

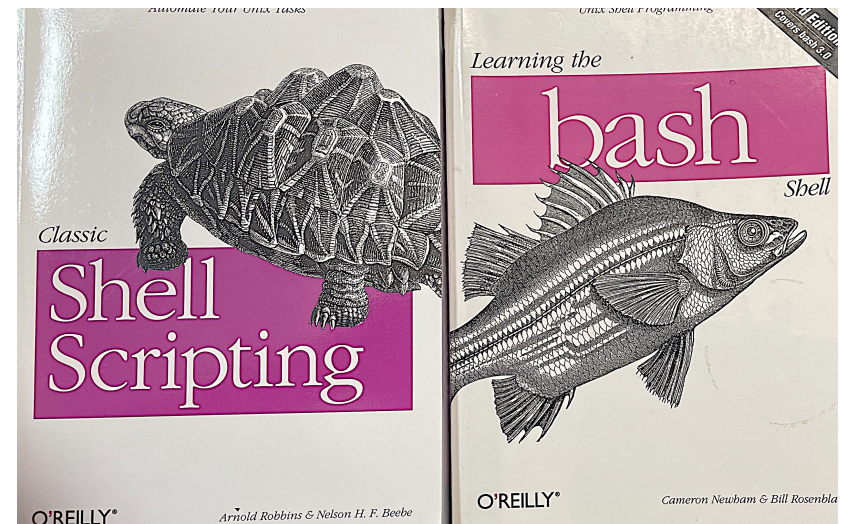
Lecture: Bash Language Part I

Photo of a new
Linux user learning
the command line



Suggested Literature

Both books are available at
<https://learning.oreilly.com>



GUI vs Terminal

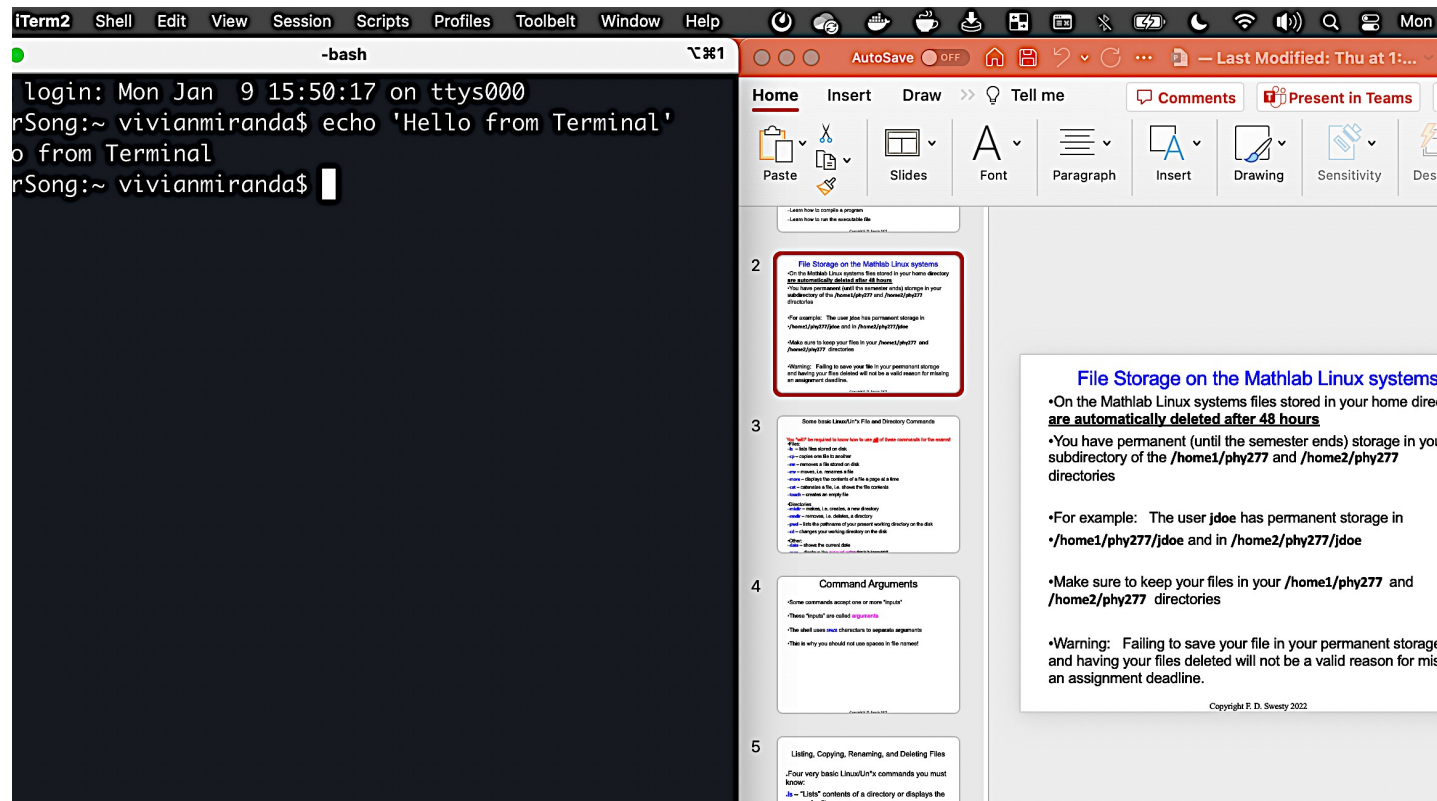
Why use the shell terminal in 2023?

Phones have only GUI! Isn't the terminal too 1970s?

The key factor
is **automation**.

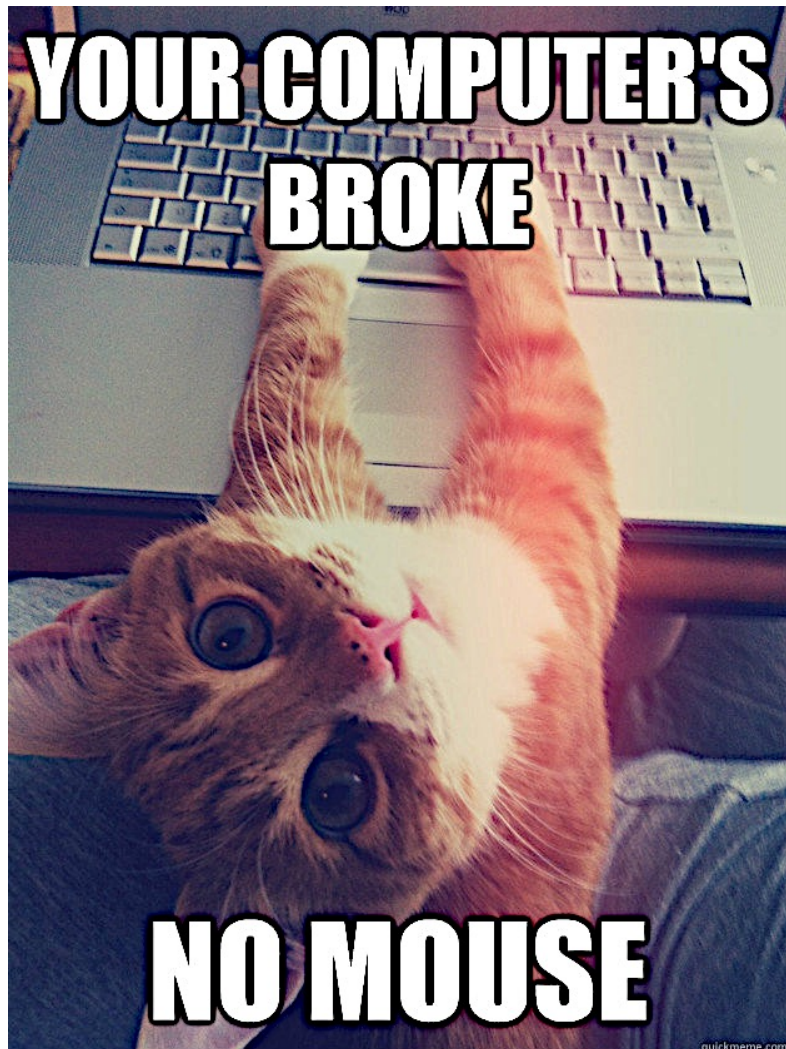
In research, we
sometimes
need **scripts** to
automate file
handling

Bash is powerful. **OPEN THE TERMINAL**



GUI vs Terminal

When you need **bash**, understanding the following lectures can save you a lot of time.



