About me

dmay@ucsd.edu

My work page
https://dmay.scrippsprofiles.ucsd.edu/

My research
https://scholar.google.com/citations?user=LkODstgAAAAJ&hl=en

- I develop novel computational methods and software to study physical processes relevant to the Earth sciences.
  - Efficient algorithms and software which exploit large scale massively parallel high-performance computing facilities and modern computational hardware.
  - Current research activities exploit non-intrusive reduced order modeling applied to Earthquakes and the thermal structure within subduction zones.
  - My research lives at the intersection of the Earth sciences, mathematics and computer science.

- I grew up on a farm in rural Australia - I learned to solve problems at a very young age.
- No one in my family ever went to university.
- I started programming a computer with my brother when I was 8 years old (for fun!).
- I've used Python for ~15 years. I use Python nearly everyday.
- I directly use Numerical Methods everyday (even in my sleep).
- I really like solving math problems with a computer.
- I teach Python and mathematics at UCSD to Earth Science students.

Introduction to Numerical Methods

Jan 27, 2025

Instructor: Dave May (dmay@ucsd.edu)

Outline

1. Motivations
2. Teaching format - what to expect
3. Learning outcomes
4. Realities of programming
5. Python time

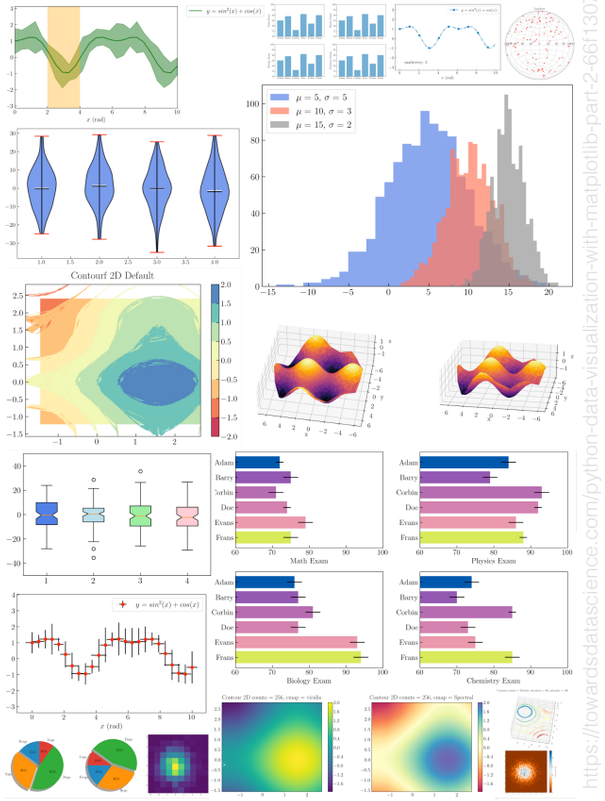
Why Python?

- Easier to learn than many other languages.
- Extremely flexible and versatile.
  - Many numerical, statistical and visualization packages.
- Freely available and cross platform (works any system).
- It is well supported, actively developed and has lots of online documentation.
- It can help you improve how you conduct your science (currently), and enable new scientific enterprises to begin.
- Python programmers are in demand.

https://codingnomads.co/why-learn-python/

Python Usages

- Data processing / filtering
- Visualisation
  - maps
  - graphs, charts, plots
  - 3D rendering
- Linear algebra
- Optimization
- Solving ordinary differential equations
- Solving partial differential equations
- Statistics
- Machine learning
- Teaching, education, scientific dissemination



www.python.org

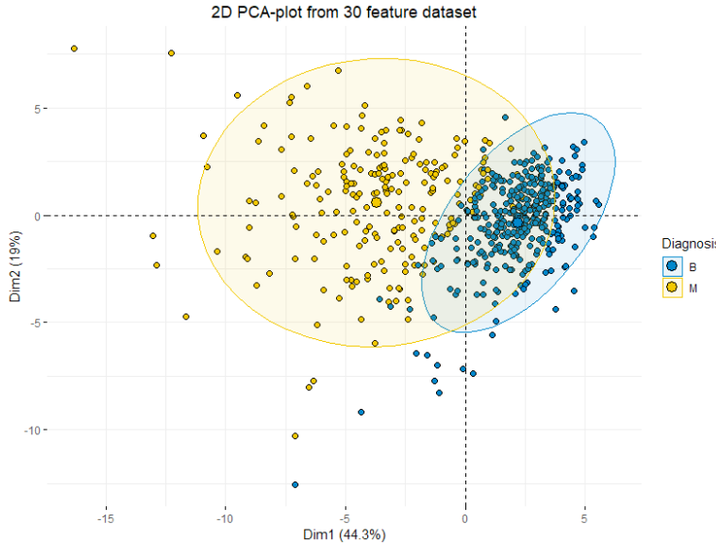
Python

High-level programming language

matplotlib

NumPy

scikit learn



jupyter

Format

- **Each topic will generally be presented in the following way**
  - A motivating problem (call it “problem P”).
  - Theory + algorithm(s) which can be used to solve problem P.
  - A problem set designed to get you to better understand the theory and to apply the theory using Python code which you will write. Problem sets will require math and coding.
  - There will be time during the day for you to work on problem sets. I will be in the class room to help you.
  - An overview / summary describing how to use existing Python packages to solve problem P.

Learning Outcomes

1. You will learn how to use Python.
2. You will learn how to apply Python to numerical methods.
3. You will learn the fundamentals of Numerical Methods.
4. You will learn fundamental algorithms in Numerical Methods to
    1. Interpolate, integrate
    2. Solve systems of linear equations
    3. Solve nonlinear equations and system of nonlinear equations
    4. Solve ODEs and PDEs
    5. Time complexity of algorithms
5. You will learn practical skills related to Numerical Methods
    1. How to translate algorithms from paper to code.
    2. How to test your code.
    3. How to measure approximation errors.

Course Philosophy

1. There is no such thing as **theoretical computer programming** or **theoretical Numerical Methods**. Numerical Methods is a **practical skill**. The only way to learn the art of Numerical Methods is to practice it. Practicing Numerical Methods does require programming.

2. Implementing Numerical Methods yourself (the algorithms) promotes understand how and why the method works.

3. Practice makes perfect.

4. Do not be deterred or discouraged by immediate failures — i.e. “code does not work”. Programming mistakes (**bugs**) are inevitable.

5. You should **only believe** code **you** can run.

6. The learning cycle is different for everybody, as is the time-scale over which you will learn to program. Everyone can learn to program.

