MIDDLESEX Community College

Tools and Technologies for Tech Writers 2024

Tips and Tricks

Notices

This document was prepared as a handout for the Middlesex Community College Tools and Technologies for Technical Writers class, Winter semester 2024.

Prepared by Zoë Lawson, course instructor.

Contents

Introduction

Over twenty years of tech writing, I've learned a thing or two that doesn't always fit into nice, descrete categories.

This document is a random collection of bits and bobs I've picked up over the years that might be useful.

Computing tidbits

I've never taken a computer science course, but I've learned a bit about computers over the years. They are great tools for doing your job. There are lots of things I've picked up over the years that might help you out.

For all that computers try to be 'easy to use' and 'not need documentation', there's a lot to learn about them. I'm not sure anyone teaches how computers work any more, without a computer science course.

As a tech writer, you're going to have to use a computer every day to do your job. You can't be afraid of it, or dislike using it. (Although we all have days where computers are terrible.) It's well worth understanding how they work so you can make them work for you all the better.

This isn't a computer science course, but I want to share some of the random things I've learned that have helped me do my job.

File Explorer configuration

Default settings for Windows File Explorer are terrible. I have customizations I make every single time I set up a system.

Both Windows and Mac systems are trying to be "helpful" and make it "easy" for you to use a computer. They're trying to make computers "less scary".

Unfortunately, sometimes this gets in the way of using them.

1. Open Windows File Explorer.
2. Open the **View** ribbon.

   If the View option isn't visible, press the **Alt** key and the menus should appear.
3. Select **Navigation Pane** and confirm **Navigation Pane** is selected.

   This should make sure you always get the tree view. When I'm moving between folders, this really helps me.
4. As long as I'm not in a folder full of images, I select **Details** for the view.
5. Optional: If I'm using a source control tool that keeps locked files read-only, I add the attributes to the view.
   a. Right-click the header row (where it says Name, Date modified, Type, and Size) and select **More**.
   b. Select **Attributes**.

   Now, if a file is read-only, I can see it's marked with R.
6. In the View ribbon, select **Options** > **Change folder and search options**.

7. Go to the **View** tab and change the following:
    - Select **Always show menus**.
    - Select **Disply the full path in the title bar**.
    - Select **Show hidden files, folders, and drives**.
    - Clear **Hide extensions for known file types**
    - Under Navigation Pane, select **Expand to open folder**.
    - Under Navigation Pane, select **Show all folders**.
8. Click **Apply to Folders**.
9. If the `Do you want all folders of this type to match this folder's view settings` click **Yes**.
10. Click **OK**.

As a tech writer, you're going to care about what types of files you have. Whether that image is a `.jpg` or `.png` is really important. You're going to end up with files named the same but with different extensions. You want to be able to see these easily.

