# GCIS-123
# Software Development & Problem Solving

*Unit 1.1: File System Basics*

RIT | Golisano College of Computing and Information Sciences

# Windows 10 Computer Literacy

If you'd like to use a non-Windows computer for your out of class work, you will need to do some extra learning on your own.

Many of the commands you will use work much the same in Linux or Mac OS. Others you may easily Google.

- The goal of this week's lectures is to establish a common, minimum level of computer literacy among the students in the class.
- We will explore:
  - The Command Line
  - The File System
  - Version Control (with Git)
  - Environment Variables
  - More Advanced Git
- You will learn to perform many tasks using only the command line.
  - After a short time, you will find using the command line to be much faster and more efficient than trying to use a graphical user interface like the File Explorer for many tasks.
- Please note that throughout this series of lectures (and the entire course) we will be using the Windows 10 operating system.
  - If your personal computer has a different operating system, some of the examples will not work.
  - It is therefore recommended that you use one of the lab computers to follow along.

1.1.0

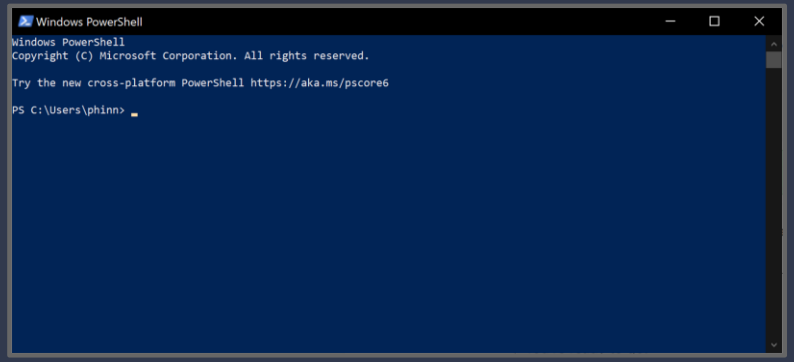
# Logging In

Even if you plan to use your own computer, take a minute to log into the lab computer in front of you to make sure that you can access it if needed.

- Depending on the lab in which your class meets, you may be required to authenticate using an account specific to the lab's domain.
  - If this is the case, you should have received an email with your username and instructions for setting a password.
  - Using your phone or laptop, follow the instructions to reset your password.
  - Once you have completed the process, you should be able to use the username and password to log into the computer in front of you.
- It is also possible that you don't need a special account to log into the lab computers.
  - Ask your instructor!

# The Command Prompt

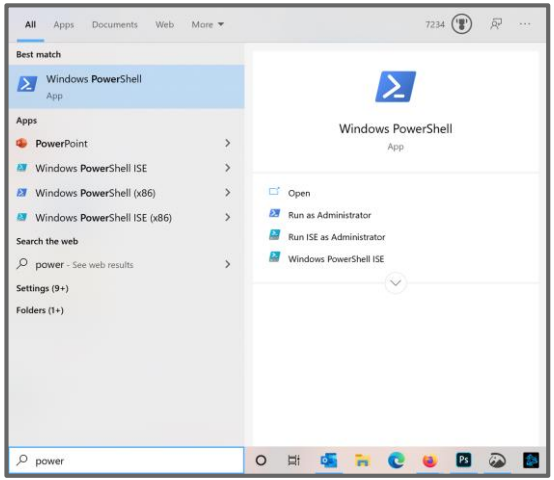
- Every major operating system includes support for a ***command prompt***.
  - The command prompt is often called a “terminal” or “command line interface” (CLI).
  - Some operating systems *only* include a command prompt!
- Many features of the operating system can be quickly and efficiently executed via the command prompt, including:
  - Running programs.
  - Opening files.
  - Creating or editing text files.
  - Copying, moving, or deleting files.
  - Creating or deleting directories.
  - etc.
- Starting the command prompt is easy:
  - Press the **Windows key** on your keyboard to open the **Start Menu**.
  - Type “`powershell`” into the search field.
  - If necessary, use the up and down arrow keys to select **Windows PowerShell** in the search results.
  - Press the enter key.



