Introduction to Shell

Nasser-Eddine MONIR

October 9th, 2024

Lecture

nasser-eddine.monir@inria.fr nasseredd.github.io

Operating Systems (OS)

Definition and Examples

- An **operating system (OS)** is the software that manages and controls the hardware and other software on a computer or device.

Windows

Developed by: Microsoft
Overview: One of the most widely used operating systems for personal computers and laptops.
Use cases: Dominant in business environments, gaming, and home computing.
Key Features: Graphical user interface (GUI), extensive software and hardware compatibility, built-in tools like Windows Explorer.

MacOS

Developed by: Apple Inc.
Overview: A Unix-certified operating system designed specifically for Apple's Mac computers.
Use cases: Popular in creative industries (graphic design, video editing, music production), education, and personal computing.
Key Features: Unix-based, intuitive GUI, strong focus on security and privacy, integration with Apple ecosystem (iCloud, Siri).

Linux

Developed by: Open-source community (initially by Linus Torvalds in 1991)
Overview: A free and open-source Unix-like operating system used for a wide range of purposes, from desktop computers to servers and embedded systems.
Use cases: Commonly used in servers, cloud infrastructure, scientific computing, and by developers.
Key Features: Highly customizable, strong security features, large community support, available in multiple distributions.

Operating Systems (OS)

Definition and Examples

- An **operating system (OS)** is the software that manages and controls the hardware and other software on a computer or device.

Windows

Developed by: Microsoft
Overview: One of the most widely used operating systems for personal computers and laptops.
Use cases: Dominant in business environments, gaming, and home computing.
Key Features: Graphical user interface (GUI), extensive software and hardware compatibility, built-in tools like Windows Explorer.

MacOS

Developed by: Apple Inc.
Overview: A Unix-certified operating system designed specifically for Apple's Mac computers.
Use cases: Popular in creative industries (graphic design, video editing, music production), education, and personal computing.
Key Features: Unix-based, intuitive GUI, strong focus on security and privacy, integration with Apple ecosystem (iCloud, Siri).

Linux

Developed by: Open-source community (initially by Linus Torvalds in 1991)
Overview: A free and open-source Unix-like operating system used for a wide range of purposes, from desktop computers to servers and embedded systems.
Use cases: Commonly used in servers, cloud infrastructure, scientific computing, and by developers.
Key Features: Highly customizable, strong security features, large community support, available in multiple distributions.

Operating Systems (OS)

Definition and Examples

- An **operating system (OS)** is the software that manages and controls the hardware and other software on a computer or device.

Windows

Developed by: Microsoft
Overview: One of the most widely used operating systems for personal computers and laptops.
Use cases: Dominant in business environments, gaming, and home computing.
Key Features: Graphical user interface (GUI), extensive software and hardware compatibility, built-in tools like Windows Explorer.

MacOS

Developed by: Apple Inc.
Overview: A Unix-certified operating system designed specifically for Apple's Mac computers.
Use cases: Popular in creative industries (graphic design, video editing, music production), education, and personal computing.
Key Features: Unix-based, intuitive GUI, strong focus on security and privacy, integration with Apple ecosystem (iCloud, Siri).

Linux

Developed by: Open-source community (initially by Linus Torvalds in 1991)
Overview: A free and open-source Unix-like operating system used for a wide range of purposes, from desktop computers to servers and embedded systems.
Use cases: Commonly used in servers, cloud infrastructure, scientific computing, and by developers.
Key Features: Highly customizable, strong security features, large community support, available in multiple distributions.

Operating Systems (OS)

Definition and Examples

- An **operating system (OS)** is the software that manages and controls the hardware and other software on a computer or device.

Windows

Developed by: Microsoft
Overview: One of the most widely used operating systems for personal computers and laptops.
Use cases: Dominant in business environments, gaming, and home computing.
Key Features: Graphical user interface (GUI), extensive software and hardware compatibility, built-in tools like Windows Explorer.

MacOS

Developed by: Apple Inc.
Overview: A Unix-certified operating system designed specifically for Apple's Mac computers.
Use cases: Popular in creative industries (graphic design, video editing, music production), education, and personal computing.
Key Features: Unix-based, intuitive GUI, strong focus on security and privacy, integration with Apple ecosystem (iCloud, Siri).

Linux

Developed by: Open-source community (initially by Linus Torvalds in 1991)
Overview: A free and open-source Unix-like operating system used for a wide range of purposes, from desktop computers to servers and embedded systems.
Use cases: Commonly used in servers, cloud infrastructure, scientific computing, and by developers.
Key Features: Highly customizable, strong security features, large community support, available in multiple distributions.

Operating Systems (OS)

Did you say Unix or Linux?

What is the difference between UNIX and LINUX ?

Unix

- **Created in 1969** at Bell Labs by Ken Thompson, Dennis Ritchie and others.
- Originally designed as a **multi-user, multi-tasking** operating system.
- Unix was written in **C** to make it portable across different machines, which was revolutionary at the time.
- **Proprietary**: Unix is a commercial product, and the original source code was owned by Bell Labs (now AT&T).

Linux

- **Created in 1991** by Linus Torvalds as a free, open-source alternative to Unix.
- Linux was designed to be a Unix-like system, but it was written from scratch without using any Unix code. This made it a clone of Unix rather than a derivative.
- **Open-source**: Linux is released under the GNU General Public License (GPL), making it free to use, modify, and distribute. The Linux kernel is maintained by a global community of developers.
- **Linux distributions** (such as Ubuntu, Fedora, and Debian) are built around the Linux kernel and often include GNU utilities, giving them the common name **GNU/Linux**.

Operating Systems (OS)

Did you say Unix or Linux?

Unix

- **Created in 1969** at Bell Labs by Ken Thompson, Dennis Ritchie, and others.
- Originally designed as a **multi-user** and **multi-tasking** operating system.
- Unix was written in **C** to make it portable across different machines, which was revolutionary at the time.
- **Proprietary**: Unix is a commercial product, and the original source code was owned by Bell Labs (now AT&T).

Linux

- **Created in 1991** by Linus Torvalds as a free, open-source alternative to Unix.
- Linux was designed to be a **Unix**-like system, but it was written from scratch without using any Unix code. This made it a **clone** of Unix, rather than a derivative.
- **Open-source**: Linux is released under the GNU General Public License (GPL), making it free to use, modify, and distribute. The Linux kernel is maintained by a global community of developers.
- **Linux distributions** (such as Ubuntu, Fedora, and Debian) are built around the Linux kernel and often include GNU utilities, giving them the common name **GNU/Linux**.

Operating Systems (OS)

Did you say Unix or Linux?

Unix

- **Created in 1969** at Bell Labs by Ken Thompson, Dennis Ritchie, and others.
- Originally designed as a **multi-user** and **multi-tasking** operating system.
- Unix was written in **C** to make it portable across different machines, which was revolutionary at the time.
- **Proprietary**: Unix is a commercial product, and the original source code was owned by Bell Labs (now AT&T).

Linux

- **Created in 1991** by Linus Torvalds as a free, open-source alternative to Unix.
- Linux was designed to be a **Unix**-like system, but it was written from scratch without using any Unix code. This made it a **clone** of Unix, rather than a derivative.
- **Open-source**: Linux is released under the GNU General Public License (GPL), making it free to use, modify, and distribute. The Linux kernel is maintained by a global community of developers.
- **Linux distributions** (such as Ubuntu, Kali, and Debian) are built around the Linux kernel and often include GNU utilities, giving them the common name **GNU/Linux**.

Distributions

Linux

- A **distribution** (often shortened to **distro**) refers to a specific version or variant of the **Linux operating system** that is packaged with the Linux kernel, system utilities, and application software.
- A **distribution** typically includes everything you need to run a complete operating system, including a package manager, graphical user interface (GUI), software libraries, and pre-installed applications.

Kali Linux

- **Purpose**: Designed for penetration testing and cybersecurity.
- **Features**: Comes pre-installed with numerous tools for ethical hacking and digital forensics.
- **Target Audience**: Security professionals and ethical hackers.

Ubuntu

- **Purpose**: A user-friendly distribution for desktops, servers, and cloud.
- **Features**: Focuses on ease of use, stability, and strong community support.
- **Target Audience**: General users, developers, and businesses.

Debian

- **Purpose**: Known for its stability and reliability.
- **Features**: Used as the foundation for many other distributions, including Ubuntu; offers free software and long-term support.
- **Target Audience**: developers, server environments, and advanced users.

Kernel

Program

- The **kernel** is a **low-level software program**, written in programming languages (commonly **C**).
- The **kernel** is the **core component** of an operating system.
- It acts as an intermediary between the **hardware** and the **software** running on a computer.
- The kernel's main job is to **manage system resources** (CPU, memory, storage, etc.) and allow different programs to interact with the hardware in a controlled way.
- **Functions**:
  - **Process management**: Handling multiple programs and processes simultaneously.
  - **Memory management**: Allocating and managing the system's memory.
  - **Device management**: Interfacing with hardware devices (e.g., hard drives, printers).
  - **File system management**: Managing files and data storage.
  - **Security and access control**: Enforcing permissions and protecting the system.

Shell

What the (s)hell?

- **Shell** is a command-line interface (CLI) that sits on top of the operating systems and provides a way for users to interact with the OS by typing commands.
- The **shell** takes the commands you type, interprets them, and then passes them to the operating system's kernel to execute them.

Bash
Bourne Again Shell

Bash was designed as a free and improved version of the Bourne shell with backward compatibility, , adding advanced features for both scripting and interactive use.

>> The default shell for many Linux distributions

Zsh
Z shell

Zsh is an extended version of the Bourne Shell (**sh**) with many additional features, including better customization options, powerful auto-completion, and a rich plugin ecosystem.

>> The default shell in macOS

PowerShell

Powershell is developed as a task automation and configuration management framework with an emphasis on **object-oriented scripting**.

>> The default shell in Windows

Shell

What the (s)hell?

- **Shell** is a command-line interface (CLI) that sits on top of the operating systems and provides a way for users to interact with the OS by typing commands.
- The **shell** takes the commands you type, interprets them, and then passes them to the operating system's kernel to execute them.

Bash
Bourne Again Shell

Bash was designed as a free and improved version of the Bourne shell with backward compatibility, , adding advanced features for both scripting and interactive use.

>> The default shell for many Linux distributions

Zsh
Z shell

Zsh is an extended version of the Bourne Shell (**sh**) with many additional features, including better customization options, powerful auto-completion, and a rich plugin ecosystem.

>> The default shell in macOS

PowerShell

Powershell is developed as a task automation and configuration management framework with an emphasis on **object-oriented scripting**.

>> The default shell in Windows

Shell

What the (s)hell?

- **Shell** is a command-line interface (CLI) that sits on top of the operating systems and provides a way for users to interact with the OS by typing commands.
- The **shell** takes the commands you type, interprets them, and then passes them to the operating system's kernel to execute them.

Bash
Bourne Again Shell

Bash was designed as a free and improved version of the Bourne shell with backward compatibility, , adding advanced features for both scripting and interactive use.

>> The default shell for many Linux distributions

Zsh
Z shell

Zsh is an extended version of the Bourne Shell (**sh**) with many additional features, including better customization options, powerful auto-completion, and a rich plugin ecosystem.

>> The default shell in macOS

PowerShell

Powershell is developed as a task automation and configuration management framework with an emphasis on **object-oriented scripting**.

>> The default shell in Windows

Shell

What the (s)hell?

- **Shell** is a command-line interface (CLI) that sits on top of the operating systems and provides a way for users to interact with the OS by typing commands.
- The **shell** takes the commands you type, interprets them, and then passes them to the operating system's kernel to execute them.

Bash
Bourne Again Shell

Bash was designed as a free and improved version of the Bourne shell with backward compatibility, , adding advanced features for both scripting and interactive use.

>> The default shell for many Linux distributions

Zsh
Z shell

Zsh is an extended version of the Bourne Shell (**sh**) with many additional features, including better customization options, powerful auto-completion, and a rich plugin ecosystem.

>> The default shell in macOS

PowerShell

Powershell is developed as a task automation and configuration management framework with an emphasis on **object-oriented scripting**.

>> The default shell in Windows

Shell

What the (s)hell?

- **Shell** is a command-line interface (CLI) that sits on top of the operating systems and provides a way for users to interact with the OS by typing commands.
- The **shell** takes the commands you type, interprets them, and then passes them to the operating system's kernel to execute them.

Bash
Bourne Again Shell

Bash was designed as a free and improved version of the Bourne shell with backward compatibility, , adding advanced features for both scripting and interactive use.

>> The default shell for many Linux distributions

Zsh
Z shell

Zsh is an extended version of the Bourne Shell (**sh**) with many additional features, including better customization options, powerful auto-completion, and a rich plugin ecosystem.

>> The default shell in macOS

PowerShell

Powershell is developed as a task automation and configuration management framework with an emphasis on **object-oriented scripting**.

>> The default shell in Windows

Terminal

Definition

- The terminal is a **program** (also known as a terminal emulator) that provides a user interface between the user and the shell. It displays the input (commands) and output (results) of the shell.
- ⚠ It doesn't process or execute commands itself; it merely **accepts user input** and **displays the output** of the shell's operations.

```
user@hostname:~$ 
```

Terminal

Definition

- The terminal is a **program** (also known as a terminal emulator) that provides a user interface between the user and the shell. It displays the input (commands) and output (results) of the shell.
- ⚠️ It doesn't process or execute commands itself; it merely **accepts user input** and **displays the output** of the shell's operations.

```
user@hostname:~$ 
```

Terminal

Definition

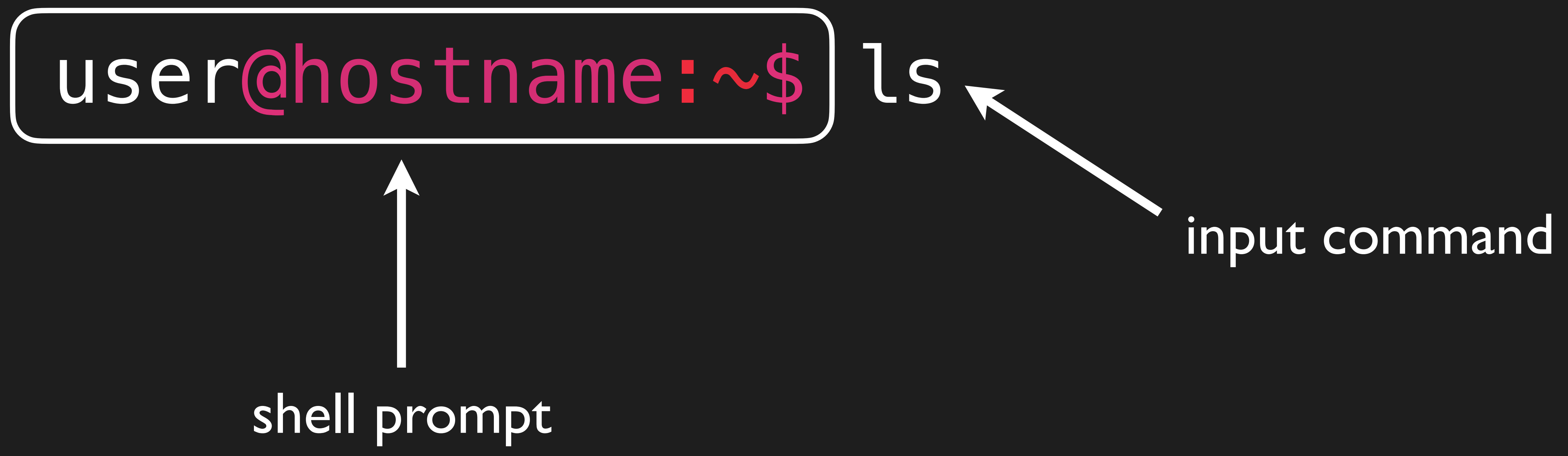
- The terminal is a **program** (also known as a terminal emulator) that provides a user interface between the user and the shell. It displays the input (commands) and output (results) of the shell.
- ⚠️ It doesn't process or execute commands itself; it merely **accepts user input** and **displays the output** of the shell's operations.

```
user@hostname:~$
```

Terminal

Definition

- The terminal is a **program** (also known as a terminal emulator) that provides a user interface between the user and the shell. It displays the input (commands) and output (results) of the shell.
- ⚠️ It doesn't process or execute commands itself; it merely **accepts user input** and **displays the output** of the shell's operations.



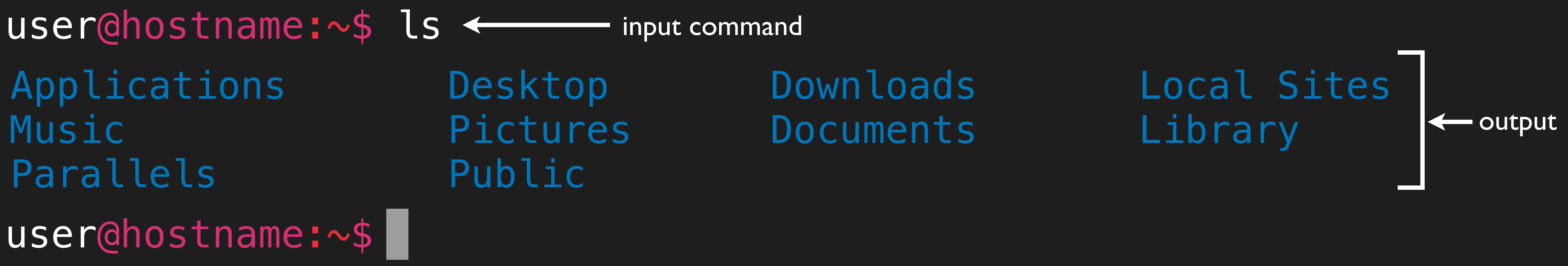
```
user@hostname:~$ ls
```

Desktop — bash

Terminal

Definition

- The terminal is a **program** (also known as a terminal emulator) that provides a user interface between the user and the shell. It displays the input (commands) and output (results) of the shell.
- ⚠️ It doesn't process or execute commands itself; it merely **accepts user input** and **displays the output** of the shell's operations.

