ACM@UAB

Command Line Workshop

Dylan Calvin

Agenda

- What is the Terminal / CLI / Shell?
- Elements of the Terminal
- Navigating files
- Creating / Moving / Copying / Removing / Renaming files
- Editing Files (vim and nano)
- Running Programs
- Stopping / Listing Running Programs
- Special Commands
- Unix Only Commands (Mac / Linux / WSL2)

Just google it

- “How to list files in folder in terminal <Operating System>”
- “How to enter folder in terminal <Operating System”
- “How to use nano/vim”
- “How to view IP address in terminal <Operating System>”
- Etc.

Google Operators – Search Filtering

- Two Periods – Filter your search by a number range (date, cost, etc.)
  - Ex: New Construction UAB 2005..2010
- Quotation Marks – Filter your search to include an exact phrase
  - Ex: Shakespeare “to be or not to be”
- Related Pages – Show results from pages like a website
  - Ex: pizza related:pizzahut.com
  - Note: This will also include subdomains of the base address (like blog.pizzahut.com)
- Wildcards – Acts as a placeholder for words you can’t figure out
  - Ex: Amazon *
- Site – Limits search results to one website
  - Ex: segmentation fault site:stackoverflow.com
  - Argument must be a domain

Before we get started (for real this time)

- The Terminal / Command Line is what interfaces with the shell.
- All 3 Terms (Terminal, Command Line, Shell) mean the same thing, for the most part.
- I might use these interchangeably, they all mean the same thing.

- If you’re taking notes, don’t worry if I’m going too fast for you to write it down.
  - There will be a link at the end of the lecture part of the workshop where you can download a copy of these slides for reference.

ACM@UAB

Why learn how to use the Terminal?

Increased Efficiency

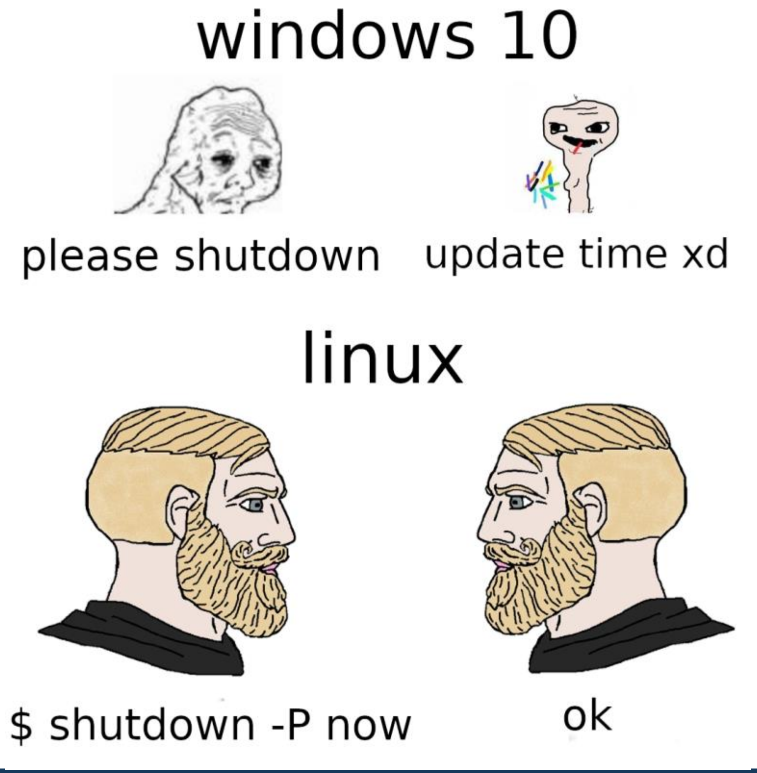
- GUI Applications take *tons* more resources to run in comparison to CLI.
- This table shows the RAM used for different Desktop Environments on Manjaro Linux.