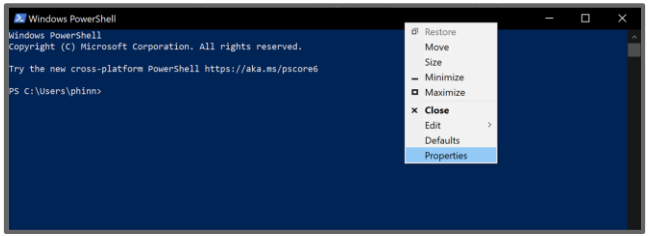
1.1.1

# Customizing the Command Prompt

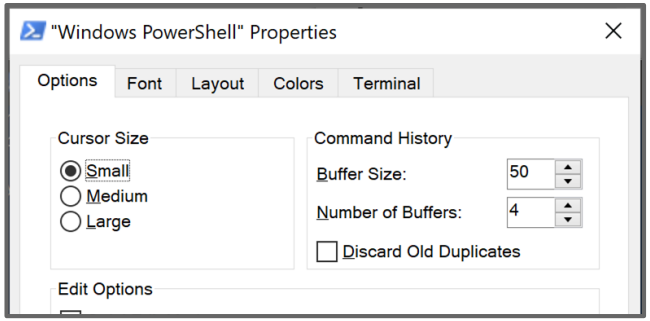
Start by customizing it to your preferred look and feel.

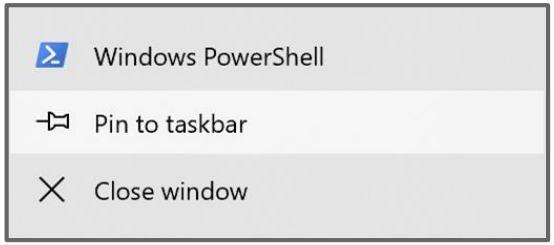



If you haven't already done so, use the Windows key to open the start menu and run the command prompt.




Right-click the window's title bar and select the ***Properties*** menu option. Play around with customizing font color & size, layout, and so on.



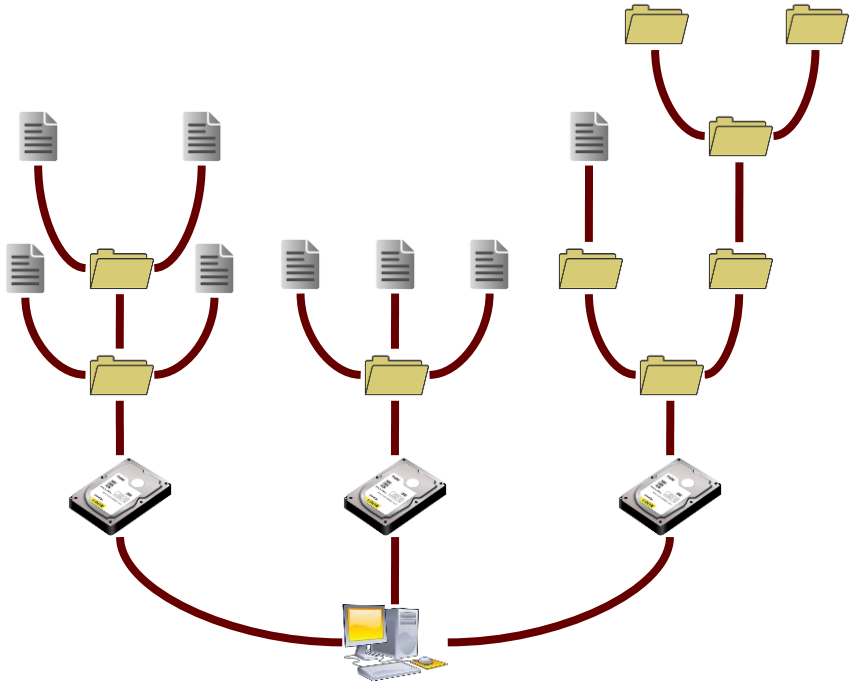




You'll be using the command prompt *a lot*, so go ahead and pin it to your taskbar (right click the icon and select ***Pin to taskbar***).

Close and re-open the command prompt using the button on the taskbar to make sure that your settings were saved.