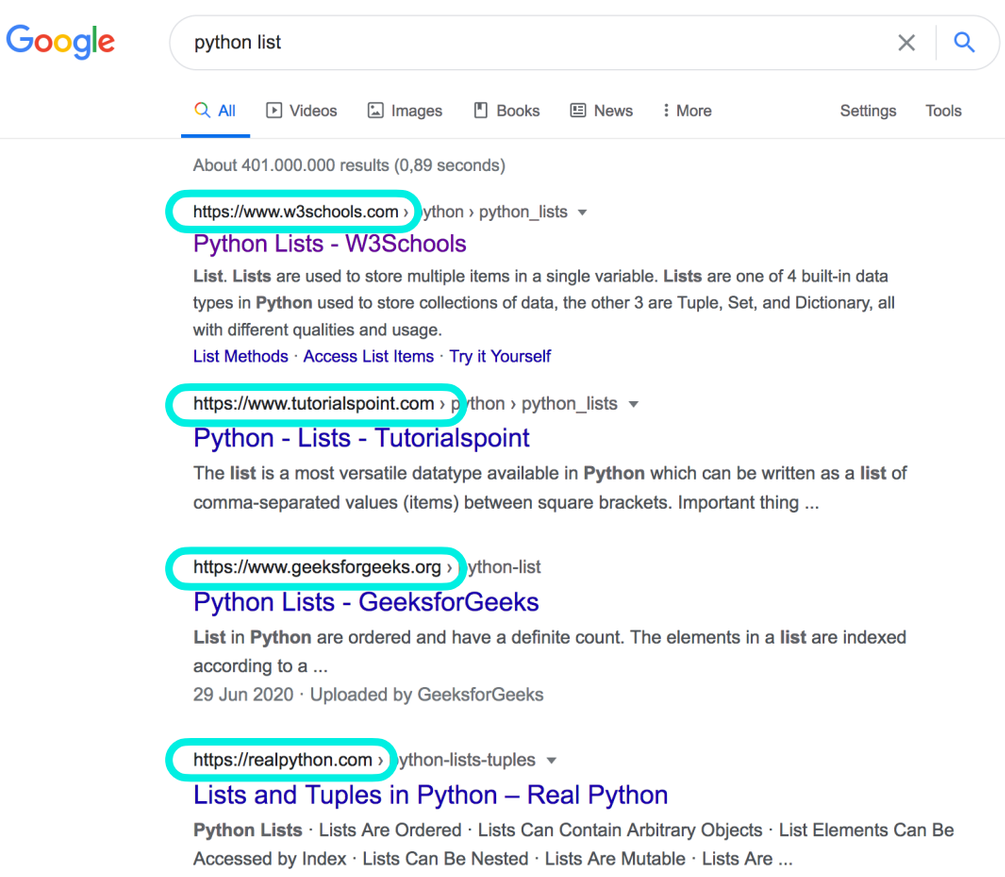
Realities of Programming

Bugs are inevitable…

1. If your code works first time you run it - be suspicious.
2. Every line of code you write can introduce a bug.
3. The most difficult bugs to identify are often trivial to fix. Don’t give up. Persevere.
4. If you see something strange in your results there is probably a bug. Be suspicious of your code.
5. A single bug can ruin your beautiful code. Be motivated to carefully debug and test your code. A single small error in the code cannot be ignored. Laziness never pays off.
6. Creating a correct and working code is possible. Perseverance does pays off.

Resources

- https://github.com/Kigali-NumericalMethods/student-material
- Me, your instructor (Dave)! I like questions - please exploit this
- Your fellow class mates
- Python itself (`help(list)`)
- The internet
- I have lots of of Python reading material which I can share with you. Please ask if you want more to read.



Time to begin

Overview of your computers file system.

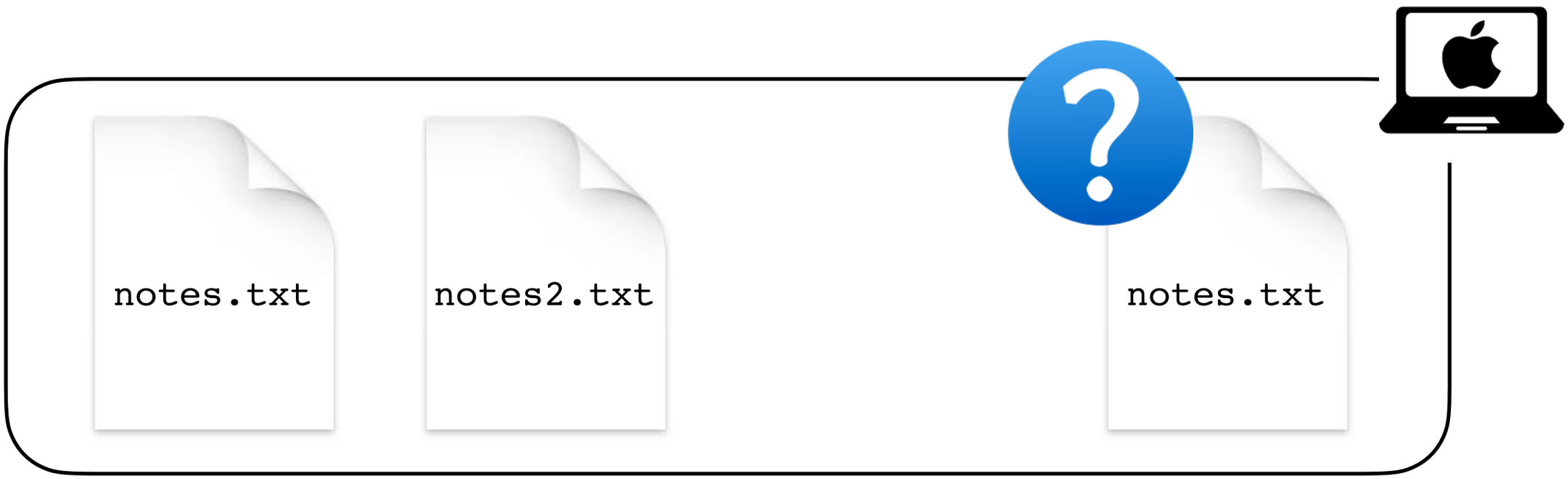
What is a file system?

- Before computers existed the terms file system, filing system and system for filing were used to describe methods of organizing, storing and retrieving paper documents.
- By 1961, the term file system was being applied to computerized filing alongside the original meaning.
- A (computers) file system provides a data storage service.

- Data is physical stored in (digital) **files**.
- File systems typically support organizing files into **directories**, also called folders, which segregate files into groups.