Personal laptop /
Local server



Via SSH to access
command line shell



Remote Server

Most servers don't have VNC to allow a remote GUI. Even with VNC, a GUI requires a lot of bandwidth.

We must be comfortable with terminals

What type of automation?

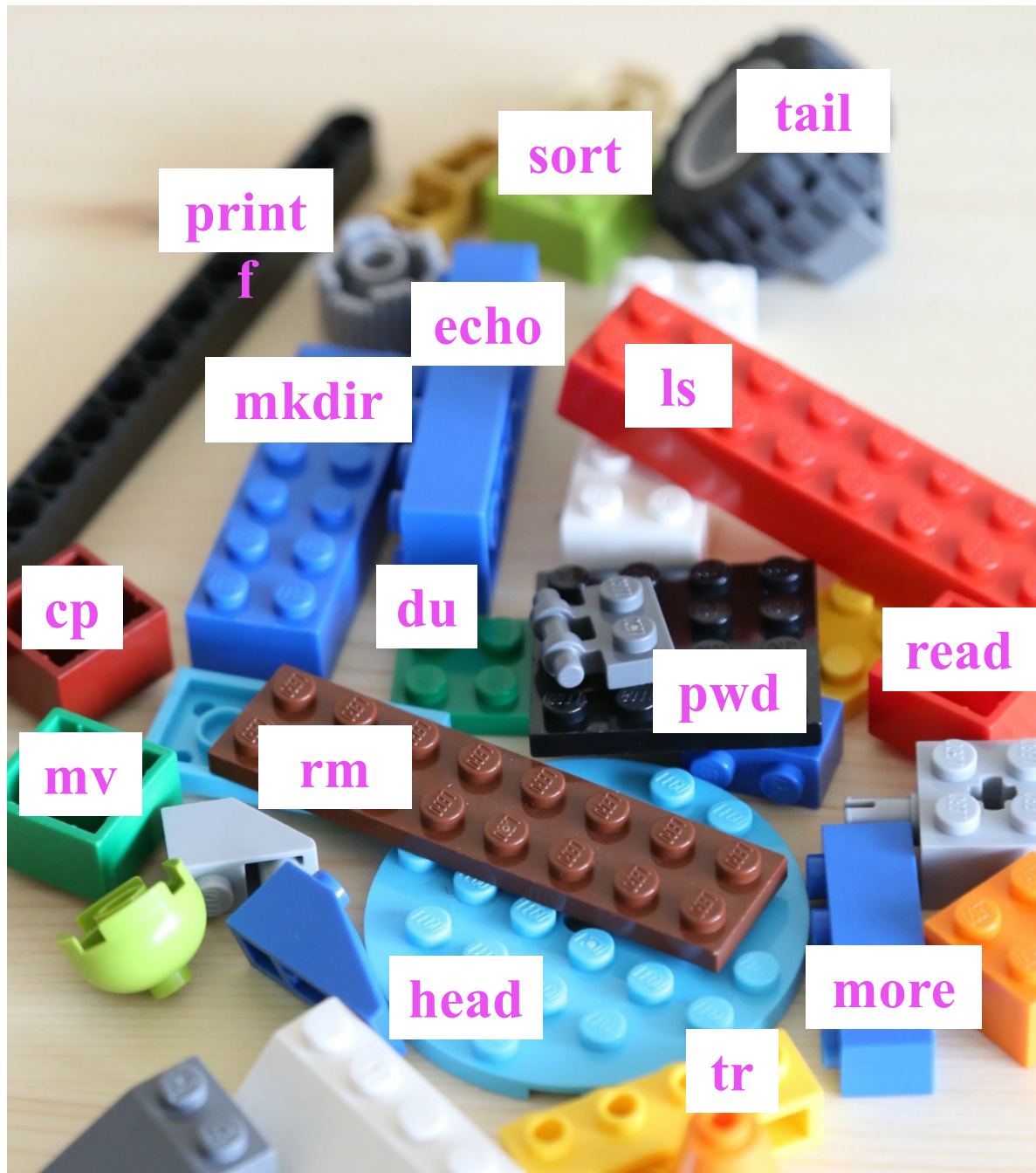
Dealing with strings is where bash shines



We will write a program that computes in $< 1s$

- How many unique words were written in Hamlet
- Show the least frequent words in Hamlet
- Show the most frequent words in Hamlet
- How many words were used just once?

Piping: bash scripts is like playing Lego



We have many small pieces (programs).

From “gluing” programs, we build pipelines.



Many shell commands are executable programs

```
*****
*
* Welcome to Docker Whovian-PHY504 docker image. I hope this VM
* will help you learn shell, C and C++
*
* This image also contains multiple gcc compilers (gcc 13 / 14)
* They contain new features about Fortran 2008 / C++23. Have fun!
*
*****

You are running this container as user with ID 1005 and group 1005,
which should map to the ID and group for your user on the Docker host. Great!

whovian@phy504:~$ gcc --version
gcc (Ubuntu 14.2.0-4ubuntu2) 14.2.0
Copyright (C) 2024 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

whovian@phy504:~$ g++ --version
g++ (Ubuntu 14.2.0-4ubuntu2) 14.2.0
Copyright (C) 2024 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

whovian@phy504:~$ bash
whovian@phy504:~$ cd ../host/
~$PHY504.pptx  .DS_Store      oldDocker/      old_notes/      PHY504.pptx
whovian@phy504:~$ cd ../host/
```

Open the
Linux
terminal!

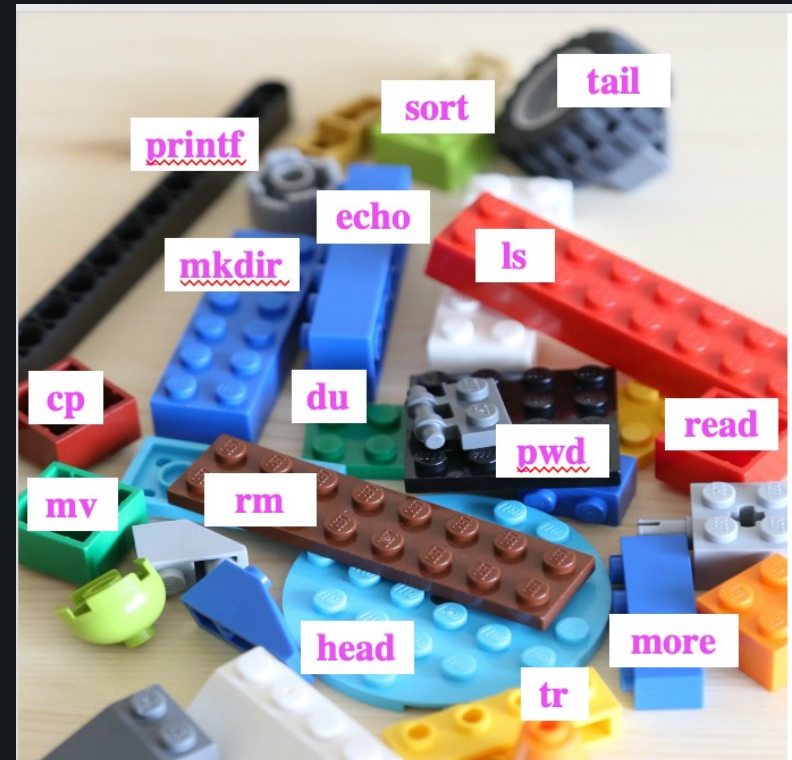
I will use
Docker to
emulate
Linux on
macOS.