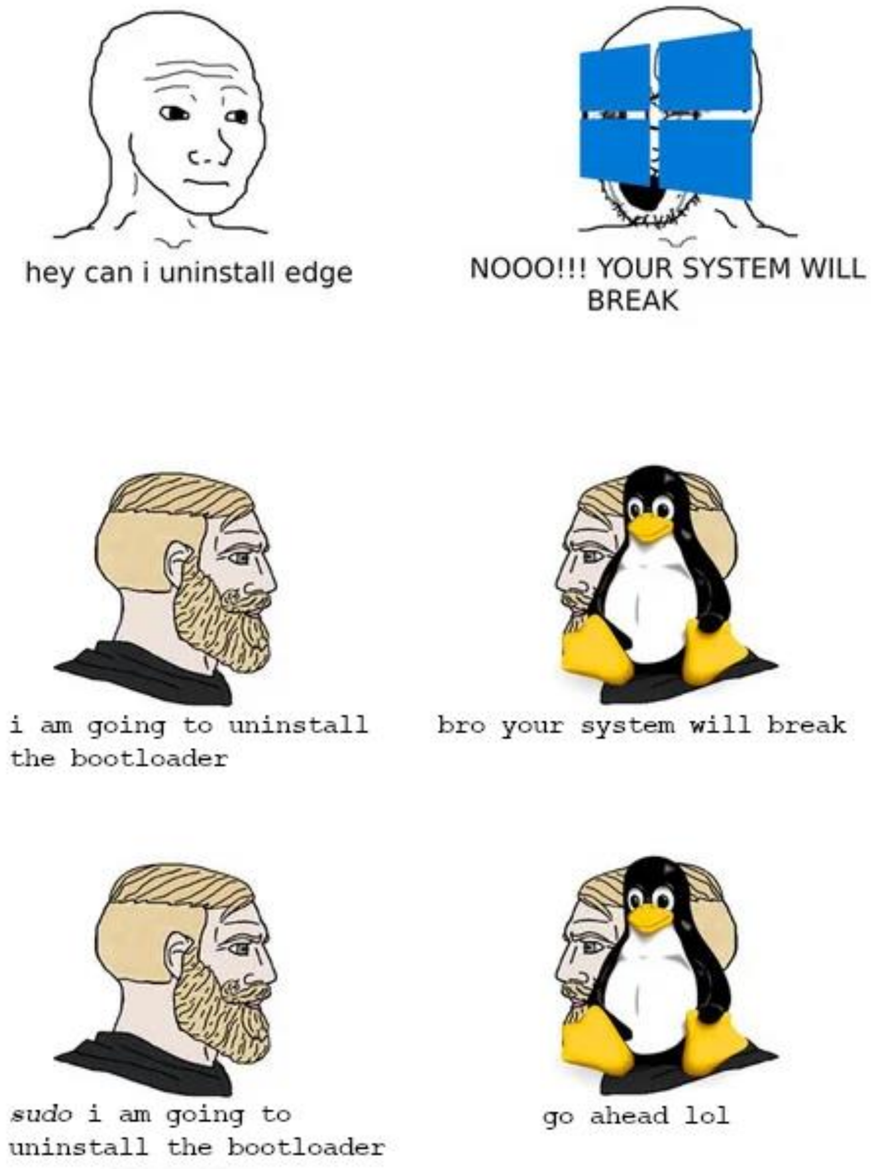
Desktop Environment	RAM Used
Base – *No Desktop Environment*	128MB
LXQt	250MB
Mate	378MB
XFCE4	390MB
KDE Plasma	455MB
GNOME	447MB
Budgie	632MB
Cinnamon	665MB

https://linuxconfig.org/manjaro-linux-system-requirements

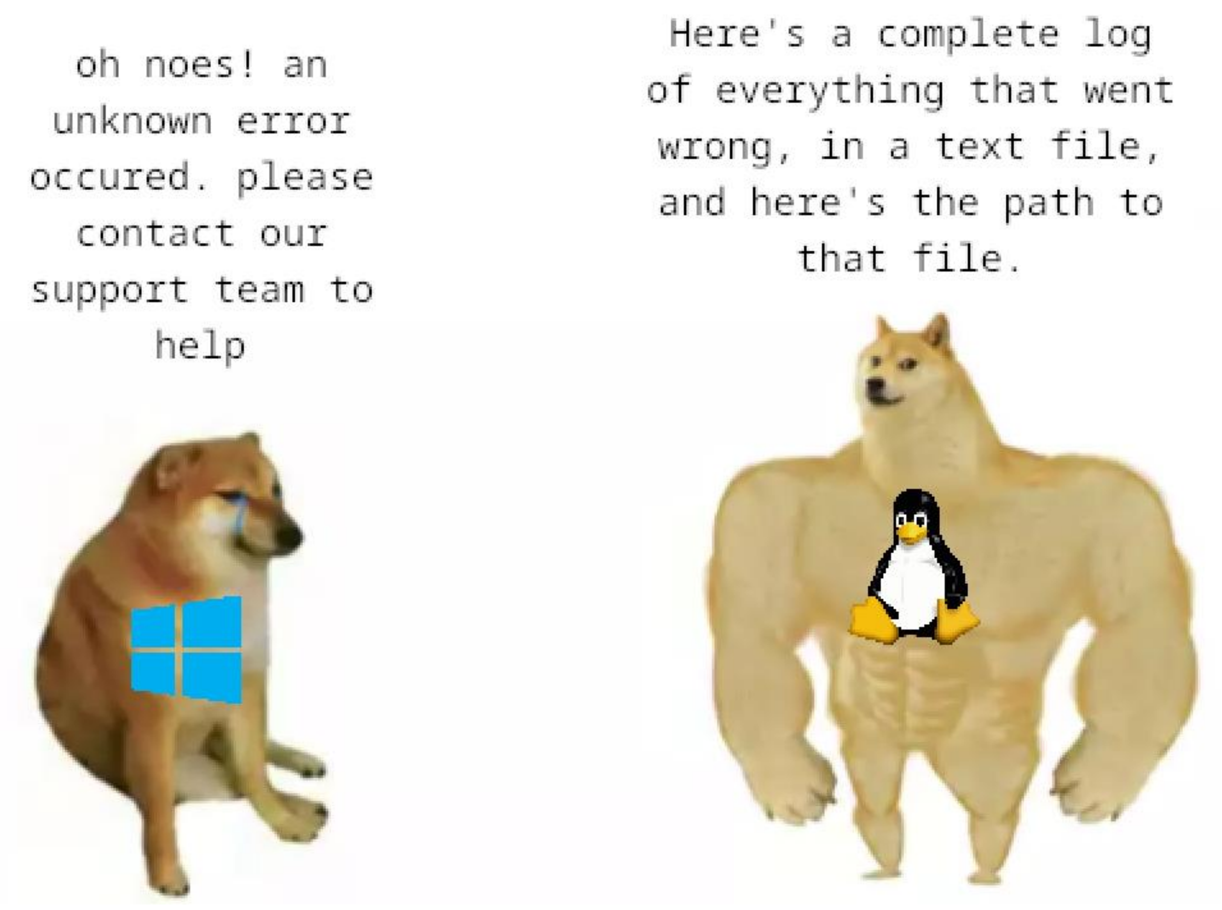
Improved Control

- Fewer "Supervisors" keeping you from doing dangerous stuff.