Files

- Files are identified by a name.
- File names must be unique.
- File names should not contain white space characters.
  - **`MyFile.txt`**
  - **`my_file.txt`**
  - **`myfile.txt`**
  - **`My   File.txt –> bad`**

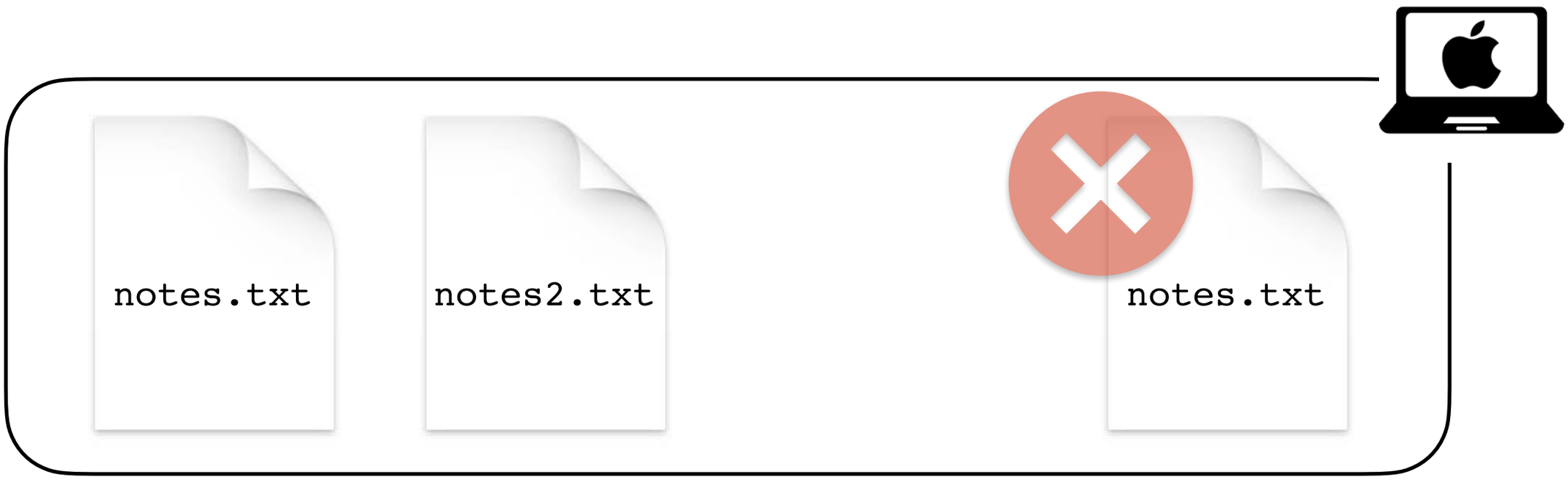


Files

- Files are identified by a name.
- File names must be unique.

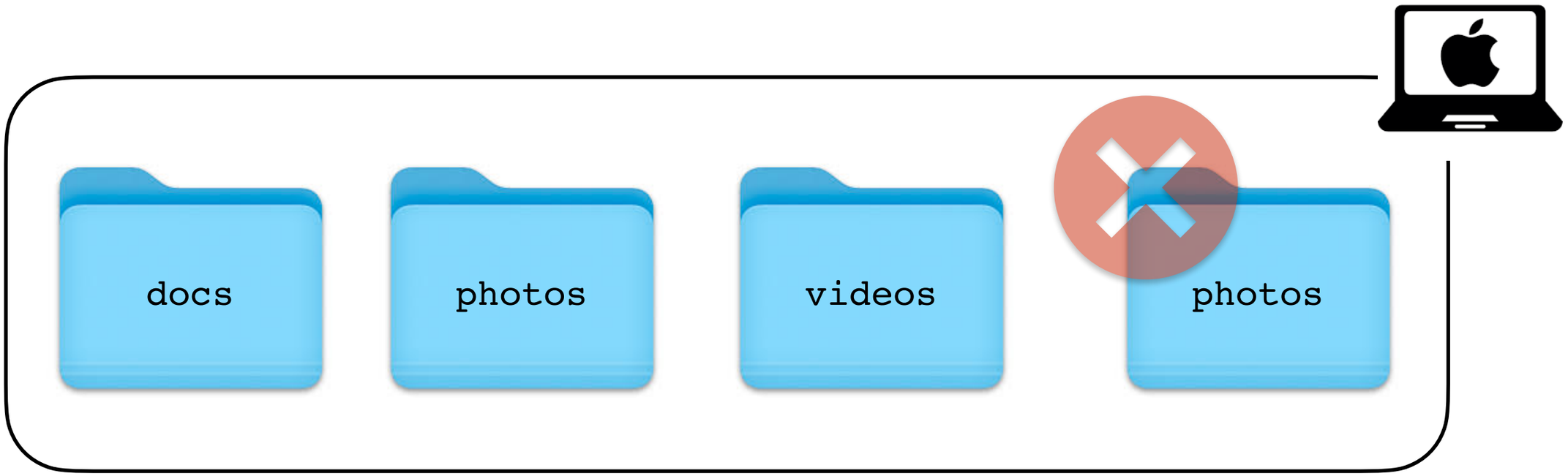
  Same name, different data - ambiguous which data you want