# The File System

- The **File System** on your computer is organized into a **tree structure**. Your PC is at the **root** of the tree.
- Your PC contains one or more **drives**.
  - Most contemporary computers use some combination of internal **solid state drives** (SSDs) and **removable drives** (e.g. USB or SATA).
- Each drive has a root **directory**.
  - Directories are often also called **folders**.
- While a directory may be empty, most contain some combination of **subdirectories** and **files**.
- Each subdirectory is itself a directory, and so may also contain its own files and subdirectories.
- In fact, directories may be nested to an arbitrary depth.



1.1.2

# Listing Drives

Executing the `gdr` command from the command line on your computer will list all of the available drives.

```
PS C:\Users\charlie> gdr
Name        Used (GB)   Free (GB) Provider      Root
----        ---------   --------- --------      ----
Alias                                 Alias
C           312.36      640.88    FileSystem    C:\
D           2746.24     48.28     FileSystem    D:\
E           0.28        0.06      FileSystem    E:\
F           464.75      11.40     FileSystem    F:\
Z           784.97      168.90    FileSystem    Z:\

PS C:\Users\charlie> _
```

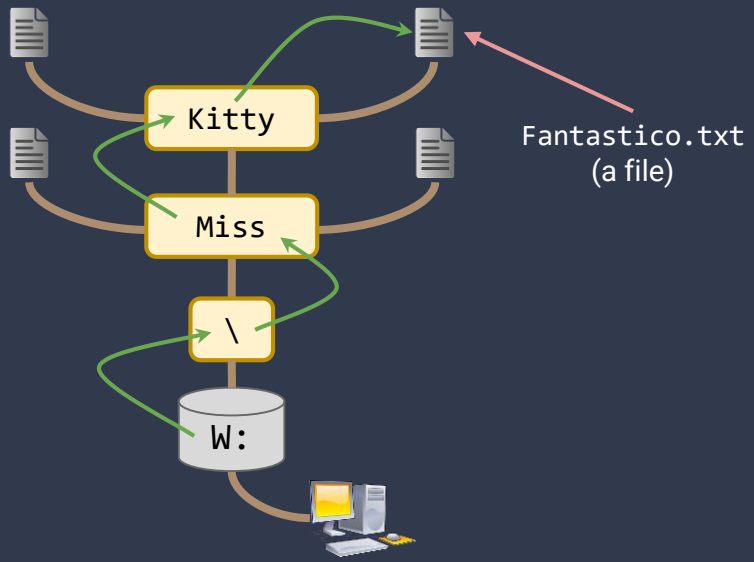
The **drive letters** will be listed on the left (along with some other *stuff*)...

And the **root directory** on each drive is shown on the right.

The used and available storage on each drive is shown (in gigabytes).

# File Paths

Consider the file named "`Fantastico.txt`" depicted below. It is in a directory named "`Kitty`", which is in a directory named "`Miss`" in the root directory on drive "`W`".




