Do not fear the Unix command line! While less user-friendly than the GUI, the command line gives you access to extremely powerful tools for file and system management, scripting, and networking. Most of the world's web servers, mobile devices, and supercomputers run on some variant of Unix or Linux; in many respects, the Internet itself is a Unix environment.

When we hear so much about the rapid pace of technological innovation and evolution, why do so many of the world's critical information systems run on an OS that was first developed in the 1960's? The simple answer is: because it works!

What do we mean by 'Command Line Interface'?

Most desktop computers, laptops, and mobile devices have a **Graphical User Interface (GUI)**. The GUI allows the user to issue commands via mouse or touchscreen clicks, drags, gestures, etc. We are so accustomed to the GUI that we usually don't notice it, and that's a testament to the skills of the many designers (including MHC alums!) who have worked to make GUIs seamless and intuitive.

But of course, your computer doesn't really contain tiny "folders" and "desktops" and "trash cans"; these are metaphors the GUI uses to remind us of stuff we use (or used to) in the physical world.

A **Command Line Interface (CLI)** is a different way of issuing commands to the computer. It's not as friendly as the GUI, but it is extremely powerful, flexible, and efficient. If you end up working with computers professionally, there will likely be times when the CLI is your only option: some computers (such as servers) typically don't have a GUI at all, or you may need to manage a computer at a remote location via shell commands.

General info

- Mac OS **Terminal.app** is found in /Applications/Utilities
- Terminal window or session is sometimes called a **shell**
- CLI is case sensitive
- Lists sort differently than they do in GUI windows (capital letters first)
- CLI is unforgiving: no spell check, no undo, no "are you sure?"
  "UNIX was not designed to stop its users from doing stupid things, as that would also stop them from doing clever things."
- Unix is a true multi-user OS, unlike Windows. Permissions matter!
- Most of these commands work the same on any Unix or Unix-like system, such as:
  - Darwin (the underlying Unix system of Mac OS and iOS)
  - BSD Unix and its offspring (FreeBSD, etc)
  - Various Linux distributions (Ubuntu, Debian, Red Hat, etc)
- Mac OS includes many Apple-specific CLI tools as well
- You can run commands immediately, or save them in a **shell script** file (.sh) for later use

CLI commands may include:

- **arguments** — the 'nouns'
- **flags** (sometimes called **switches**) — the 'adjectives' — things like:
    - **-R or -r** = recursive (do this command to a whole directory, including sub-directories)
    - **-H** = human-readable (eg, display file sizes in MB or GB, not bytes)
    - **-v** = verbose (tell me what you're doing)
    - Flags are somewhat standard between commands, but not always. **RTFM!**
- wildcards, variables, regex

Basic filesystem navigation

`pwd`	print working directory (where am I?)
`cd`	change directory — use tab key to autocomplete paths!
`ls`	list contents of a directory
`ls -l`	list a directory "long" (formatted in a column)
`ls -al`	include invisible files
`ls -alh`	human-readable sizes
`.`	this directory
`..`	parent directory
`~`	current user's home directory (default starting point)

How to deal with white spaces in file paths

- **escape** with backslash
- **enclose** the whole path in single quotes
- try to avoid using them in file & directory names whenever possible

How CLI relates to the Finder GUI (Mac OS only)

Drag & drop a file or folder into Terminal window to resolve its full path

`open .` open current directory

`open /some/directory/path`

`open /Applications/SomeApp.app` to launch a GUI app from Terminal

More filesystem tools

`mkdir -p`	create a directory path, including any intermediate directories
`touch`	update an existing file's timestamp, or create a new file
`mv`	move / rename file or directory
`cp`	copy file or directory

`rm`	delete file or directory — **BE CAREFUL!**

Other useful things to know

`nano`	simple command-line text editor
`(ctrl)-C`	STOP! Interrupts whatever the shell is doing
`history`	show previous commands — scroll back with up-arrow
!!	re-run last command
!	with line number, re-run a command from history
`man`	manual — space to page down; q to exit
`sudo`	run a command with admin privileges
`rsync`	sync one directory to another (local or remote) — great for backup
`diff`	compare two files
`grep`	search for pattern — piping your output to grep can narrow down results
>>	redirect output to a file. example: **`date >> ~/Desktop/thisFile.txt`** >> appends data to the file > "clobbers" the file before writing to it (one > single)

Permissions

UNIX assigns a set of **permissions** to every file and directory. Permissions control who is allowed to do what. There are three classes designated **u**, **g** and **o** (user, group, and others) and three permission levels designated **r**, **w** and **x** (read, write and execute). Each of the three classes can have any combination of the three permissions. This means there are 2^9 (512) possible combinations of permissions, though in reality many of these have no practical application..