  Same name, same data - pointless / wasting space

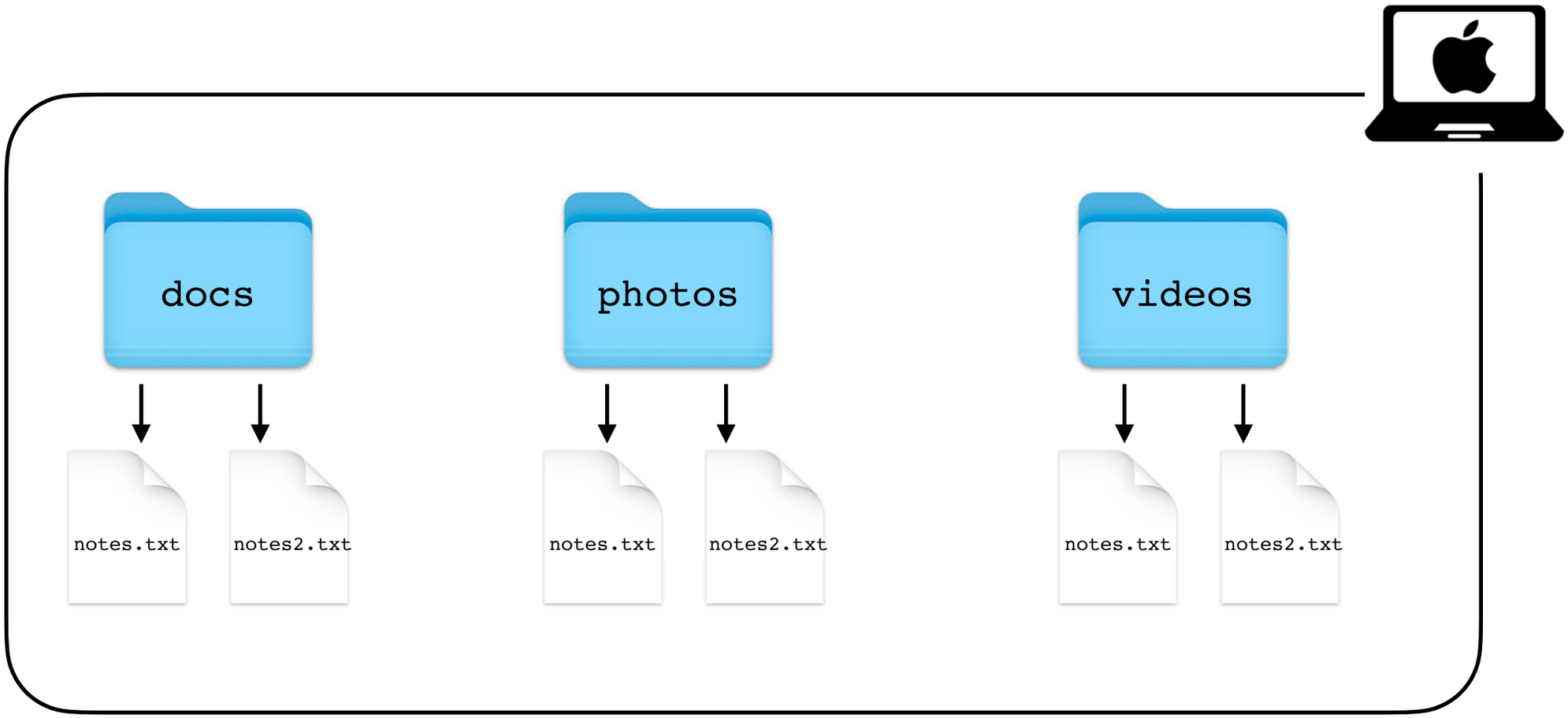


Directories

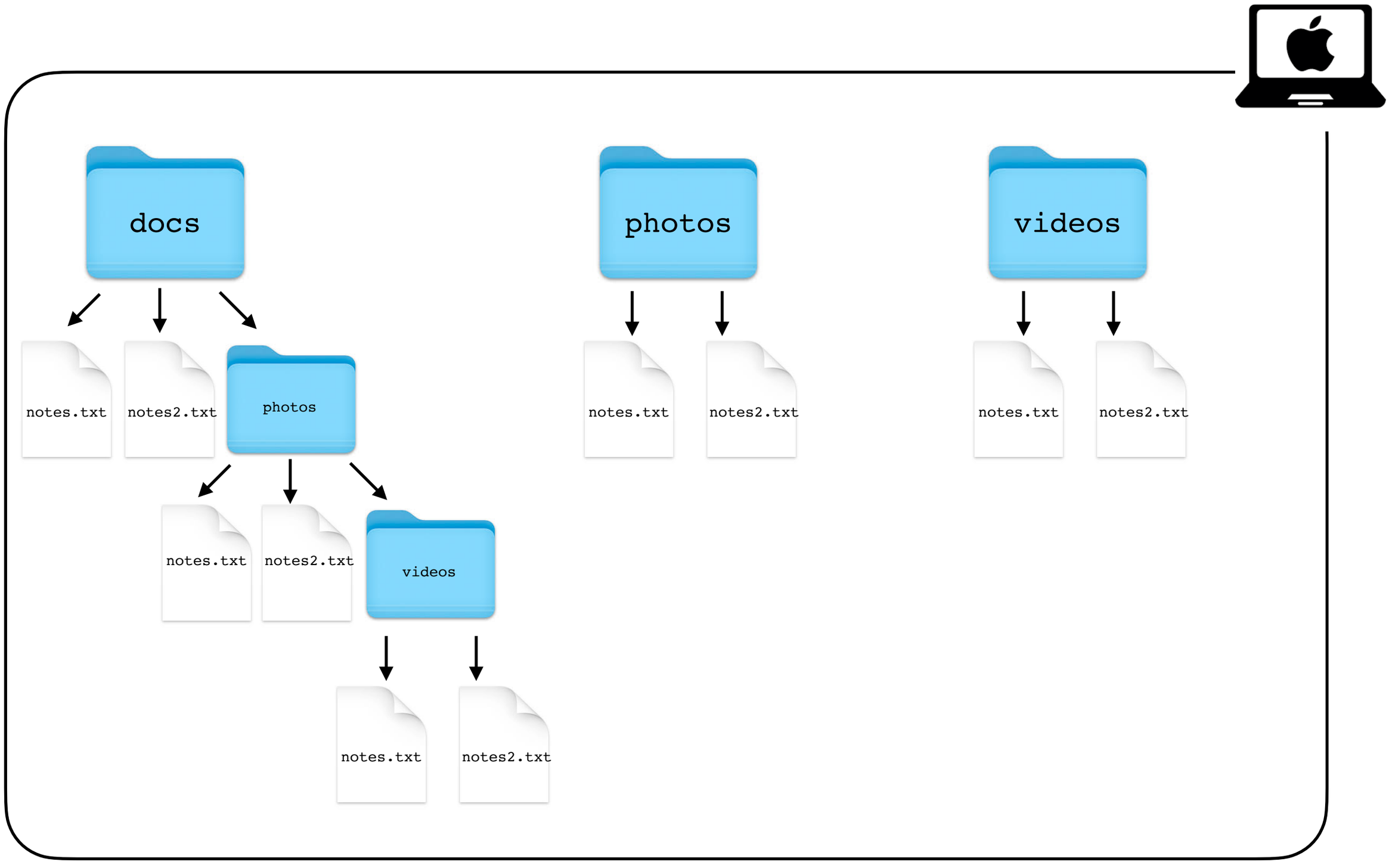
- Directories are identified by a name.
- Directory names must be unique.
- Directory names should not contain white space characters.



Is this ok?