absolute path to the file Fantastico.txt

```
W:\Miss\Kitty\Fantastico.txt
```

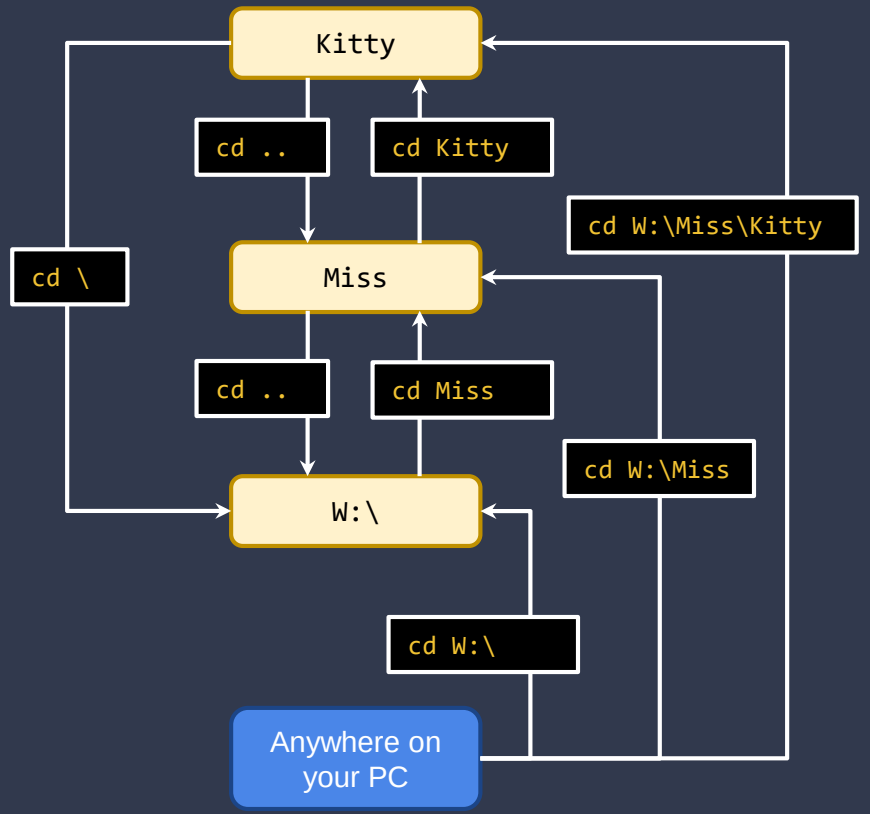
- Every drive in your PC is identified by a **drive letter**, e.g. `C`, `D`, or `Z`.
  - A removable drive (e.g. a USB flash drive) will be automatically assigned an unused letter.
- You may switch between drives on your computer

  e.g. `C:`
  - Case does not matter.
- Every file or directory in your file system is uniquely identified by its **absolute path**.
  - The path begins with the letter of the drive on which the file resides.
  - It includes the name of each directory and subdirectory.
  - It ends with the name of the file.
  - The names of directories and files are separated by a file separator, e.g. `'\'`.

# Navigating the File System

- You may also move from one drive to another by typing the drive letter followed by a colon, e.g. `C:`, `W:`, etc.
  - This will move your command prompt into the last directory you used on the drive, or the root directory if you have not used the drive yet.
- Once on a drive, you may change from one directory to another using the `cd` command.
  - When used with the name of a subdirectory, cd will move into that directory, e.g. `cd Kitty`
  - `cd \` will move to the root directory on the current drive.
  - `cd ..` will move *up* one directory (e.g. from a subdirectory into its parent directory).