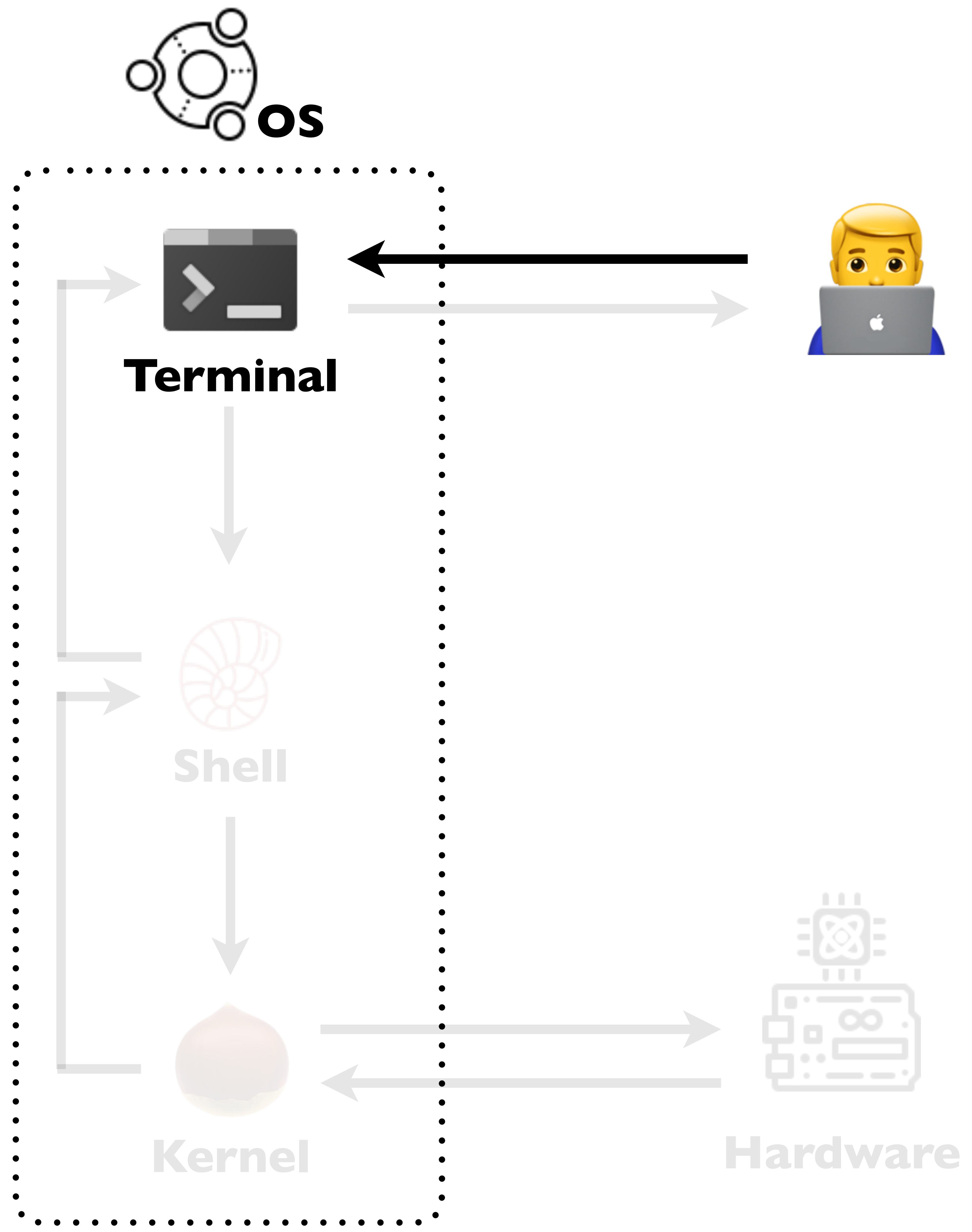


```
user@hostname:~$ ls
Applications        Desktop     Downloads        Local Sites
Music               Pictures    Documents        Library
Parallels           Public
user@hostname:~$ 
```

Shell

Step-by-Step Process

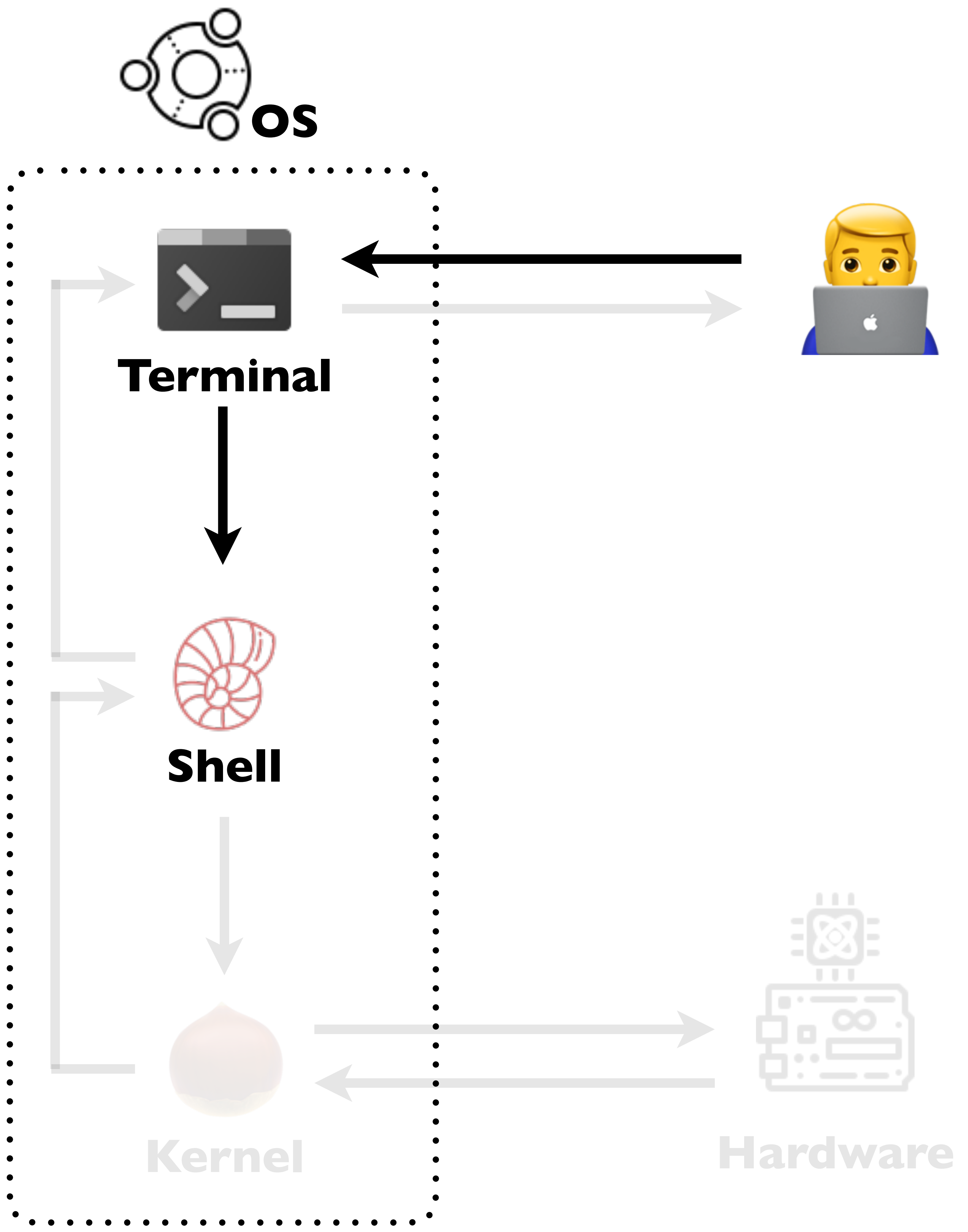
1. User Enters a Command in the **Terminal**
2. Terminal Passes the Command to the **Shell**
3. **Shell** Interprets the Command
4. **Shell Forks** a New Process
5. **Shell** send the Command to the Kernel for **Execution** (exec)
6. **Kernel** Manages System Resources for the command
7. Command **Executes** and Produces **Output**
8. **Shell** Sends the **Output** to the **Terminal**
9. **Shell** Returns to **Waiting State**



Shell

Step-by-Step Process

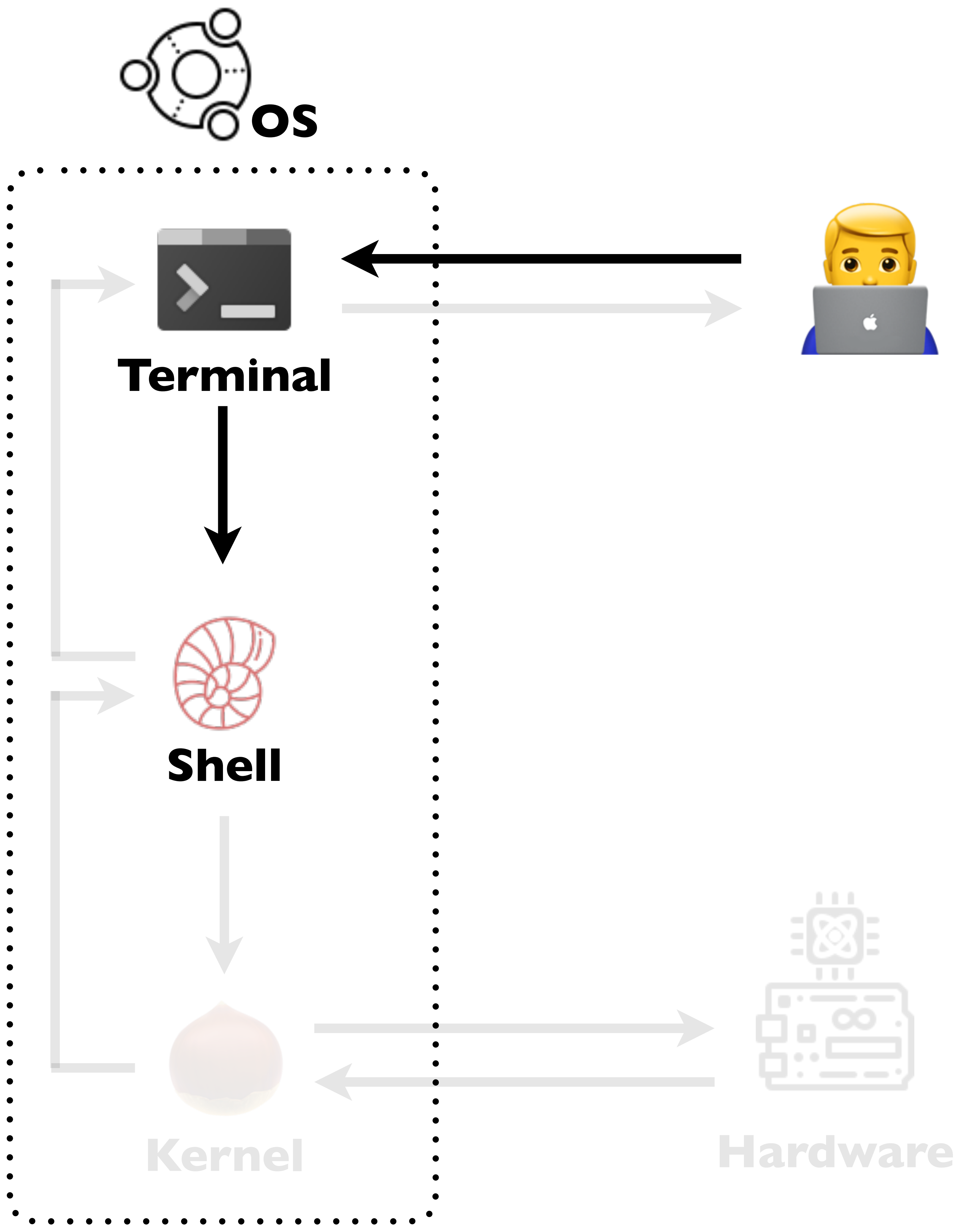
1. User Enters a Command in the **Terminal**
2. Terminal Passes the Command to the **Shell**
3. **Shell** Interprets the Command
4. **Shell Forks** a New Process
5. **Shell** send the Command to the Kernel for **Execution** (exec)
6. **Kernel** Manages System Resources for the command
7. Command **Executes** and Produces **Output**
8. **Shell** Sends the **Output** to the **Terminal**
9. **Shell** Returns to **Waiting State**



Shell

Step-by-Step Process

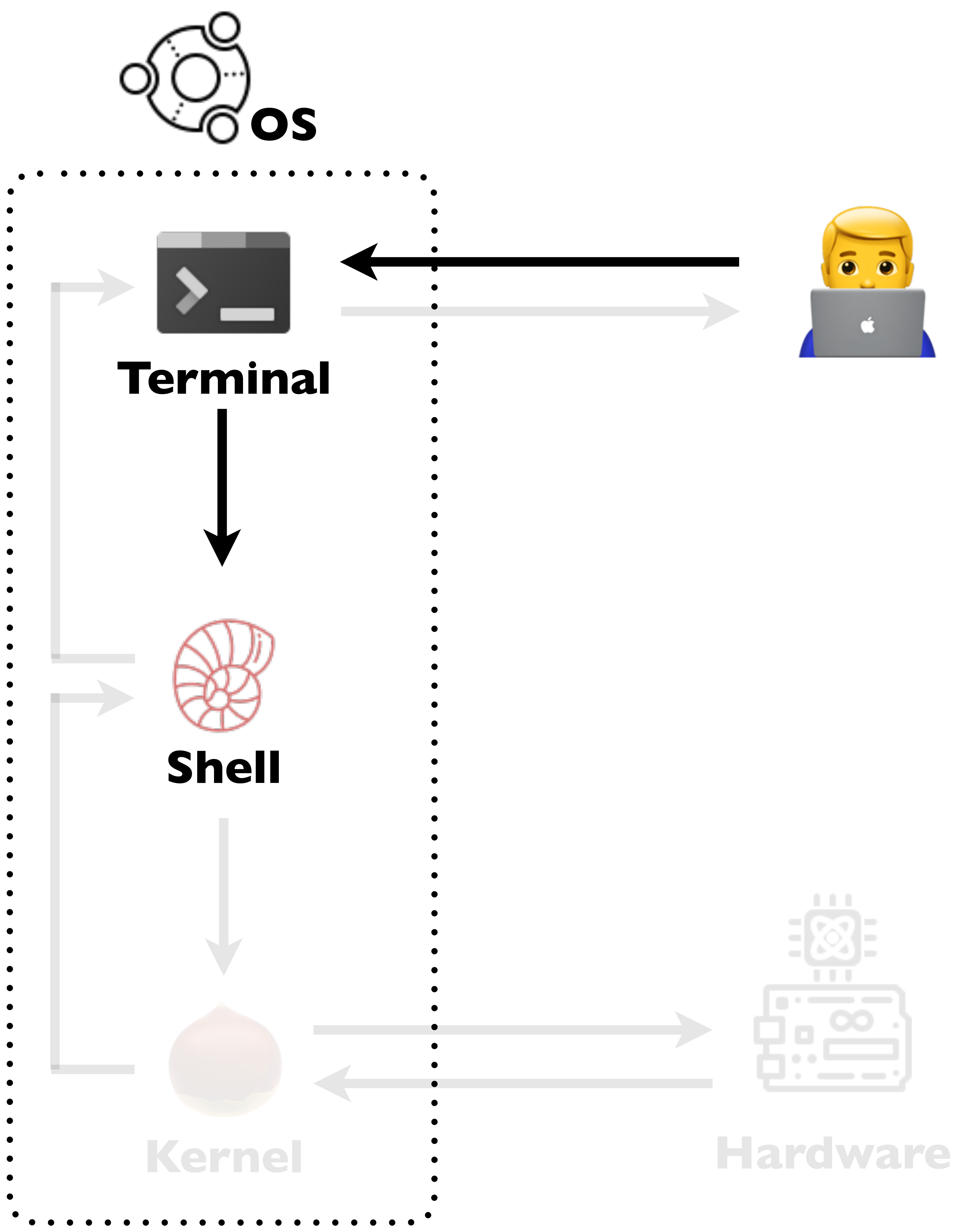
1. User Enters a Command in the **Terminal**
2. Terminal Passes the Command to the **Shell**
3. **Shell** Interprets the Command
4. **Shell Forks** a New Process
5. **Shell** send the Command to the Kernel for **Execution** (exec)
6. **Kernel** Manages System Resources for the command
7. Command **Executes** and Produces **Output**
8. **Shell** Sends the **Output** to the **Terminal**
9. **Shell** Returns to **Waiting State**



Shell

Step-by-Step Process

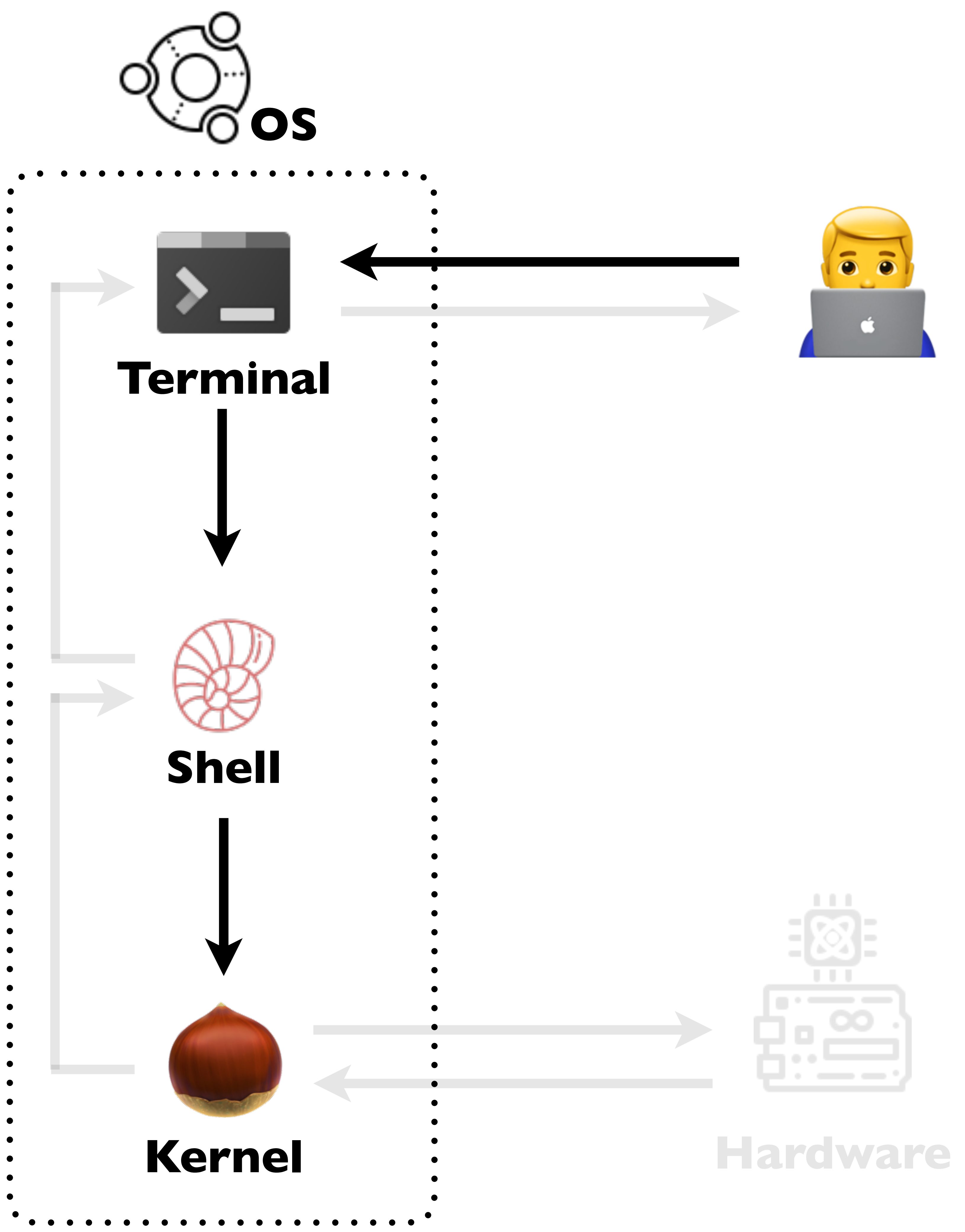
1. User Enters a Command in the **Terminal**
2. Terminal Passes the Command to the **Shell**
3. **Shell** Interprets the Command
4. **Shell Forks** a New Process
5. **Shell** send the Command to the Kernel for **Execution** (exec)
6. **Kernel** Manages System Resources for the command
7. Command **Executes** and Produces **Output**
8. **Shell** Sends the **Output** to the **Terminal**
9. **Shell** Returns to **Waiting State**