Paths

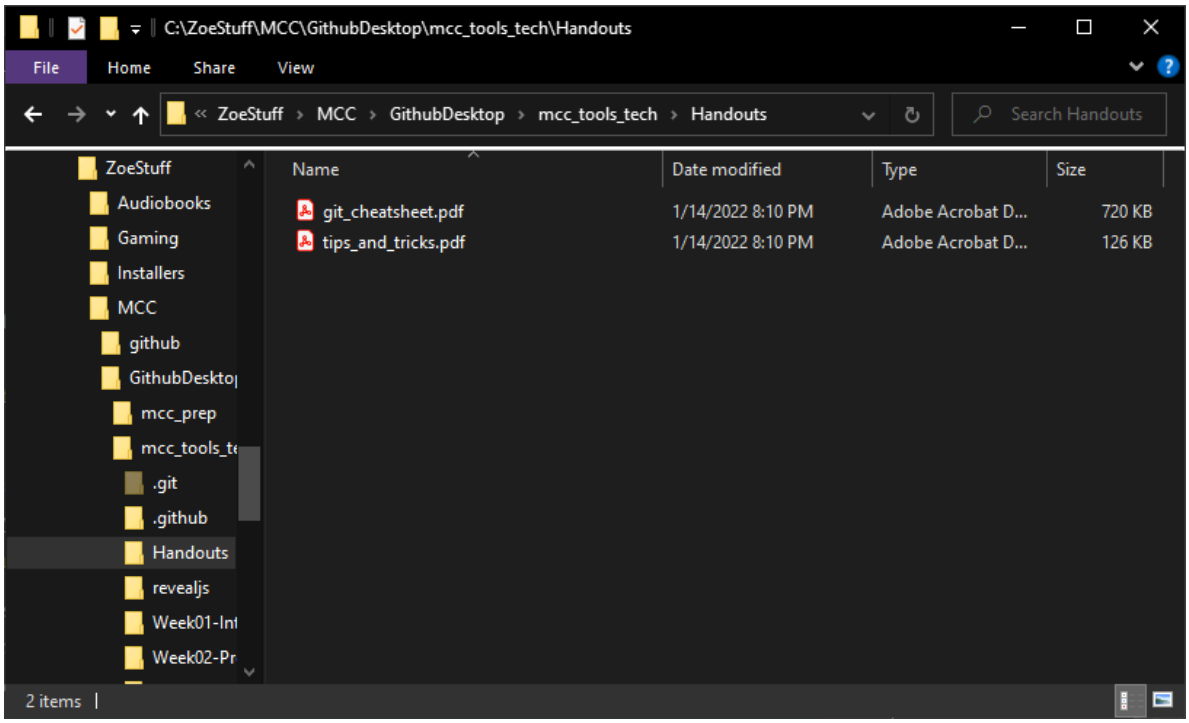
When working with computers, you often have to "write directions" on where to find a file. These are called 'paths'.

Computers have tons of information in them. This data and the tools to view and use the data is organized for humans into files and containers usually called folders or directories. The actual bits and bytes may be scattered all over a harddrive, but logically, computers present this data to us as files and folders. Files can be in folders, but folders cannot be in files. Folders can be nested inside of other folders.

Humans need to be able to find files and applications, so we need some sort of addressing system. Computers use bits and bytes and pointers n' things that are very hard to remember, unless you like remembering hex codes. So, someone figured out that you can write out the "path" to the file using folder names.

In the Unix world, someone decided to use forward slashes (/) as the separator between folder and file names. To avoid copyright infringement (or to be annoying) DOS (i.e. Windows) uses back slashes (\). This has caused all sorts of confusion and annoyance and bugs in the computer world ever since.

Most computer languages like to use forward slashes. Notice URLs, they use forward slashes. Notice all the paths in GitHub, they use forward slashes. Often, when linking to files for cross-references or images, you use forward slashes.



This is a screenshot of my GitHub Desktop folder location for this handout. You can see the path in the title bar: `C:\ZoeStuff\MCC\GithubDesktop\mcc_tools_tech\Handouts`. Since this is a Windows system, it's using backslashes. In the Navigation Pane/Tree View, you can see all the folders and how they nest.

If I want to know the precise location of this handout on my computer, it is `C:\ZoeStuff\MCC\GithubDesktop\mcc_tools_tech\Handouts\tips_and_tricks.pdf`. This is called a *fully qualified path*. This tells me exactly where the file is, on my computer.

This is great...as long as I am working on my computer. If you want to find the file on your computer, it's not quite as useful.

Sometime people use *variables* in path descriptions. Common formatting conventions include italic text or surrounding with angle brackets. So I might write about this location as `<GitHub Desktop Repo Location>/mcc_tools_tech/Handouts/tips_and_tricks.pdf`. The string `<GitHub Desktop Repo Location>` indicates whatever directory I picked for cloning the repository. That could be the default of `C:\Users\User Name\Documents\GitHub` or the path I picked, `C:\ZoeStuff\MCC\GitHubDesktop`.

If you pick a formatting convention for variables, make sure you include some non-formatting related indicator. If you just make variables italic, a screen reader may not recognize the font change. If you include something not format related, such as adding the angle brackets (< >), it makes the difference more accessible.

This format of writing paths is extremely common in technical documentation. The only difficulty is standardizing how you refer to your variables so you don't have a mishmash of *Install_dir*, *Product Installation Directory*, and *Installation_Dir*.