Consider the directory structure from the previous example: `W:\Miss\Kitty\`



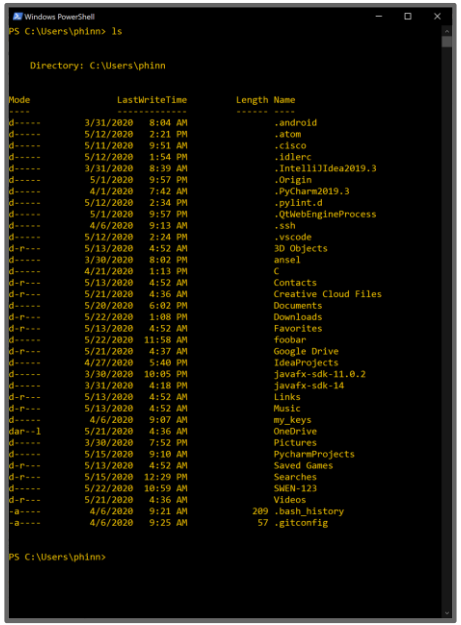
# Listing Files

- Once you are in a directory, you can use the `ls` command to list the files in the directory.
  - "`ls`" is short for "list."
- The listing provides a lot of detailed information about the contents of the directory.
  - The **mode** indicates details about the attributes of each file.
    - `d` indicates that it is a directory.
    - `a` indicates that the file has been archived (backed up) since the last update.
    - `r` indicates that the file is read only.
  - The **last write time** is the last time that the file was updated.
  - The **length** is the number of bytes of data in the file.
  - The **name** is, well, the name.

```
PS C:\users\charlie\Downloads> ls
  Directory: C:\users\charlie\Downloads
Mode          LastWriteTime     Length  Name
----          -------------     ------  ----
d-----   4/2/2020   8:33 PM             nightmare_hero
d-----  5/12/2020   8:36 AM             hello-world
--r---  5/22/2020  10:57 AM     352609  The File System.pdf
-a----  5/22/2020   9:20 AM     241580  More Git.pdf
-ar---  5/16/2020   9:26 AM    1353127  batman-logo.png
-a----  5/12/2020   1:57 PM  198109216  AtomSetup-x64.exe
-a----  5/22/2020   1:08 PM   46891904  Git-2.26.2-64-bit.exe
-a----  5/12/2020   1:54 PM         24  hello.py
-a----  5/12/2020   2:34 PM        204  hw02.py
-a----  5/20/2020  12:25 PM       2138  set.txt


PS C:\users\charlie\Downloads> _
```

1.1.3

# Changing Directories



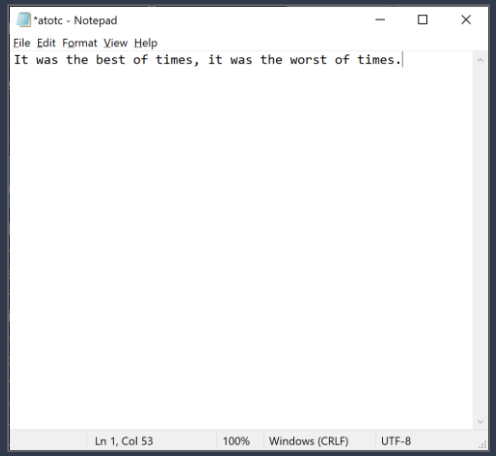

Practice using the `cd` command to change drives or directories, and the `ls` command to list files in a directory.

- Navigate each of the directories below and list the files inside of each.

***Hint***: you can type the first few letters in the name of the directory and press the TAB key to autocomplete, e.g. `cd Doc` → `cd .\Documents\`

  - Your user directory, e.g. `C:\Users\George`
  - `Documents` (in your user directory)
  - `Downloads` (in your user directory)
  - `Pictures` (in your user directory)
  - `C:\Program Files`
  - `C:\Program Files (x86)`

# File Types




Notepad is the default text editor for the Windows operating system and it can be used to create and edit text files.

You can run Notepad from the command prompt followed by the name of the file that you want to create or edit.

```
C:\users\charlie> notepad atotc.txt
```

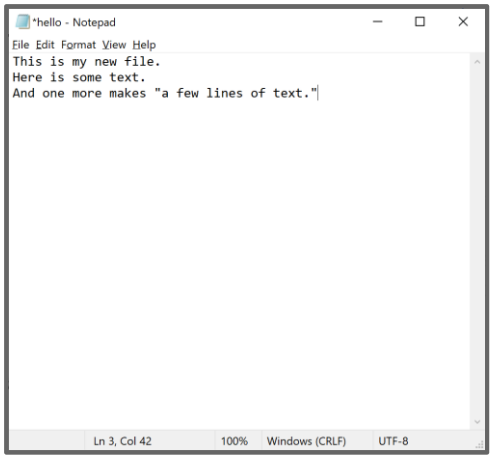
- Files are used to store data such as text, images, video, or executable applications.
- The file type is usually indicated using a file extension - the last part of the filename after a dot (`.`).
  - For legacy purposes, most file extensions are three letters.
- Some examples of file extensions include:

| Extension | File Type |
|---|---|
| `txt` | A text file, containing only characters |
| `pdf` | Portable Document Format |
| `png, gif, jpg` | Image files |
| `zip` | A ZIP compressed archive |
| `py` | A Python program (or script) |
| `html, htm` | A Hypertext Markup Language file |

- Files highlighted in yellow use different/special file extensions, but are all ***text files***.

1.1.4

# Creating Text Files




Lets practice by using Notepad to create, save, and then reopen a text file.

- Launch a new command prompt.
- Navigate to the `Documents` directory.
- Use `notepad` to create a file named "`hello.txt`".
  - When prompted, choose to create the file.
- Add a few lines of text to your file, save it (shortcut: ***CTRL-S***), and exit (shortcut ***ALT-F*** to open the File menu and ***X*** to exit) Notepad.
- Use `ls` to verify that your file has been created.
- Use `notepad` again to open your file and see the text.
- Close the command prompt.

