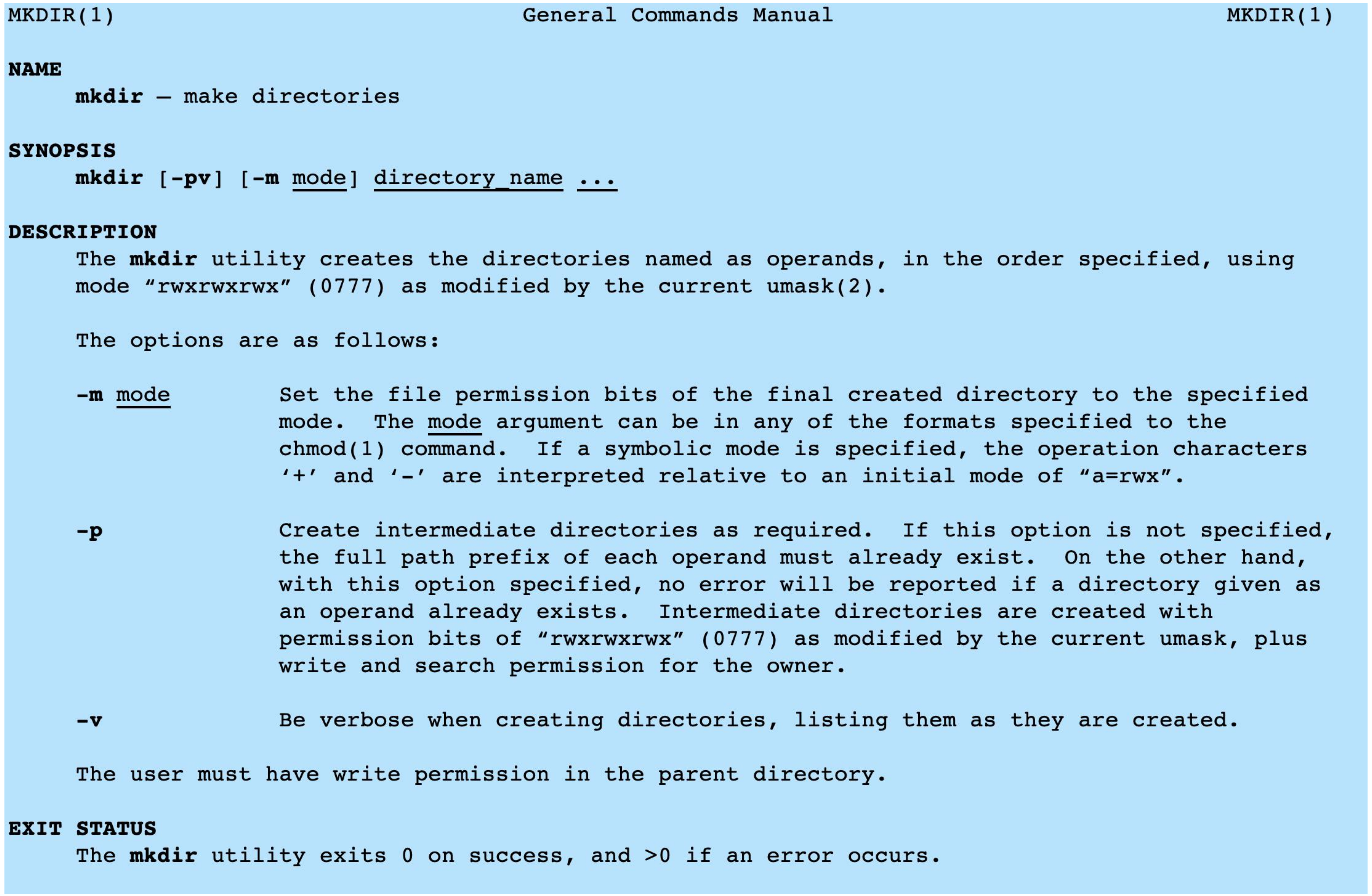
The **touch** command creates a file



```
milenavalentini$ touch file_1.txt
```

