Getting Started in Bash

There is no user-friendly interface for many of the programs that the lab uses, so you'll have to become familiar with navigating via command line and Bash.

What is Bash? What is a shell?

Bash (which is an acronym for 'Bourne-Again SHell') is a simple programming language used for navigating and manipulating file systems in UNIX. The shell it refers to is the command language interpreter for the GNU programming system (Unix). Windows operates not with Bash but MS-DOS, which is why the commands differ from what you use on a Mac (which is Unix-based). There are Bash ports for MS-DOS as well, so you can use Bash on a Windows machine.

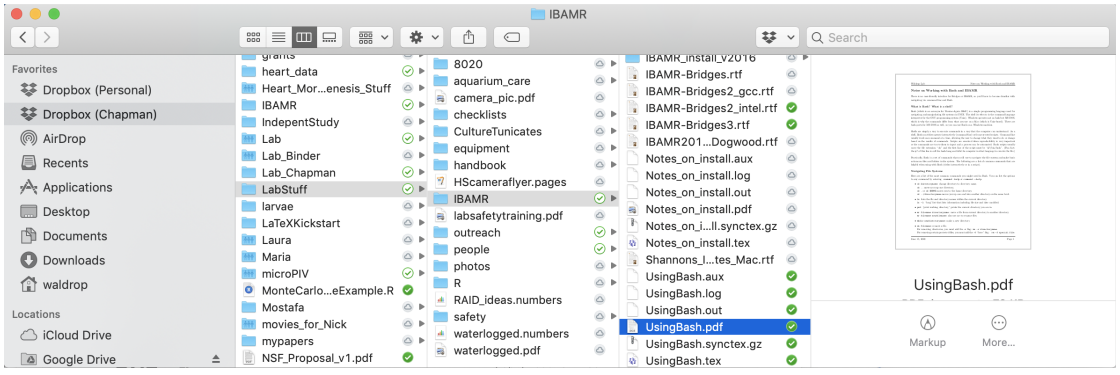
Shells are simply a way to execute commands in a way that the computer can understand. As a shell, Bash can either operate interactively (command-line) or from executed scripts. Command line usually work one command at a time, allowing the user to change what they want to do or change based on the results of commands. Scripts are executed when reproducibility is very important or the commands are too tedious to input and a process can be automated. Bash scripts usually carry the file extension ".sh" and the first line of the script must be "#!/bin/bash". (Fun fact: the #! of this line is call the hash-bang and tells the computer in what language to execute the file.)

Practically, Bash is a set of commands that you'll use to navigate the file system and make basic actions on files and folders in the system. The following are a list of common commands that are helpful when using with Bash (either interactively or in a script).

Navigating File Systems

Files on computers are organized into directories, represented by folders. These subsections of files are important because when the computer executes commands or looks for files, it scans within a 'working directory.' If all the files on a computer where in the same directory, this would take a long time! You should therefore use directories to organize your files and make the computer do less work when you run code. (In other words, do not store all your files in your Download folder!)

Usually, when you interface on Windows or Mac, you do so through a graphical user interface (GUI) like Finder:



This shows you the file "UsingBash.pdf" on my computer in different nested folders. However, in a Bash Terminal, it would look like:

```
waldrop — -bash — 80×24
Last login: Mon Jun 15 11:33:38 on ttys000
CPSC-WALDROP-MBP:~ waldrop$
```

Not as much information, right? But you will need to learn to navigate via Bash with some common commands!

Here are a list of the most common commands you might need in Bash to navigate file systems. You can list the options to any command by entering: `command -help` or `command --help`.