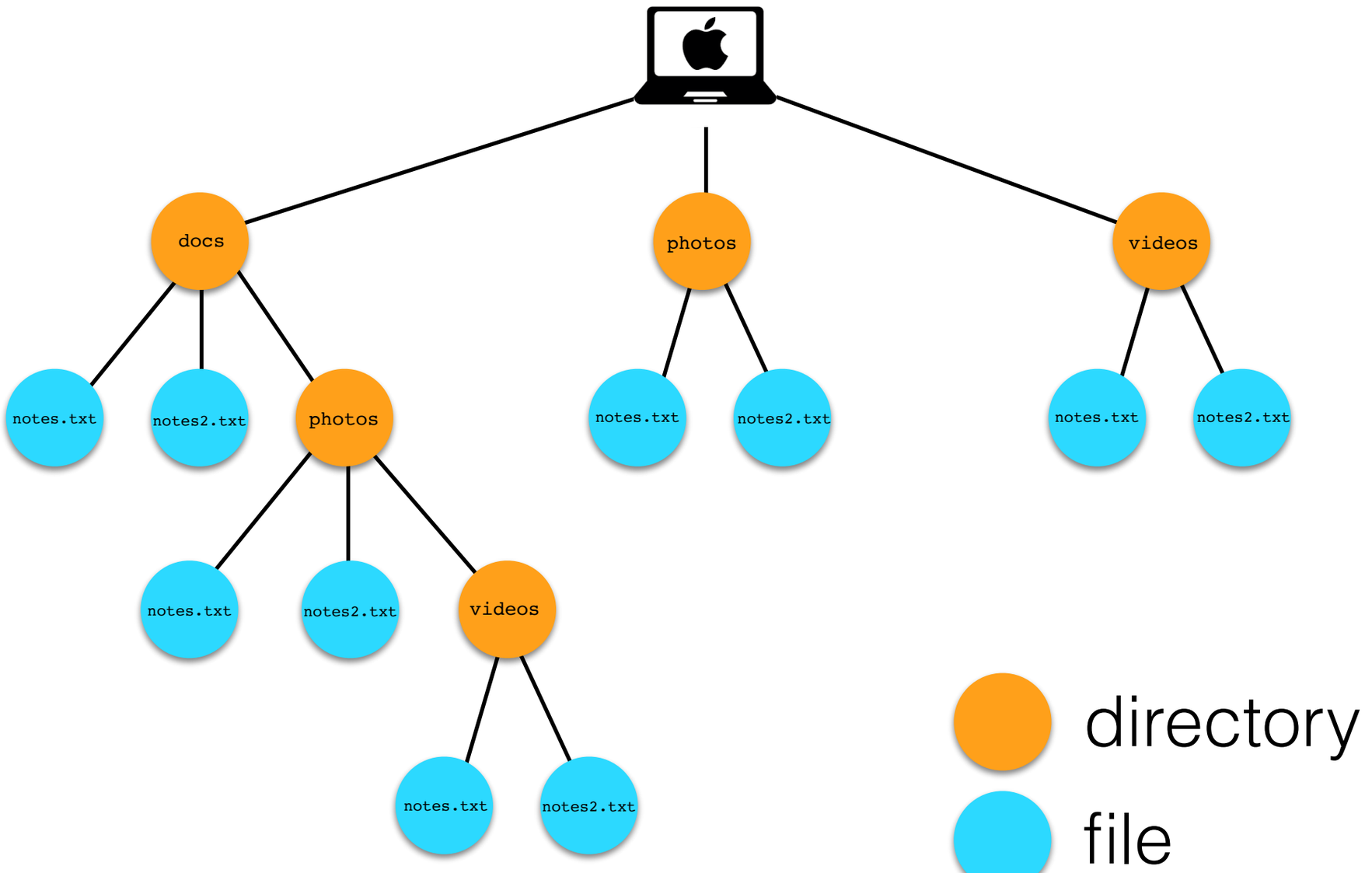

docs
photos
videos
notes.txt
notes2.txt
notes.txt
notes2.txt
notes.txt
notes2.txt

Is this ok?

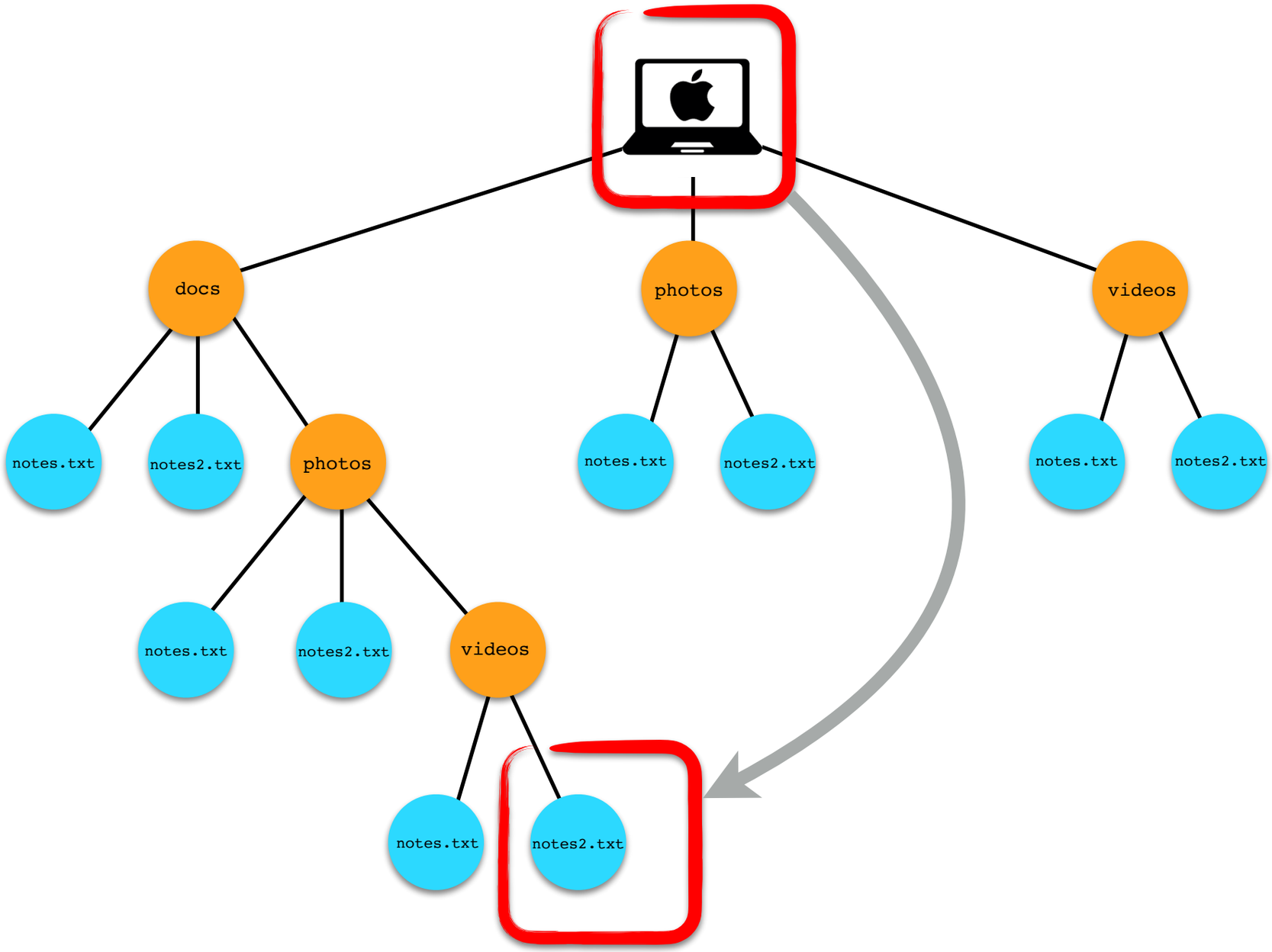

docs
notes.txt
notes2.txt
photos
notes.txt
notes2.txt
videos
notes.txt
notes2.txt
photos
notes.txt
notes2.txt
videos
notes.txt
notes2.txt

