Introduction to Unix

Data Science Lecture Series: Essential Tools

W. Evan Johnson, Ph.D.
Professor, Division of Infectious Disease
Director, Center for Data Science
Rutgers University – New Jersey Medical School

2023-03-13

Section 1

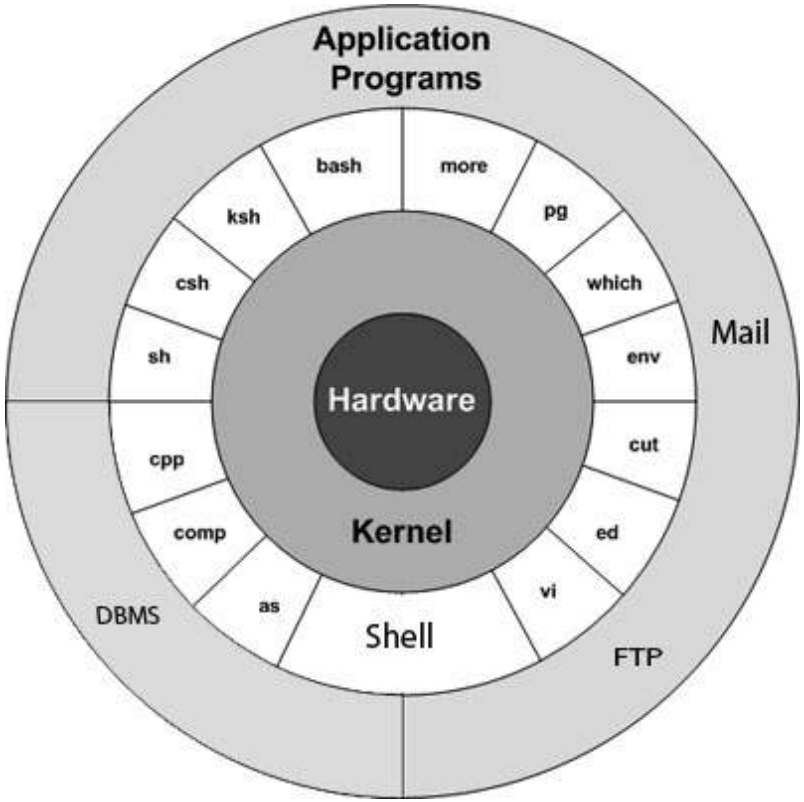
Introduction to Unix

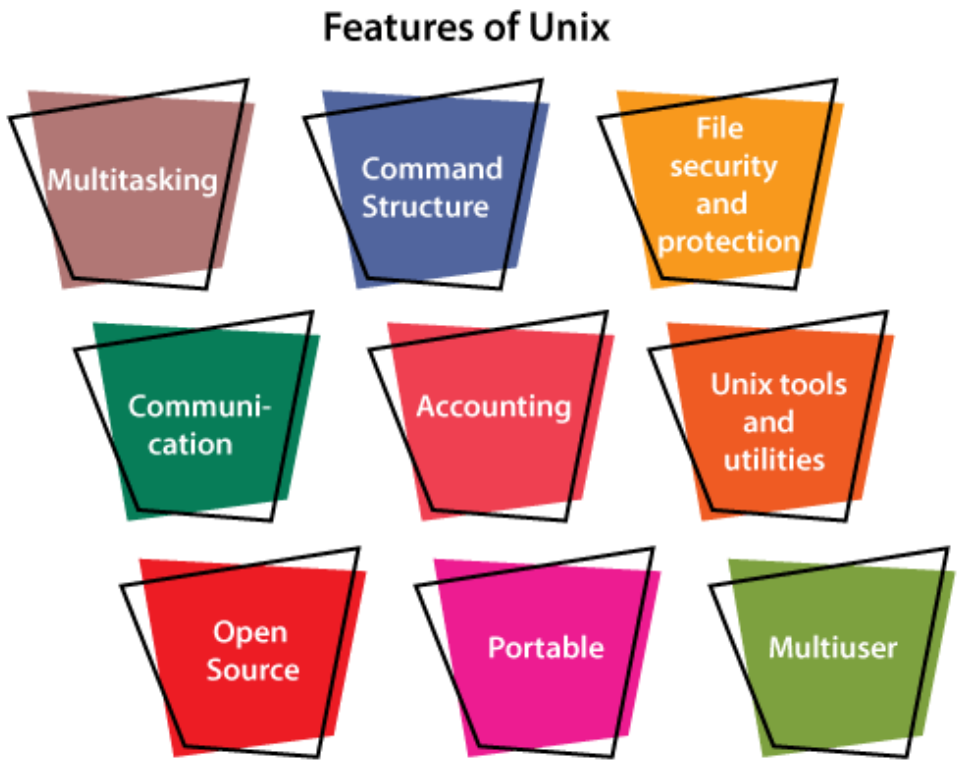
Unix Introduction

Unix is a family of multitasking, multiuser computer operating systems that derive from the original AT&T Unix. Originally developed in 1969 at Bell Labs

Unix Introduction

The Unix operating system consists of many libraries and utilities along with the master control program, the kernel.