- `pwd`: "print working directory," prints the current (working) directory you are in:

```
CPSC-WALDROP-MBP:IBAMR waldrop$ pwd
/Users/waldrop/Dropbox (Chapman)/LabStuff/IBAMR
```

- `ls`: lists the file and directory names within the current directory. This list is comparable to the one in Finder!

```
CPSC-WALDROP-MBP:IBAMR waldrop$ ls
Bosque-IBAMR-Install-clang.rtf          IBAMR2016_Dogwood.rtf
Bosque-IBAMR-Install_Notes.txt          IBAMR_Install_August_2016.pdf
Bosque.aux                              IBAMR_install_v2016
Bosque.log                              Notes_on_install.aux
Bosque.out                              Notes_on_install.log
Bosque.pdf                              Notes_on_install.out
Bosque.synctex.gz                       Notes_on_install.pdf
Bosque.tex                              Notes_on_install.synctex.gz
Bridges.aux                             Notes_on_install.tex
Bridges.log                             Shannons_IBAMR_Install_Notes_Mac.rtf
Bridges.out                             UsingBash.aux
Bridges.pdf                             UsingBash.log
Bridges.synctex.gz                      UsingBash.out
Bridges.tex                             UsingBash.pdf
Finder.png                              UsingBash.synctex.gz
IBAMR-Bridges.rtf                       UsingBash.tex
IBAMR-Bridges2_gcc.rtf                  bash1.png
IBAMR-Bridges2_intel.rtf                data_analysis
IBAMR-Bridges3.rtf                      github-clone.pdf
CPSC-WALDROP-MBP:IBAMR waldrop$
```

`ls -l`: "long" list that lists information including file size and date modified.
`ls -a`: lists "a"ll files in the directory, even the hidden ones. Hidden files often start with a period, like ".bash_profile" and won't show up on a normal list. Using the flag -a will force them to show up.
You can also combine these flags together to create a long list of all files:
`ls -al`

- `cd directoryname`: change directory to directory name.
  `cd ..` moves you up one directory,

```
CPSC-WALDROP-MBP:IBAMR waldrop$ cd ..
CPSC-WALDROP-MBP:LabStuff waldrop$ ls
8020                    aquarium_care           outreach
CultureTunicates        camera_pic.pdf          people
HScameraflyer.pages     checklists              photos
IBAMR                   equipment               safety
R                       handbook                waterlogged.numbers
RAID_ideas.numbers      labsafetytraining.pdf   waterlogged.pdf
```

  `cd ../directoryname` moves you up one and into another directory on the same level.

```
CPSC-WALDROP-MBP:IBAMR waldrop$ cd ../equipment/
CPSC-WALDROP-MBP:equipment waldrop$ ls
Bosque_order            RAID                    lab_documentation
HSCamera                Stereoscope             laptop
```

  `cd .` or `cd $HOME` moves you to the home directory.

```
CPSC-WALDROP-MBP:~ waldrop$ cd $HOME
CPSC-WALDROP-MBP:~ waldrop$ ls -a
.                               .oracle_jre_usage
..                              .rstudio-desktop
.CFUserTextEncoding             .ssh
.DS_Store                       .subversion
.RData                          .thumbnails
.Rapp.history                   .visit
.Renviron                       Applications
.Rhistory                       Creative Cloud Files
.Trash                          Desktop
.Xauthority                     Documents
.bash_history                   Downloads
.bash_profile                   Dropbox
.bash_profile~                  Dropbox (Chapman)
.bash_sessions                  Dropbox (Personal)
.cache                          Google Drive File Stream
.config                         Library
.cups                           Movies
.dropbox                        Music
.emacs.d                        OrthoSample.png
.gitconfig                      Pictures
.idlerc                         Public
.ipython                        get-pip.py
.jupyter                        just_ARandQ.csv
.local                          visit0000.curve
.matplotlib                     visitlog.py
```

Manipulating Files and Folders

At some point, it will be necessary to create, delete, copy or move files and folders using the command line. Here are a list of helpful commands:

- `mkdir newdirectoryname`: make a new directory.

```
[CPSC-WALDROP-MBP:IBAMR waldrop$ mkdir ANewFolder                              ]
[CPSC-WALDROP-MBP:IBAMR waldrop$ ls                                            ]
 ANewFolder                          IBAMR_install_v2016
 Bosque-IBAMR-Install-clang.rtf      Notes_on_install.aux
 Bosque-IBAMR-Install_Notes.txt      Notes_on_install.log
```

- `cp filename newcopyfilename`: copy one file to another with a new name that you must specify.
  To copy the file to a new location (not in the current directory):
  `cp filename /location/newcopyfilename`.

```
[CPSC-WALDROP-MBP:IBAMR waldrop$ cp Bosque-IBAMR-Install_Notes.txt ANewFolder/Ins]
tall_Notes.txt
[CPSC-WALDROP-MBP:IBAMR waldrop$ ls ANewFolder/                                ]
Install_Notes.txt
```

- `mv filename directoryname`: move a file from current directory to another directory.
  `mv filename newfilename`: also use mv to rename files.