While using string variables is great for tech doc, it's not super useful for computers. Yes, there are variables in programming languages, and you absolultely use them in paths in programming, but

generally speaking, writing tech doc isn't programming. So there's another type of path you use all the time, a *relative path*.

A relative path is directions to a file based on where you're starting.

- `filename.txt` - the file is in the same folder as where you are starting.
- `directory/filename.txt` - the file is in the directory folder. The directory folder is in the same folder where you are starting.
- `../differentDirectory/filename.txt` - the file is in the differentDirectory folder, which is in the "parent" folder to where you are starting. With a .., you go "up" a folder.

For example, I'm starting in the Handouts folder and I want to get to the Week 2 homework folder. The Week 2 folder is not in the Handouts folder, so I go up a folder.

```
../
```

This puts me into the `mcc_tools_tech` folder. Goody! The Week 2 folder is in this directory, and I want to go into it.

```
../Week02-ProgressiveInfo
```

Now that I'm here in the Week 2 folder, there's the homework folder I want. So the relative path from the `Handouts` folder to the Week 2 Homework folder is:

```
../Week02-ProgressiveInfo/Homework
```

This relative path only works from certain locations, such as from the Handouts directory. However, this path will always work from the Handouts directory to the Week 2 Homework folder in the mcc_tools_tech repository, no matter what computer I'm on, or where you clone the repository.

Learning how to make relative paths between files is really useful. You use them all the time for cross-references and links to images.

Regular Expressions

Regular expressions (regex) are a way of augmeting searches to be more specific.

Regular expressions are a common "language" used to search for text strings. Many different scripting and programming languages use them. Some find and search and replace tools also allow them. (Notepad++ is a text file tool that uses them.) You can even sometimes do some search and replace.

There are several different flavors of regex. But there many options that are relatively standard. Here's a few common expressions:

Regex are case-sensitive. Searching for `Z` is different than searching for `z`.

`.`	Any single character
`[]`	A set or range of characters. `[xyz]` finds the character x, y, or z. `[a-z]` finds any lowercase letter between a and z. You can also search for numbers, such as `[0-9]`. A common expression is [a-zA-Z] which finds

	any upper case or lower case letter. [a-zA-Z0-9] finds any alpha-numeric character.
*	Zero or more times. In general `.*` finds most anything.
\	Escape character. `.` finds most anything. `\.` finds a period. If you need to find a backslash, you can double it, `\\`.
^	When used inside square brackets, not whatever. `[^a-z]` finds any character that isn't a lowercase letter.

There are many, many other options, but you can find your favorite cheat sheet, or the flavor you need for whatever application you're using on the internet.

Some sites to get you started:

- https://www.rexegg.com/regex-quickstart.html
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions/Cheatsheet
- https://cheatography.com/davechild/cheat-sheets/regular-expressions/

Formatting tidbits

Typography and layout is part art, part science. Again, there's too much to go into. However, if you play with text formatting, there are some things to be aware of.

Here's a collection of random facts to file and forget about fonts and formatting.

- Fonts with extra flourishes on them are called *serif* fonts. A serif is a leftover from when letters were made with pens with nibs. The extra thickening or tick or swoosh at the end of a line is a serif. Common practice believes that serif fonts are easier to read in print materials.

  Common serif fonts include Times, Times New Roman and Palatino. Most typewriter-like fonts such as Courier are also serif fonts.
- Smooth fonts without flourishes are called *sans-serif* fonts. Common practice believes that sans-serif fonts are easier to read on screens.

  Common sans-serif fonts include Arial, Calibri, and Helvetica.
- With the advent of computers, most fonts are variable-width. A lower-case i or l takes up less space than an m. Most typewriters were monospaced. All letters took up the same amount of space on the page, no matter the size of the character within that space.

  When dealing with code or command line tools, where you're either mimicking an old terminal screen or white space matters, monospace fonts are often used.