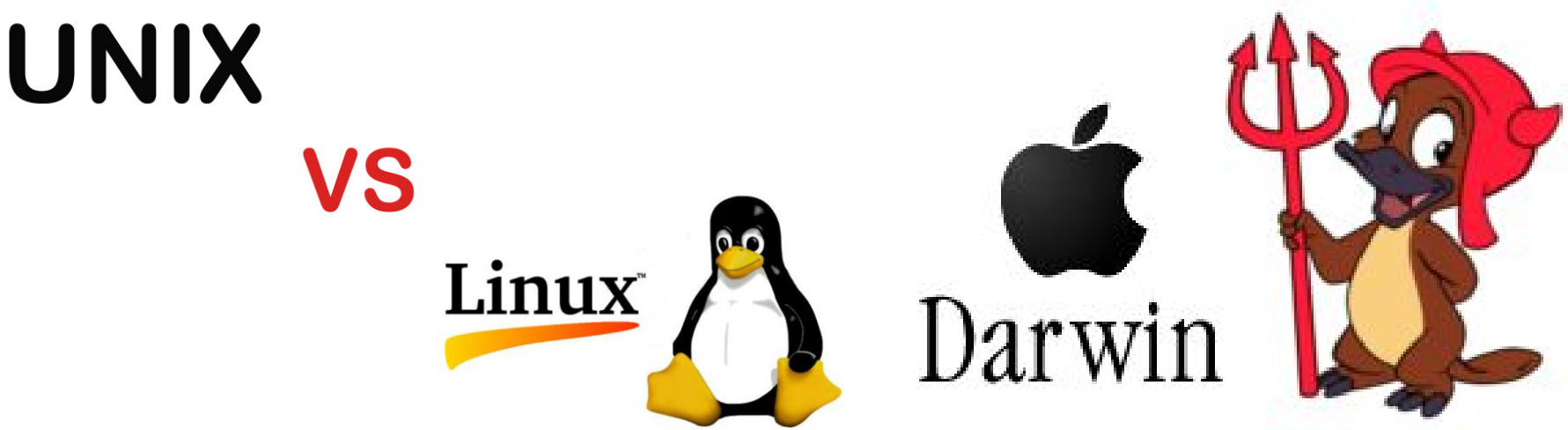




History of operating systems

Most operating systems can be grouped into two lineages:

- Microsoft's Windows NT-based operating systems
- "Unix-like" operating systems (e.g., Linux, Mac OS X, Android, iOS, Chrome OS, Orbis OS used on the PlayStation 4, firmware on your router)



Unix and Data Science

Unix is the operating system of choice in data science.

We will introduce you to the Unix way of thinking using an example: how to keep a data analysis project organized, and later how to do things more efficiently.

We will learn some of the most commonly used commands along the way.