Many shell commands are executable programs

location of many system-wide programs

```
whovian@phy504:/bin$ cd /bin
whovian@phy504:/bin$ ls -l bash cat ls grep echo sort pwd head rm cp mv tr more
```

-rwxr-xr-x	1	root	root	1183448	Apr	18	2022	bash
-rwxr-xr-x	1	root	root	43416	Sep	5	2019	cat
-rwxr-xr-x	1	root	root	153976	Sep	5	2019	cp
-rwxr-xr-x	1	root	root	39256	Sep	5	2019	echo
-rwxr-xr-x	1	root	root	199136	Jan	29	2020	grep
-rwxr-xr-x	1	root	root	47480	Sep	5	2019	head
-rwxr-xr-x	1	root	root	142144	Sep	5	2019	ls
-rwxr-xr-x	1	root	root	43160	Feb	7	2022	more
-rwxr-xr-x	1	root	root	149888	Sep	5	2019	mv
-rwxr-xr-x	1	root	root	43352	Sep	5	2019	pwd
-rwxr-xr-x	1	root	root	72056	Sep	5	2019	rm
-rwxr-xr-x	1	root	root	117376	Sep	5	2019	sort
-rwxr-xr-x	1	root	root	51544	Sep	5	2019	tr



Other locations of many executable programs (including shell) are **/usr/bin** and **/usr/local/bin**

Get the location of shell programs

```
whovian@phy504:/bin$ type -a echo
echo is a shell builtin
echo is /usr/bin/echo
echo is /bin/echo
whovian@phy504:/bin$ type -a more
more is /usr/bin/more
more is /bin/more
whovian@phy504:/bin$ type -a cp
cp is /usr/bin/cp
cp is /bin/cp
whovian@phy504:/bin$ type -a mv
mv is /usr/bin/mv
mv is /bin/mv
```



- **type -a name** displays all locations containing the executable name
- Two locations may hold copies of the same program (here, **/bin** is just a **symbolic link** of **/usr/bin**).
- How does the system know where to look them up (and the priority of execution)?

```
whovian@phy504:/bin$ echo "$PATH"
/opt/conda/bin:/usr/local/nvidia/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Shell programs are written in C

```
int
main (int argc, char **argv)
{
    int c;
    bool ok;
    bool make_backups = false;
    char *backup_suffix_string;
    char *version_control_string = NULL;
    struct cp_options x;
    bool copy_contents = false;
    char *target_directory = NULL;
    bool no_target_directory = false;

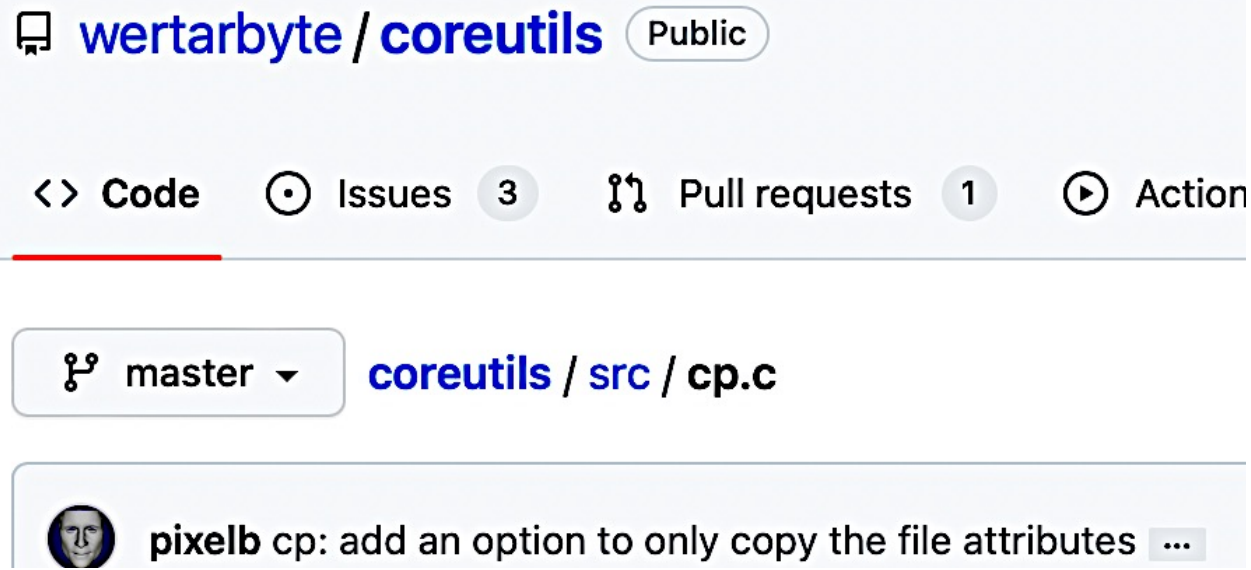
    initialize_main (&argc, &argv);
    set_program_name (argv[0]);
    setlocale (LC_ALL, "");
    bindtextdomain (PACKAGE, LOCALEDIR);
    textdomain (PACKAGE);

    atexit (close_stdin);

    selinux_enabled = (0 < is_selinux_enabled ());
    cp_option_init (&x);

    /* FIXME: consider not calling getenv for SIMPLE_BACKUP_SUFFIX unless
       we'll actually use backup_suffix_string. */
    backup_suffix_string = getenv ("SIMPLE_BACKUP_SUFFIX");