A refined statement

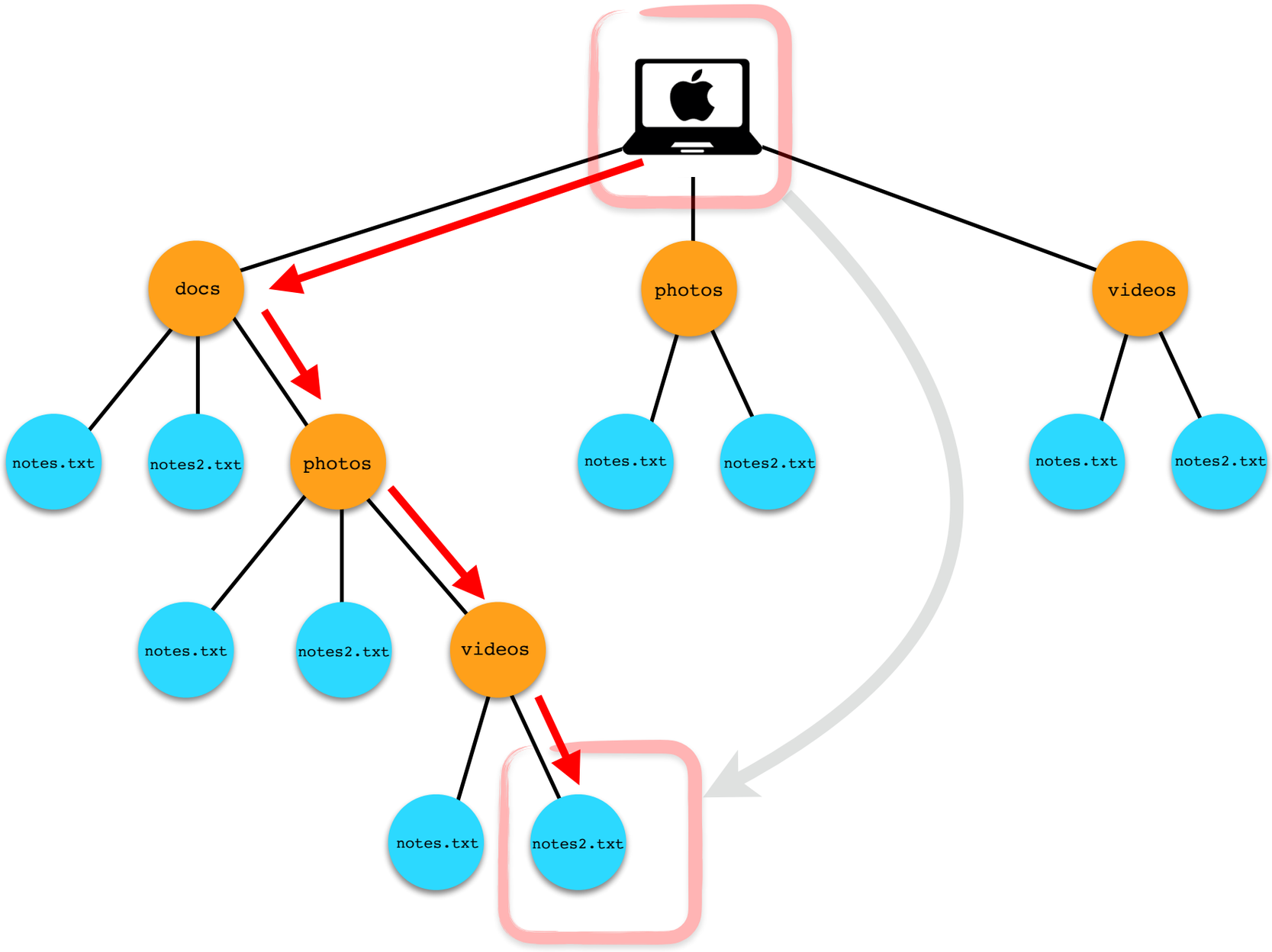
- Files within a directory must have a unique name
- Sub-directories with the same common parent directory must have a unique name.



File access


docs
photos
videos
notes.txt
notes2.txt
photos
notes.txt
notes2.txt
notes.txt
notes2.txt
notes.txt
notes2.txt
videos
notes.txt
notes2.txt