Section 2

Learning Unix

Unix Resources

Here are some resources more information:

- https://www.codecademy.com/learn/learn-the-command-line
- https://www.edx.org/course/introduction-linux-linuxfoundationx-lfs101x-1
- https://www.coursera.org/learn/unix

There are many reference books as well. Here are some particularly clear, succinct, and complete examples:

- https://www.quora.com/Which-are-the-best-Unix-Linux-reference-books
- https://gumroad.com/l/bite-size-linux
- https://jvns.ca/blog/2018/08/05/new-zine--bite-size-command-line/

Naming conventions

Before you start, pick a name convention that you will use to systematically name your files and directories. The Smithsonian Data Management Best Practices has "five precepts of file naming and organization":

- Have a distinctive, human-readable name that gives an indication of the content.
- Follow a consistent pattern that is machine-friendly.
- Organize files into directories (when necessary) that follow a consistent pattern.
- Avoid repetition of semantic elements among file and directory names.
- Have a file extension that matches the file format (no changing extensions!)

The Tidyverse Style Guide[1] is highly recommended!

[1] https://style.tidyverse.org/

Section 3

The Terminal

Basic Unix Commands

Instead of clicking, dragging, and dropping files and folders, we will be typing Unix commands into the terminal. (Similar typing commands into the R console!)

You will need access to a terminal[2]. Once you have a terminal open, you can start typing into the *command line*, usually after a $ or % symbol. Once you hit enter, Unix will try to execute this command. For example:

```
echo "hello world"
```

```
## hello world
```

[2] https://rafalab.github.io/dsbook/accessing-the-terminal-and-installing-git.html

Basic Unix Commands

The command `echo` is similar to `cat` in R. Executing this line should print out `hello world`, then return back to the command line.

Notice that you can't use the mouse to move around in the terminal. You have to use the keyboard. To go back to a previous command, you can use the up arrow.

Section 4

Files and directories

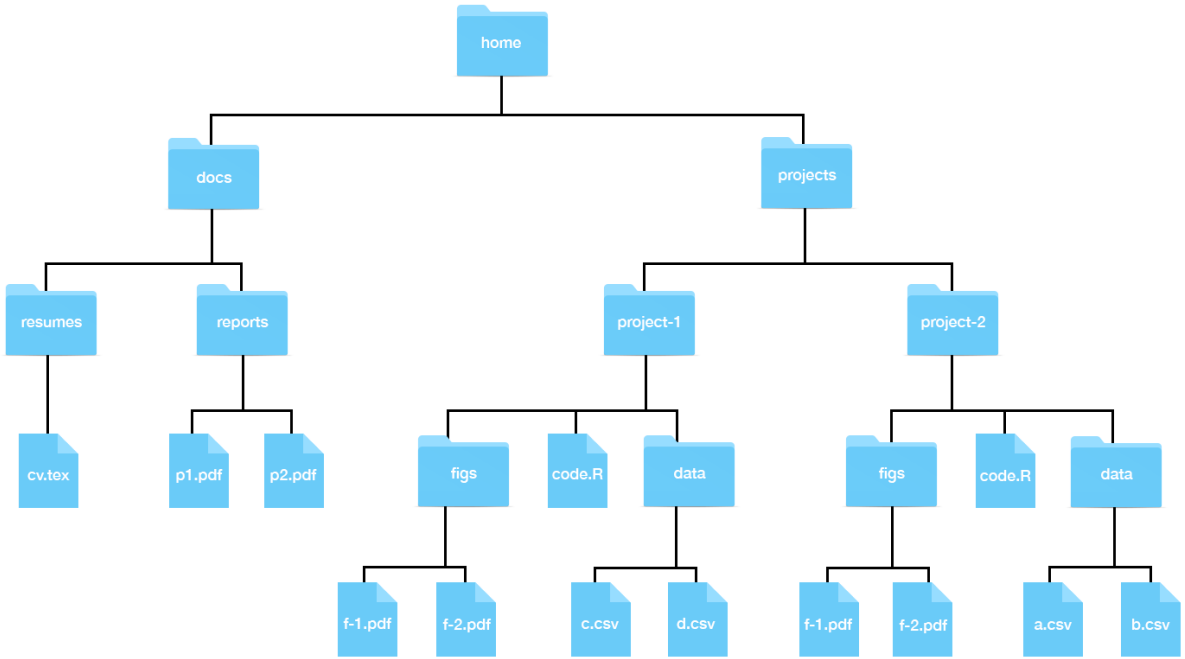
The filesystem

We refer to all the files, folders, and programs on your computer as the **filesystem**. Keep in mind that folders and programs are also files (for later discussion). We will focus on files and folders for now and discuss programs, or **executables**, in a later section.