Shell

Step-by-Step Process

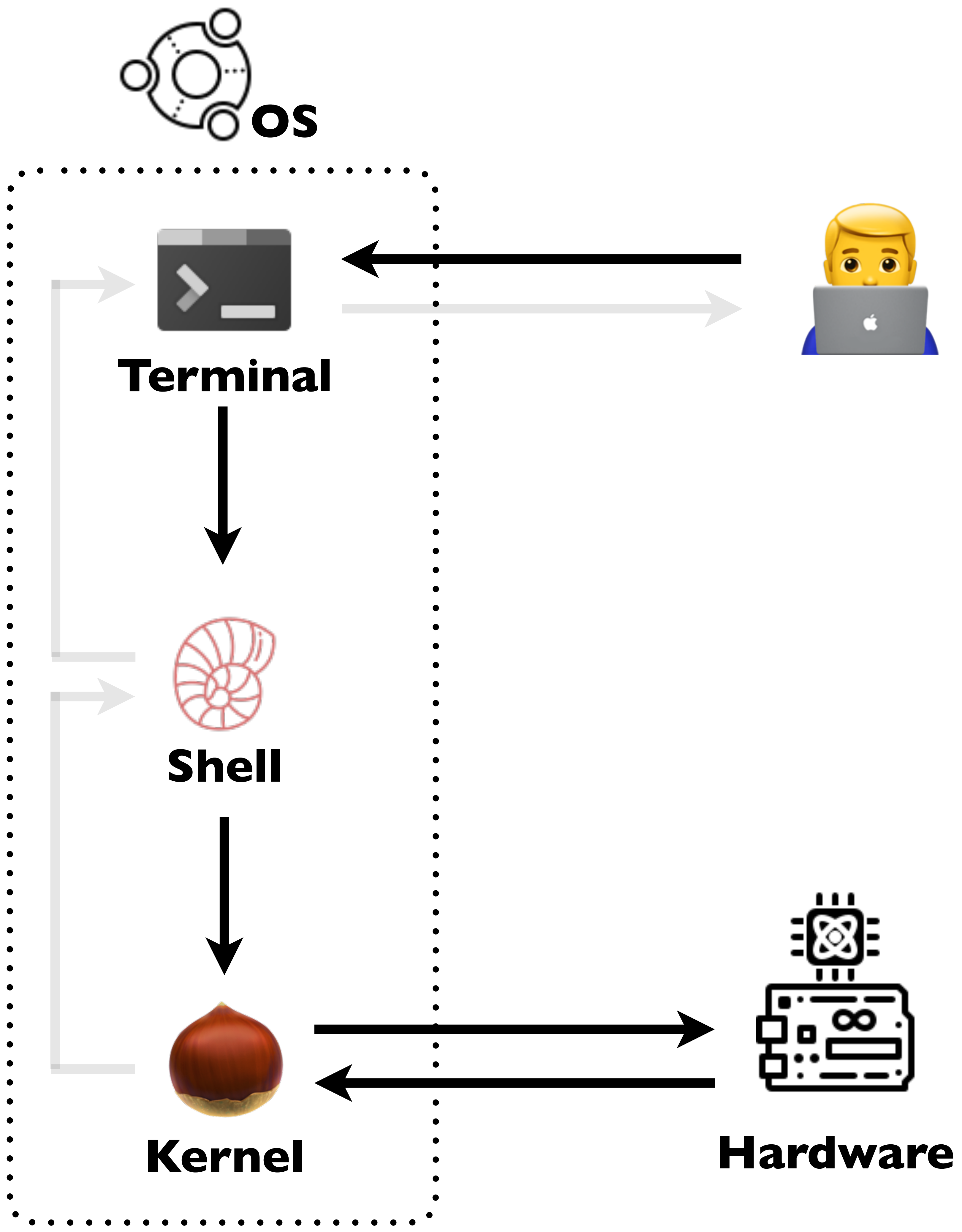
1. User Enters a Command in the **Terminal**
2. Terminal Passes the Command to the **Shell**
3. **Shell** Interprets the Command
4. **Shell Forks** a New Process
5. **Shell** send the Command to the **Kernel** for **Execution** (exec)
6. **Kernel** Manages System Resources for the command
7. Command **Executes** and Produces **Output**
8. **Shell** Sends the **Output** to the **Terminal**
9. **Shell** Returns to **Waiting State**



Shell

Step-by-Step Process

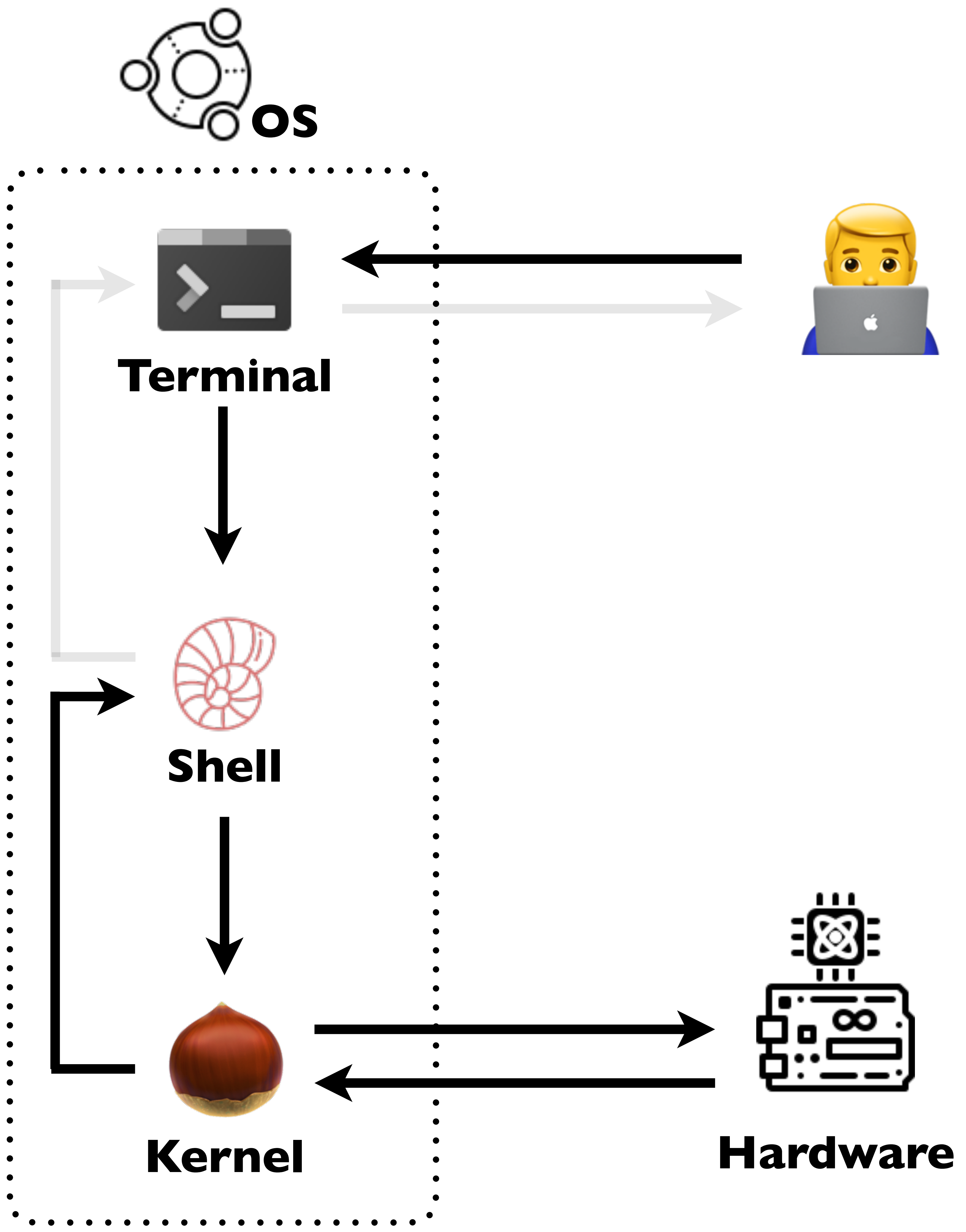
1. User Enters a Command in the **Terminal**
2. Terminal Passes the Command to the **Shell**
3. **Shell** Interprets the Command
4. **Shell Forks** a New Process
5. **Shell** send the Command to the Kernel for **Execution** (exec)
6. **Kernel** Manages System Resources for the command
7. Command **Executes** and Produces **Output**
8. **Shell** Sends the **Output** to the **Terminal**
9. **Shell** Returns to **Waiting State**



Shell

Step-by-Step Process

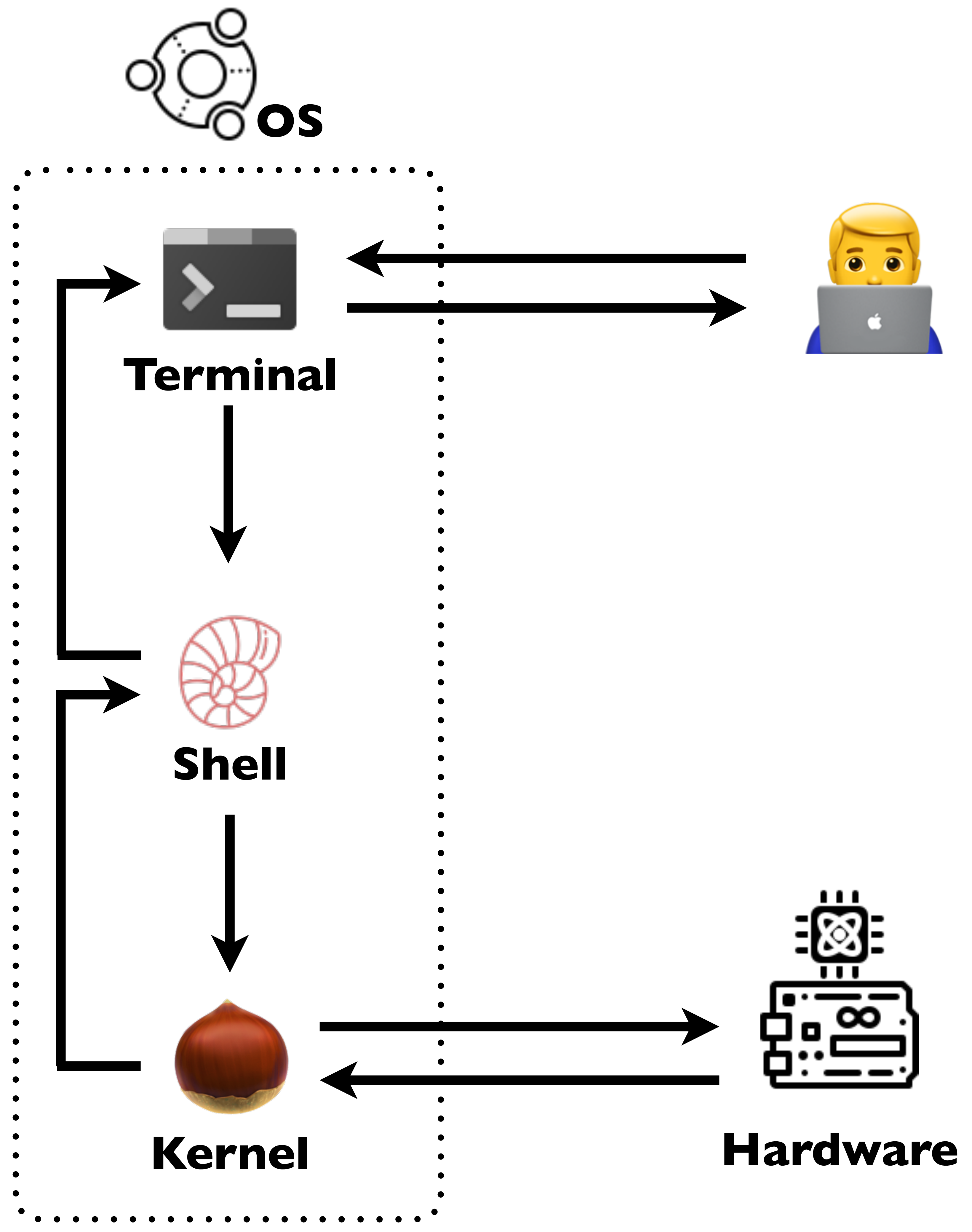
1. User Enters a Command in the **Terminal**
2. Terminal Passes the Command to the **Shell**
3. **Shell** Interprets the Command
4. **Shell Forks** a New Process
5. **Shell** send the Command to the Kernel for **Execution** (exec)
6. **Kernel** Manages System Resources for the command
7. Command **Executes** and Produces **Output**
8. **Shell** Sends the **Output** to the **Terminal**
9. **Shell** Returns to **Waiting State**



Shell

Step-by-Step Process

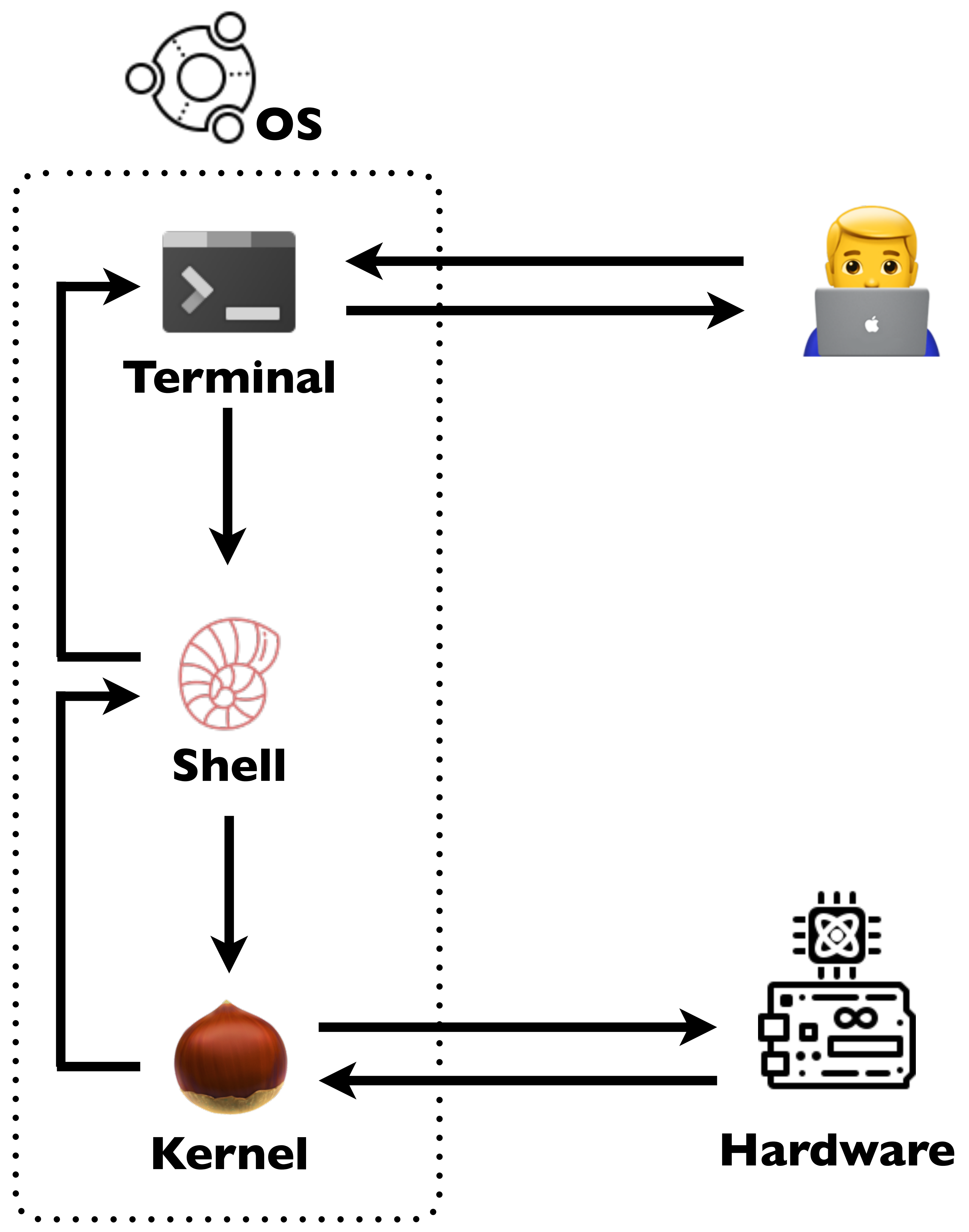
1. User Enters a Command in the **Terminal**
2. Terminal Passes the Command to the **Shell**
3. **Shell** Interprets the Command
4. **Shell Forks** a New Process
5. **Shell** send the Command to the Kernel for **Execution** (exec)
6. **Kernel** Manages System Resources for the command
7. Command **Executes** and Produces **Output**
8. **Shell** Sends the **Output** to the **Terminal**
9. **Shell** Returns to **Waiting State**



Shell

Step-by-Step Process

1. User Enters a Command in the **Terminal**
2. Terminal Passes the Command to the **Shell**
3. **Shell** Interprets the Command
4. **Shell Forks** a New Process
5. **Shell** send the Command to the Kernel for **Execution** (exec)
6. **Kernel** Manages System Resources for the command
7. Command **Executes** and Produces **Output**
8. **Shell** Sends the **Output** to the **Terminal**
9. **Shell** Returns to **Waiting State**



Shells

General Syntax

Shell

```
Desktop/ $  <cmd> -opt1 -opt2 -opt3  arg1 arg2 arg3
```

- Desktop/ : current directory
- $: regular user
- <cmd> : shell command
- -opt1 -opt2 -opt3 : options
- arg1 arg2 arg3 : arguments

Shells

General Syntax

Shell

```
Desktop/ $  <cmd> -opt1 -opt2 -opt3  arg1 arg2 arg3
```

- Desktop/ : current directory
- $: regular user
- <cmd> : shell command
- -opt1 -opt2 -opt3 : options
- arg1 arg2 arg3 : arguments

Shells

General Syntax

Shell

```
Desktop/ $  <cmd> -opt1 -opt2 -opt3  arg1 arg2 arg3
```

- Desktop/ : current directory
- $: regular user
- <cmd> : shell command
- -opt1 -opt2 -opt3 : options
- arg1 arg2 arg3 : arguments

Shells

General Syntax

Shell

```
Desktop/ $  <cmd> -opt1 -opt2 -opt3  arg1 arg2 arg3
```

- Desktop/ : current directory
- $: regular user
- <cmd> : shell command
- -opt1 -opt2 -opt3 : options
- arg1 arg2 arg3 : arguments

Shells

General Syntax

Shell

```
Desktop/ $  <cmd> -opt1 -opt2 -opt3  arg1 arg2 arg3
```

- Desktop/ : current directory
- $: regular user
- <cmd> : shell command
- -opt1 -opt2 -opt3 : options
- arg1 arg2 arg3 : arguments

Shells
Example

Shell

```
Desktop/ $  <cmd> -opt1 -opt2 -opt3  arg1 arg2 arg3
```

Example

```
Desktop/ $  grep -i -r -n  "error" /var/log /home/user/logs
```

```
Desktop/ $  grep -irn  "error" /var/log /home/user/logs
```

Shells

Example

Shell

```
Desktop/ $  <cmd> -opt1 -opt2 -opt3  arg1 arg2 arg3
```

Example

```
Desktop/ $  grep -i -r -n  "error" /var/log /home/user/logs
```

```
Desktop/ $  grep -irn  "error" /var/log /home/user/logs
```