hey can i uninstall edge
NOOO!!! YOUR SYSTEM WILL BREAK
i am going to uninstall the bootloader
bro your system will break
sudo i am going to uninstall the bootloader
go ahead lol

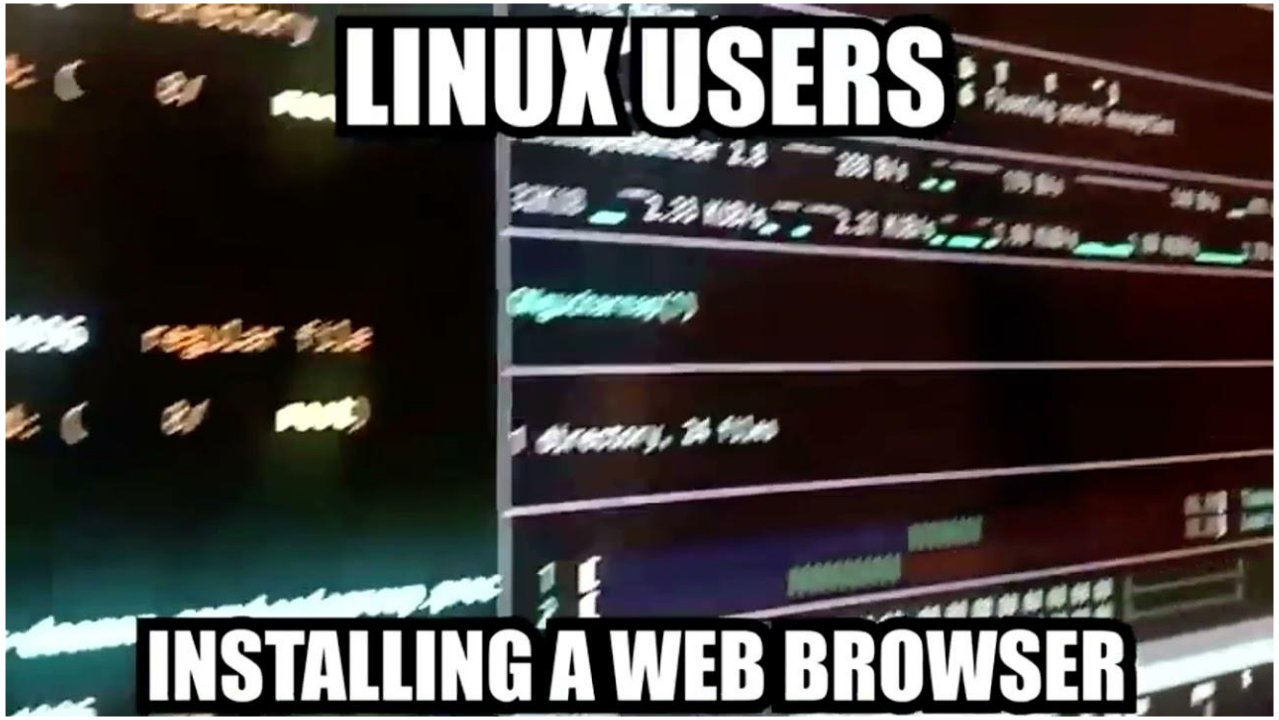
Better Error Messages (usually)


oh noes! an unknown error occured. please contact our support team to help
Here's a complete log of everything that went wrong, in a text file, and here's the path to that file.

ACM@UAB

Reasons to not use the Terminal for everything

Harder to accomplish simple tasks in Terminal



Fewer Supervisors is a double-edged sword

- Can *Very Easily* break your operating system if you have no clue what you're doing.
- Just play it safe and don't mess with any system files, you'll be ok.

```
localhost% sudo pacman -Syu
:: Synchronizing package databases...
 core                 197.5 KiB  67.3 KiB/s 00:03 [##############################] 100%
 extra                  2.2 MiB  78.1 KiB/s 00:29 [##############################] 100%
 community              5.2 MiB  59.3 KiB/s 01:30 [##############################] 100%
 alarm                 54.7 KiB   193 KiB/s 00:00 [##############################] 100%
 aur is up to date
:: Starting full system upgrade...
:: Replace bzr with extra/breezy? [Y/n] y
resolving dependencies...
looking for conflicting packages...

Packages (12) bat-0.18.1-1  breezy-3.2.0-1  bzr-2.7.0-3 [removal]  dav1d-0.9.0-1
              elfutils-0.184-1  libelf-0.184-1  linux-api-headers-5.12.3-1
              pacman-5.2.2-4  python-configobj-5.0.6.r110.g3e2f4cc-1
              python-patiencediff-0.2.2-1  sudo-1.9.7-1  virglrenderer-0.9.1-1

Total Download Size:   12.12 MiB
Total Installed Size:  68.31 MiB
Net Upgrade Size:      -1.18 MiB

:: Proceed with installation? [Y/n] y
:: Retrieving packages...
 linux-api-headers-...  1071.3 KiB  34.6 KiB/s 00:31 [##############################] 100%
 libelf-0.184-1-aarch64  413.0 KiB   155 KiB/s 00:03 [##############################] 100%
 elfutils-0.184-1-a...   562.5 KiB  21.6 KiB/s 00:26 [##############################] 100%
 pacman-5.2.2-4-aarch64  330.3 KiB  38.7 KiB/s 00:10 [############------------------]  44%
```

```
                GNU GRUB  version 2.02~beta3-5+deb9u1

   Minimal BASH-like line editing is supported. For the first word, TAB
   lists possible command completions. Anywhere else TAB lists possible
   device or file completions.


grub> _
```

ACM@UAB

What is the Terminal?