The first concept top understand is how your filesystem is organized; a series of nested folders, each containing files, folders, and executables.

Directories and subdirectories

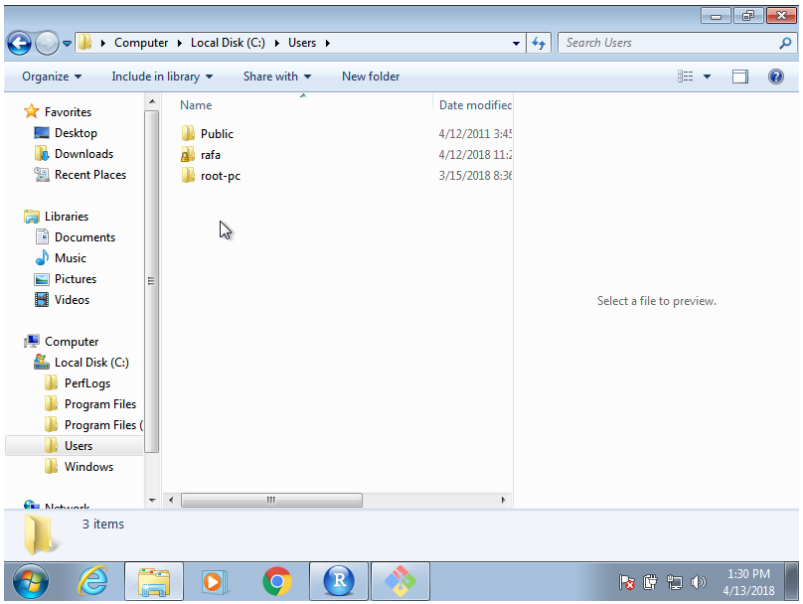
Here is a representation of the structure we are describing:

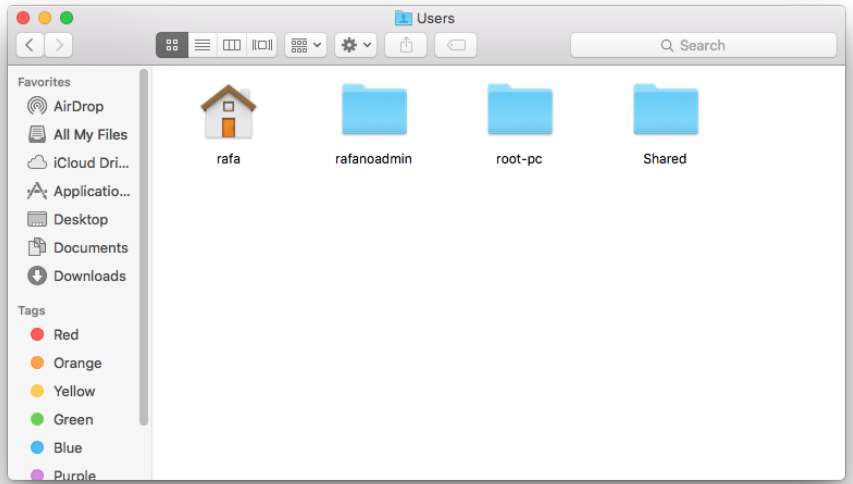


In Unix, we refer to folders as **directories**. Directories inside other directories referred to as **subdirectories**.