Shells

Users

User

- regular user
- root user

Shell

```
$
```

```
#
```

Shells

Basic Commands

Print

```
$   echo “Hello World”
```

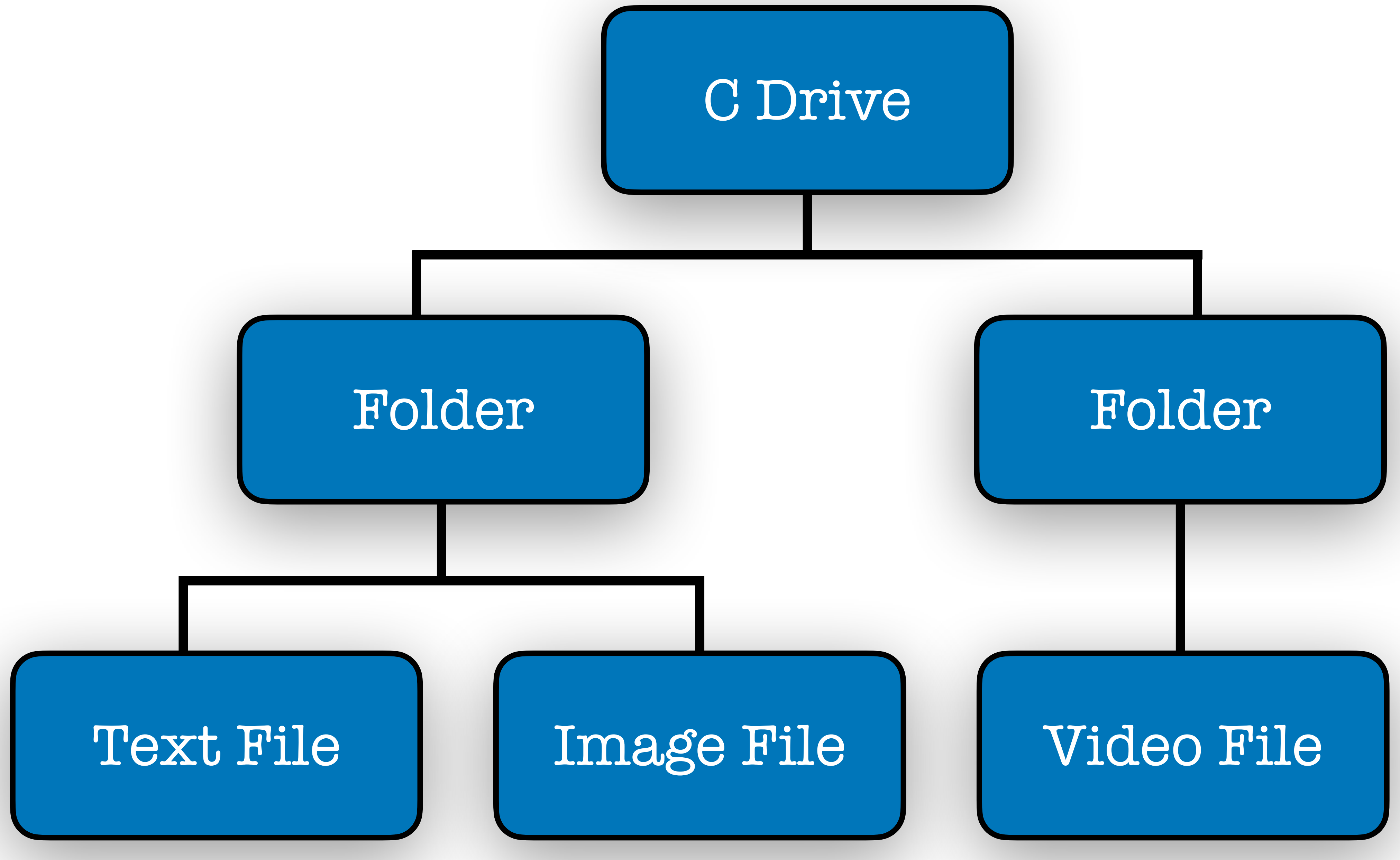
Help/Documentation

```
$   man ls
```

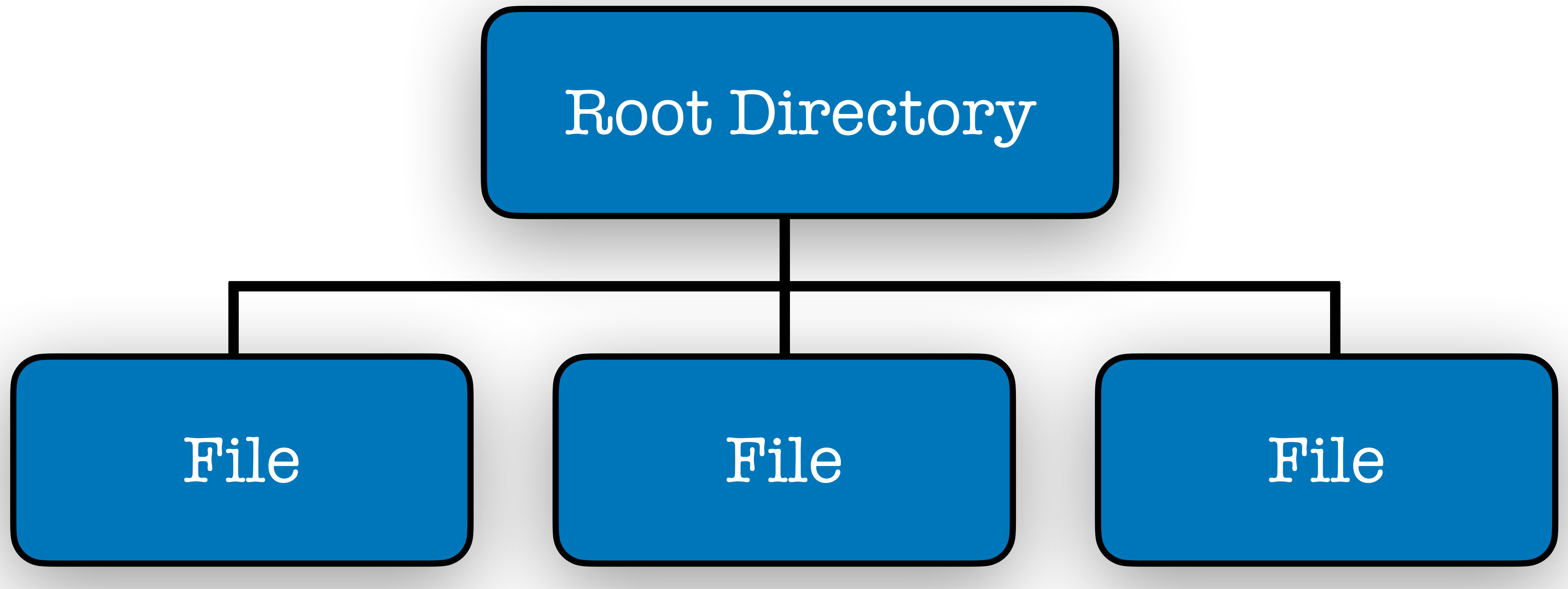
File System

Windows vs Unix-like OS

Windows



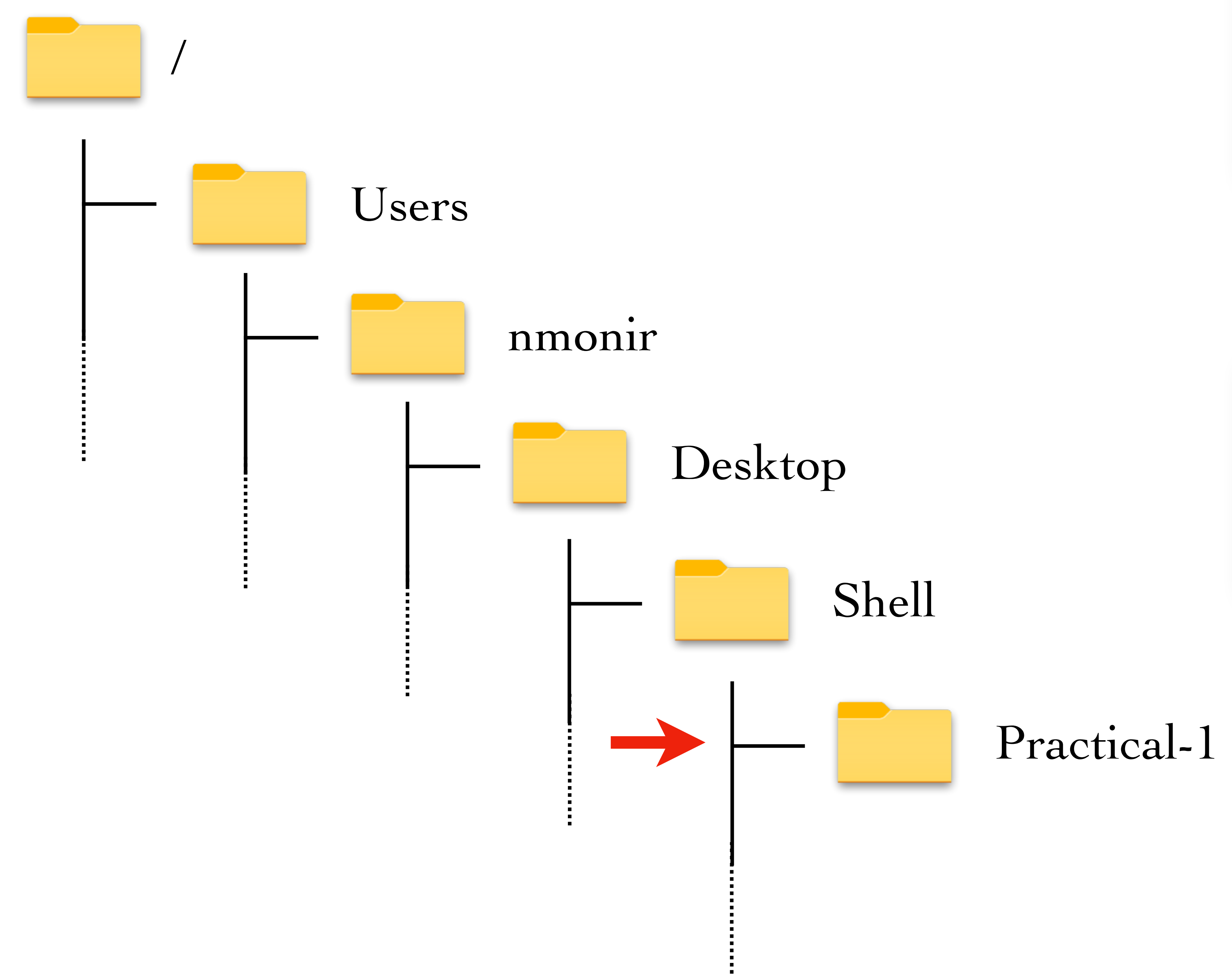
Unix-like OS



Navigating the File System

Absolute Path vs Relative Path

Tree



Absolute Path

```
>> /Users/nmonir/Desktop/Shell/Practical-1/
```

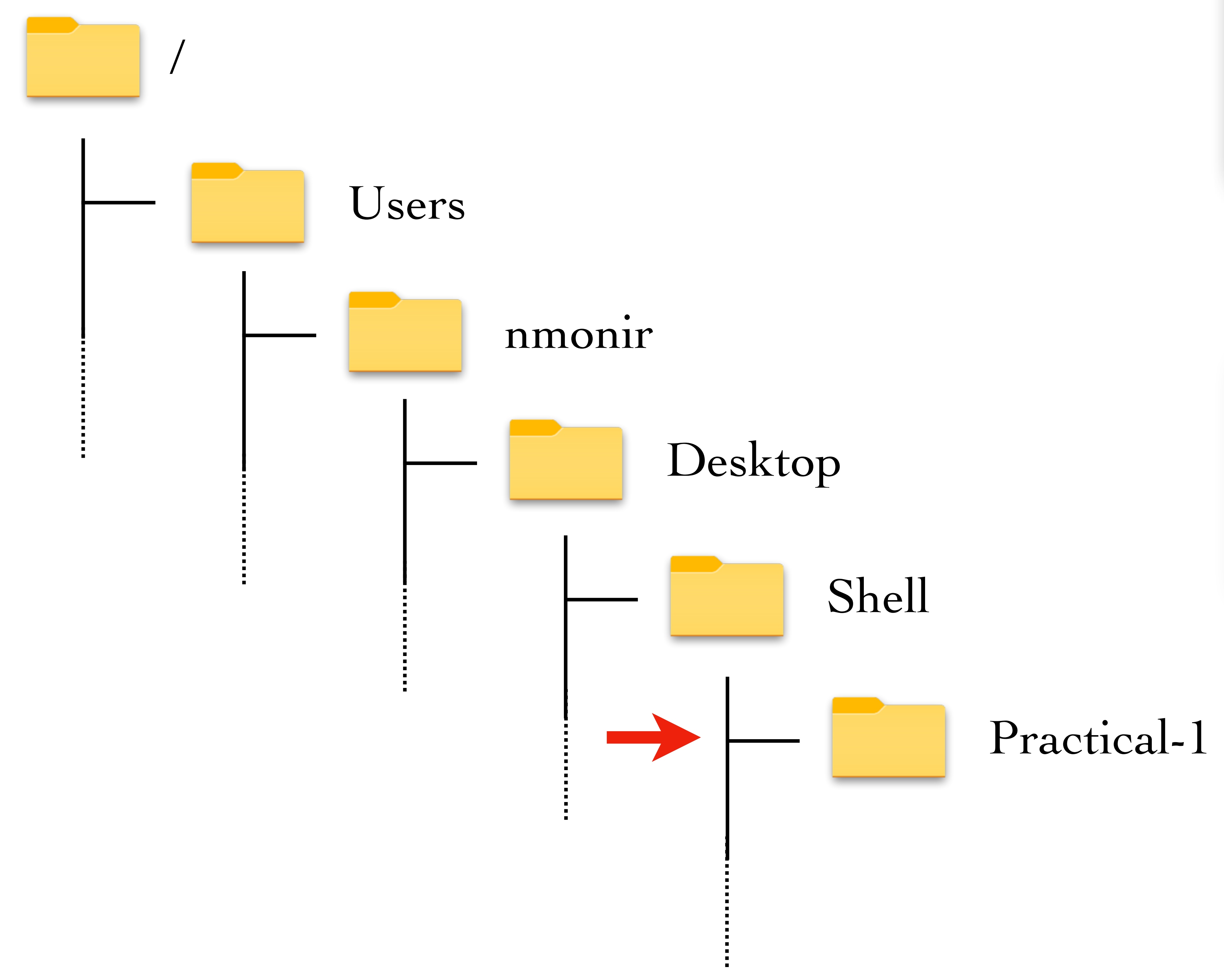
Relative Path

```
>> Practical-1/
```

Navigating the File System

Absolute Path vs Relative Path

Tree



Absolute Path

```
>> /Users/nmonir/Desktop/Shell/Practical-1/
```

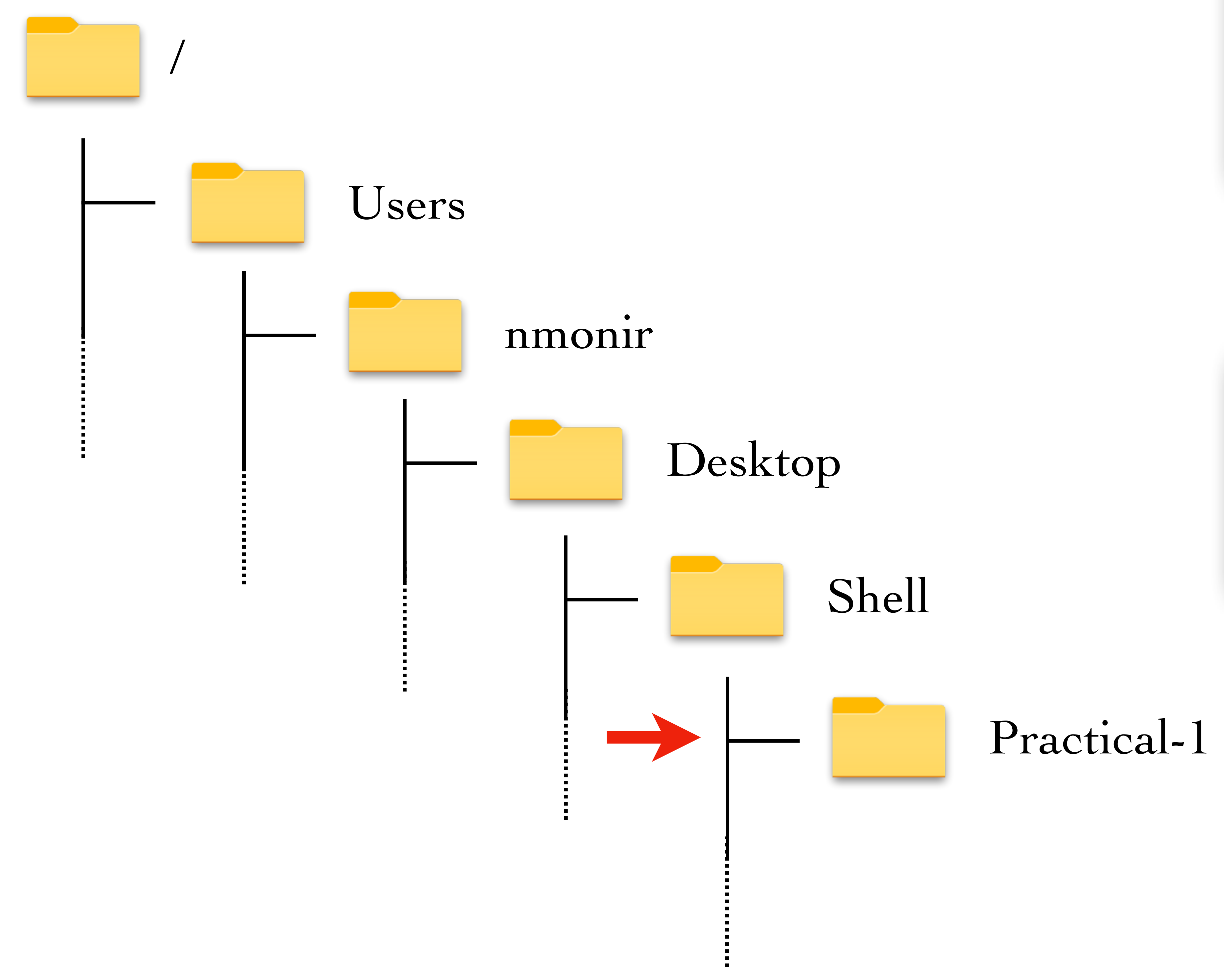
Relative Path

```
>> Practical-1/
```

Navigating the File System

Absolute Path vs Relative Path

Tree



Absolute Path

```
>> /Users/nmonir/Desktop/Shell/Practical-1/
```

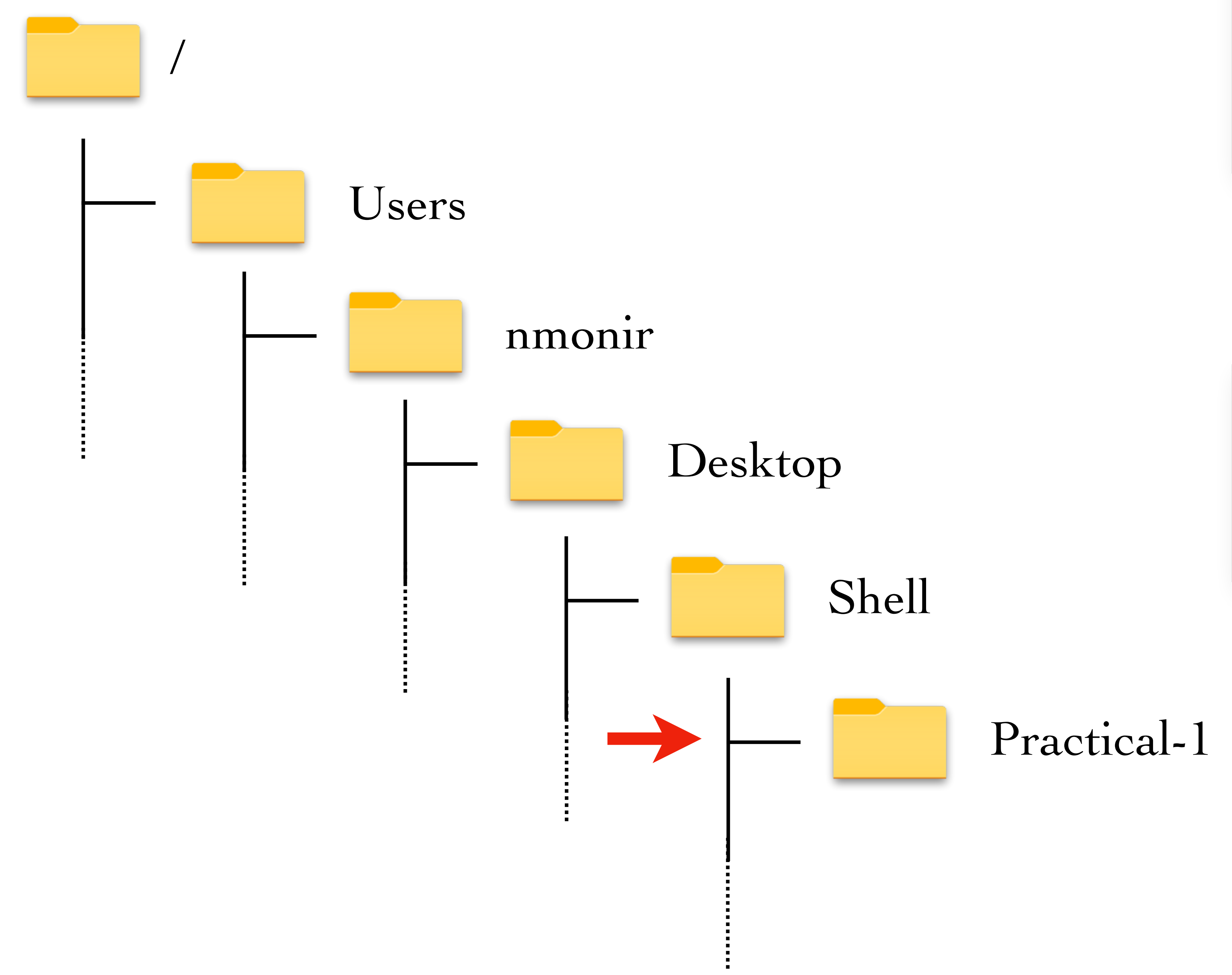
Relative Path

```
>> Practical-1/
```

Navigating the File System

Print Working Directory

Tree



Shell

```