The Terminal is sort of like the "Backend" Of the Operating System.

- The GUI of an Operating System is an Application that interfaces with the shell
  - Or directly with system calls, if configured to.

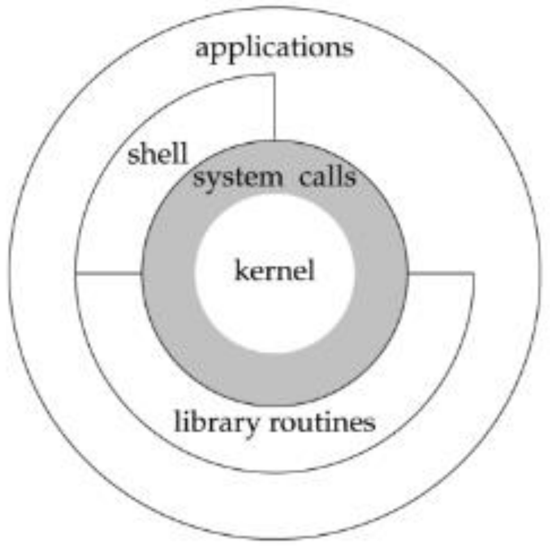


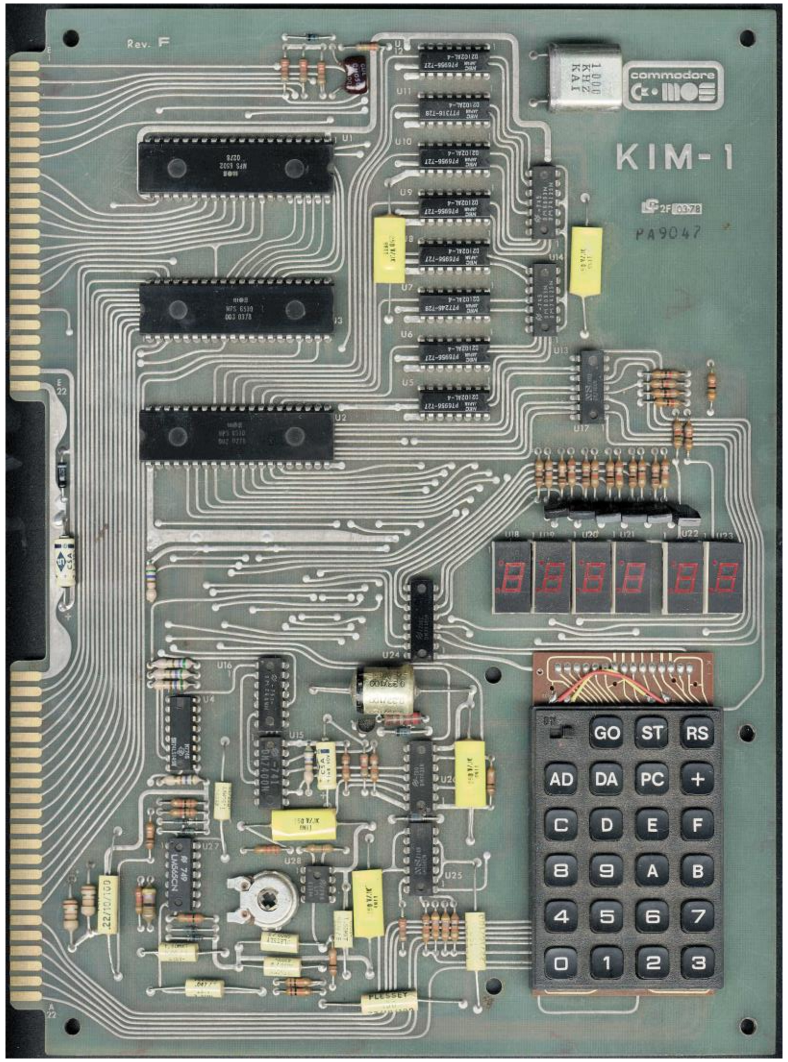
Figure 1.1 Architecture of the UNIX operating system

History of the Command Line / Terminal

- In the beginning, there was *no GUI or Terminal*
  - Computers were programmed with machine code being entered directly into memory