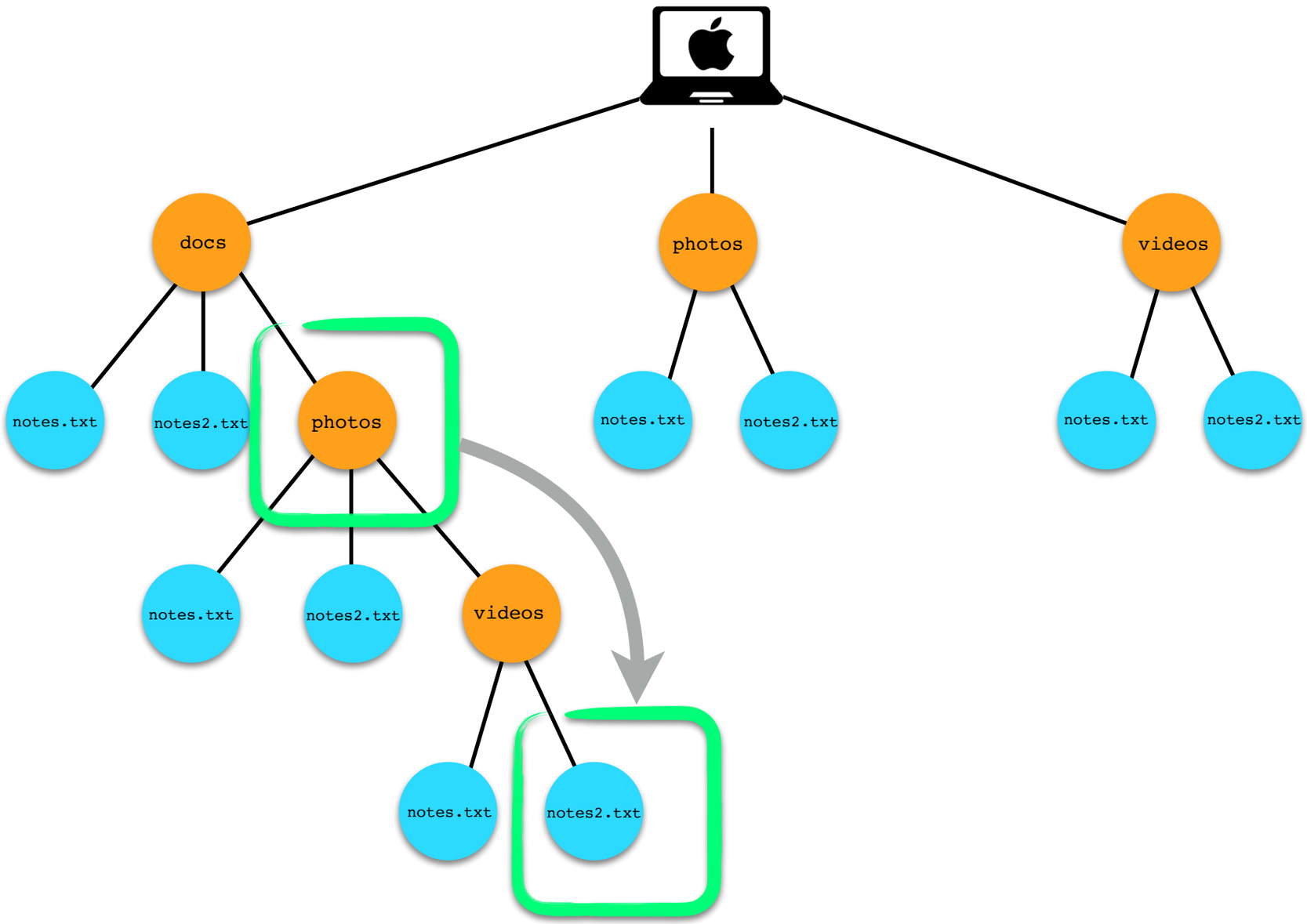
File access



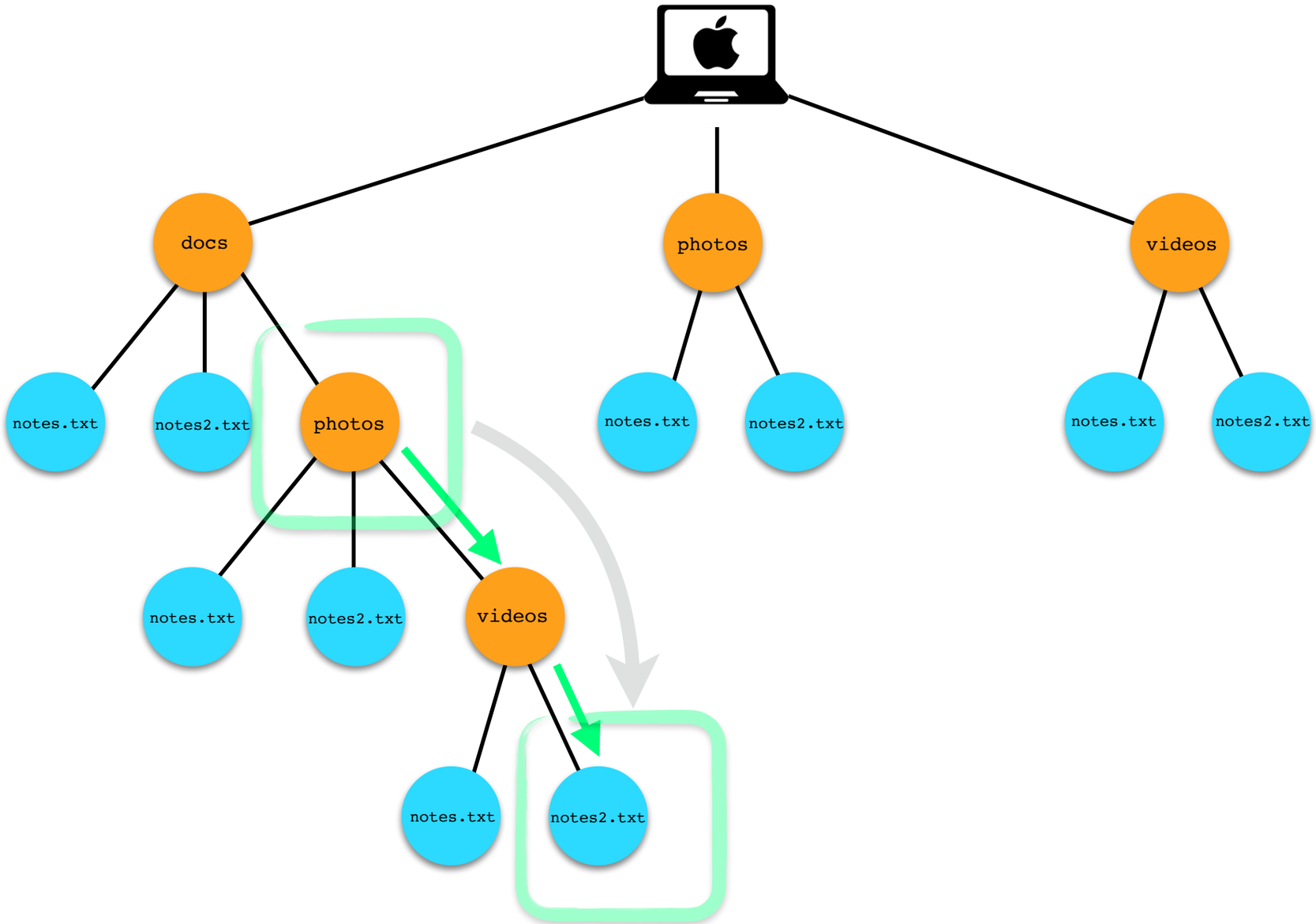
Absolute path

`docs / photos / videos / notes2.txt`

File access


docs
photos
videos
notes.txt
notes2.txt
photos
notes.txt
notes2.txt
notes.txt
notes2.txt
notes.txt
notes2.txt
videos
notes.txt
notes2.txt