The home directory

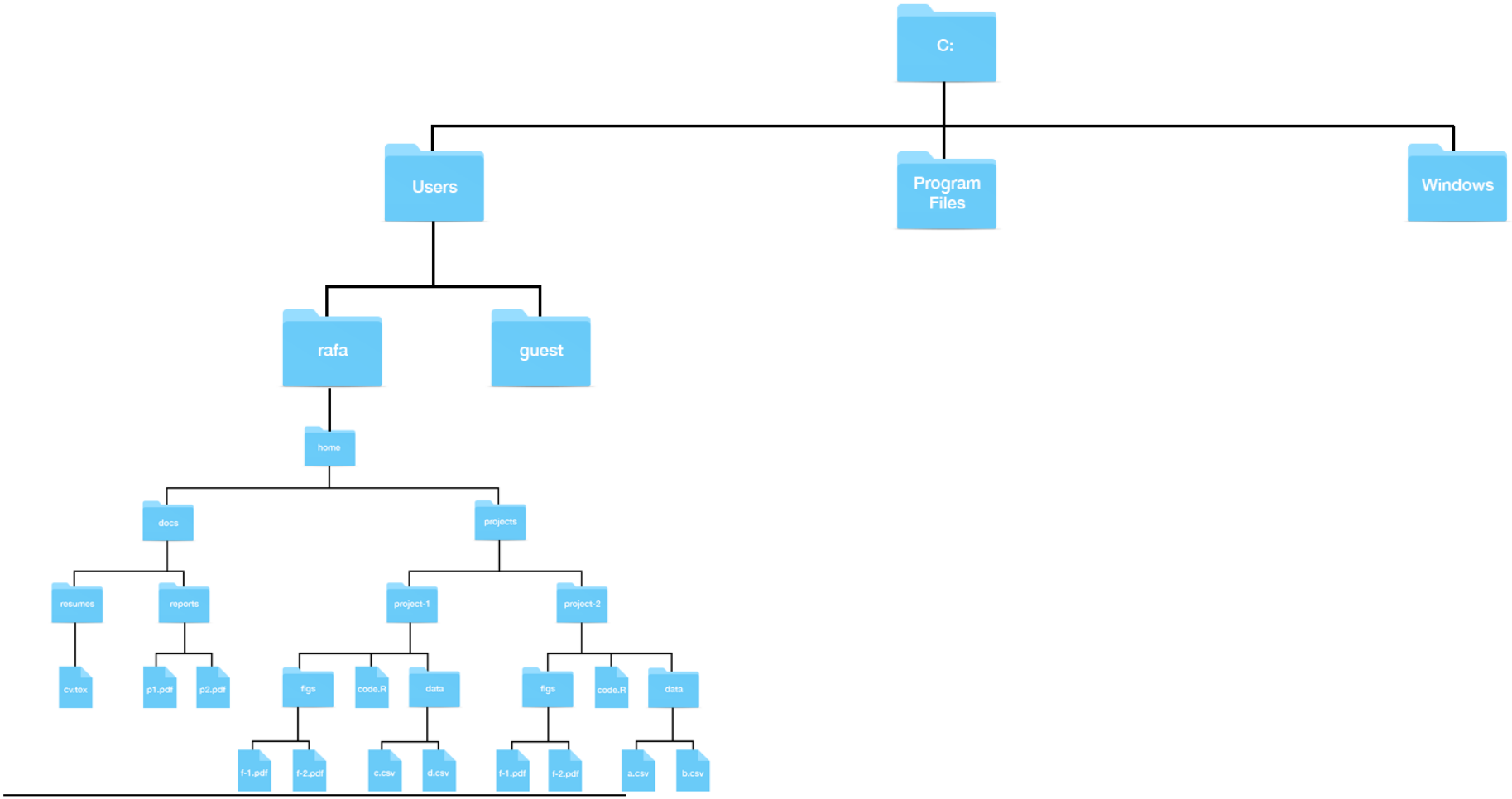
The **home directory** is where your stuff is kept, as opposed to the system files, which are kept elsewhere. The name of your home directory is likely the same as your username. For example:





Windows filesystem structure

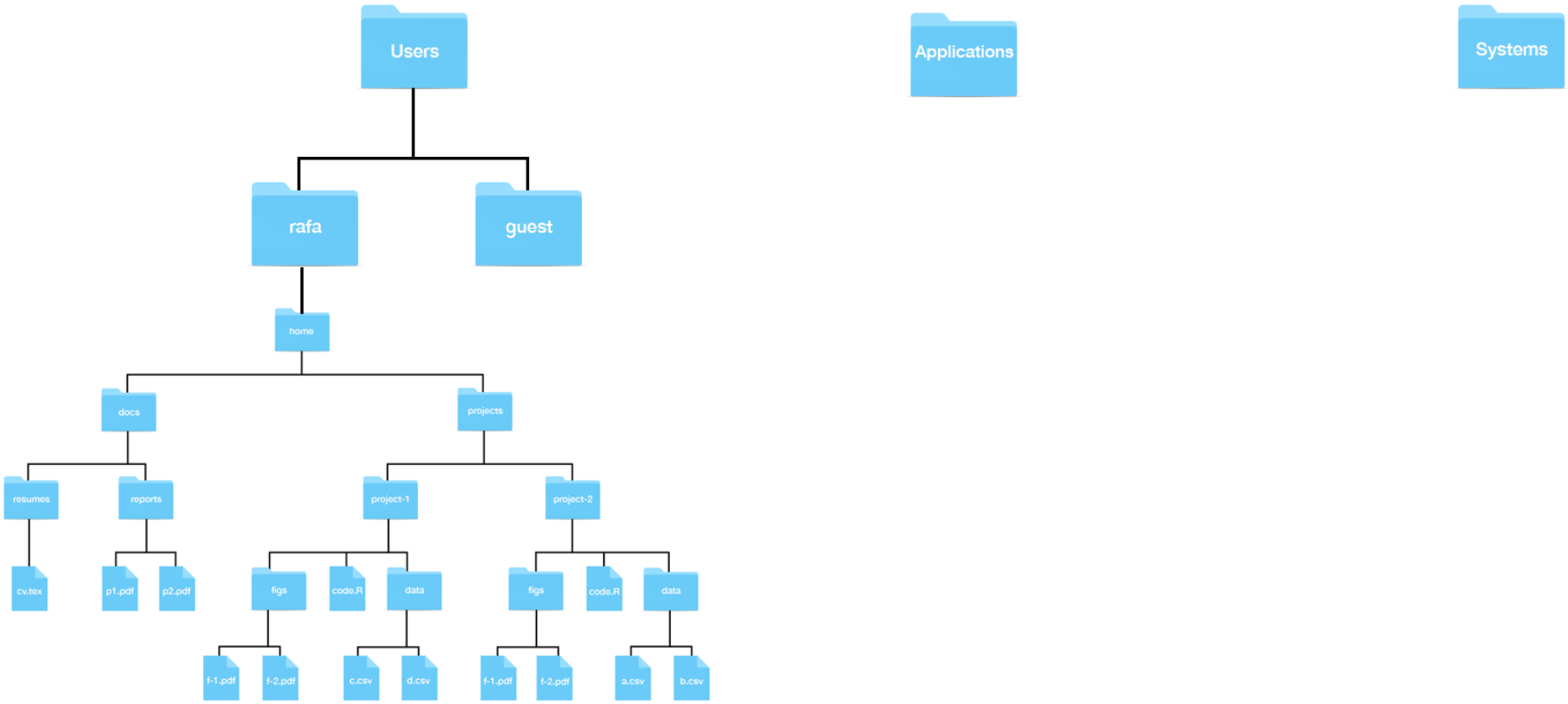
The Windows filesystem structure[3] looks like this:



[3] **Windows Users:** The typical R installation will make your `Documents` directory your home directory in R. This will likely be different from your home directory in Git Bash.

Mac filesystem structure

The Mac filesystem structure looks like this:



Working directory

In Unix, the concept of a current location is indispensable. We refer to this as the **working directory**. Each terminal window you have open has a working directory.