Practical-1/ $  pwd
```

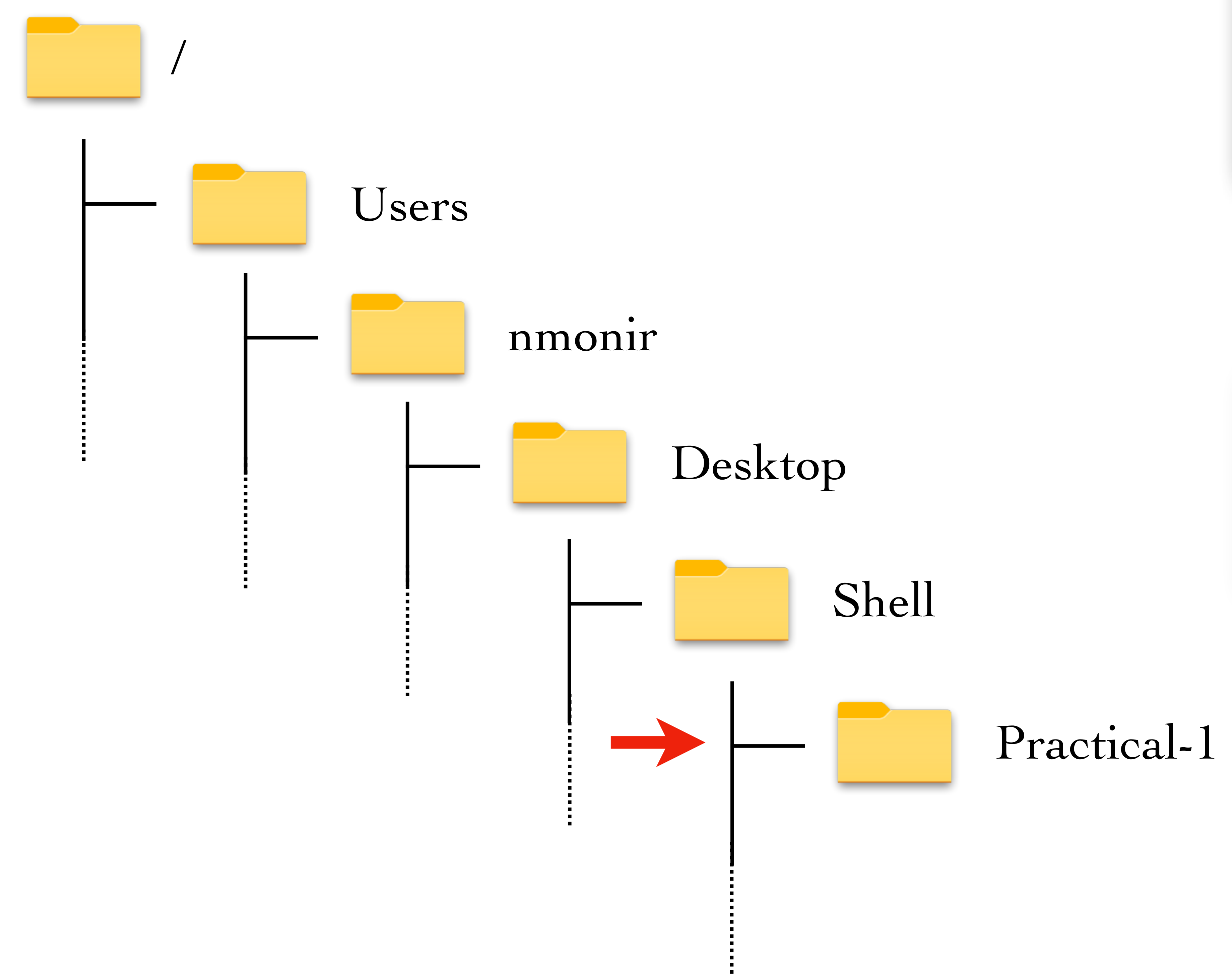
Output

```
>> /Users/nmonir/Desktop/Shell/Practical-1/
```

Navigating the File System

Print Working Directory

Tree



Shell

```
Practical-1/ $ pwd
```

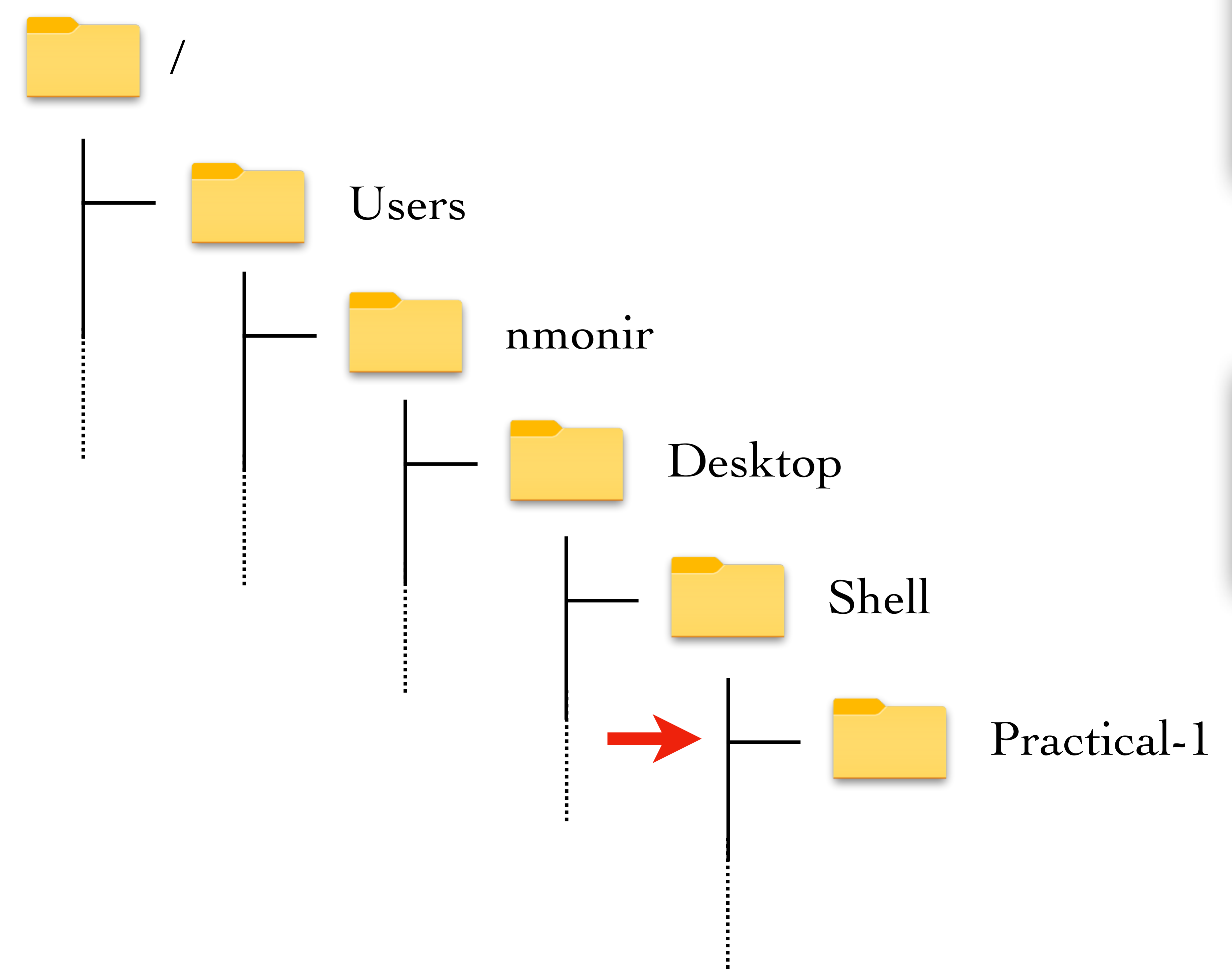
Output

```
>> /Users/nmonir/Desktop/Shell/Practical-1/
```

Navigating the File System

Print Working Directory

Tree



Shell

```
Practical-1/ $ pwd
```

Output

```
>> /Users/nmonir/Desktop/Shell/Practical-1/
```

Navigating the File System

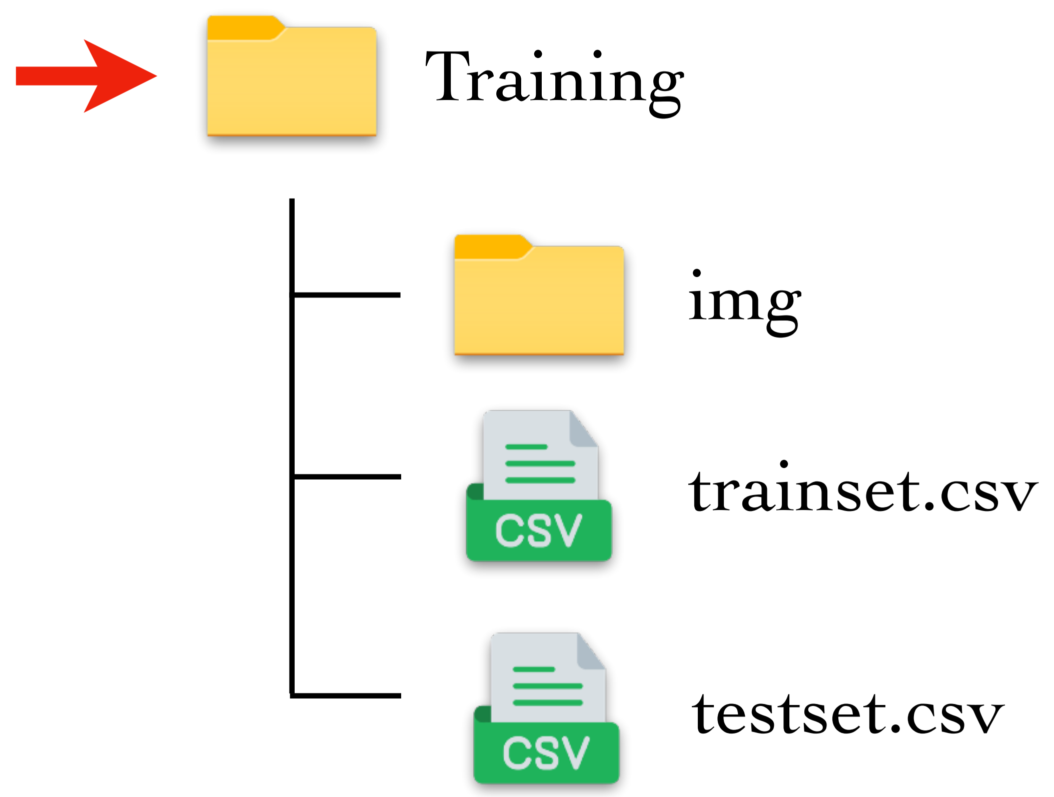
Working Directories

/ Root working directory	. Current working directory
~ User working directory	.. Previous directory

Navigating the File System

List (files and directories)

Tree



Shell

```
Training/$ ls
```

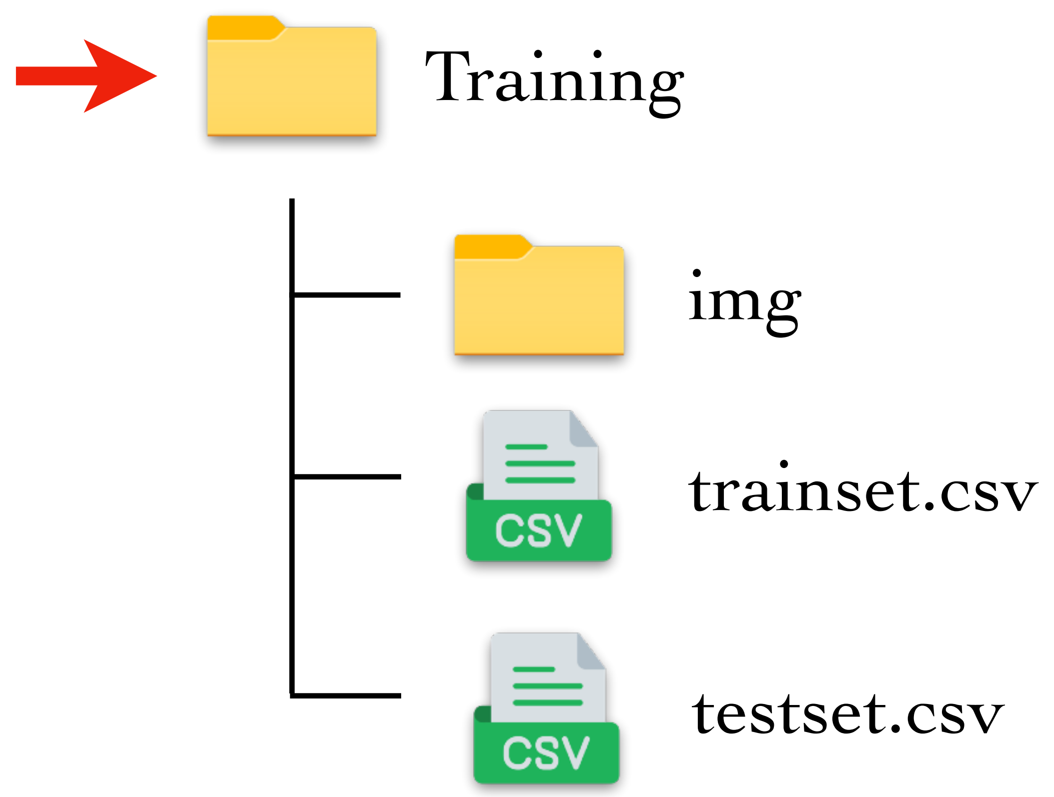
Output

```
>> img   trainset.csv   testset.csv
```

Navigating the File System

List (files and directories)

Tree



Shell

```
Training/$ ls
```

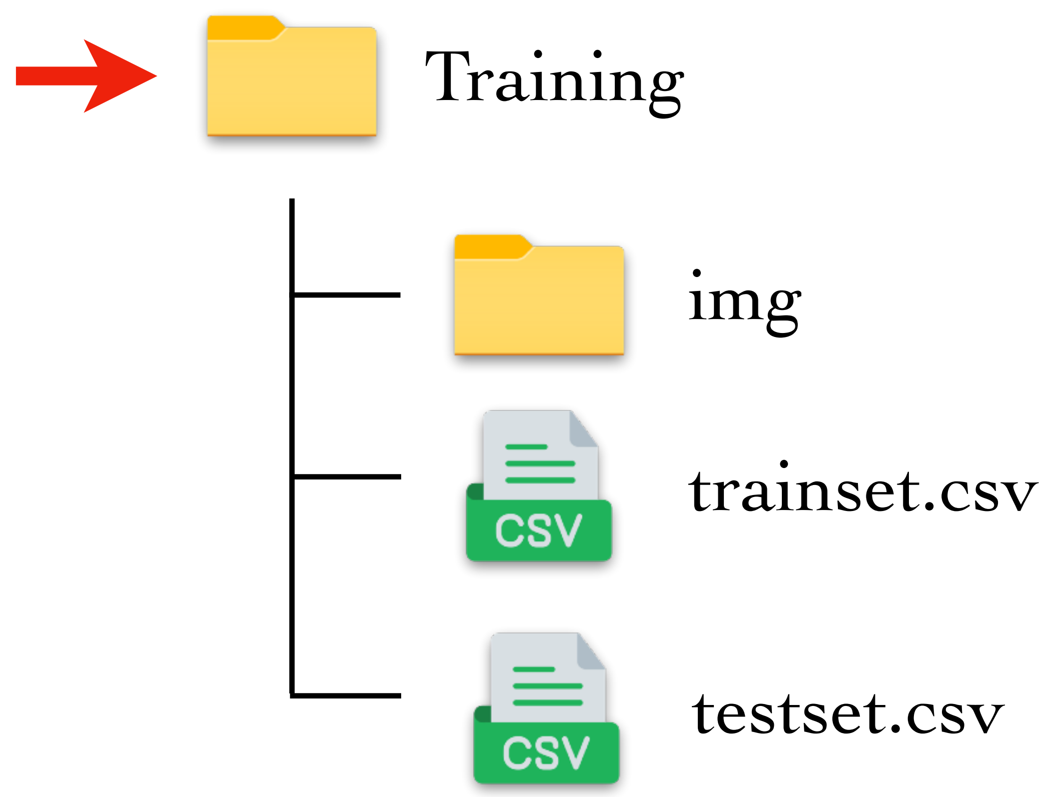
Output

```
>>  img   trainset.csv   testset.csv
```

Navigating the File System

List (files and directories)

