Command Line Commands for Mac

By: Andrew Krause

Basic Commands:

ls - shows you everything in your directory

touch - used with a file such as test.txt will create a new file in your directory (ex. **touch test.txt**)

cat - reads the contents of a file and displays them on the terminal.

whoami - returns your username in the terminal

man - can be used before any command to learn more about that particular command (ex. Typing 'man echo' in your terminal will tell you about the echo command).

date - gives you the current day and time

- used with -**u** gives you Coordinated Universal Time (UTC).

-a - lists both visible and invisible files and directories (used with **ls**)

-p - helps us create nested directories. Usually used with mkdir (ex. $ mkdir -p <whatever is nested>).

-l - gives a more detailed view of files and directories in current directory including number of bytes (used with **ls**)

-S - sorts the results of last by file size (used with **ls**)

-h - gives the size of the file in human readable terms (used with **ls**)

file - used often with a path to a file. The command states the type of the file (directory or other)

-t - sorts last results by the last time of modification (used with **ls**)

-r - reverse the results of last (used with **ls**)

unzip - extracts a file or files from a zipped file

-v - prints the results of mkdir to the console (ex. $ mkdir -v foo).

-f - forcing a link if you get an error that "the file already exists" when attempting to link files

ln - allows to point or link one file to another

diff - used with two files; compares the contents of the two files and, if there are any differences between the two, displays those differences on the console (ex. diff test1.txt test2.txt)

*****- matches zero or more characters in a file name so *.txt matches a.txt but not any.txt.

? - matches any single character to a filename, so *.txt matches all files ending in .txt.

-s - creates a link to a directory (can be used with **ls**)

cat - lists contents of file or directory

cd - allows you to change or navigate to a different directory (ex. $ cd ~/Applications).

- If you ever want to navigate back to your home directory, you can call cd without any arguments

.. - used with cd command ($ cd ..) navigates to the parent directory of the current directory

"-" - used with cd command ($ cd -) navigates back to the working directory you were in

-v - used with CS commands

cp - copying files and directories (format is usually: **cp <what you want to copy> <where that copy will go>**).

- When used with **-r**, you can copy a directory and all of its contents

emacs/nano -nw NAME OF FILE - displays file content editor in a separate window on the terminal

rm - deletes a file, folder, or directory. Usually used with the -v flag to list what was deleted.

- (If you hit the TAB key, a command that you have begun to type can be completed for you. This is useful when typing out long commands.)
- **IMPORTANT:** It is highly recommended that you use -**i** with the rm command to double check with yourself if you really want to delete the file or directory you are considering deleting:
- (Example: **rm -i <name of file or directory you want to delete>**). This will prompt you to hit ‘y’ (yes) or ‘n’ (no) to either delete or not delete.

rm -r - deletes a directory and all of the files contained within that directory. Usually used with the -v flag to list what was deleted.

mkdir - creates a new directory

top - shows overall system utilization, including top resource using processes

ps - shows a list of living processes

kill - used when you need to kill or stop a process; if process won’t stop, use **kill -9**

fg[n] - reattaches to the specified job (see jobs command below) or if specified, the most recent job

jobs -lists the currently running background processes

***.txt** - used with cop command. Will copy every file with the extension .txt.

-v - verbose output. Prints the verbose output regarding copied files.

-R - recursively copies the directories contents to the new directory.

-**Q** or **q** - use this command when you want to quit running a process such as a page that explains how to use commands on your terminal.

-f - force overwriting a file. Forces the copying of a source file to a target file. Useful when different users own different files.

-i - confirm overwriting a file. cp will prompt you to confirm that you are about to overwrite another file if you use the -i flag.

echo $SHELL - shows you what shell you are currently in on your terminal.

mv - moves files. The process of moving files is almost identical to copying files. Usually used with -v to list where the files were moved (can achieve same effect by using cp and rm commands).

- Also used to rename files. (Format is usually **mv <name of what you want to move> <where you want to move it>**)
- Also, if you want me move something to the current working directory, use the command exactly how you just formatted it above, except add a '**.**' (for current) or '**..**' (for parent) afterwards.

pipe operator (|) - make the output of one command the input of another command.

grep - lists all files with whatever extension is listed afterwards (ex. grep _ lists all files with an underscore).

pwd - print working directory

IMPORTANT: When you use **"."** with the **mv** command, you can move a file from a location like downloads into your current working directory

sed - changes all extensions or marks (ex. $ ls -a ~ | grep _ | sed "s/_/-/g" will change all the underscore to dashes).

> - write data to a file (ex. $ ls -a ~ | grep _ > underscores.txt).

< - read data from a file

CONTROL A - takes you to the beginning of the line of commands

CONTROL E - takes you to the end of the line of commands

CONTROL U - clears the line of commands

clear - clears everything on the terminal so you are left with a blank screen

CONTROL L - shortcut does clearing everything on terminal

echo - prints whatever you type after it on the screen

exit - used for exiting a terminal window

CONTROL D - shortcut for exiting a terminal window

CONTROL C - IMPORTANT. Use this command whenever you have executed something incorrectly or are stuck to get back to the original command prompts

CONTROL O - Use this command when you want to write data to a disk after writing in a text editor within the terminal.

CONTROL X - quit an editor and return to shell.