    while ((c = getopt_long (argc, argv, "abdfHilLnprst:uvxPRS:T",
                           long_opts, NULL))
           != -1)
```



C language is the base
of Linux kernel and
shell scripts

cp code is >1000 lines!
Shell can do so much
with just a few lines

EDITORS – EMACS NANO and VIM

- Popular editors on Unix: **Emacs** & **vim** & **nano**.
- Editor functions are available as keyboard commands on a command-line interface.
- They enable editing while remotely logged in.
- I use **nano** (simpler). Not all HPC systems have **nano**

VIM

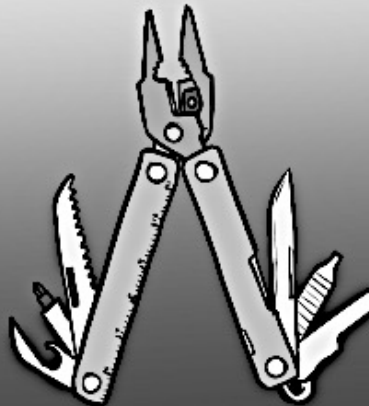
usable in just about any environment.

does one thing, well.



EMACS

flexible, customizable, and packed with every feature known to man.



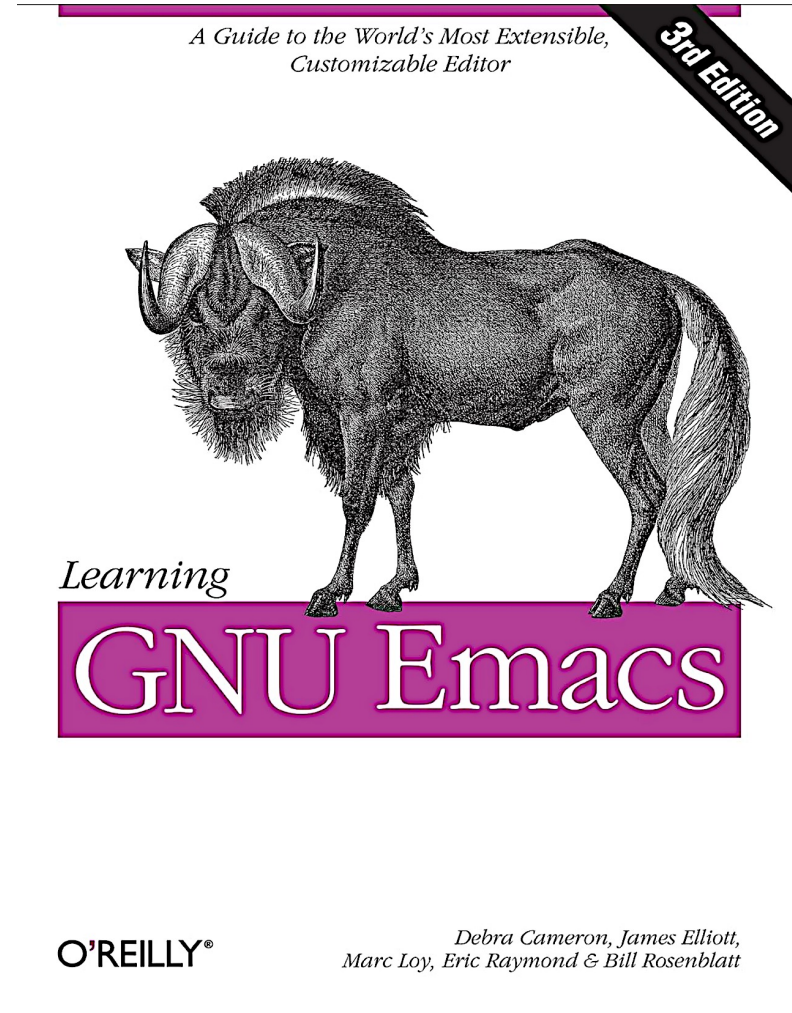
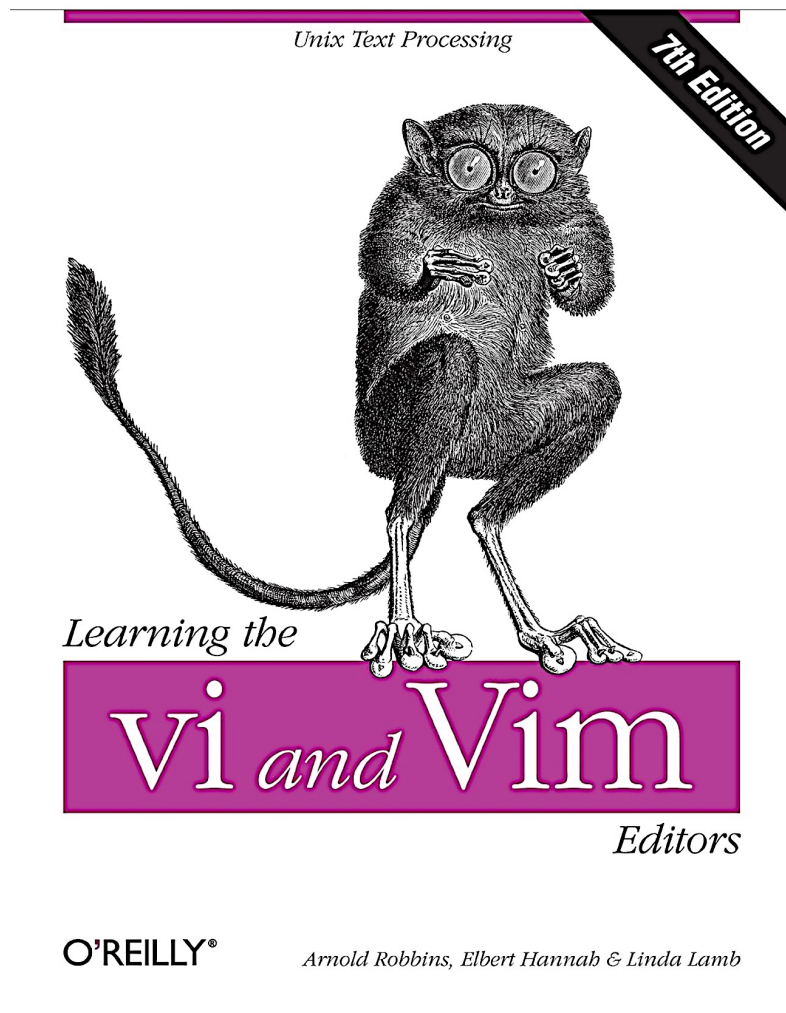
NANO

mostly used by people who do not know what they are doing; or psychopaths.



EDITORS – EMACS NANO and VIM

- Popular editors on Unix: **Emacs** & **vim** & **nano**.
- All editor functions are available as keyboard commands on a command-line interface.



EDITORS – EMACS NANO and VIM

- Popular editors on Unix: **Emacs** & **vim** & **nano**.
- Be careful with the rabbit hole (for now)! You can spend months learning these editors.



Interview with an Emacs Enthusiast [Colorized]



Programmers are also human
216K subscribers

Subscribe

38K



Interview with a VIM Enthusiast



Programmers are also human
216K subscribers

Subscribe

33K



Share

Clip

Save



Open GNU nano

```
com.docker.cli 1
GNU nano 4.8    New Buffer    Modified
Hi, this is GNU nano editor
There are some instructions below on how to manipulate this file.
If you want to exit the file, then press CTRL+X.
Then decide if you want to save the file or not!

Nano is not just black and white - there is a special line you
can add so that nano knows how to place colors (depending on
the language)

^G Get Help    ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify
^X Exit        ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell
```

HW: Practice working with GNU nano

GNU nano is not just B&W



```
com.docker.cli
GNU nano 4.8 sless.sh
#!/bin/bash
s1="HELLO WORLD"
s2="HELLO"

if [[ "$s1" < "$s2" ]]; then
    echo "s1 is less than s2"
elif [[ "$s1" > "$s2" ]]; then
    echo "s2 is less than s1"
else
    echo "s1 and s2 are equal"
fi