# Copying Files

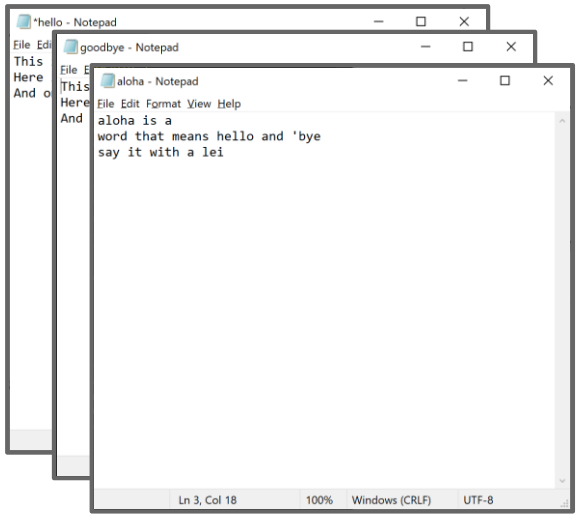
- The `cp` command can be used to create a copy of a file. It requires at least one argument: source.
  - For example, `cp C:\story.txt` will copy the file with the specified path into the current directory; the copy will be named `story.txt`.
- An optional second argument can be used to specify the name (or path) of the copy.
  - For example `cp story.txt tale.txt` will create a copy of the file `story.txt` in the current directory; the copy will be named `tale.txt`.
- A directory cannot contain two files with exactly the same name; trying to copy a file into a directory that already has a file with that name will cause an error.

```
PS C:\Users\charlie> cp atotc.txt dickens.txt
PS C:\Users\charlie> ls

            Directory: C:\users\charlie

Mode          LastWriteTime  Length  Name
----          -------------  ------  ----
-a----  5/23/2020  10:54 AM      52  atotc.txt
-a----  5/23/2020  10:54 AM      52  dickens.txt

C:\Users\charlie> cp atotc.txt
cp : Cannot overwrite the item
C:\users\charlie\atotc.txt with itself.

PS C:\Users\charlie> cp C:\afile.txt here.txt
PS C:\Users\charlie> cp here.txt Documents\there.txt
```

If everything goes well, the `cp` command is silent (it will not produce any output). Use `ls` to see your copy.

1.1.5

# Copying Files



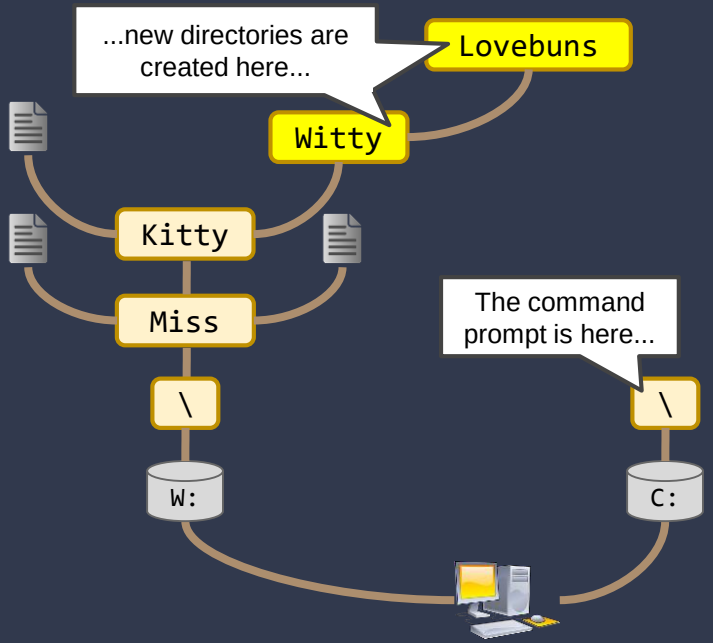

Practice using the `cp` command to make two copies of the same file.