Command line

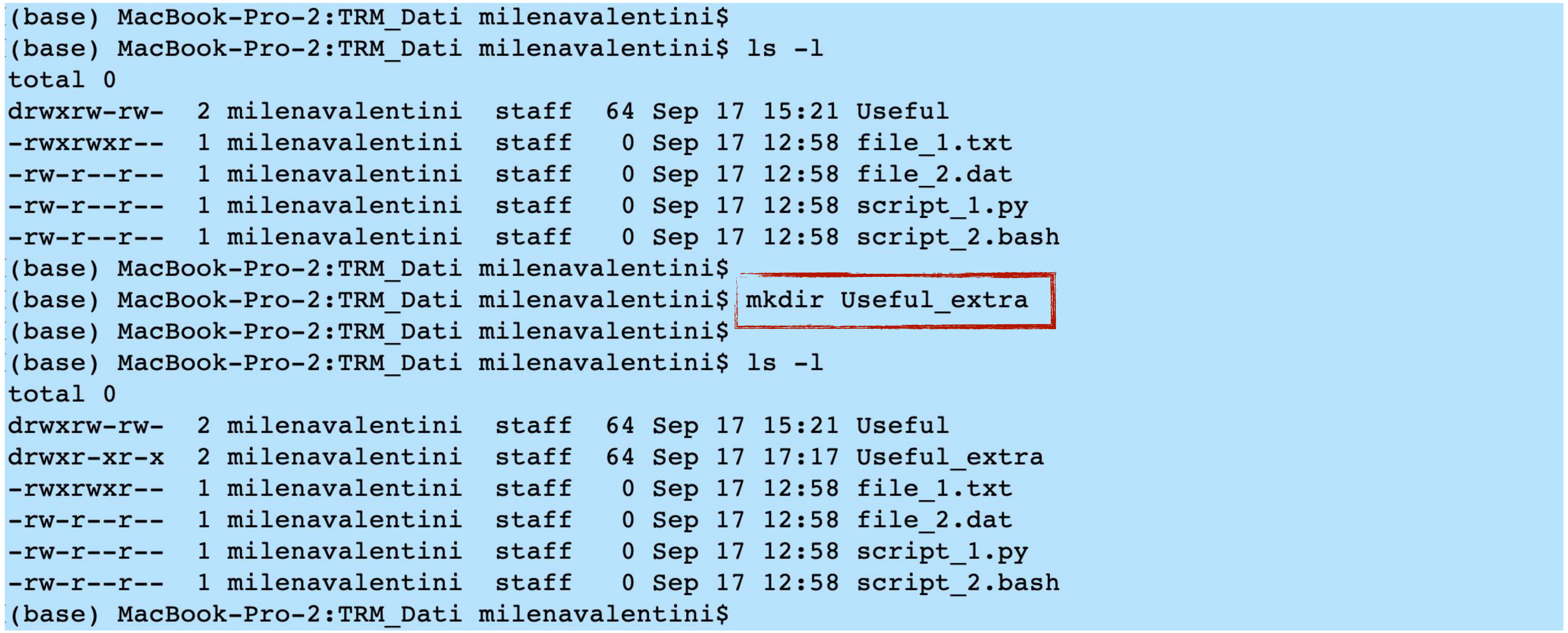
The **mkdir** command creates a directory



```
MKDIR(1)                      General Commands Manual                     MKDIR(1)

NAME
     mkdir – make directories

SYNOPSIS
     mkdir [-pv] [-m mode] directory_name ...

DESCRIPTION
     The mkdir utility creates the directories named as operands, in the order specified, using
     mode “rwxrwxrwx” (0777) as modified by the current umask(2).

     The options are as follows:

     -m mode        Set the file permission bits of the final created directory to the specified
                    mode.  The mode argument can be in any of the formats specified to the
                    chmod(1) command.  If a symbolic mode is specified, the operation characters
                    ‘+’ and ‘-’ are interpreted relative to an initial mode of “a=rwx”.

     -p             Create intermediate directories as required.  If this option is not specified,
                    the full path prefix of each operand must already exist.  On the other hand,
                    with this option specified, no error will be reported if a directory given as
                    an operand already exists.  Intermediate directories are created with
                    permission bits of “rwxrwxrwx” (0777) as modified by the current umask, plus
                    write and search permission for the owner.

     -v             Be verbose when creating directories, listing them as they are created.

     The user must have write permission in the parent directory.

EXIT STATUS
     The mkdir utility exits 0 on success, and >0 if an error occurs.
```

Command line

The **mkdir** command creates a directory — examples:



```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l
total 0
drwxrw-rw-  2 milenavalentini  staff  64 Sep 17 15:21 Useful
-rwxrwxr--  1 milenavalentini  staff   0 Sep 17 12:58 file_1.txt
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 file_2.dat
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_1.py
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_2.bash
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ mkdir Useful_extra
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls -l
total 0
drwxrw-rw-  2 milenavalentini  staff  64 Sep 17 15:21 Useful
drwxr-xr-x  2 milenavalentini  staff  64 Sep 17 17:17 Useful_extra
-rwxrwxr--  1 milenavalentini  staff   0 Sep 17 12:58 file_1.txt
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 file_2.dat
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_1.py
-rw-r--r--  1 milenavalentini  staff   0 Sep 17 12:58 script_2.bash
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```