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell
```

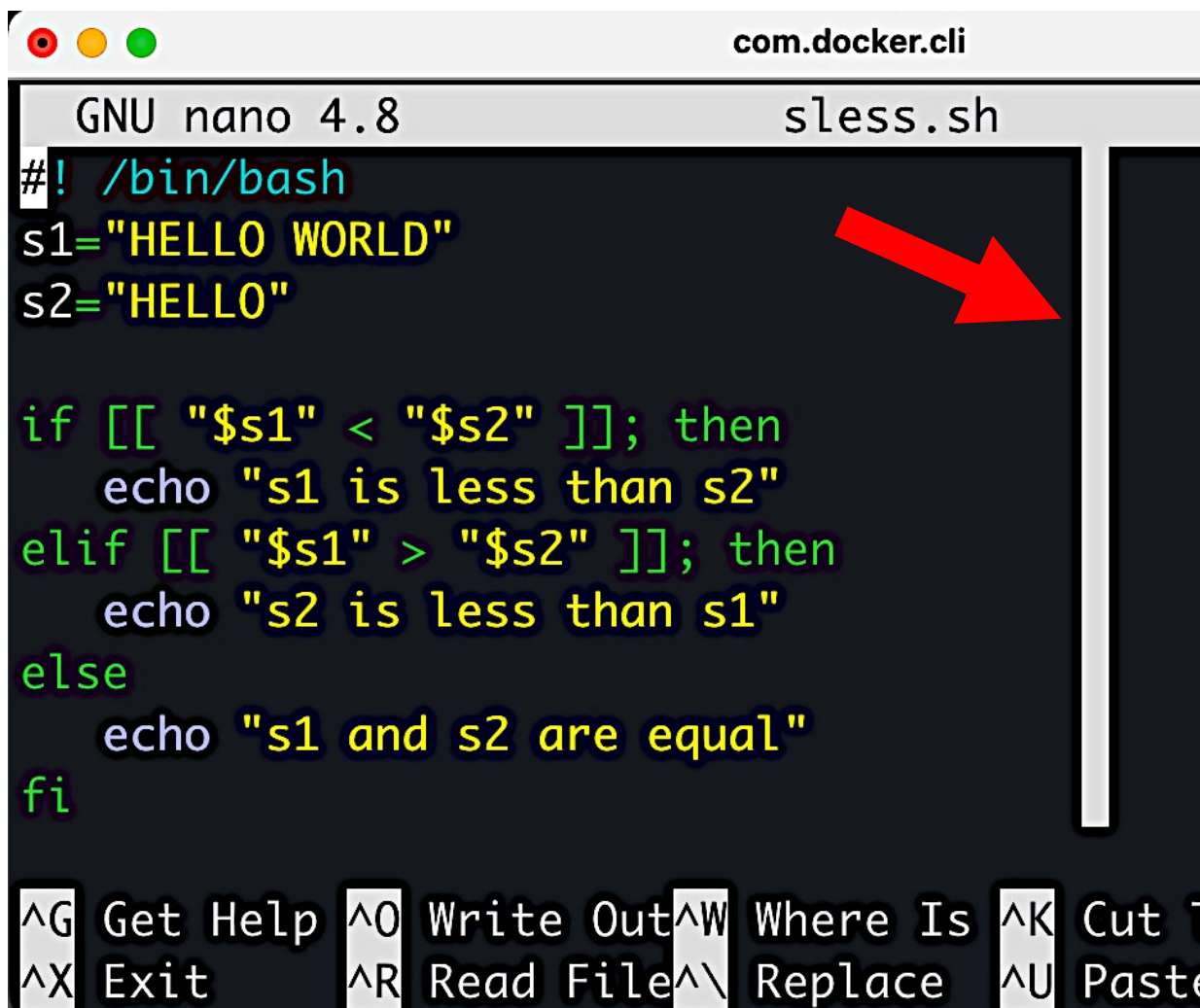
Magic first line that tells the system (and GNU nano) the file is a bash script

How to limit page size in nano?

Add a mark with **-J num** option (just a mark)

Options: core features that make **bash** powerful

```
whovian@phy504:~/host$ nano -J 40 sless.sh
```



```
com.docker.cli
GNU nano 4.8 sless.sh
#!/bin/bash
s1="HELLO WORLD"
s2="HELLO"

if [[ "$s1" < "$s2" ]]; then
    echo "s1 is less than s2"
elif [[ "$s1" > "$s2" ]]; then
    echo "s2 is less than s1"
else
    echo "s1 and s2 are equal"
fi

^G Get Help ^O Write Out ^W Where Is ^K Cut T
^X Exit ^R Read File ^\ Replace ^U Paste
```

In research, avoid using Nano with an SFTP connection to the server.

In many places (SBU), 2-factor authentication is required for **sftp**.

man & **--help** commands are your friend!

If you don't know what a bash command does, use the **man** command. If you understand what a command does but do not know all its options, use **--help**.

```
whovian@phy504:~$ type --help
type: type [-afptP] name [name ...]
    Display information about command type.

For each NAME, indicate how it would be interpreted if used as a
command name.

Options:
  -a      display all locations containing an executable named NAME;
          includes aliases, builtins, and functions, if and only if
          the '-p' option is not also used
  -f      suppress shell function lookup
  -P      force a PATH search for each NAME, even if it is an alias,
          builtin, or function, and returns the name of the disk file
          that would be executed
  -p      returns either the name of the disk file that would be executed,
          or nothing if `type -t NAME' would not return `file'
  -t      output a single word which is one of `alias', `keyword',
          `function', `builtin', `file' or `', if NAME is an alias,
          shell reserved word, shell function, shell builtin, disk file,
          or not found, respectively

Arguments:
  NAME      Command name to be interpreted.

Exit Status:
Returns success if all of the NAMEs are found; fails if any are not found.
```

On macOS, not all
bash programs have
the option (**ls** does
not have it).

On Docker, most
commands do have it
--help

man & --help commands are your friend!

MacOS vs Docker and the --help

```
Last login: Mon Jan 27 10:19:55 on ttys001
RiverSong:~ vivianmiranda$ ls --help
ls: unrecognized option '--help'
usage: ls [-@ABCFGHILOPRSTUWabcdefghiklmnopqrstuvwxy1%,] [--color=when] [-D format] [file ...]
RiverSong:~ vivianmiranda$
```



```
whovian@phy504:~$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
-a, --all                do not ignore entries starting with .
-A, --almost-all        do not list implied . and ..
    --author              with -l, print the author of each file
-b, --escape              print C-style escapes for nongraphic characters
    --block-size=SIZE    with -l, scale sizes by SIZE when printing them;
                        e.g., --block-size=M; see SIZE format below

-B, --ignore-backslash  ng with ~
-c                        me (time of last
                        n);
                        name;
                        first

-C                        below
                        their contents
                        cs' dired mode
                        der
                        to entries WHEN
                        '*'
                        -x, long -l,
                        rtical -C

--color[=WHEN]
-d, --directory
-D, --dired
-f
-F, --classify[=W
    --file-type
    --format=WORD
```



Different systems may have different rules. macOS is based on the BSD (Berkeley Software Distribution) Unix, while Linux adopts the GNU Core Utils as the bash distribution. They are not 100% compatible.

```
whovian@phy504:~/host$ bash --help
GNU bash, version 5.0.17(1)-release-(x86_64-pc-linux-gnu)
Usage: bash [GNU long option] [option] ...
       bash [GNU long option] [option] script-file ...
```

GNU long options:

- debug
- debugger
- dump-po-strings
- dump-strings
- help
- init-file
- login
- noediting
- noprofile
- norc
- posix
- pretty-print
- rcfile
- restricted
- verbose
- version

Shell options:

- ilrsD or -c command or -O shopt_option (invocation only)
- abefhkmnptuvxBCHP or -o option

Type `bash -c "help set"' for more information about shell options.

Type `bash -c help' for more information about shell builtin commands.

Use the `bashbug' command to report bugs.

bash home page: <<http://www.gnu.org/software/bash>>

General help using GNU software: <<http://www.gnu.org/gethelp/>>

Even the word
bash (which is a
program) accept
--help

.bashrc, .bash_profile. What are they?

- They are files that initialize your environment.
- What is the diff between .bashrc and .bash_profile?
 - .bashrc loads at the beginning of every session
 - .bash_profile loads only at the OS startup
- .bashrc is an excellent resource for creating aliases of complicated but frequently used commands