- Launch a new command prompt.
- Navigate to the `Documents` directory.
- Use `cp` to create two copies of your "`hello.txt`" file.
  - Name them whatever you'd like.
- Use Notepad to edit and change the contents of the files.
  - Don't forget to save (***CTRL-S***)!
- When you are finished, close the command prompt.

# Making Directories

The `mkdir` command allows you to create new directories anywhere in the file system.

```
C:\> mkdir W:\Miss\Kitty\Witty\Lovebuns
```




- The `mkdir` command can be used to make a new directory with a specified name in the current directory.
  - For example, assuming that you are in the directory with the path `C:\Users\charlie`, then the command `mkdir Weasley` will make a subdirectory named "`Weasley`".
  - The full path to the new directory would be `C:\Users\charlie\Weasley`
- You may also create the same directory from anywhere in the file system using an absolute path.
  - `mkdir C:\Users\charlie\Weasley`

# Moving Files

- The `mv` command will let you do exactly that.
- Like `cp`, `mv` can be used with at least one argument: the path to the file to move.
  - For example, `mv C:\story.txt` will move the file named `story.txt` from `C:\` to the current directory.
- An optional second argument can be used to specify the name (or path) of the destination.
  - For example `mv C:\story.txt W:\tale.txt` will move the file `story.txt` from `C:\` to `W:\` and rename it to `tale.txt` at the same time.
  - The `mv` command can also be used to rename files in place, e.g. `mv old.txt new.txt` will change the name of the file "`old.txt`" to "`new.txt`" in the current directory.

A file that is moved is not necessarily ***physically*** moved from one location to another on the storage media.

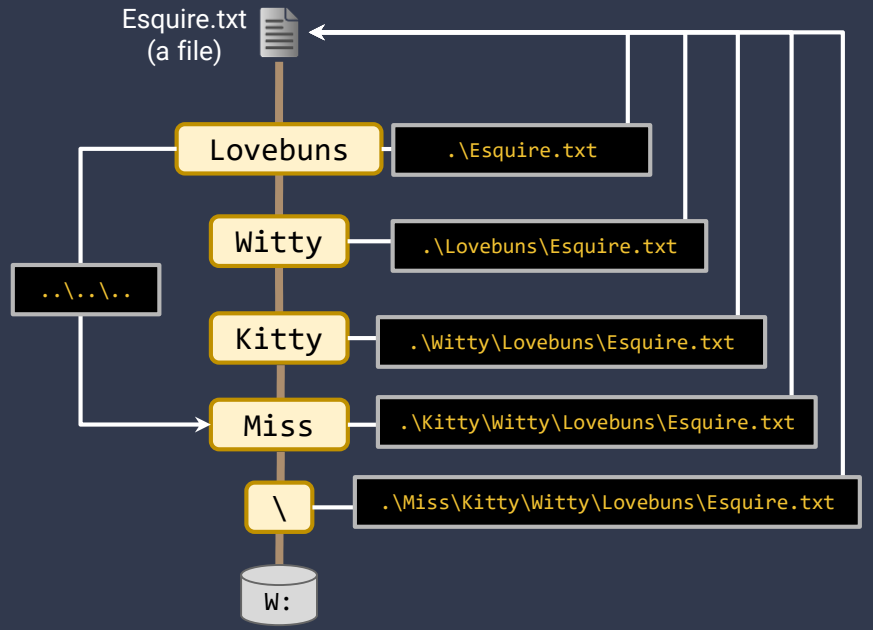
It is more often the case that the alias used to refer to the file's location is changed from one name to another.

Conversely, your operating system may physically move a file from one place to another ***without*** changing its name.