- `rm filename`: remove a file.
  For removing directories, you must add the `-r` flag: `rm -r directoryname`.

```
[CPSC-WALDROP-MBP:IBAMR waldrop$ rm -r ANewFolder/                             ]
[CPSC-WALDROP-MBP:IBAMR waldrop$ ls                                            ]
 Bosque-IBAMR-Install-clang.rtf      Notes_on_install.log
 Bosque-IBAMR-Install_Notes.txt      Notes_on_install.out
```

  The r stands for "recursive", you are telling bash to find all the files in the directory and delete them and any other files in any other folders within the folder you are deleting.
  For removing certain protected files, you must add the `-f` "force" flag: `rm -f special.file`.
  BEWARE of the `rm` command: it does NOT send things to the trash can, it deletes them permanently! No going back!

Editing and Manipulating Individual Files

Bash comes with several different text editors that work within the Bash shell to help you editing files directly. They work best on files that don't carry encryption (like .xlsx, .docx, or .mat files), i.e. that are regular text files. The most common editors are:

- emacs: terminal editor emacs. It gives you editing power within the terminal. Scroll up, down, and sideways using arrows, to save: `clt+s` and `clt+x`, to exit: `clt+x` and `clt+c`. emacs comes with a huge variety of commands, for more information, see this reference card: https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf (Note: opens a PDF.)

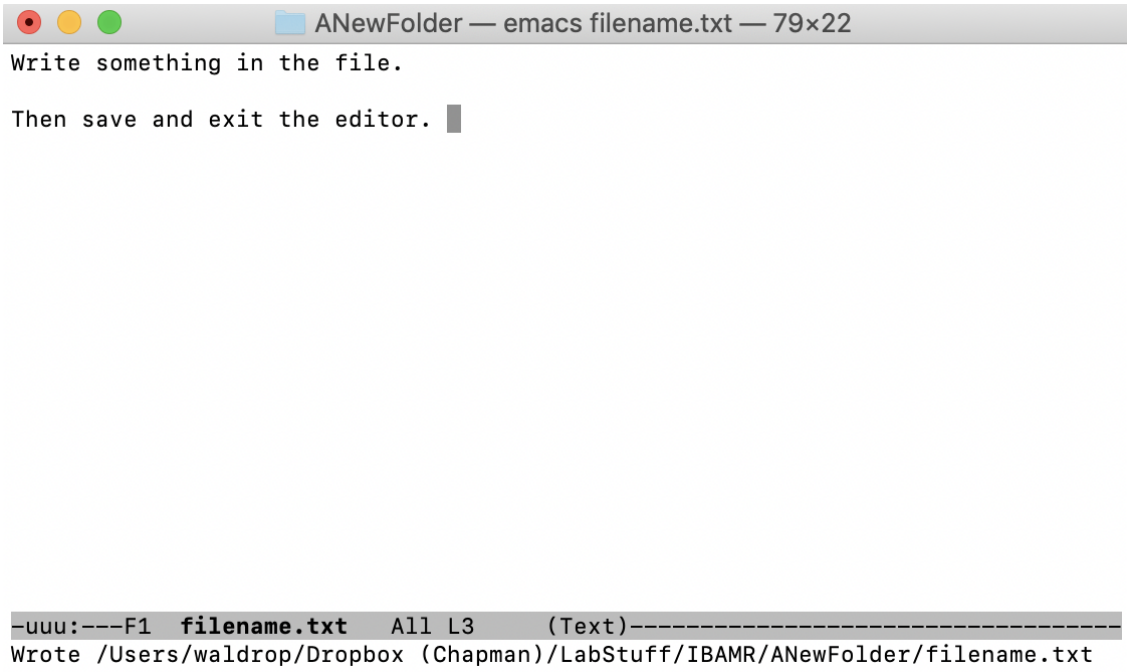
- vim: terminal editor vim. It also gives you editing power within the terminal with a few more bells and whistles. If you're up for learning it, it can be useful, but there is more of a learning curve than with emacs. For more information, see: https://vim.rtorr.com/.