```
alias ocelote='ssh -o ServerAliveInterval=60 vivianmiranda@hpc.arizona.edu'  
alias arizona='ssh -o ServerAliveInterval=60 vivianmiranda@amypond.cosmoatuofa.com'  
alias arizona2='ssh -o ServerAliveInterval=60 vivianmiranda@128.196.208.65'  
  
alias amypond0='ssh -X -o ServerAliveInterval=60 vivianmiranda@192.168.1.5'  
alias amypond='ssh -XY -o ServerAliveInterval=60 vivianmiranda@129.49.63.5'  
alias amypond1='ssh -o ServerAliveInterval=60 grads@129.49.63.5'  
alias amypond2='ssh -o ServerAliveInterval=60 undergrads@129.49.63.5'  
alias amypond3='ssh -o ServerAliveInterval=60 postdocs@129.49.63.5'
```

Basic File Manipulation

Creating New Directories

New directories can be created with the **mkdir** (make directory)

Syntax:

mkdir *<name_of_new_directory>*

Example:

mkdir codes

Creates a new directory named **codes**

```
whovian@phy504:~$ mkdir --help
```

```
Usage: mkdir [OPTION]... DIRECTORY...
```

```
Create the DIRECTORY(ies), if they do not already exist.
```

```
Mandatory arguments to long options are mandatory for short options
```

-m, --mode=MODE	set file mode (as in chmod), not a=rwx - umask
-p, --parents	no error if existing, make parent directories as
-v, --verbose	print a message for each created directory
-Z	set SELinux security context of each created to the default type
--context[=CTX]	like -Z, or if CTX is specified then set the or SMACK security context to CTX
--help	display this help and exit

The **ls** command

ls displays all files in the current directory (with information)

- **-d** list directories, not files (**options can tweak command behavior**)
- **-S** (yes CAPS): sort files by size, largest first
- **-t** sort by modification time, newest first
- **-U** (yes CAPS) do not sort
- **-l** use long listing format (with size in bytes)
- **-h** (human-readable)! That means converting bytes to Kb, Mb, Gb

```
whovian@phy504:~/host$ ls -Shl
```

```
total 428K
```

```
-rw-r--r-- 1 whovian users 175K Jan 13 00:22 hamlet.txt
-rw-r--r-- 1 whovian users 124K Jan 12 02:35 es.txt
-rw-r--r-- 1 whovian users 556 Jan 12 21:21 compile_des_y3
-rw-r--r-- 1 whovian users 416 Jan 13 00:38 wf.sh
-rw-r--r-- 1 whovian users 391 Jan 17 16:45 employees.txt
-rw-r--r-- 1 whovian users 342 Jan 20 01:22 address_list.txt
-rw-r--r-- 1 whovian users 341 Jan 20 01:06 address_list_backup.txt
-rw-r--r-- 1 whovian users 326 Jan 17 04:02 caseloop.sh
-rw-r--r-- 1 whovian users 283 Jan 17 03:31 whileloop3.sh
```

Yes! Concatenate options

The **ls** command

ls displays all files in the current directory (with information)

- **-d** list directories, not files (**options can tweak command behavior**)
- **-S** (yes CAPS): sort files by size, largest first
- **-t** sort by modification time, newest first
- **-U** (yes CAPS) do not sort
- **-l** use long listing format (with size in bytes)
- **-h** (human-readable)! That means converting bytes to Kb, Mb, Gb

```
whovian@phy504:~/host$ ls -S
```



without -l = short

hamlet.txt	caseloop.sh	whileloop.sh	myfunc2.sh
es.txt	whileloop3.sh	hello.txt	example
compile_des_y3	whileloop2.sh	ifelse3.sh	example2
wf.sh	sless.sh	forloop.sh	ifelse2.sh
employees.txt	table.txt	ifelse.sh	portugues-numbers
address_list.txt	sorting2.txt	singers.txt	forloop2.sh
address_list_backup.txt	singers2.txt	sorting3.txt	wine.txt

The **ls** command

One Useful acronym is **latr** (long, all, by time and reverse)

```
whovian@phy504:~/host/C$ ls -latr | tail -n 10
```



-rw-r--r--	1	whovian	users	436	Jan	26	18:49	functions8.c
-rw-r--r--	1	whovian	users	274	Jan	26	22:15	functions9.c
-rwxr-xr-x	1	whovian	users	16760	Jan	26	22:15	functions9
-rw-r--r--	1	whovian	users	270	Jan	26	22:43	functions10.c
-rwxr-xr-x	1	whovian	users					ut
-rwxr-xr-x	1	whovian	users	16728	Jan	26	22:45	functions10
-rw-r--r--	1	whovian	users	282	Jan	26	22:54	functions11.c
-rwxr-xr-x	1	whovian	users	16824	Jan	26	22:54	functions11
drwxr-xr-x	86	whovian	users	2752	Jan	27	02:18	.
drwxr-xr-x	53	whovian	users	1696	Jan	27	13:34	..

What is that? Piping!

Why is it useful? Sometimes, you need to check the permissions of files you just worked on.

How to access lab
machines remotely

Bash: Power comes with responsibility.
You can erase your files with the wrong command.

Terminal with Matlab Linux systems

On the Matlab Linux systems, files stored in your home directory
are automatically deleted after 48 hours (except MySBFiles)

You have permanent storage under **MySBFiles** folder

```
vinmirandabr@portal:~$ ls
Desktop  MySBFiles  PDF
vinmirandabr@portal:~$ cd ~/MySBFiles/
vinmirandabr@portal:~/MySBFiles$ ls
vinmirandabr@portal:~/MySBFiles$
```

Keep your files in your **/home/MySBFiles** (~ short for **/home/**)

Warning: Failing to save your file in your permanent storage.

SSH Protocol: accessing your account remotely

Access Transfer of files from one machine to another can be accomplished via SSH protocol.

```
whovian@phy504:~$  
whovian@phy504:~$ ssh vinmirandabr@portal.mathlab.stonybrook.edu  
Failed to add the host to the list of known hosts (/home/whovian/.ssh/known_hosts)  
vinmirandabr@portal.mathlab.stonybrook.edu's password:  
Creating directory '/home/campus.stonybrook.edu/vinmirandabr'.  
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-136-generic x86_64)  
  
For graphical logons, the following shortcuts have been added to your  
desktop:  
  
* Virtual SINC Site      * MySBFiles      * Firefox Web Browser  
* Google Chrome          * Logout
```

Your MySBFiles area can be accessed at ~/MySBFiles. You should save your work to this area or your Google Drive area (accessible through browser). Your home directory on this computer is not shared with the computers in the Math SINC site.

SSH Protocol: accessing your account remotely

If you are using Windows OS, you can access ssh command via Docker or PSCP (PuTTY Secure Copy

```
whovian@phy504:~$ ssh vinmirandabr@portal.mathlab.stonybrook.edu
Added host to the list of known hosts (/home/whovian/.ssh/known_hosts).
vinmirandabr@portal.mathlab.stonybrook.edu's password:
Creating directory '/home/campus.stonybrook.edu/vinmirandabr'.
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-136-generic x86_64)
```



For graphical logons, the following shortcuts have been added to your desktop:

- * Virtual SINC Site
- * Google Chrome
- * MySBFiles
- * Logout
- * Firefox Web Browser

Your MySBFiles area can be accessed at ~/MySBFiles. You should save all of your work to this area or your Google Drive area (accessible through a web browser). Your home directory on this computer is not shared with the other computers in the Math SINC site.

Your home directory on this computer will be removed 48 hours after creation. Due to system maintenance there are cases where the removal will occur prior to the 48 hour interval.

SSH Protocol – transferring files with **scp**

Transfer of files from one machine to another can be done via SSH protocol using the SSH **scp** (Secure CoPy) command:

scp **user1@machine1:frompath** **user2@machine2:topath**

This syntax can be simplified if either the "from" or the "to" files are located on the current machine.