Altair 8800, 1974

History of the Command Line / Terminal

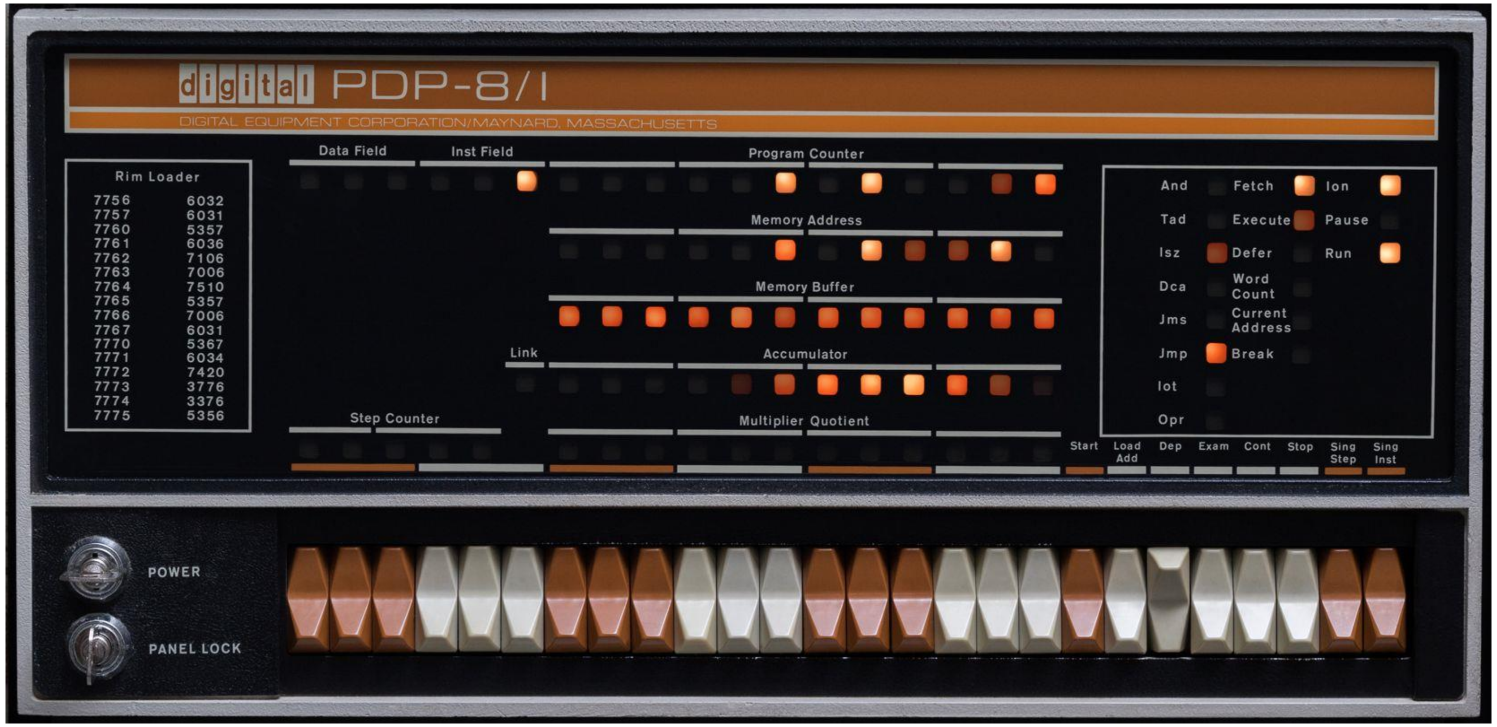


- In the beginning, there was *no GUI or Terminal*
  - Computers were programmed with machine code being entered directly into memory
    - Altair 8800, KIM-1, PDP-8/11, etc.

Commodore KIM-1, 1976

History of the Command Line / Terminal

- In the beginning, there was *no GUI or Terminal*
  - Computers were programmed with machine code being entered directly into memory



7756	6032
7757	6031
7760	5357
7761	6036
7762	7106
7763	7006
7764	7510
7765	5357
7766	7006
7767	6031
7770	5367
7771	6034
7772	7420
7773	3776
7774	3376
7775	5356

DEC PDP/8, 1965

History of the Command Line / Terminal

- In the beginning, there was *no GUI or Terminal*
  - Computers were programmed with machine code being entered directly into memory

DEC PDP/11, 1970

History of the Command Line / Terminal

- There were several different interpreters for those machines, but they all used different methods and were not standardized.
  - Interpreters used a predefined, hard coded set of commands that could not be changed without redistributing a new version of the software.
  - No internet yet, couldn’t just download an update.
- In 1964, Louis Pouzin coined the term *Shell* for the Multics Operating System.
  - He had an idea of “using commands somehow like a programming language”
- In 1965, a Multics document describes the shell as
  - “a common procedure called automatically by the supervisor whenever a user types in some message at his console, at a time when he has no other process in active execution under console control. This procedure acts as an interface between console messages and subroutine [in the supervisor].”

https://en.wikipedia.org/wiki/Shell_%28computing%29

History of the Command Line / Terminal

- In 1971, Ken Thompson developed the Thompson Shell for the first version of UNIX.
  - The UNIX Shell set the new standard for all interpreter-like interfaces.
  - Machines that had not been obsoleted usually adopted UNIX as an operating system.
- In 1975, John Mashey augments the Thompson Shell to improve shell scripting
  - It was called the Programmer's Workbench (PWB) Shell, as it was distributed from 1975 – 1977 with PWB Unix
- Both of these are short lived, as they are quickly(ish) succeeded by…

https://en.wikipedia.org/wiki/Shell_%28computing%29
https://en.wikipedia.org/wiki/Unix_shell

History of the Command Line / Terminal

- The Bourne Shell (sh)
  - First Distributed with UNIX 7 in 1979
  - Parts of the Shell Script was influenced by ALGOL 68 (ending nested structures with reversed keywords, like if … fi)
  - Still used today, sort of.
- The Bourne Again Shell (bash)
  - Written as part of the GNU Project in 1989, it is an improvement on the Bourne Shell.
  - Includes more features, and is the Default Interactive Shell for most Linux Systems
- The C Shell (csh)
  - Written by Bill Joy, a graduate student at Berkeley, wrote the C shell in the C language in 1978.
  - It was widely distributed with BSD Linux.
- The Z Shell (zsh)
  - A more modern shell (1990) written in C that is backwards compatible with bash.
    - Think like C Shell and Bash combined
  - The Default shell in MacOS since Catalina

https://en.wikipedia.org/wiki/Shell_%28computing%29
https://en.wikipedia.org/wiki/Unix_shell