COMMAND L - clears one line of commands at a time

CONTROL C - exits out of an actively running program; used often to get out of trouble if you enter an incorrect command and encounter issues; if hitting CONTROL (Ctrl) C does not work, then usually hitting ESCAPE (ESC) on your computer should work.

chsh -s /bin/NAME OF SHELL - changes the default shell to whatever the name of your input is (as long as it is a name of a real shell)

sleep - used with the amount of time (in seconds) that you want the terminal to sleep (ex. sleep 5 will make terminal sleep for five seconds)

ESC F - moves one word at a time

ESC B - moves one word at a time

say - makes your computer speak out loud (ex. say "hello")

nc towel.blinkenlights.nl 23 - allows you to watch a pixelated version of Star Wars episodes IV - VI on the terminal

defaults write com.apple.Finder AppleShowAllFiles TRUE; killall Finder - shows hidden files on your computer. Useful when attempting to navigate to a folder that you cannot see on your laptop interface

defaults write com.apple.Finder AppleShowAllFiles FALSE; killall Finder - hides hidden files on your computer. Useful when attempting to simplify the interface on your laptop interface

Switching Shells - (not an argument) to switch from the shell you are using, first check and see which shell you are in.
Method1:
- The is can be done using the **echo $0** command.
- After you find what shell you are in and want to switch it, type out **cat /etc/**

shells. This will bring up a list of the shells you can switch to.
- Lastly, type the name of the shell you want to switch to (Ex: **bash**) and hit enter.

Method2:
- To switch the shell you are using, first type in **chsh -s /bin/NAME OF SHELL TO SWITCH TO**.
- Next, after you enter your password, enter the command called **exec su - $USER**. Then enter your password again.

Method3:
- Simply navigate to Terminal -> Preferences -> General Window; then change the "Shells open with:" space to whatever shell you want (Ex: /bin/bash).

Cybersecurity:

ipconfig getifaddr en0 - obtains your local IP address for Wi-Fi connections

ipconfig getifaddr en1 - obtains your local IP address for wired connections

curl ifconfig.me - gets your remote IP address as seen by other users online

ping - gets the attention of an IP address; displays the connection speed between your source computer and a specified destination computer (also for testing connection speed between your computer and a website)

CAREFUL. The following is part of how to administer a DDoS attack:

abt install hping3 - only need to use this once; used to download a hacking tool for various hacking strategies

sudo hping -1 --flood - with the correct software downloaded, this command begins a DDoS attack on a computer. To complete the DDoS attack, you need to use multiple IP addresses with this command to attack the site you are targeting.

To stop a DDoS attack, turn off ICMP on your server (turn off ping). This stops the server from receiving pings so it will not be overloaded by a DDoS attack.

Random Fun Things to Do:

Animate a Christmas Tree
- First, make sure that wget is installed: If it is not installed, use the command: **brew install wget**
- After wget is installed, use the command: **wget -d -c -o "christmas.sh" "https://raw.githubusercontent.com/sergiolepore/ChristBASHTree/**

master/tree-EN.sh”

- Next, run the command: **chmod +x christmas.sh**
- Lastly, run: **./christmas.sh**

Python Commands:

python3 - allows you to run Python in the command line

exit() - exits out of using Python commands in the command line

idle3 - opens a separate shell that you can program in with Python

pip3 - package a manger for Python that allows you to add new libraries

python - - version - allows you to see the preinstalled version of Python on your Mac

idle - opens the older version of python in a shell

Side Notes:

CMD + **Option** + **J** opens the console in your Chrome browser.

CMD + **Shift** + **3** takes a picture of the entire screen.

CMD + **Shift** + **4** allows you to determine what part of the screen you want to capture in your picture.

Different combinations of commands can yield different results. Examples:
ls -F / - gives us a listing of files and directories in the root directory
ls -help - how to find how to use a command

Terminal Commands for Java:

javac PROGRAM NAME.java - compiles a program in java to prepare to be run

java PROGRAM NAME.java - runs compiled program. (Make sure you use the command listed before this to compile the code first.)

Running programs with arguments - (not an argument) to run a program that takes in command line arguments such as text files, first compile the program using **javac programName.java**. Next, run the program with the **java programName.java** command followed by the names of the arguments (such as text files) on the same line. For example, assume you have two arguments called test1 and test2. You would run your program that takes in the two arguments by typing in: **java programName.java test1 test2**.

Terminal Commands for C:

gcc -Wall -o PROGRAM NAME PROGRAM NAME.c - this command prepares to run a C program. (Keep in mind that this creates a prepared file in your folder that runs the program.)

./PROGRAM NAME - this command runs the prepared program. (Make sure command listed above this one is run first. Also, if you have already ran the command above this one, you usually do not need to run it again. Simply run ./ PROGRAM NAME, and your program should run.)

To open a file, use the text editor program of your choice and then the file you want to open is the argument: **nano test_prog.c** (we use **nano** for your terminal in this case).

In UNIX (may work in java as well), you can use the **>** and **<** characters to show file redirection. If you want to output data to a file from a program, you would use something like:
./programToRun > textFileForOutput.txt.
If you want to bring in data from a file into a program, you would use something like:
./programToRun < textFileForInput.txt.

To make the process of compiling and running programs in your terminal easier, you can use a Makefile to complete certain processes for you (template for a basic Makefile is in a separate document). The command called **make** can compile your program for you while the command called **make clean** can delete the compiled code. This is useful when you want to simplify the process of running and compiling.