Tree



Shell

```
Training/$ ls
```

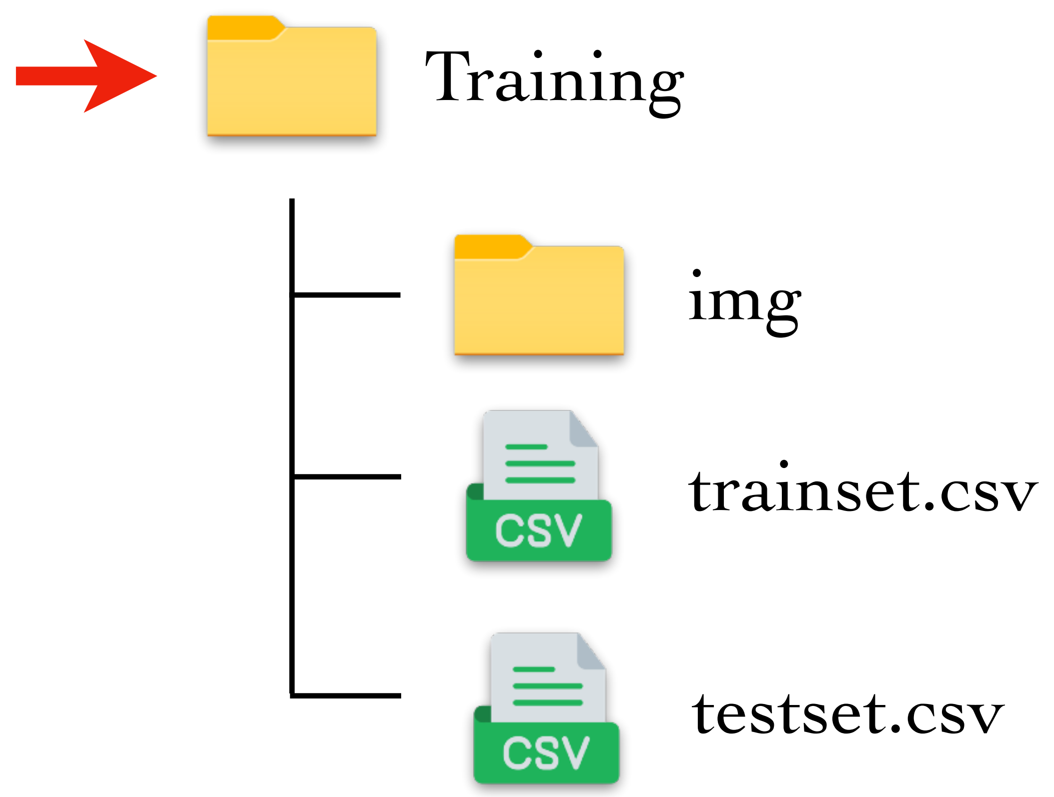
Output

```
>>  img   trainset.csv   testset.csv
```

Navigating the File System

Change Directory

Tree



Shell

```
Training/$ cd img/
```

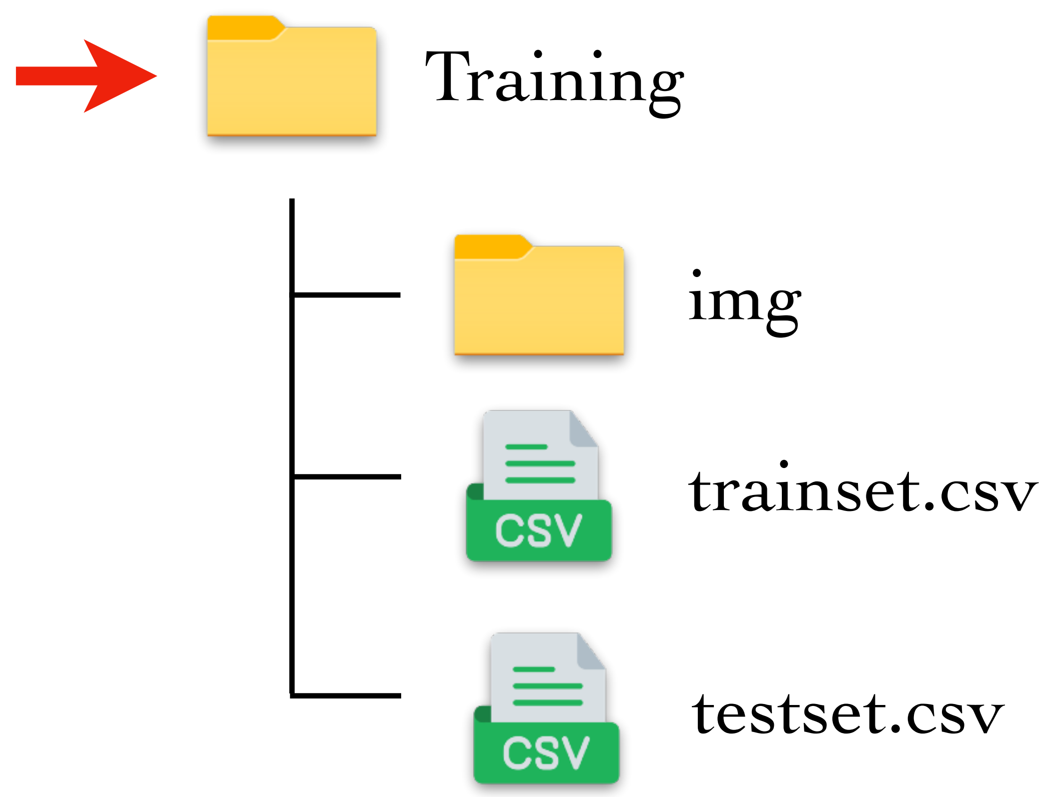
Output

```
img/$
```

Navigating the File System

Change Directory

Tree



Shell

```
Training/$ cd img/
```

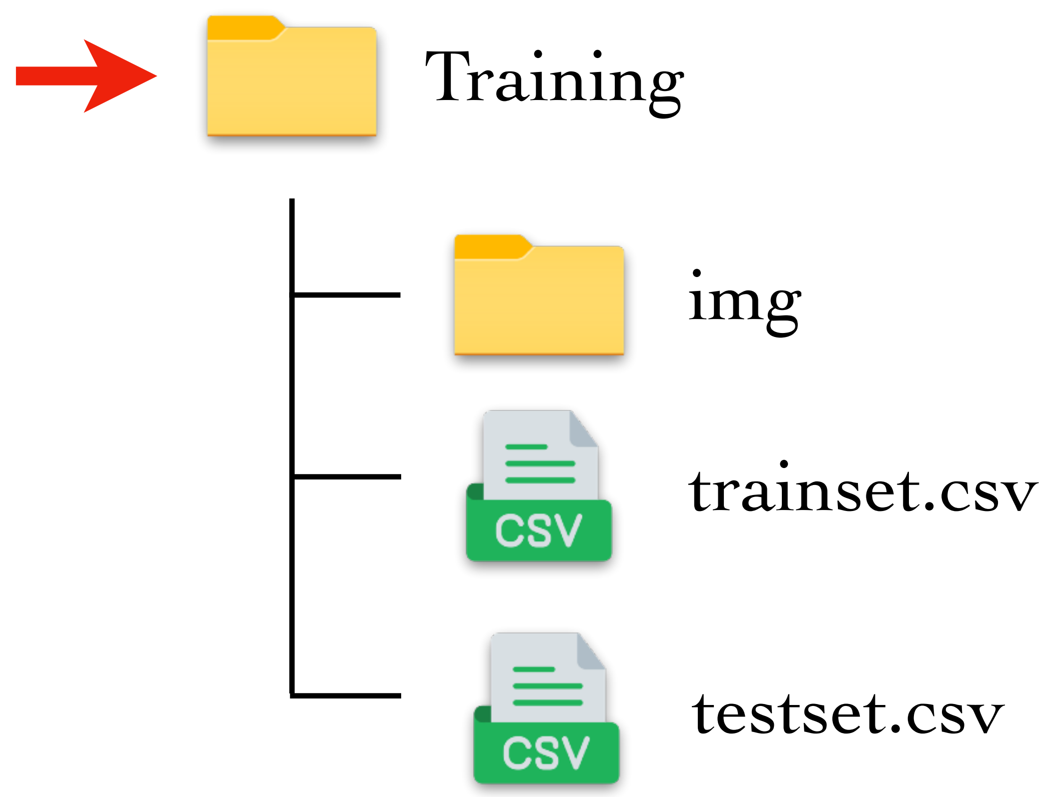
Output

```
img/$
```

Navigating the File System

Change Directory

Tree



Shell

```
Training/$ cd img/
```

Output

```
img/$
```

Shells

Question Time ?

I am in my user working directory. What happens if I run the following command ?

```
...$ cd ~
```

A Change to Root directory

B Change to Desktop directory

C One step back

D Nothing

Shells

Question Time ?

I am in my user working directory. What happens if I run the following command ?

```
...$ cd ~
```

A Change to Root directory

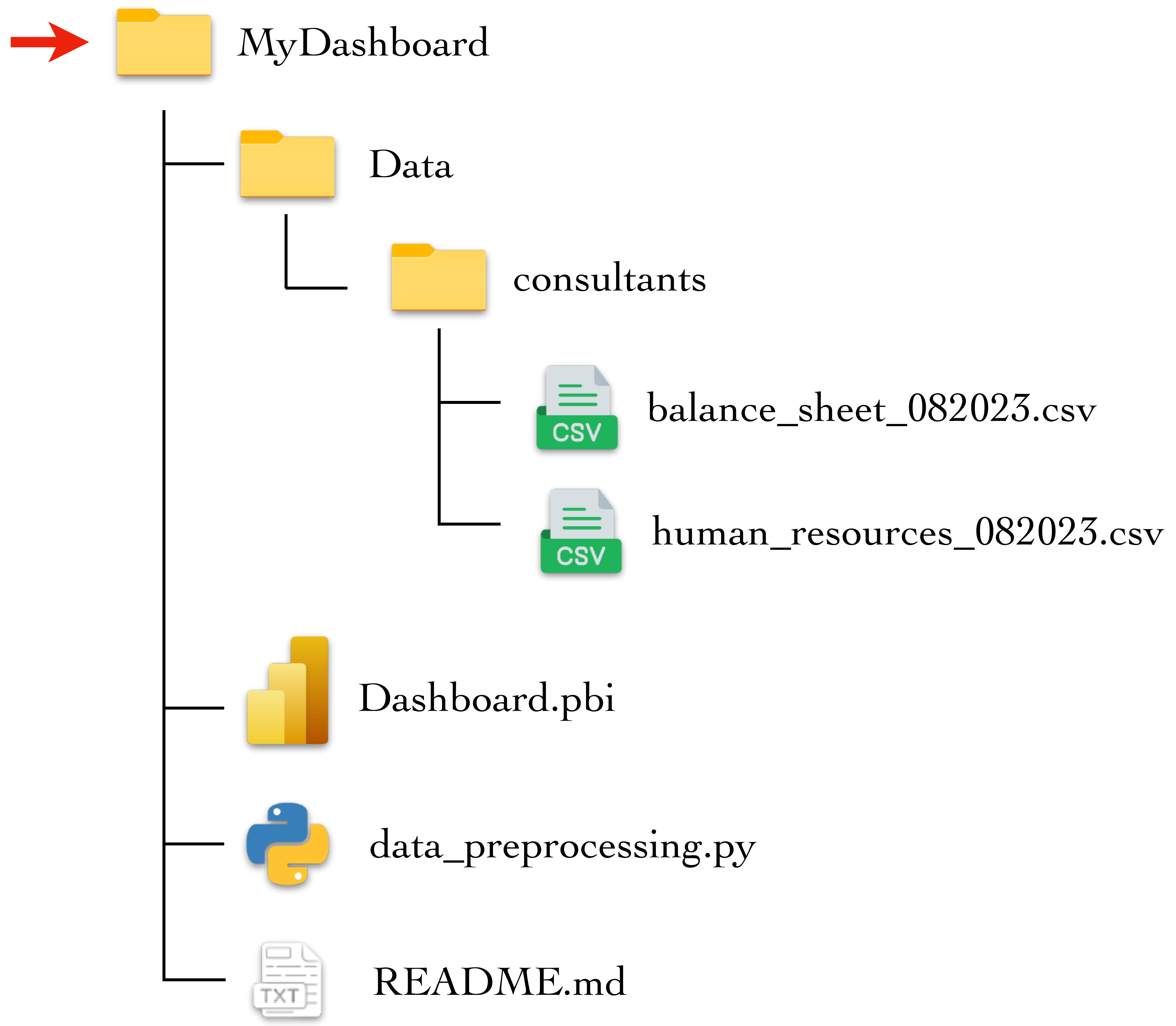
B Change to Desktop directory

C One step back

D Nothing

Navigating the File System

Question 1

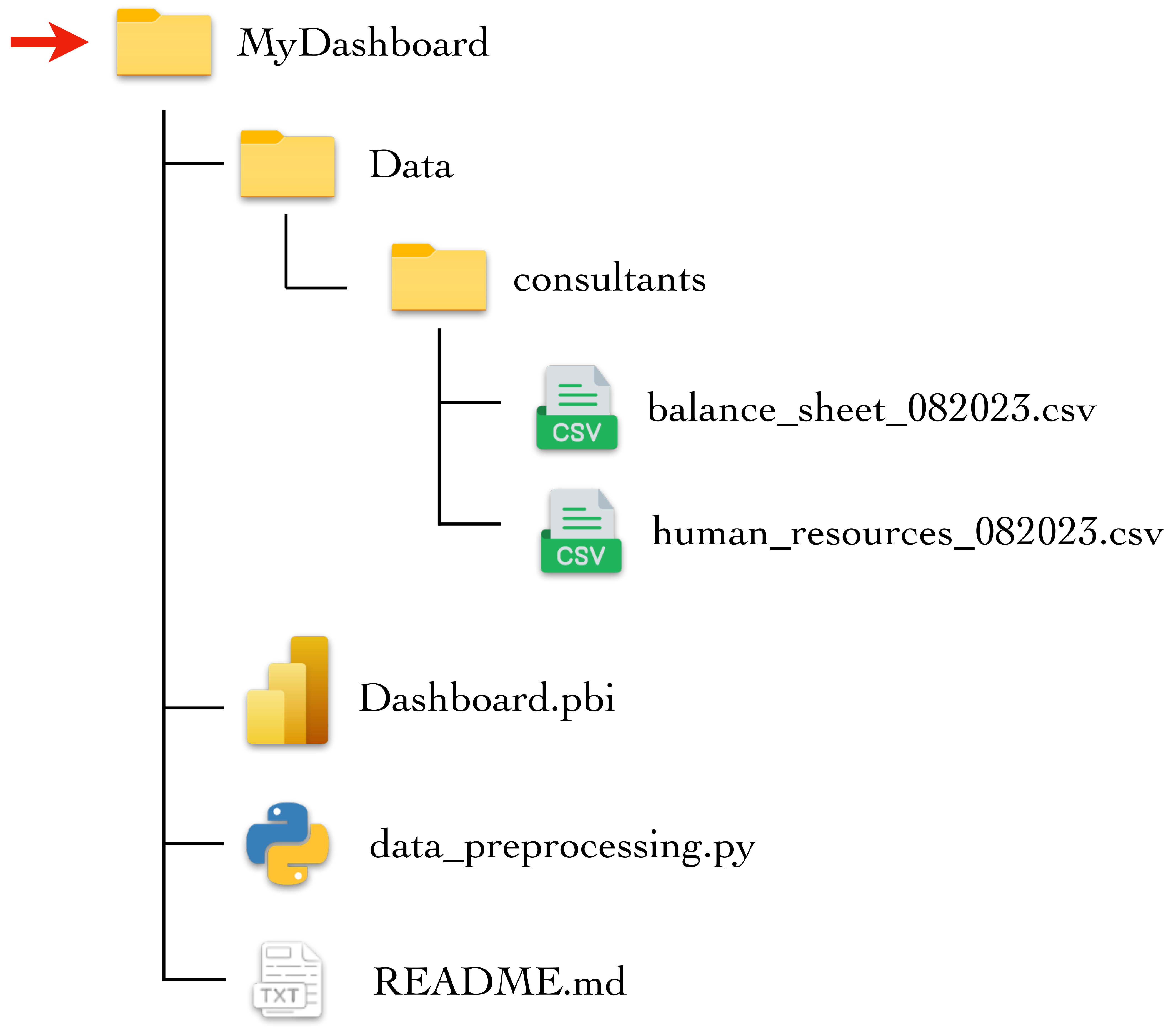


What is the depth of balance_sheet_082023.csv ?

A	1
B	2
C	3
D	4

Navigating the File System

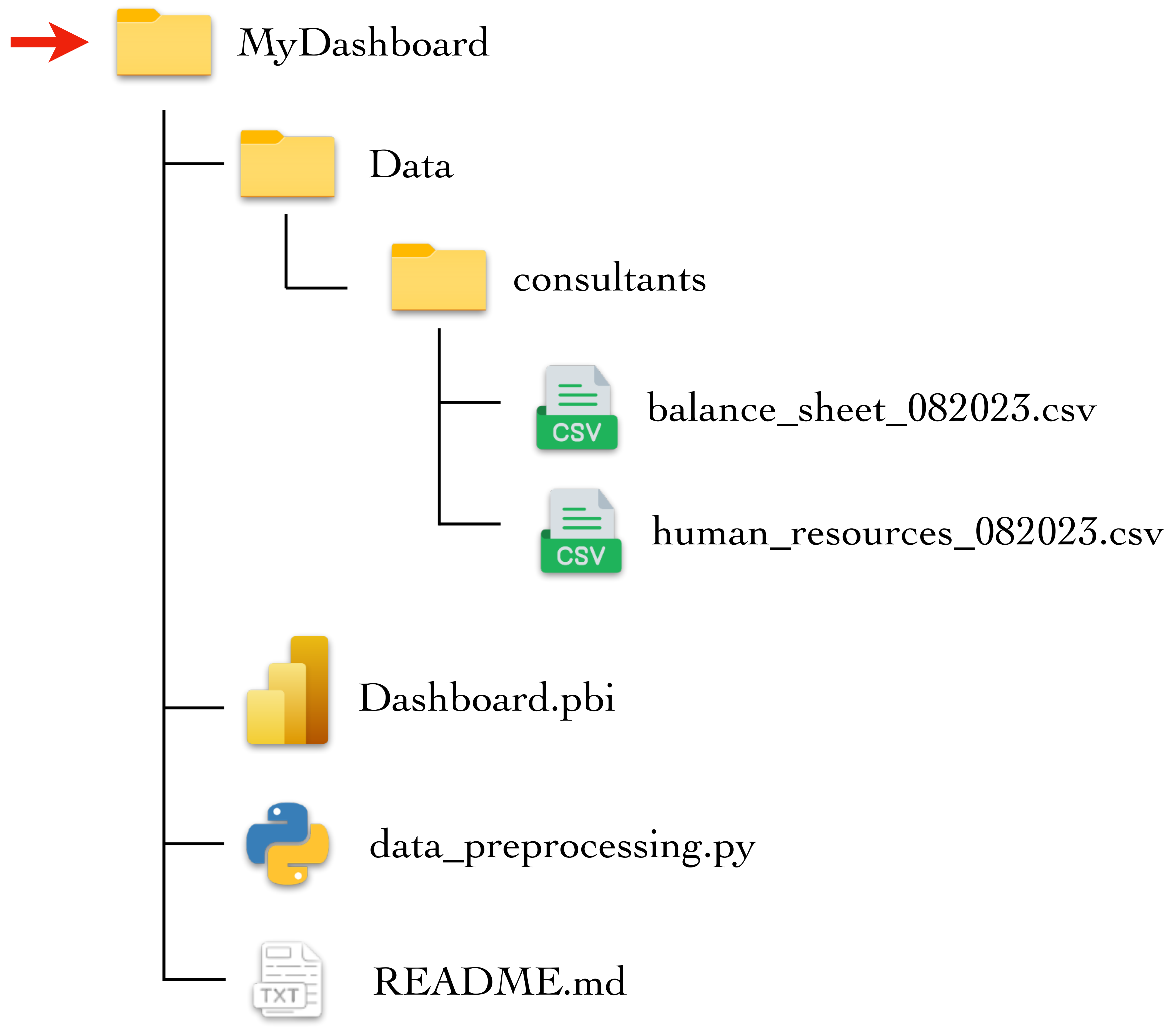
Question 1



What is the depth of balance_sheet_082023.csv ?