```
whovian@phy504:~$ ls ~/host  
es.txt  
whovian@phy504:~$  
whovian@phy504:~$ scp ~/host/es.txt vinmirandabr@portal.mathlab.stonybrook.edu:~/MySBFiles/
```

Use **pwd** to find out where that directory is

To copy entire directory trees instead of single files, add the option **-r**

HW. The Bandit: a wargame

This is part I of many of this fun wargame from OverTheWire.

Although OverTheWire is a reputable site, you must only run that HW inside the docker *phy504_2025_secure* container.

The VM will protect you by isolating itself from the host machine

- [The Bandit](#) wargame is aimed at absolute beginners. It will teach the basics needed to be able to play other wargames.
- This game, like most other games, is organized in levels. You start at Level 0 and try to “beat” or “finish” it. Finishing a level results in information on how to start the next level.
- You will encounter many situations in which you have no idea what you are supposed to do. **Don't panic! Don't give up!**

HW. The Bandit: a wargame

This is part I of many of this fun wargame.

- The purpose of this game is for you to learn the basics. Part of learning the basics, is reading a lot of new information. If you've never used the command line before.
- First, if you know a command but don't know how to use it, try the **manual** by entering **man <command>** (not available on Docker). For example, **man ls** to learn about the **ls** command. The **man** command also has a manual, try it! When using **man**, press q to quit.
- Second, if there is no man page, the command might be a **shell built-in**. In that case use the "**help <X>**" command. E.g. **help cd**
- Also, your favorite **search-engine** is your friend.

HW. The Bandit: a wargame levels 0/1

- **Level 0:** The goal of this level is for you to log into the game using SSH. The host to which you need to connect is bandit.labs.overthewire.org, on [port 2220](#). The username is **bandit0**, and the password is **bandit0**. Once logged in, go to the Level 1.

Commands you may need to solve level 0: **ssh**

- **Level 1:** The password for the next level is stored in a file called **readme** located in the home directory. Use this password to log into bandit1 using SSH. Whenever you find a password for a level, use SSH (on port 2220) to log into that level and continue the game.

Commands you may need to solve level 1: **ls, cd, cat, file, du, find**

Open-Source Bibliography

There are a lot of free opensource material online – including notes from Prof. Zingale

<https://zingale.github.io/phy504/>

Study them!!! They also contain homework I will adopt in this class

Latex free opensource material (hyperlink) – not complete list

- [LaTeX for Complete Novices](#)
- [Using LaTeX to Write a PhD Thesis](#)
- [LaTeX for Administrative Work](#)
- [The not so short introduction to latex \(or latex in 280 minutes\)](#)
- There is also a StackOverflow type of site just to answer latex questions
 - The TEX EXCHANGE (<https://tex.stackexchange.com>)

Open-Source Bibliography

Bash free opensource material (hyperlinks) – not complete list

- [Bash GNU reference notes](#)
- [Bash Guide for Beginners](#) (long ebook)