- nano: terminal editor nano. Yet another editor that is less popular than the other two, but still plenty useful. For more information, see: https://www.nano-editor.org/dist /latest/cheatsheet.html.
- atom: terminal editor by the folks at Github. For more information: https://atom.io/.

I don't care which one you use, but be aware, I use emacs so therefore know it the best. I may not be able to help you troubleshoot other editors.

Here are some helpful commands for editing and manipulating files (mostly in emacs, but you can substitute the other editors as you wish):

- `touch filename.txt`: Touch will create a new, empty file.
- `emacs filename.txt`: will open the new file in emacs. Substitute "vim" or "nano" for emacs to use the other editors.



- `cat filename.txt`: cat will print out the entire contents of the file in the shell window.

```
CPSC-WALDROP-MBP:ANewFolder waldrop$ cat filename.txt
Write something in the file.

Then save and exit the editor.
CPSC-WALDROP-MBP:ANewFolder waldrop$
```

- For very long files, as this example file allpara.txt, there are a few commands that are useful.
  `head allpara.txt`: Head will print the first 10 lines of any file.
  `tail allpara.txt`: Tail will print the last 10 lines of any file. This command is especially

helpful for checking the IBAMR log files, tail will give you the log output's last few lines so that you can see where the simulation is and if it has finished successfully (if not, it will likely give you the error).
You can also specify more or fewer lines to be printed by using the -n flag: `head -n 20 allpara.txt` will print the first 20 lines of the file.

```
CPSC-WALDROP-MBP:ANewFolder waldrop$ head allpara.txt
0.35191583      0.675   1.25
0.4342759       0.51622868      1.25
0.4342759       0.675   0.8169873
0.4342759       0.675   1.25
0.4342759       0.675   1.6830127
0.4342759       0.83377132      1.25
0.56440976      0.46198592      1.25
0.56440976      0.51622868      0.8169873
0.56440976      0.51622868      1.25
0.56440976      0.51622868      1.6830127
CPSC-WALDROP-MBP:ANewFolder waldrop$
```

- `more allpara.txt`: More is very helpful for looking through the length of a longer file. It operates the same as the man environment in that you can scroll through the file without printing it all out. The down arrow key will advance the file one line at a time. The up arrow key will advance back towards the beginning one line at a time. The letter key d will advance by paging down and f will reverse by paging up. The b letter key will take you to the beginning of the file. The q letter key will exit the environment back to the command line.

- `cut`: Cut will cut a file along columns or whatever else you specify. Use the -f flag to tell it what column to cut by. The -d flag will specify a character to cut by other than a tab, which is default. As an example:
`cut -f 2 allpara.txt`: prints out the second column of numbers from allpara.txt. Here is a screenshot of the top few lines:

```
CPSC-WALDROP-MBP:ANewFolder waldrop$ cut -f 2 allpara.txt
0.675
0.51622868
0.675
0.675
0.675
```

- `grep`: a search tool that will find regular expressions. Grep will return the entire lines of files that have the specified characters. It can be very powerful when searching huge files.
`grep 0.675 allpara.txt`: returns all the lines that have the value 0.675. Here is a screenshot of the top few lines:

- `awk`: Awk is super powerful and awesome, but I won't get into it here. Look it up if you have the need to edit specific lines in files!

