GW ACM UNIX Cheat Sheet

Moving Around

COMMAND	WHY?
cd [directory]	Change directory- use this command to go to a folder in the current location
cd ..	By passing the "..", you're saying "go back one directory". Use this to move go back
ls	List the contents of the current directory
pwd	Print working directory- prints the full path of your current directory

File Basics

COMMAND	WHY?
cp [input file] [output location] [-r]	Copy the input file to the specified location. Using "-r" allows for recursively copying contents of directories
mv [input file/directory] [output location]	Move the specified input file or directory to the specified output location
mkdir [directory name]	Create a directory in the current location
rmdir [directory name]	Delete an *empty* directory
rm [file name]	Delete a file
rm -r [directory name]	Delete recursively- this deletes a directory and its contents. NOTE: you might need to add "sudo" at the front of the command

vim Basics

What is **vim**? It's a text editor that lets you edit files from the command line.

COMMAND	WHY?
vim [filename]	Open vim with the selected file (can be a file in current directory or new file name)

GW ACM UNIX + Git Workflows Cheat Sheet

Commands for Inside vim

COMMAND	WHY?
i	Enter “insert” mode. You can only write/delete text when in insert mode
esc (escape key)	Exits the current mode or command
:w	“Write”- saves the file
:wq	Write (save) and quit
:q	quit

THE GIT WORKFLOW

1. **Make Some Changes -** Edit some files locally! (Using whatever editor you prefer)
2. **Tell Git You Made Some Changes -** Using “git add [file]”, tell git that you want to keep track of the changes you just made. Using “git add” tells git that these files should be added to the “stage”, and will be a part of your next commit
3. **Tell Other People You Made Some Changes -** When you’re done with your commit, use **“git commit -m [message]”** to write a commit message.

Tips:

- **Write meaningful commit messages!** “Made some changes” is a useless commit message, especially if something breaks...
- **Commit often!** Any new major piece of code should be its own commit, so that if something goes wrong, it’s easy to track down what caused the problem
- **Make sure your repo is up-to-date!** (and “git pull” a lot!) If you’re making commits to an out-dated repo, then you’re almost certainly going to run into merge conflicts