- A 1
- B 2
- [✓ C 3]
- D 4

Navigating the File System
Question 2



Which command allows you to display the list of files in consultants ?

A ls consultants

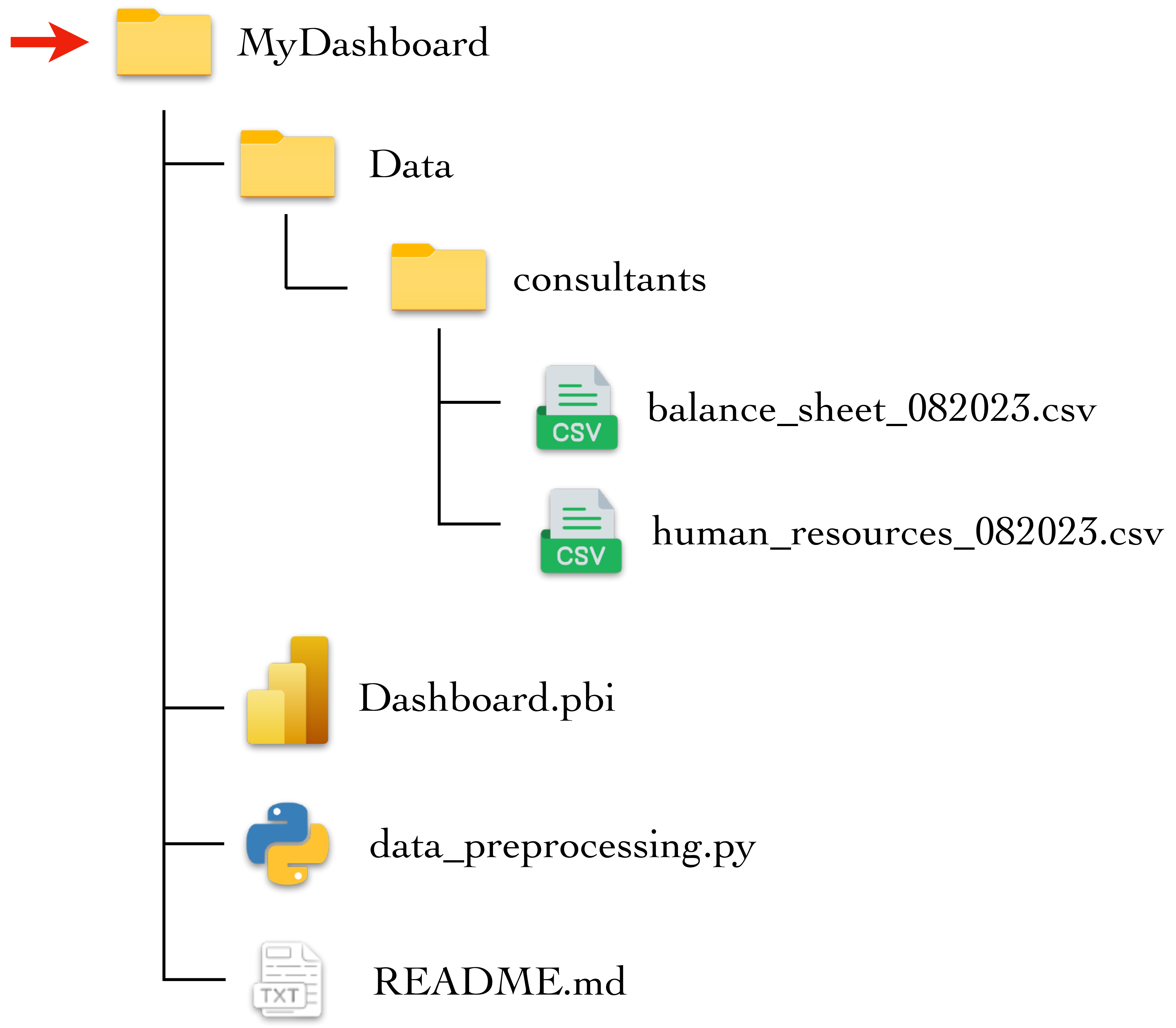
B cd Data/Consultants

C ls Data/Consultants

D ls Data/consultants

Navigating the File System

Question 2



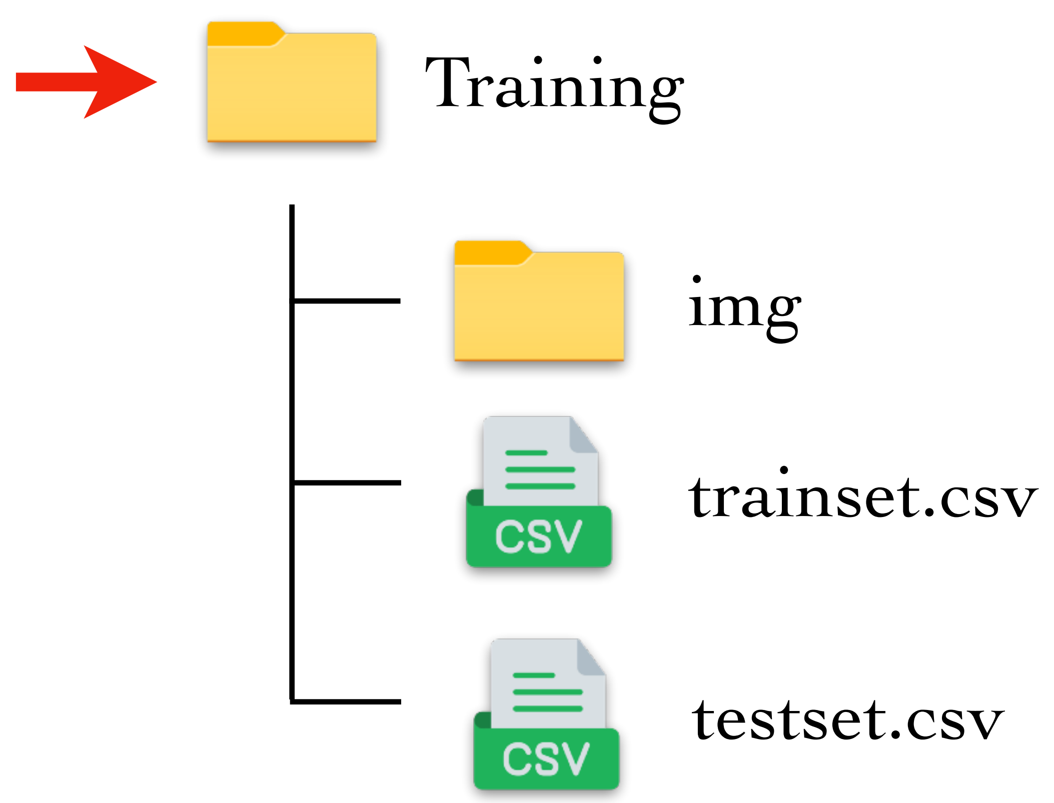
Which command allows you to display the list of files in consultants ?

- A ls consultants
- B cd Data/Consultants
- C ls Data/Consultants
- [✓ D ls Data/consultants]

File Operations

Creating a file (1/2)

Tree



Shell

```
Training/$ touch README.md
```

Output

```
Training/$
```

File Operations

Creating a file (2/2)

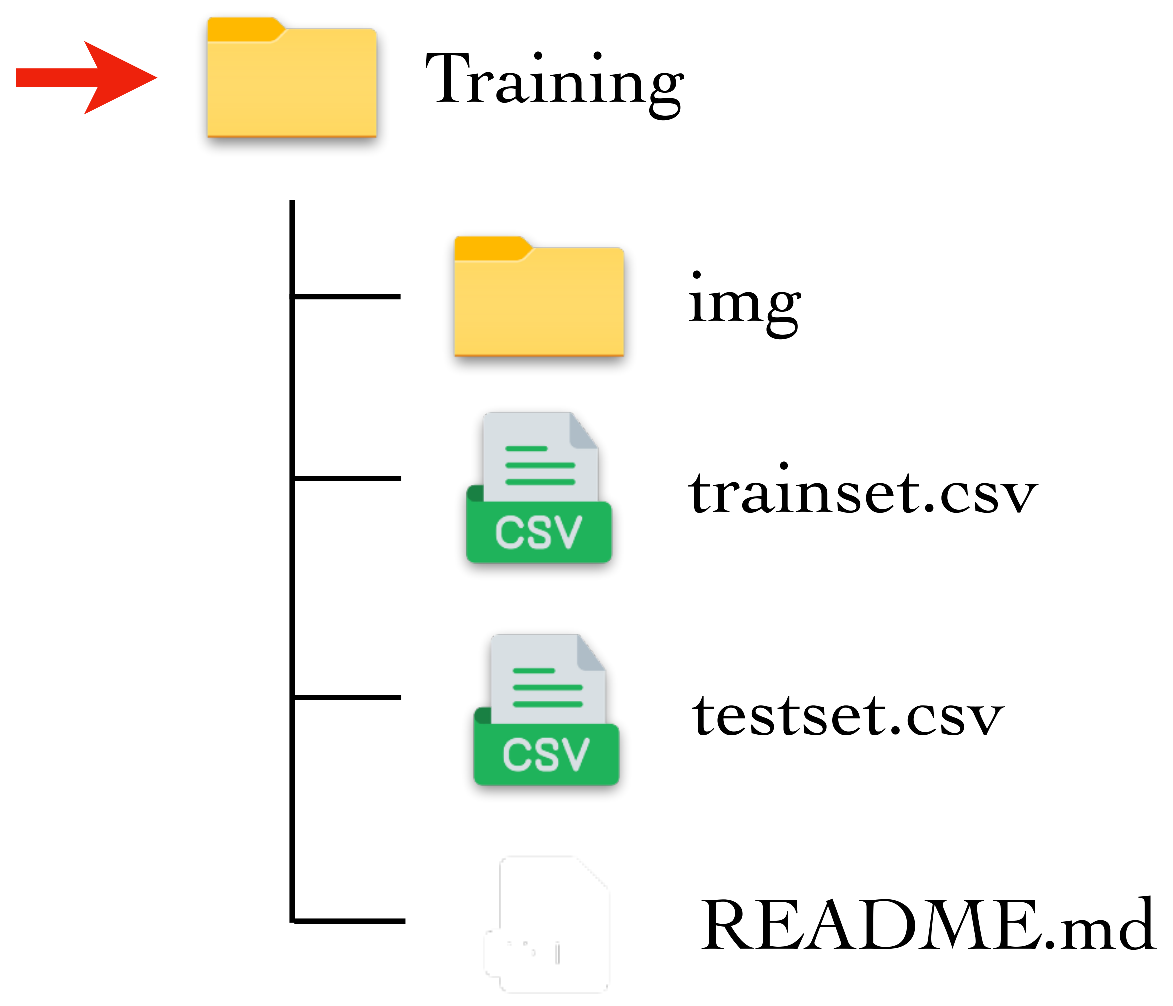
Shell

```
Training/$ touch README.md
```

Output

```
Training/$
```

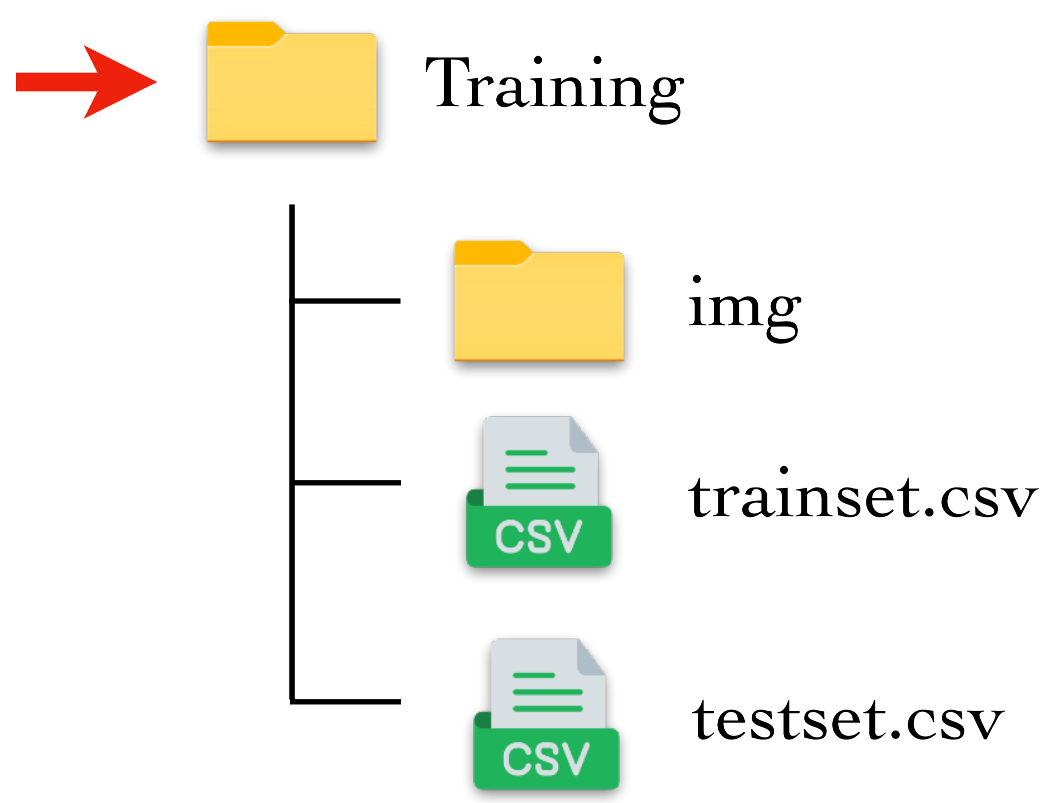
Tree



File Operations

Creating a directory (1/2)

Tree



Shell

```
Training/$ mkdir models
```

Output

```
Training/$
```

File Operations

Creating a file (2/2)

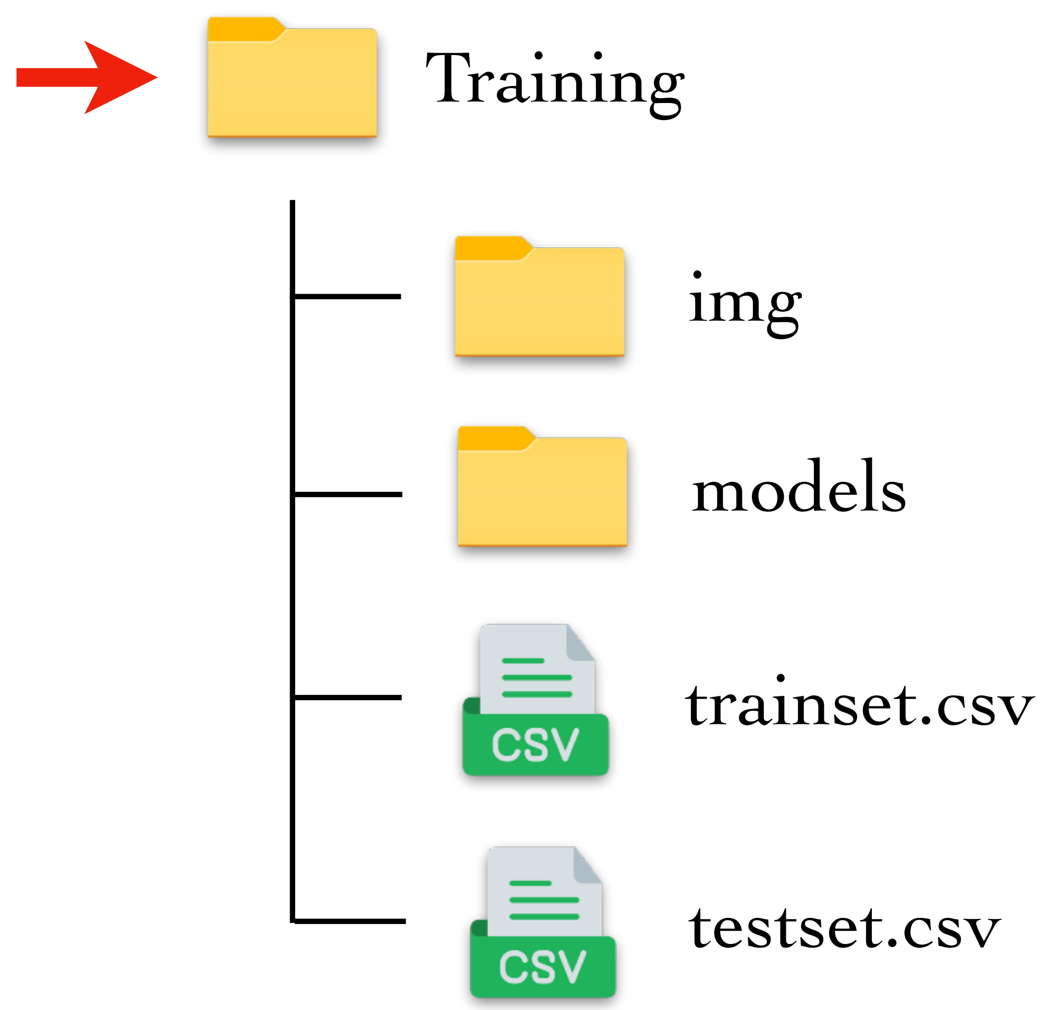
Shell

```
Training/$ mkdir models
```

Output

```
Training/$
```

Tree



File Operations
Copying a file/directory (1/2)