  Common monospace fonts include Courier, Courier New, and Consolas. (Yes, Courier is both a serif font and a monospaced font. Consolas is a sans-serif monospace font.)
- Once upon a time, computer terminals were limited to 80 characters wide. Most code samples don't want to go wider than that. For print, humans generally read best at 60-70 characters per line, assuming a single column on the page. Digitally, things get weird with screen size, glare, flicker, scrolling, etc. But if you've ever tried to read small print that goes across your entire HD monitor, you know that's hard to read.
- A few more typography terms you might want to know because sometimes you care when defining formatting:
  - kerning - the spacing between two letters.
  - leading - the spacing between lines.
  - widow - the last line of a paragraph is on the next page.
  - orphan - the first line of a paragraph is on the previous page.
  - justification - how the text lines up with the margins. Left justified text aligns along the left margin (this is the most common justification in English). Fully justified means the text lines up against both the left and right margins. This can lead to some really terrible word spacing on occasion.
- Typography uses weird sizes. There's some logical ones, like inch and centimeter, but then you get into some fun ones:
  - pixel (px) - a single dot on a monitor. Once upon a time, CRT monitors were pretty much always 96 dots (or pixels) per inch. Now, with HD monitors, screen resolution is highly variable. Using pixels with fonts is generally a bad idea, because the size of the object change based on the screen resolution. However, they're important to be aware of for graphics and some parts of CSS.

    - point (pt) - 1/72 of an inch. A monospace 72 pt font should be an inch square. A 36 pt font is half an inch high. Fonts are defined in points. Often spacing in and around lines of text are also defined in points, since they're related.
    - pica (pc) - 1/6 of an inch. There are 12 points in a pica. I've rarely seen this used, but it's usually in the list of measurements you can use.
    - em - A width dependant on the size of the font. Traditionally, it's the width of a capital M, which is generally the widest letter. Today, it's the same as the size of the font. So, for a 14 pt font, an em would be 14 points wide. This can be useful because the spacing will always be relative to the size of the font, instead of being prescribed by a set distance.
- If you're ever designing a template, find a *lorem ipsum* generator. There are a ton of great ones online. When you're doing layout, you need sample text to work with. While you might have existing content lying around, often you don't. Lorem ipsum is some gobbledygook text that has a similar spacing and cadence as actual English. It can help you fill out paragraphs to see what happens with line wrapping, etc.

Habits to lose

If all you've ever done is write academic papers or the occasional business email, there are probably some formatting habits you'll need to break yourself of.

In no particular order:

- Stop creating formatting as you go.

  You have to apply existing styles as you write, but don't go about creating the styles as you go.

  As you write, you are creating new paragraphs, lists, numbered steps, etc. Avoid designing how things should look as you go. You don't want to be spending your time remembering that a second level heading should be Arial font, 24pt, with 36 pt above and 18 pt below, bold, and the official corporate shade of green. At most, you should just be applying some "Heading 2" style.
- Trust the template.

  At large companies, there is a team that defines the corporate look and feel. There may be an official corporate font. There are all sorts of rules about color schemes and page layout. You may hate it. But you have to use it. Do not tweak it. Do not add extra things so that it 'looks better'.

  Even at small companies, when writing, you want to focus on your content, not the formatting. Learn your style guide, and apply the standards as defined. Change the template, don't format as you go.

  If you don't like the styles, minimally see how you can make suggestions. If you really care, see if you can get on the committee that makes the decisions.

  Lots of tools get transformed before publishing. If you don't apply the correct styles, those transformations might break. Then you get to undo all the fancy changes you made.
- Avoid using spaces and carriage returns to make formatting work.

  Since most fonts are now variable width, you generally no longer need to use two spaces after a period. The font does that work for you. (In fact, many tools now automatically undo extra spaces

between sentences.) If you have spaces for other types of alignment, if the font changes, the alignment may change.

Similarly, don't use extra lines between paragraphs for spacing. If you need more space between paragraphs, fix the spacing in the template.

Learn if it's possible to insert page breaks in your tool. Page breaks can be a thing you add, or maybe you have different styles that certain ones "start on a new page" or "page break before". Also learn which styles have "keep with next" or "keep with previous".

- Know that in certain authoring tools, you have to completely let go of formatting.

  Markup languages, such as HTML, DITA, Docbook, Markdown, or ReST have no formatting. You're authoring some sort of source code that will be transformed by something. You just have to trust that the people designing and programming the transforms make output that works.

Resources

A short collection of some books, web sites, and tools I've found very useful over the years.