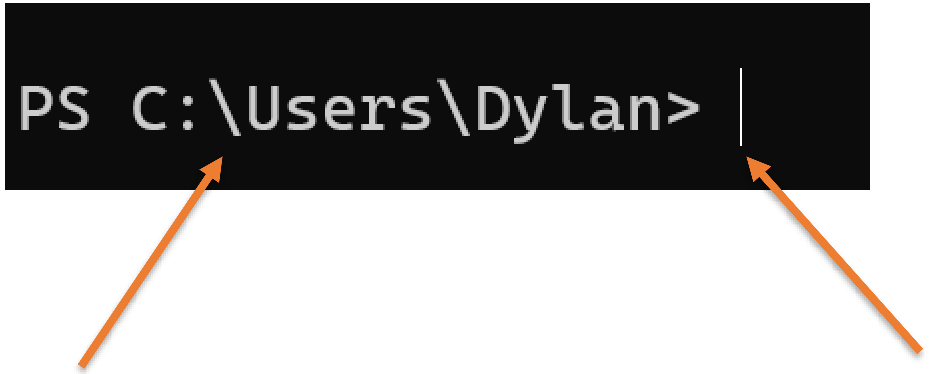
ACM@UAB

Elements of the Terminal

The Prompt

- Contains useful information you might want to know at any time
  - Usually, most important info is the Current Working Directory
- Should be customizable, not going over that
  - If you break it, it’s not my fault.

The Prompt – Windows (Powershell or CMD)

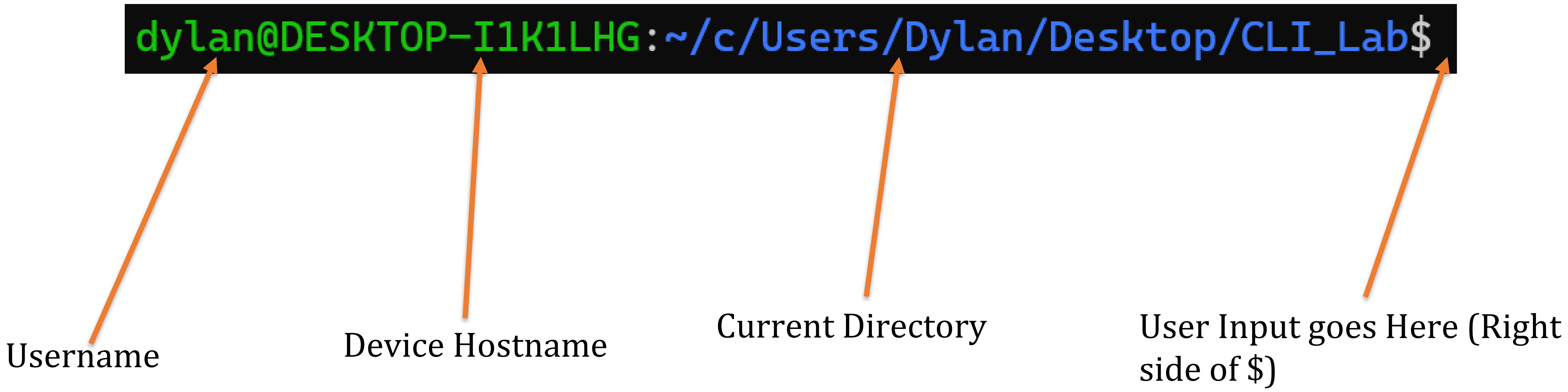


```
PS C:\Users\Dylan>
```

Current Directory

User Input goes Here (Right side of >)

The Prompt – Unix



```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab$
```

Note: If your Current Directory in the prompt is just ~, this means you're in your home directory.

This is typically /home/username in Linux, and /Users/username in MacOS.

The Cursor

- If you have a Line for a cursor, text will be inserted at the cursor.
- If you have a Block for a cursor, text will be inserted to the left of the cursor.

Terminal Command History

- If you press the up arrow on your keyboard, you can reuse commands you have already typed

- If you go up in your command history, you can press the down arrow to go back down to a blank prompt (or whatever you typed before pressing the up arrow)

Autocomplete in Terminal

- A recent-ish feature, pressing the Tab key will attempt to autofill the current “phrase” in the Terminal.
    - Works for commands and filenames *only*.

Before we get into commands…

- If you see something surrounded by <> or [] signs (like <name> or [name]), that is something you need to replace *before* running the command

- For example, if the slide says `ssh <blazerid>@moat.cis.uab.edu`, you need to replace the entire `<blazerid>` part with whatever goes there (In this case, your blazerID).
  - Ex: My ssh command would be `ssh dylcal13@moat.cis.uab.edu`

Before we get into commands…

- Lastly, some commands will be UNIX (Linux/Mac) Only.
- I'll do my best to include Windows equivalent commands if they don't exist in Powershell
- Also note that powershell != cmd, they behave differently
  - Now, I'd probably recommend using Powershell, it can do all the things CMD does and more.
  - Unless otherwise specified, all commands that are not UNIX specific should be able to run in Powershell (*not* CMD).

When in Doubt, just Google it.