- [**An Introduction to the Linux Command Shell For Beginners**](#) (pocket book)

C free opensource material (hyperlinks) – not complete list

- [The GNU C Programming Tutorial](#) (awesome – I will take examples/exercises from it)
- [Stanford CS library C language pocketbook](#)
- [Hal Open Science - Modern C by Jens Gustedt](#) (awesome – I will take examples from it)

C++ free material (hyperlinks) – not complete list

- [C++ Annotations by Frank B. Brokken](#) (good – I will take examples from it)

Physics / Numerical Problems in Python (awesome!)

- [The NUMFSY project](#)

Open-Source Bibliography

Bash free opensource material (hyperlinks) – not complete list

- [Bash GNU reference notes](#)
- [Bash Guide for Beginners](#) (long ebook)
- [**An Introduction to the Linux Command Shell For Beginners**](#) (pocket book)

C free opensource material (hyperlinks) – not complete list

- [The GNU C Programming Tutorial](#) (awesome – I will take examples/exercises from it)
- [Stanford CS library C language pocketbook](#)
- [Hal Open Science - Modern C by Jens Gustedt](#) (awesome – I will take examples from it)

C++ free material (hyperlinks) – not complete list

- [C++ Annotations by Frank B. Brokken](#) (good – I will take examples from it)

Physics / Numerical Problems in Python (awesome!)

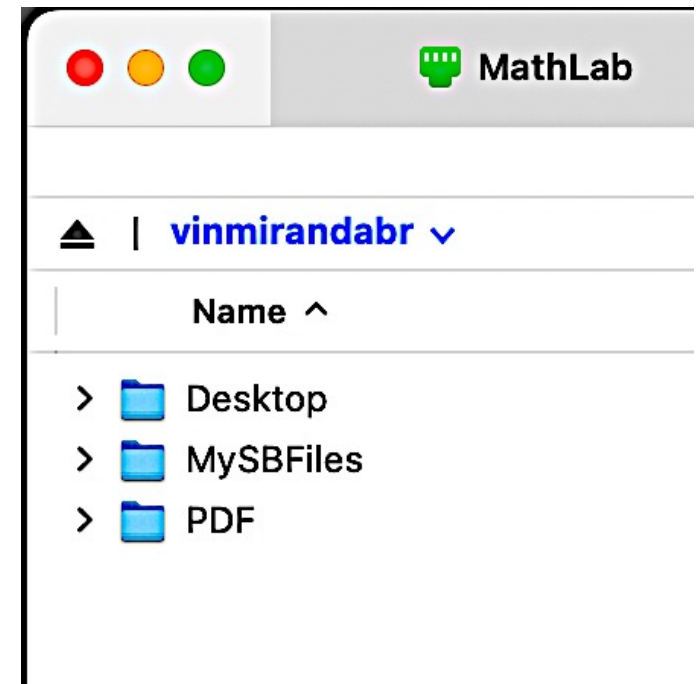
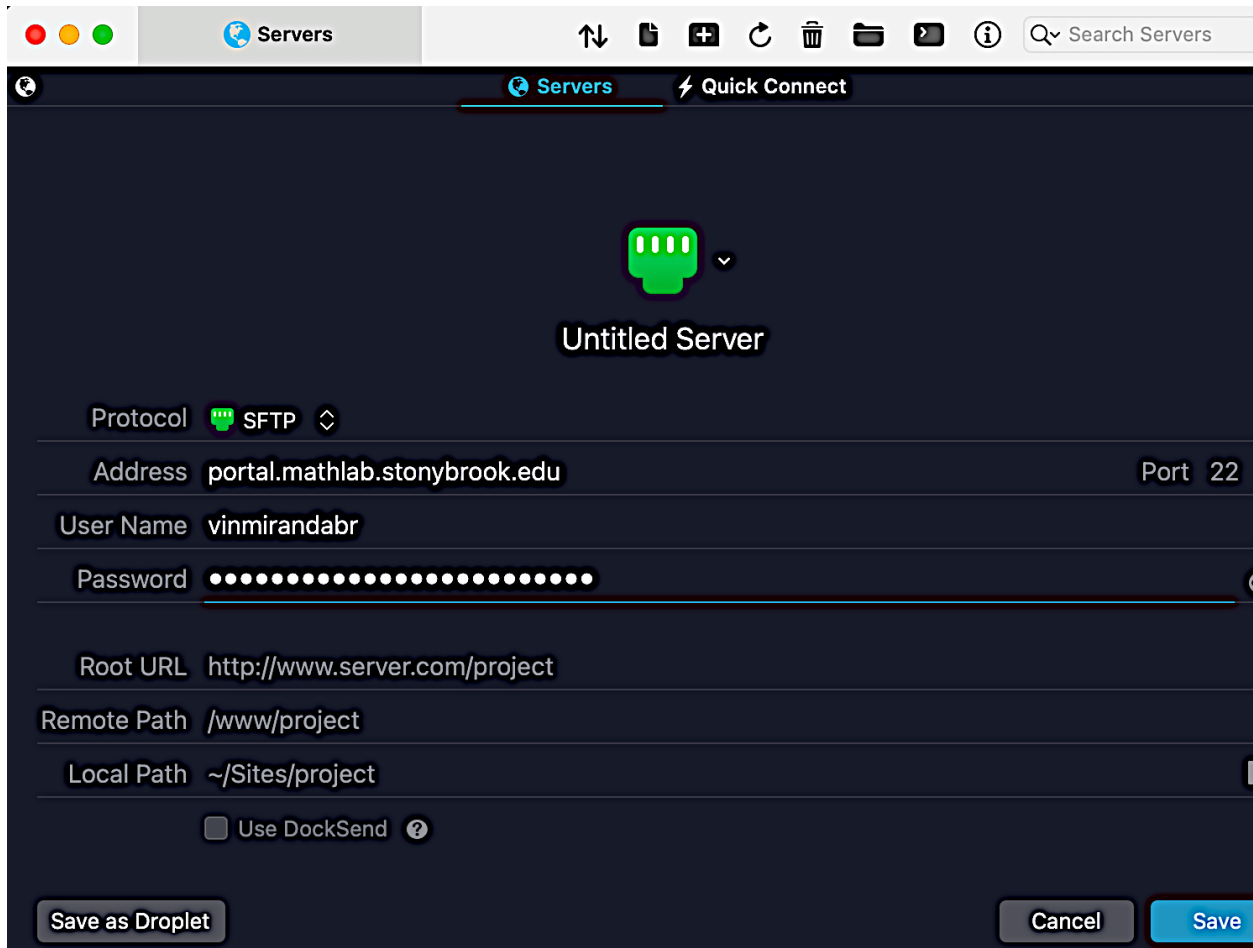
- [The NUMFSY project](#)

Additional Reading

- Medium article on what is a Docker (<https://medium.com/swlh/what-exactly-is-docker-1dd62e1fde38>)
- (Optional): a more advanced reading on how to write Docker containers: <https://medium.com/@dalibor91/containers-part-1-starting-with-docker-basics-b04b72c70129>

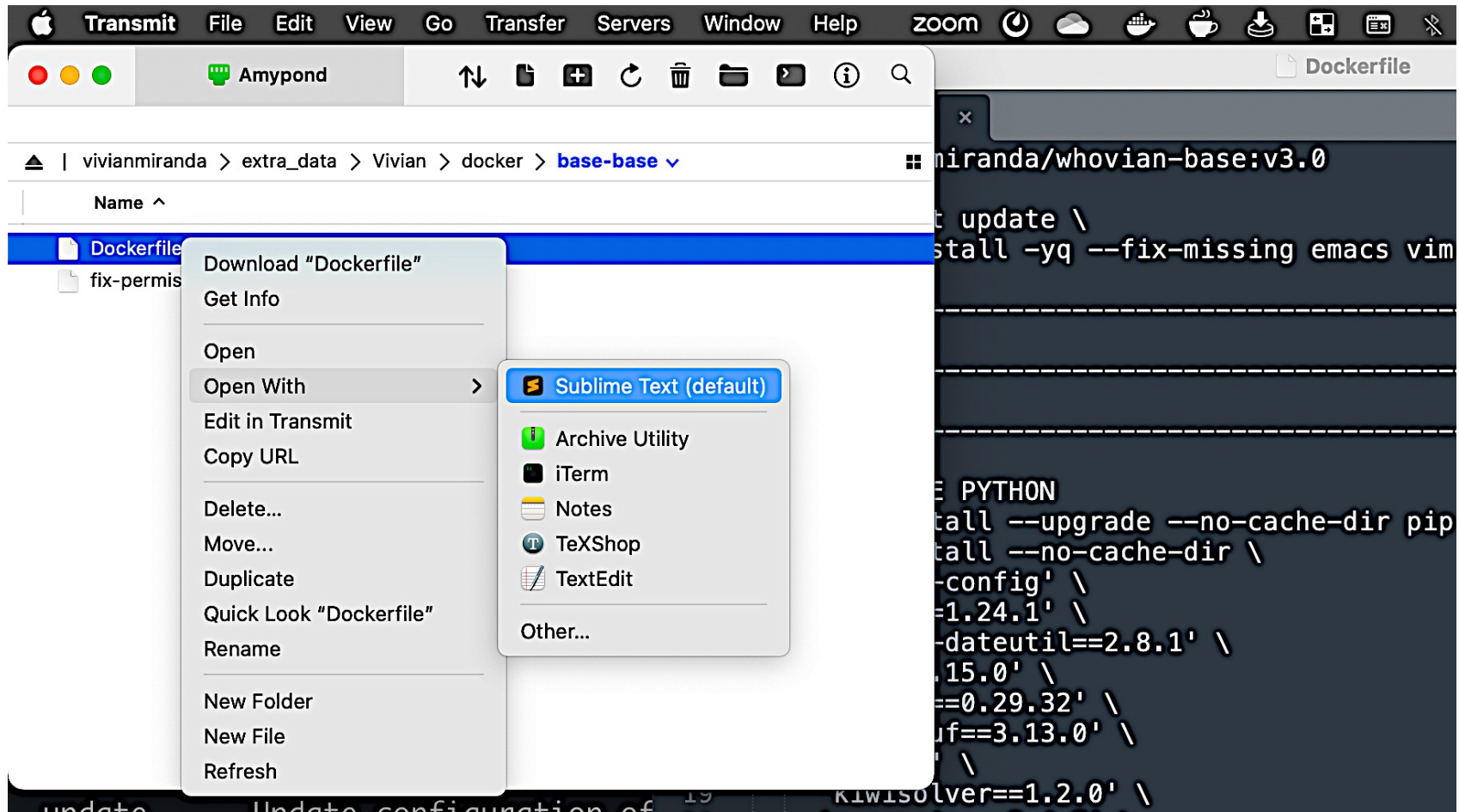
AR: GUI EDITORS – GEDIT/SUBLIME

- My favorite editor is [Sublime Text](#) (GUI editor)
- SFTP: you can open files on the server using a local GUI editor
- Free SFTP programs: [WinSCP](#) (Win), [CyberDuck](#) (MacOS)
- I use the expensive (50 dollars) [Transmit 5](#) (MacOS)



AR: GUI EDITORS – GEDIT/SUBLIME

- My favorite editor is [Sublime Text](#) (GUI editor)
- SFTP: you can open files on the server using a local GUI editor
- Free SFTP programs: [WinSCP](#) (Win), [CyberDuck](#) (MacOS)
- I use the expensive (50 dollars) [Transmit 5](#) (MacOS)



AR: Mac Users: **Bash** vs **zsh**

- We will teach bash, but Mac comes with **zsh**. You need to be comfortable switching between them. One possibility is to use the **exec bash** command (or use **DOCKER!**)

```
Last login: Mon Jan 27 08:21:24 on ttys001
RiverSong:~ vivianmiranda$ exec bash
bash-3.2$ which echo $SHELL
/bin/echo
/bin/bash
bash-3.2$
```

- If you want bash to be the default shell language, you should to
 1. Add the following lines to your hidden **.bash_profile** file (bash uses **.bashrc** to set your configuration when the terminal starts)
 2. Create the hidden file **.bashrc** with the shell command **touch ~/.bashrc** (the dot in front of the filename is important).
 3. Type **exec bash --login** in your terminal.

Seems complicated? Use **DOCKER!**

```
bash-3.2$ more ~/.bash_profile
if [ -r ~/.bashrc ]; then
    source ~/.bashrc
fi
```