How do we know our working directory? We use the Unix command: `pwd`, which stands for *print working directory*:

```
pwd
```

Paths

The string returned by `pwd` is the **full path** of the working directory. Every directory has a full path. Later, we will learn about **relative paths**, which tell us how to get to a directory from the working directory.

In Unix, the shorthand `~` is a nickname for your home directory. So, for example, if `docs` is a directory in your home directory, the full path can be written `~/docs`.

Section 5

Basic Unix Commands

Some Basic Unix Commands

To see the content of your home directory, open a terminal and type:

```
ls
```

You can also add **options** (‘-’ and ‘-’), **arguments**, and **wild cards** (‘*’) to change function behavior:

```
ls -l
ls -a
ls -t
ls -r
ls -lart
ls -l *.txt
```

Some Basic Unix Commands

To create and remove directories, use the `mkdir` and `rmdir` functions, for example (use `ls` in between to see the directory come and go):

```
mkdir projects
mkdir junk
rmdir projects
```

Note: `rmdir` does not work if the directory is not empty.
Pro tip 1: The up arrow retrieves your previous commands.
Pro tip 2: You can auto-complete by hitting tab.

Some Basic Unix Commands

To change the working directory use the `cd` command, which stands for *change directory*. For example:

```
cd projects
```

To check that it worked, use `pwd`. Now try:

```
cd .
cd ..
cd ../..
cd /
cd ~
cd ~/projects
cd ../junk
```

Some Basic Unix Commands

In Unix, we move files from one directory using the `mv` command:

```
mv path-to-file path-to-destination-directory
```

Warning: `mv` will not ask "are you sure?" if your move results in overwriting a file. Some `mv` examples:

```
mkdir ~/junk ~/projects/resumes
touch ~/projects/resumes/cv.tex
mv ~/projects/resumes/cv.tex ../../
mv ~/cv.tex ~/mycv.tex
mv ~/projects/mycv.tex junk/
mv ~/junk/mycv.tex ../projects/resumes/
```

Some Basic Unix Commands

The command `cp` behaves similar to `mv` except instead of moving, we copy the file.

```
cp ~/projects/resumes/mycv.tex ~/mycv.tex
```

So in all the `mv` examples in the prior slide, you can switch `mv` to `cp` and they will copy instead of move. However to copy entire directories, add the recursive (`-r`) option:

```
cp -r ~/projects/resumes ~/junk/
```

Some Basic Unix Commands

In Unix, we remove files (and directories) with the `rm` command.

```
rm filename
```

Warning: Unlike throwing files into the trash or recycle bin, `rm` is permanent. Be careful! Note the following:

```
rm mycv.tex
rm junk/*.tex
rm -r projects junk
```

Some Basic Unix Commands

Unix uses an extreme version of abbreviations, which makes it hard to guess how to call commands. However, Unix includes complete help files or **man pages** (man is short for manual). In most systems, you can type `man` followed by the command name to get help. For example:

```
$ man ls
```

This command is not available in some of compact implementations of Unix (e.g., Git Bash). Alternatively, we can type the command followed by `-help`:

```
ls --help
```

Advanced Unix functions

Most Unix implementations include a large number of powerful tools and utilities. It will take time to become comfortable with Unix, but as you struggle, you will find yourself learning just by looking up solutions on the internet.\vskip .1in

In your Extra Practice, you will cover more advanced Unix functions such as `chmod`, `less`, `vim`, `grep`, `ln`, `tar`, `ssh`, and `pipes`. (and `git`!!!) Stay tuned!

Session info

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Ventura 13.2.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] digest_0.6.31   lifecycle_1.0.3 magrittr_2.0.3  evaluate_0.19
##  [5] rlang_1.0.6     stringi_1.7.8   cli_3.5.0       rstudioapi_0.14
##  [9] vctrs_0.5.2     rmarkdown_2.19  tools_4.2.2     stringr_1.5.0
## [13] glue_1.6.2      xfun_0.36       yaml_2.3.6      fastmap_1.1.0
## [17] compiler_4.2.2  htmltools_0.5.4 knitr_1.41
```