- Special operators: there are a variety of special operators that will help combine commands in powerful ways. Here are a few:

```
CPSC-WALDROP-MBP:ANewFolder waldrop$ grep 0.675 allpara.txt
0.35191583      0.675   1.25
0.4342759       0.675   0.8169873
0.4342759       0.675   1.25
0.4342759       0.675   1.6830127
0.56440976      0.675   0.6690525
```

- `>`: redirects output into a new file.
  `cut -f 2 allpara.txt > pamp.txt`
- `>>`: same as `>` except the output file, if it exists, will be appended by the new output.
- `>|`: same as `>` except the output file will be overwritten by the new output.
- `|`: pipes one command into another. Bash will run both commands, one after the other, and show you the output of both.
  `ls -l | grep IBAMR` lists the files in a directory and returns only those with "IBAMR" in the name:

```
CPSC-WALDROP-MBP:IBAMR waldrop$ ls -l | grep IBAMR
-rw-r--r--@  1 waldrop  staff    11799 Dec 13  2016 Bosque-IBAMR-Install-clang.r
tf
-rw-r--r--@  1 waldrop  staff     9199 Dec 13  2016 Bosque-IBAMR-Install_Notes.t
xt
-rw-r--r--@  1 waldrop  staff    12445 Dec 21  2016 IBAMR-Bridges.rtf
-rw-r--r--@  1 waldrop  staff    10876 Jun 19  2017 IBAMR-Bridges2_gcc.rtf
-rw-r--r--@  1 waldrop  staff    16603 Apr 10 16:50 IBAMR-Bridges2_intel.rtf
-rw-r--r--@  1 waldrop  staff    14782 Apr 12 17:45 IBAMR-Bridges3.rtf
-rw-r--r--@  1 waldrop  staff    10630 Sep 15  2018 IBAMR2016_Dogwood.rtf
-rw-r--r--@  1 waldrop  staff   143916 Oct  3  2016 IBAMR_Install_August_2016.pd
f
drwxr-xr-x@  9 waldrop  staff      288 Aug 20  2019 IBAMR_install_v2016
-rw-r--r--@  1 waldrop  staff    14207 Jun 15  2017 Shannons_IBAMR_Install_Notes
_Mac.rtf
CPSC-WALDROP-MBP:IBAMR waldrop$
```

- `;`: the semicolon will allow you to issue two commands in one line, with command 1 running and completing before running command2:
  `command1 ; command2`
- `&`: similar to the semicolon, the ampersand will allow you to issue two commands; however, command 1 will run in the background and command 2 will run in the foreground, starting before command 1 finishes.
  `command1 & command2`

These aren't all the possible commands, just the ones I find most useful on a day-to-day basis.

Other Useful Tips and Tricks

- `~`: This is called the tilde expansion and stands for the home directory. Using it is a shortcut like `$HOME`.
- `*`: The asterisk is called a wildcard in a file or directory name. Bash will match files containing any phrase and a wildcard standing for any other characters:

  Similarly, adding a question mark `?` will match only a single character in that position.

```
CPSC-WALDROP-MBP:IBAMR waldrop$ ls IBAMR*.rtf
IBAMR-Bridges.rtf               IBAMR-Bridges3.rtf
IBAMR-Bridges2_gcc.rtf          IBAMR2016_Dogwood.rtf
IBAMR-Bridges2_intel.rtf
CPSC-WALDROP-MBP:IBAMR waldrop$
```

- \: a backslash before a character that usually means something else will allow that character to retain its literal value. For example, a space usually breaks up a command and indicates the start of arguments, so changing directories with names that include spaces is difficult. Using a backslash makes this possible:

```
CPSC-WALDROP-MBP:~ waldrop$ cd Dropbox\ \(Chapman\)/
CPSC-WALDROP-MBP:Dropbox (Chapman) waldrop$ pwd
/Users/waldrop/Dropbox (Chapman)
CPSC-WALDROP-MBP:Dropbox (Chapman) waldrop$
```

  In this example, both the space in the directory name and the two parentheses need backslashes in order to be read properly.

- `clt + c`: kill a process running in the current window (make it stop doing the thing).

- Up/down arrows: scrolls through previously entered commands (useful when you want to repeat a command without typing it out completely).

- Tab key: will complete a directory or filename if it exists (useful if you are too lazy or make too many mistakes when typing out a filename). Only works for unique file names. Hitting tab twice quickly will give you a list of options for completing your selection as it is entered so far.

- To access the help documentation for any command, enter `man` (stands for "manual") followed by the function:
  `man cd`
  To exit the help documentation, hit the q key.

There is a lot more to Bash than what's covered here, but these should help get you started!