Tech writing sites

There are many, many resources out there, and they're constantly changing. These are some that I've enjoyed over the years.

There are a bunch of great people out there who have made talking about tech writing their job. They often provide free webinars that touch on up and coming trends in the tech writing world.

These are some general "Tech writing" sites in no particular order.

The Content Wrangler https://thecontentwrangler.com/

Scott Abel started writing and lecturing on tech writing many years ago. His blog turned into a great resource of information about tech writing.

TC Dojo https://www.single-sourcing.com/products/tcdojo/

I've caught a few webinars hosted by this group and have enjoyed them.

Write the docs https://www.writethedocs.org/slack/

This is a Slack channel, so you have to make a Slack account, but a good communities are good communities.

These are pages I bookmark because they're just useful

HTML Color Picker https://www.w3schools.com/colors/colors_picker.asp

Every now and then you'll need to enter HTML colors. Having a color picker in your tool box is convenient.

Color Palette Generator https://paletton.com

I'm a terrible web designer, and usually I'm handed color schemes by my company, but sometimes I'll want to design something. A color palette generator is useful. Or fun to play with.

HTML reference https://www.w3schools.com/html/

You're going to have to hand code HTML every now and then. Know a reference (or two) for when you forget how some tags go together.

CSS reference https://www.w3schools.com/css/

Hand-in-hand with HTML is CSS. You will have to design the look and feel of something at some point in time. CSS is the standard these days.

	Even if you're not producing HTML output, many tools still use CSS-like properties.
Batch scripting reference	https://ss64.com/nt/ The command line for Windows can be used from the command prompt. These commands can be put together in a script (usually with the `.bat` or `.cmd` extension). You can also just run these commands from the command prompt. There are some networking commands that are useful, and the copy and rename commands are invaluable.
DITA spec	http://docs.oasis-open.org/dita/dita/v1.3/dita-v1.3-part3-all-inclusive.html Since I love DITA and try to use it for everything, I have the specification bookmarked. It gives a definition for every element. If you don't use DITA, you don't need this. If you use some other public standard, such as Markdown, Docbook, or ReST, bookmark the standard.
Stackoverflow	https://stackoverflow.com/ I don't actually have this site bookmarked, but be aware of the glory that is Stackoverflow. I probably end up googling something that puts me on Stackoverflow at least once a week, probably more often. It's a site where folks ask questions, and other folks answer them. The collective knowledge here is fantastic.

On my bookshelf

I love the internet, but I will always love actual books.

A dictionary	Yes, I google things all the time, but a good dictionary is always a good thing to have available.
A thesaurus	Same as a dictionary, having a physical thesaurus is useful.
Style guides	There are many style guides available. Get the one that your company decides to follow. However, collecting them isn't terrible. Many include some basic 'introduction to tech writing' content than can be useful. Also, sometimes when you're trying to convince others to change a style or rephrase something, being able to point to someone else's official style guide is great. Many of these seem to be going online. • Microsoft Manual of Style - now only available online: https://docs.microsoft.com/en-us/style-guide/welcome/ • IBM Style Guide - https://www.amazon.com/IBM-Style-Guide-Conventions-Writers/dp/0132101300
Developing Quality Technical Information: A	Written by many authors from IBM, it provides a lot of great guidance for improving your writing.

Handbook for Writers and Editors	https://www.oreilly.com/library/view/developing-quality-technical/9780133119046/
Docs for Developers	An engineer's Field Guide to Technical Writing. It's designed for developers to learn the basics of tech writing if they can't hire a writer. Therefore, it's a nice, concise guide on how to be a tech writer. https://docsfordevelopers.com/
Unix in a Nutshell	If you need to use a Unix system, this book is a great place to get started. https://www.oreilly.com/library/view/unix-in-a/0596100299/
Web design in a Nutshell	You can get all your web design stuff from the internet, but this book helped me understand CSS and other basics. https://www.oreilly.com/library/view/web-design-in/0596009879/

Open source tools

There are a ton of great free tools available for you to use.

Always, check with your employer to figure out what their policy is for third-party open source tools. You may not be allowed to use them, or only tools under specific licenses (GNU vs Apache, for example), or they may have a special place to download 'blessed' versions.