Copying a file

```
$ cp <file-path> <destination>
```

<destination>

<destination-file> : new name

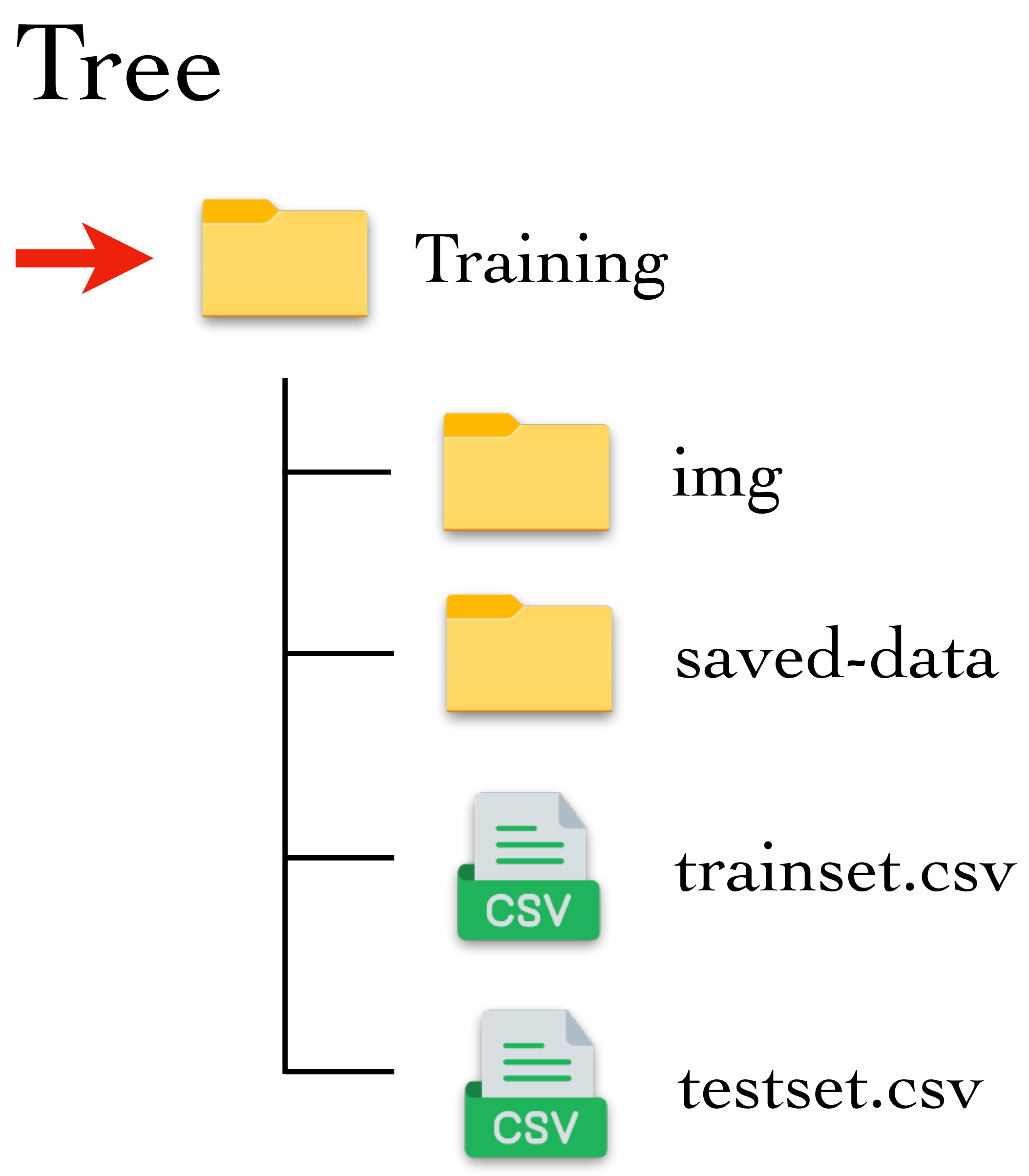
<destination-directory> : same name

Copying a directory

```
$  cp -r <directory-path> <destination-directory>
```

File Operations

Copying a file/directory (2/2)



Copying a file

```
Training/$  cp trainset.csv trainset_copy.csv
```

```
Training/$  cp trainset.csv saved-data/
```

Copying a directory

```
Training/$  cp -r  img/ img-copy/
```

File Operations

Moving a file/directory

Moving a file to another directory

```
$ mv <file-path> <destination-directory>
```

Moving a directory to another directory

```
$  mv <directory-path> <destination-directory>
```

File Operations

Renaming a file/directory

Renaming a file

```
$ mv <file-name> <new-file-name>
```

Renaming a directory

```
$  mv <directory-name> <new-directory-name>
```

File Operations

Removing a file/directory

Removing a file

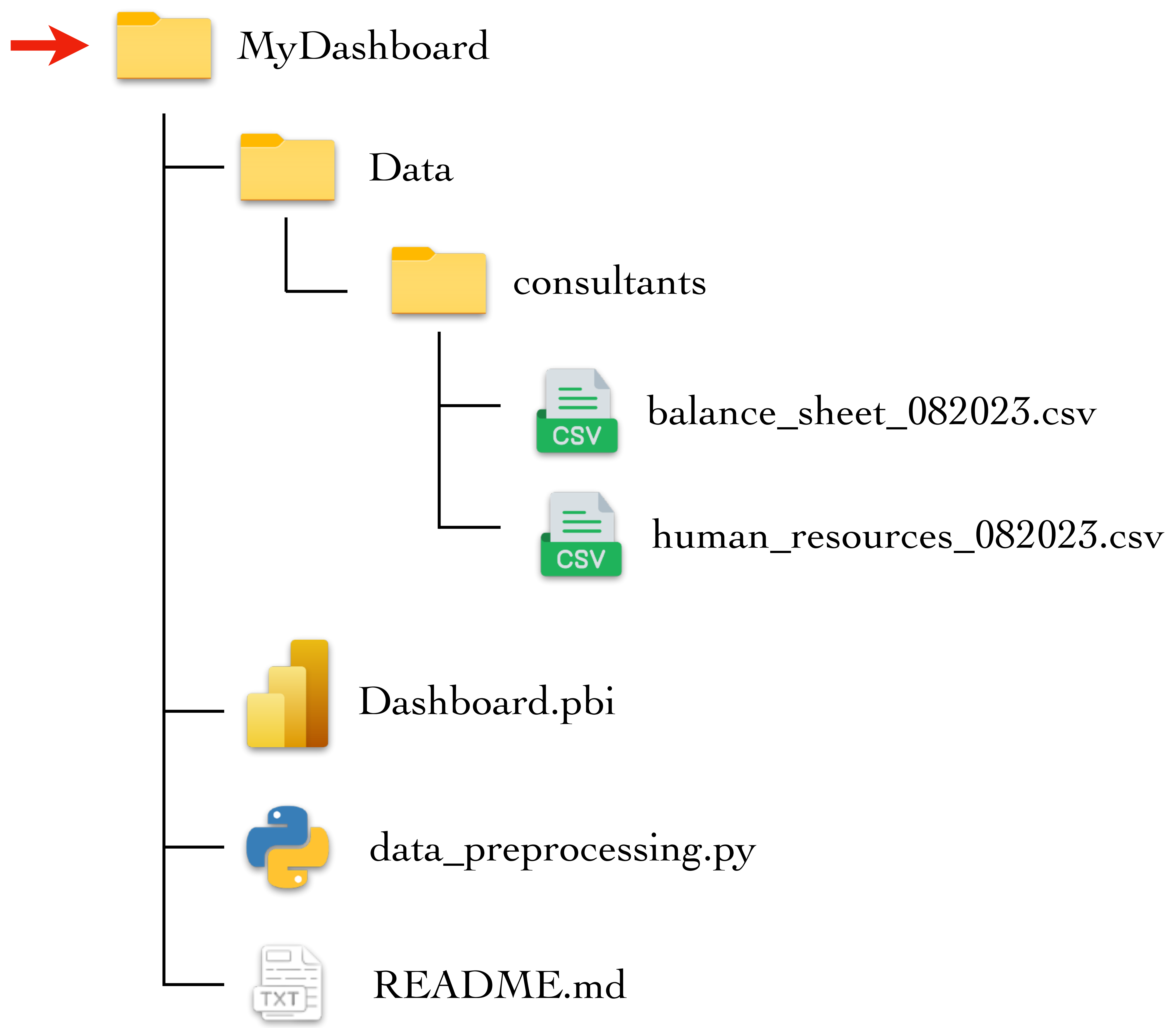
```
$ rm <file-path>
```

Removing a directory

```
$  rm -r <directory-path>
```

File Operations

Question 1



How to remove all CSV files?

A rm files.csv

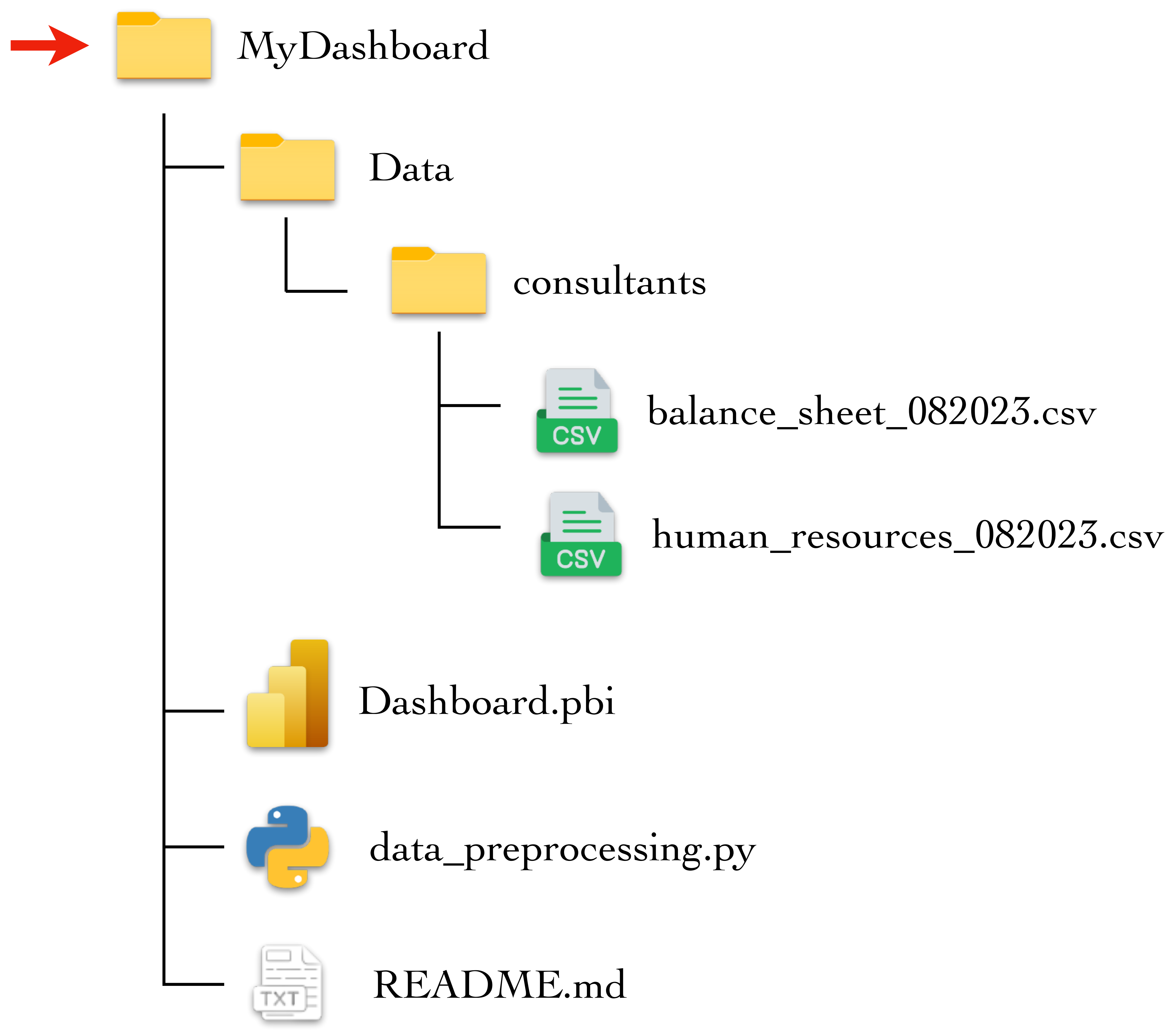
B rm Data/consultant/*.csv

C rm -r Data/consultant/*.csv

D rm *.csv

File Operations

Question 1

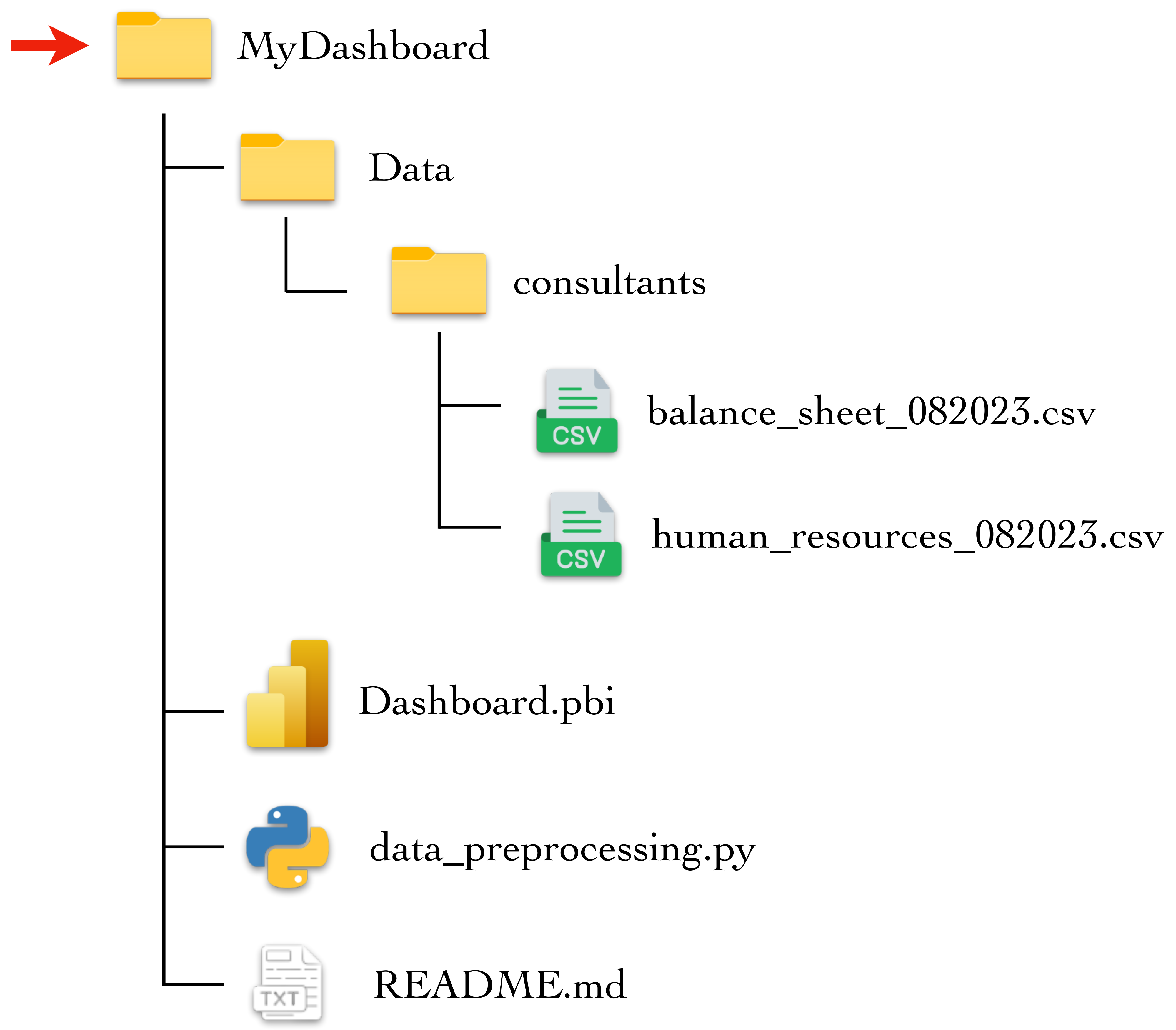


How to remove all CSV files?

- A rm files.csv
- [✓ B rm Data/consultant/*.csv]
- C rm -r Data/consultant/*.csv
- D rm *.csv

File Operations

Question 2



Create img as a subfolder of Data ?

A mk Data/img

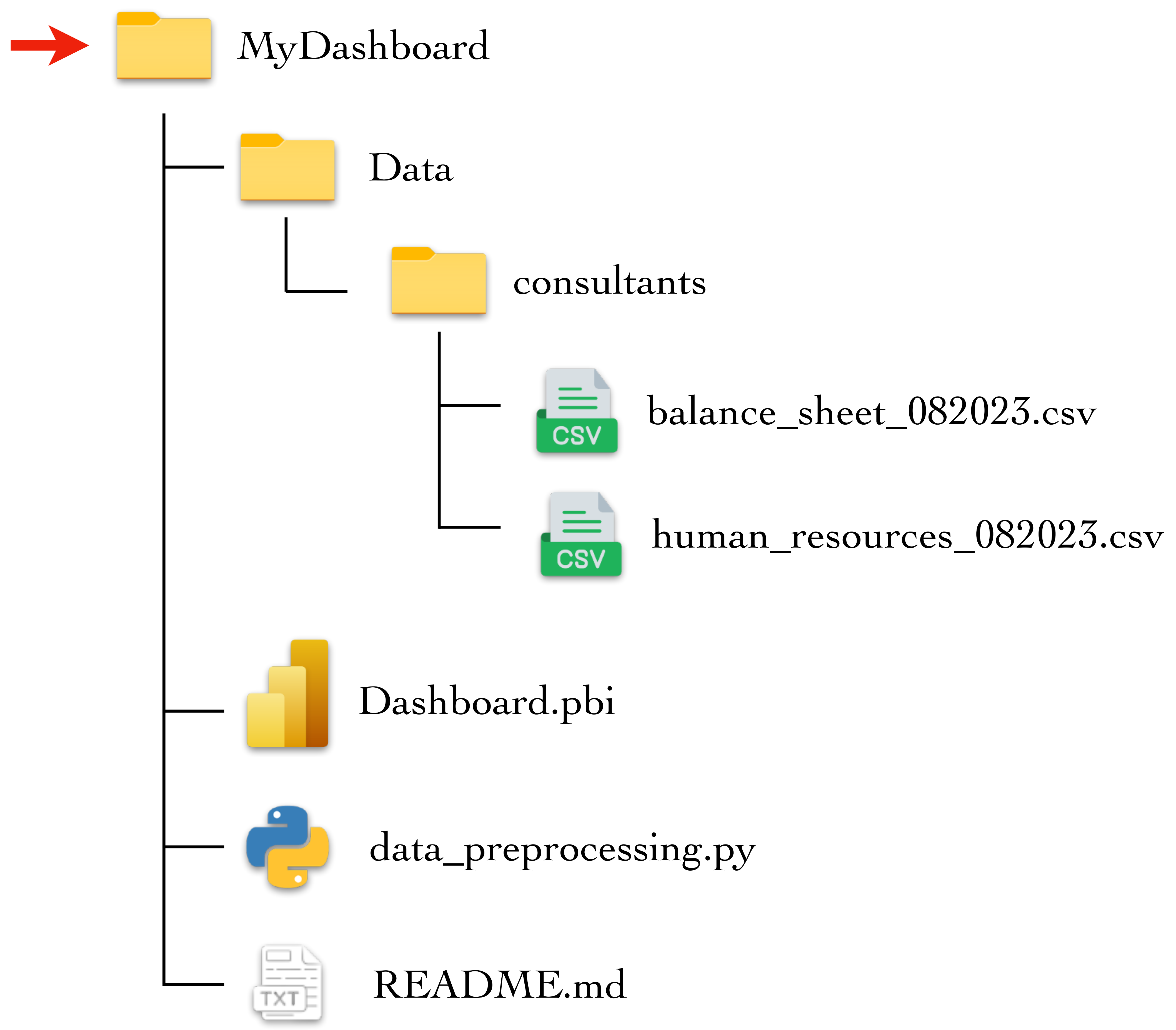
B mk img

C mkdr Data/img

D mkdir Data/img

File Operations

Question 2



Create img as a subfolder of Data ?

A mk Data/img

B mk img

C mkdr Data/img

[✓ D mkdir Data/img]

Text Files
Display

Shell

```
$ cat <file-path>
```

Text Files

Write and Redirect

Write and Redirect (Create/Erase and Add)

```
$ cat file1.txt > file2.txt
```

Write and Redirect (Create or Append)

```
$ cat file1.txt >> file2.txt
```

Text Files

Searching

General command

```
$ grep ”keyword” <file-path>
```

Text Files

Searching Examples

Case-Insensitive Search

```
$ grep -i ”apple” file.txt
```

Whole words only

```
$ grep -w ”apple” file.txt
```

Multiple files

```
$ grep ”apple” *.txt
```

Recursively in directories

```
$ grep -r ”apple” *.txt
```

Text Files

Sort & Uniq

Sorting

```
$ sort <file-path>
```

Keep unique (distinct) adjacent lines

```
$ uniq ”keyword” <file-path>
```