# Relative Paths

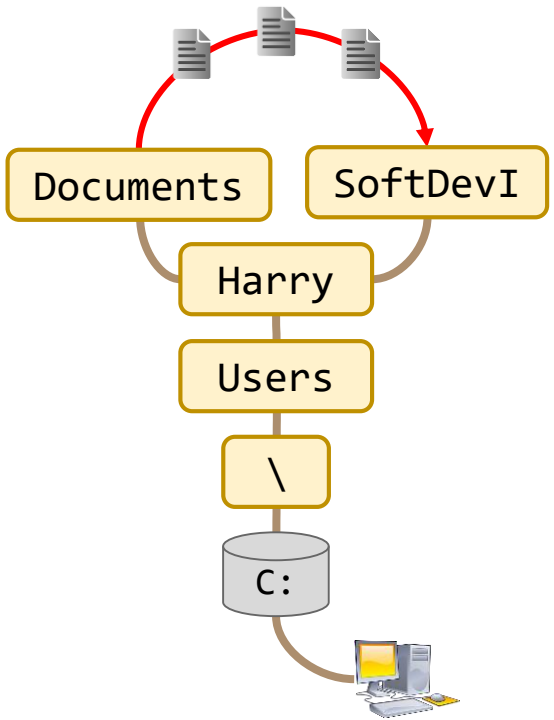
- So far we have referred to files either by name or by using the absolute path to the file.
- Files may also be referenced using a **relative path**, that is a path ***relative*** to the current directory; it specifies how to get ***there*** from ***here***.
- For example, consider the file named `Esquire.txt` that is depicted to the right.
  - The **relative path** to the file from the `Kitty` directory is: `.\Witty\Lovebuns\Esquire.txt`.
  - Note that the dot (`.`) is a shortcut to refer to the current directory.
- A relative path may include both `.` and `..` (to refer to a parent directory), e.g. the path to `Miss` from `Witty` would be `..\..` (the parent of its parent).

Consider the file depicted below with the absolute path `W:\Miss\Kitty\Witty\Lovebuns\Esquire.txt`
What is the **relative path** from each directory?




Remember: `..` can be used to create paths that move in the opposite direction.

1.1.6




# Making Directories and Moving Files

Practice using `mkdir` to make directories and `mv` to move files between directories.

- Launch a new command prompt.
- Create a new directory structure under your user directory with the relative path `SoftDev1\Unit01\Day01`
  - e.g. `C:\Users\Harry\SoftDevI\Unit01\Day01`
- Move all 3 of the text files that you created previously into the new directory.
  - e.g. `mv .\Documents\hello.txt .\SoftDevI\Unit01\Day01`

# Wildcards & Deleting Files

- A **wildcard** (*) can be used to find multiple files with names that match a certain pattern, e.g.:
  - `*.txt` matches all files with a `.txt` extension.
  - `hel*` matches all files with names that begin with "`hel`" such as "`hello.txt`" or "`help.png`".
  - `*if*` matches all files with "`if`" anywhere in the name such as "`if_only.txt`", "`riff.jpg`", or "`tears.gif`".
  - And so on.
- Wildcards can be used in combination with commands such as `ls`.
  - e.g. `ls *.txt` will list all of the text files in the current directory.

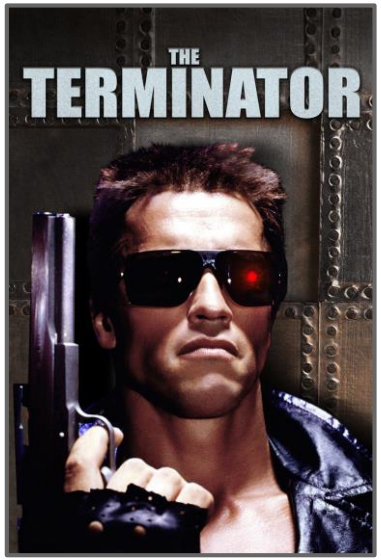
- The `rm` command can be used to delete a file by name, e.g. `rm hello.txt`
- Wildcards can be used with `rm`, and so `rm *.txt` will delete *all* of the files with a `.txt` extension in the current directory.

You may be used to being able to recover deleted files, but files deleted from the command line line ***are not*** moved to the Recycle Bin!

That means that you should use caution when using the `rm` command to delete anything, especially when using a wildcard (*)!

1.1.7

# Deleting Files




Remember! Deleting files from the command prompt deletes them permanently! Be careful!

- Launch a new command prompt.
  - If necessary, navigate to your user directory.
- Delete two of the text files in your `Day01` folder.
  - Challenge: can you use a *relative path* and a *wildcard* (*) to delete both files from your user directory without deleting the third?
- List the files in the directory to verify that your files have been deleted.