These are the tools I'm aware of. There may be newer ones or better ones available, but these should get you started.

7zip

A zipping tool with a command line.

Once upon a time, Windows did not come with the **Send to compressed folder** option. There was WinZip, but you had to pay for it.

7zip is a free tool that makes zip files. Windows now has a built-in utility, but it doesn't have a lot of options. If you install 7zip, you can make zip files and also update contents of zip files. 7zip includes an integration with File Explorer, so it's easy to zip up or extract files.

The big plus is that it comes with a command line. If you want to get into scripting, being able to automatically zip up files is extremely useful. Most real scripting languages such as Python include zipping libraries, but Windows scripting options didn't. (Unix generally comes with `zip` or `gzip` commands.)

Reputable download

https://www.7-zip.org/

Audacity

If you get into making videos with an audio track, you may want some additional audio editing tools beyond what comes with your video tools. Audacity is a decent free audio editing tool.

If you need to edit audio, Audacity is great. I was introduced to it when I got a USB turn table and I was converting record albumns to MP3s. I could use Audacity to divide up the record into tracks.

If you're doing video with a voice over, you can use Audacity to clean up your audio clips.

Reputable download

https://www.audacityteam.org/

Gimp

If you want a free image editing application, this one has a bunch of features.

Photoshop is one of the standards, but Adobe products are not cheap. I have not used gimp much because I have spent money on other tools, but many folks I know have used it and are very happy with it.

Reputable download

https://www.gimp.org/

Imagemagick

This is a command line tool, so may not be your cup of tea, but if you have to do bulk conversion, it may be worth looking into.

This is a very, very powerful tool. It can work with many different types of files and convert them to other types. It can do all sorts of resizing, resampling, etc.

I've used this tool to automate resizing images. If you want to have different sized images with different resolutions for PDF vs online, you can use this tool to automatically take one image and create the different versions you need, or for making thumbnails. It's also great for conversion work if you have to update a whole bunch of JPG files to PNG files.

Reputable download

https://imagemagick.org/index.php

Inkscape

Resizing images is pain, so having a good vector image editor is useful.

There are two major types of images, bitmaps and vector diagrams. Oversimplifed, bitmaps images are files that save the information as color information and pixel position. Vector diagrams are image files that save the information as shapes and math with properties. In general, photos are bitmaps. Illustrations are probably vector diagrams. Photoshop and CorelPhoto edit bitmaps. Adobe Illustrator and Corel Draw are vector editing applications.

Inkscape is an open source vector drawing program. You can save them as SVG files. Today, most browsers can render SVG images. Because SVG images are vector based, you can zoom into them and they will always look good.

Reputable download

https://inkscape.org/

kdiff3

At some point in time, you will be working text files. Your going to need to compare text files to figure out what's going on.

There are many different comparison tools out there. This is one I found years and years ago that I have been very happy with.

Reputable download

Source: https://invent.kde.org/sdk/kdiff3

If you go to the README, you can find the link to the download.

https://download.kde.org/stable/kdiff3/?C=M;O=D

Notepad++

You have to edit text files. Or tidy up stuff that's easier to work with as text. Or change a file's encoding. Notepad++ is a fantastic text editor.

You are going to need a text editor. Sometimes you will be working with text files, such as properties files or raw HTML. Other times you will be cutting and pasting between applications that add extra bits. (Have you ever copied something from a web page and pasted it into a Word file and got extra junk?) Yes, all computers have some sort of text editor, but plain old Notepad doesn't have extra features. Having a good text editor is invaluable.

Notepad++ is my favorite text editor. It's familiar with a whole bunch of common coding languages and provides highlighting. You can change file encoding (e.g. English vs Russian) and also change line endings. You can search and replace across files. You can perform searches using regular expressions.

Reputable download

https://notepad-plus-plus.org/

Pencil

This tool was introduced to me as a UI mockup tool. You can also use it to make basic diagrams.

Being able to throw together a UI design is sometimes part of your job. Being able to put together basic workflow diagrams is a common requirement for techdocs. This is a nice tool to be aware of and can be worth using.

Reputable download

https://pencil.evolus.vn/