Typically, the **user** is the owner and has the highest level permissions to any given file or directory. **Group** may have the same or slightly lower level of permissions. **Others** (sometimes called **world**) permissions are typically lowest of all. There are, of course, exceptions.

You'll see the ownership and permissions info when you **`ls -l`**.

Permissions should maintain the highest level of security possible while still letting authorized users do what they need to do. Manage permissions using the **`chmod`** & **`chown`** commands. Much more about permissions here.

Remember: Unix won't prevent you from "locking yourself out of the house". On the CS dept. lab Macs, you can't break anything outside of your own home directory. But if you're the admin of your own Mac / Linux laptop, you can theoretically bork up your permissions to the point where the computer becomes unusable.

Before you start experimenting with Unix permissions on your own computer, create a second admin account for emergency use. Then if you make a mess of your permissions, you can log in as the other admin and fix them.

`chown`	change the owner/group of file(s) or folder(s)

`chmod`	change read/write permissions of file(s) or folder(s)

System info & management

`ifconfig`	show network interface info (Ethernet, wireless, etc)
`uptime`	how long since last reboot
`whoami`	show currently logged in user
`du`	disk usage
`top`	show CPU usage, etc
`ps -ax`	show all running processes
`pgrep`	show Process ID (PID) for a given process, e.g. **`pgrep Finder`**
`kill`	kill one or more processes (by PID)
`killall`	kill one or more processes by name

Network tools

`ping`	see if a host or IP address is reachable via the network
`host`	resolve a hostname to an IP address via DNS lookup
`dig`	get info about your network's DNS server
`curl`	connect to a remote server via http, ftp, etc
`ssh`	"secure shell" — log into a remote host (if allowed)
`scp`	"secure copy" — copy files between computers (if allowed)
`traceroute`	show hops to remote server
`whois`	show owner of an internet domain, IP address, or IP range

Apple-specific and 3rd party tools

In addition to these generic Unix commands, many Apple-specific command-line tools are built into every Mac. These replicate many of the functions of Apple's built-in apps and utilities. A small sample:

`ditto`	smart file copying utility
`osascript`	run Applescript commands from the Terminal
`sips`	basic image-processing utility (convert formats, scale, rotate, etc)
`afinfo`	get info about audio files
`afconvert`	convert audio file formats

Just like GUI apps, there are vast numbers of extra CLI tools available to extend the capabilities of your Unix system. Many of these are free. Installation may work in various ways: some are packaged in a familiar "installer" executable, some require running a script in your Terminal, and some may have to be built from source code. Here are a few cool ones:

`goosh`	the unofficial google shell. Search Google from the Terminal!

`ia` command line tool for the Internet Archive API.

`gam` Google Apps Manager lets you talk to Google Apps API from the Terminal. An essential tool if you are ever the admin of a Google Apps domain for school or business.

Resources

- Basic UNIX command glossary and cheat-sheet from Software Carpentry
- RTFM! Built-in manuals: **`man`** + [command name]
- A whole document about nothing but permissions.
- the List Of Useful Bash Keyboard Shortcuts
- For any given UNIX command, someone has probably already figured out how to do what you're trying to do. Search online for [command name] + "UNIX examples"
- "never fear / the command line" workshop notes (ooh — recursive!) and slideshow
- Regex (regular expressions) examples
- The incredibly useful ss64 command-line index (not just Unix / Linux, but also DOS prompt, mysql, and more)
- A guide to working with calendars and dates
- Apple's shell scripting primer — most of this will also work on Linux
- More useful Mac-specific Terminal commands
- Unix.com (search before you post! They are merciless)
- A directory full of random files and subdirectories for you to download and mess around with
- Mike Sierra's excellent "Learn UNIX in a hurry" documentation at Deep End Of The Pool
- Quick references for editing your PATH variable and customizing your bash_profile