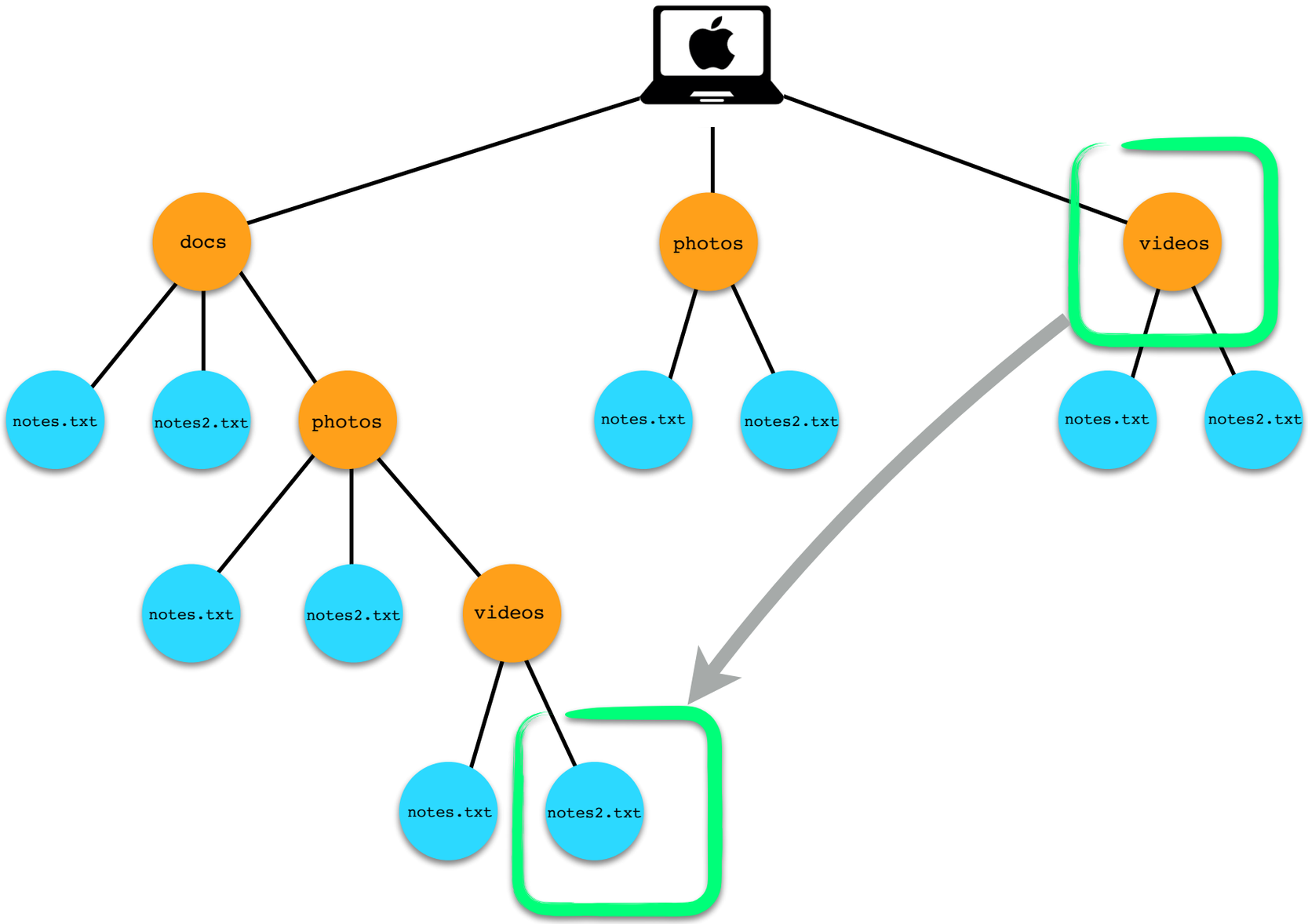
File access



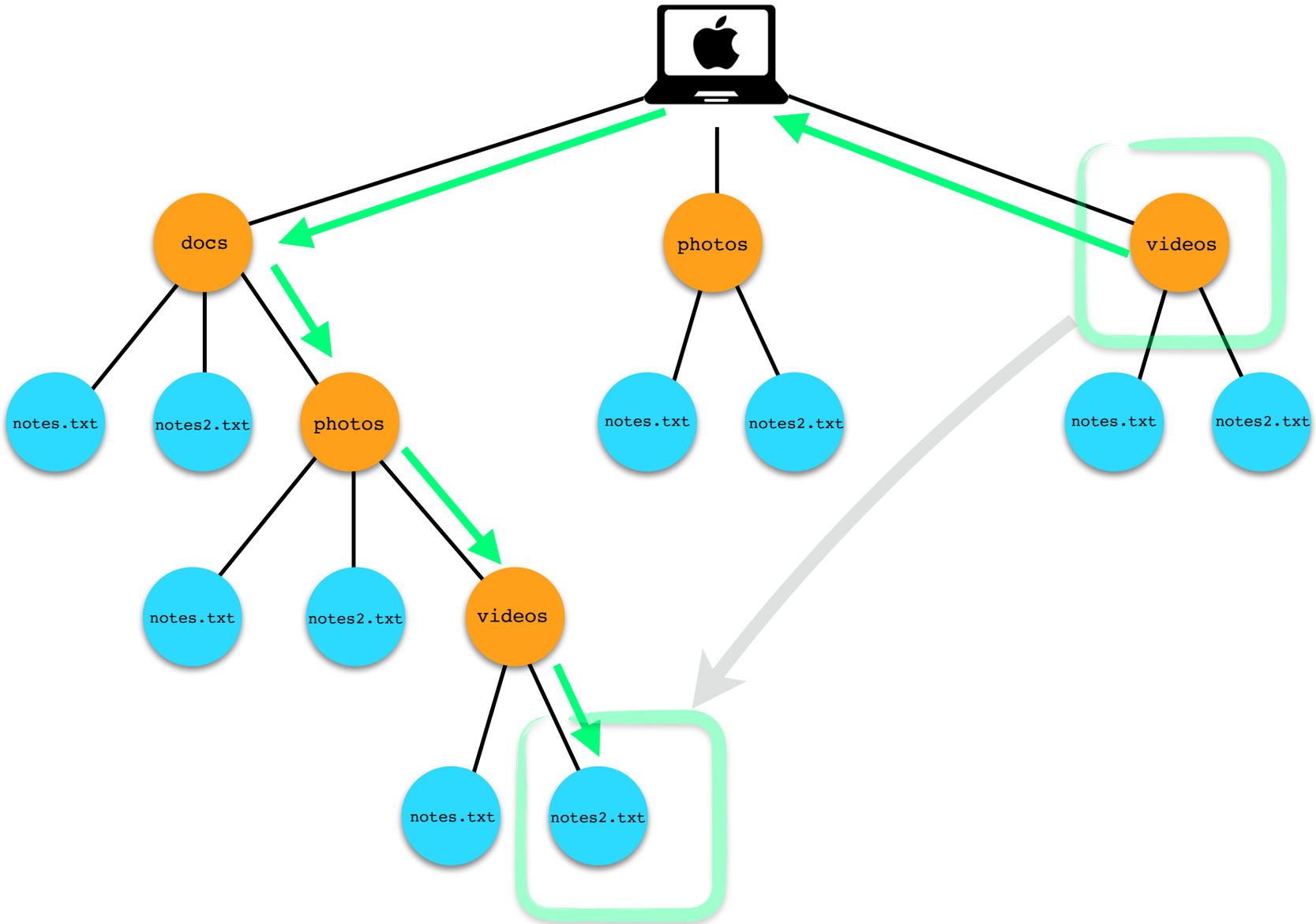
Relative path

`videos / notes2.txt`

File access


docs
photos
videos
notes.txt
notes2.txt
photos
notes.txt
notes2.txt
notes.txt
notes2.txt
notes.txt
notes2.txt
videos
notes.txt
notes2.txt

File access



Relative path

```
.. / docs / photos / videos / notes2.txt
```