- If you need a windows equivalent to a UNIX command…
- Google "<the command> for windows".
    - For example, chmod does not exist in Windows Powershell (or CMD).
        - I would look up "chmod for windows"

Me when I can't remember the Linux command:

When in Doubt, just Google it. (Man Pages)

- The Man Pages are like a bible for UNIX commands.
- Extremely detailed documentation about every command, including option flags and behaviors.
- To find a man page, in the terminal you can type `man <command>` to pull it up in the terminal, or you can google "man page <command>"
  - Any site should work, but personally I prefer linux.die.net for the way they format the pages.

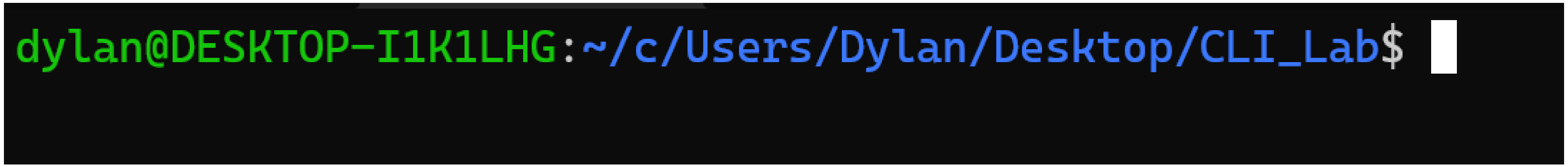
ACM@UAB

Navigating Folders / Files

clear

- Clears all text on the screen (Command History does not change)
- Usage: `clear`

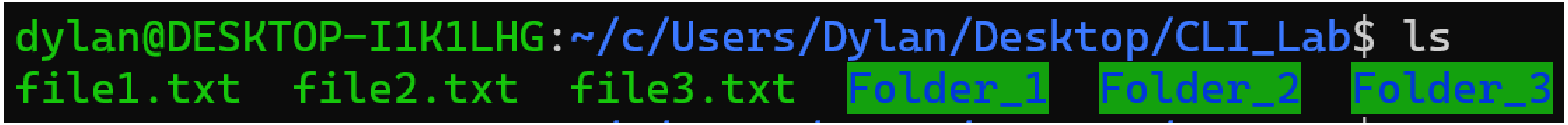
- Example:



```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab$ 
```

ls

- Shows what files are in the current folder you are in
- Usage: `ls`
  - Option Flags:
    - -h: Human Readable
    - -l: Long form Listing
    - -a: List hidden files as well (files starting with a .)
    - Flags can be chained together (Ex: ls –lah)
    - This is not an exhaustive list, there are more flags.
- Example Output:

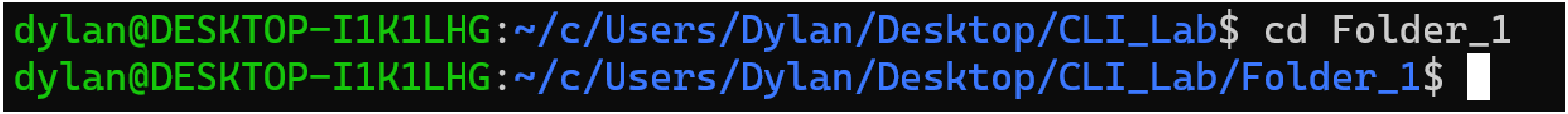


```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab$ ls
file1.txt  file2.txt  file3.txt  Folder_1  Folder_2  Folder_3
```

cd

- Used to enter/exit folders
- Usage: `cd <foldername or path>`
  - . Means “here”
  - .. Means “back”

- Example:

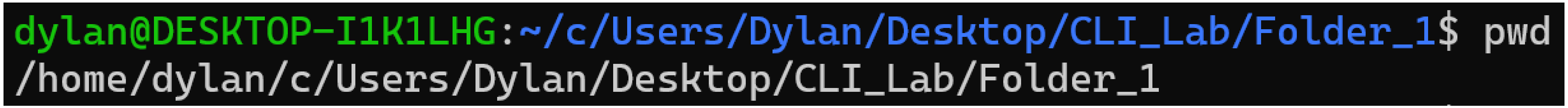


```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab$ cd Folder_1
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/Folder_1$
```

pwd

- Prints the current working directory
- Usage: `pwd`

- Example:



```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/Folder_1$ pwd
/home/dylan/c/Users/Dylan/Desktop/CLI_Lab/Folder_1
```

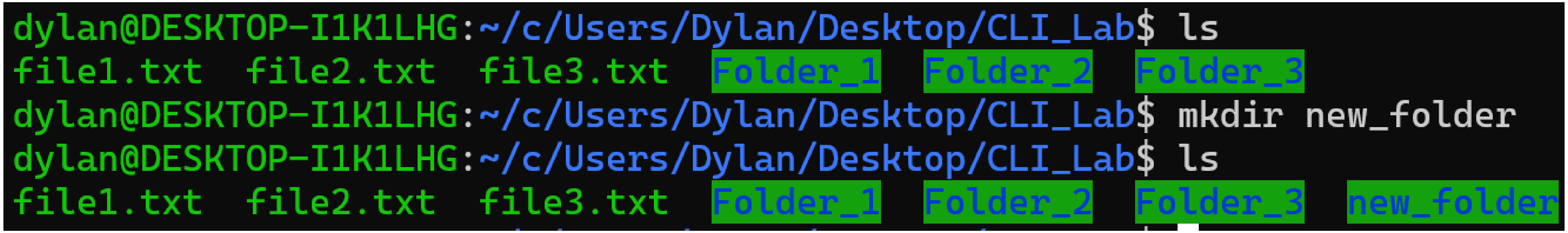
ACM@UAB

Creating Files & Folders

mkdir

- Creates a new folder using the name provided
- Usage: `mkdir <folder_name>`

- Example:

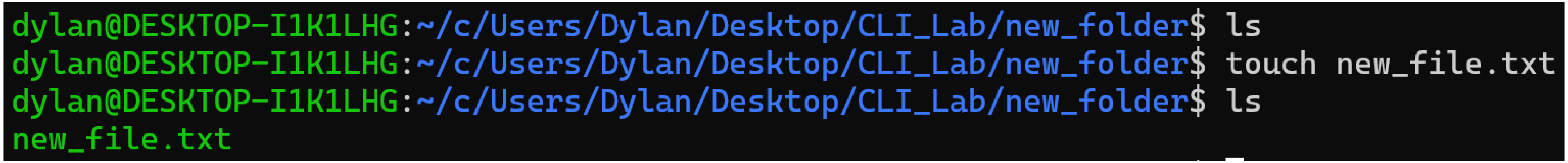


```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab$ ls
file1.txt  file2.txt  file3.txt  Folder_1  Folder_2  Folder_3
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab$ mkdir new_folder
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab$ ls
file1.txt  file2.txt  file3.txt  Folder_1  Folder_2  Folder_3  new_folder
```

touch (UNIX Only)

- Creates a new file using the name provided
- Usage: `touch <file_name>`

- Example:



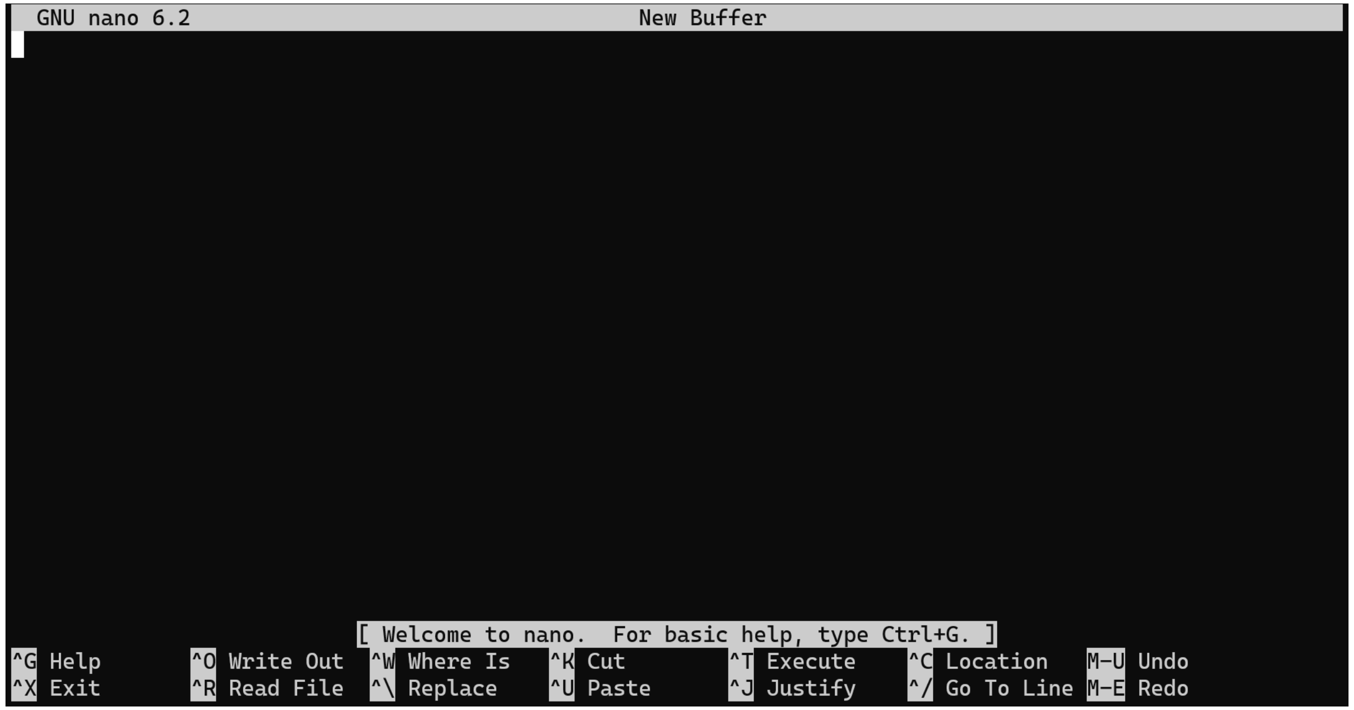
```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ls
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ touch new_file.txt
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ls
new_file.txt
```

ACM@UAB

Editing Files - Nano

What is nano?

- A Text-Based Text Editor used in the command line (developed in 1999)
- Not as powerful as vim, but easier to use and still useful.
- Nano is UNIX only (unless you can find a way to install nano on windows)



```
  GNU nano 6.2                                          New Buffer



                                         [ Welcome to nano.  For basic help, type Ctrl+G. ]
^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line    M-E Redo
```

Opening Nano

- Just type `nano <file_name>`
  - If the file already exists, it will open it in nano.
  - If the file does not exist, it will open nano and name the new file the file_name you gave it.
- You can also just type `nano` to open a new file with no name.

Using Nano

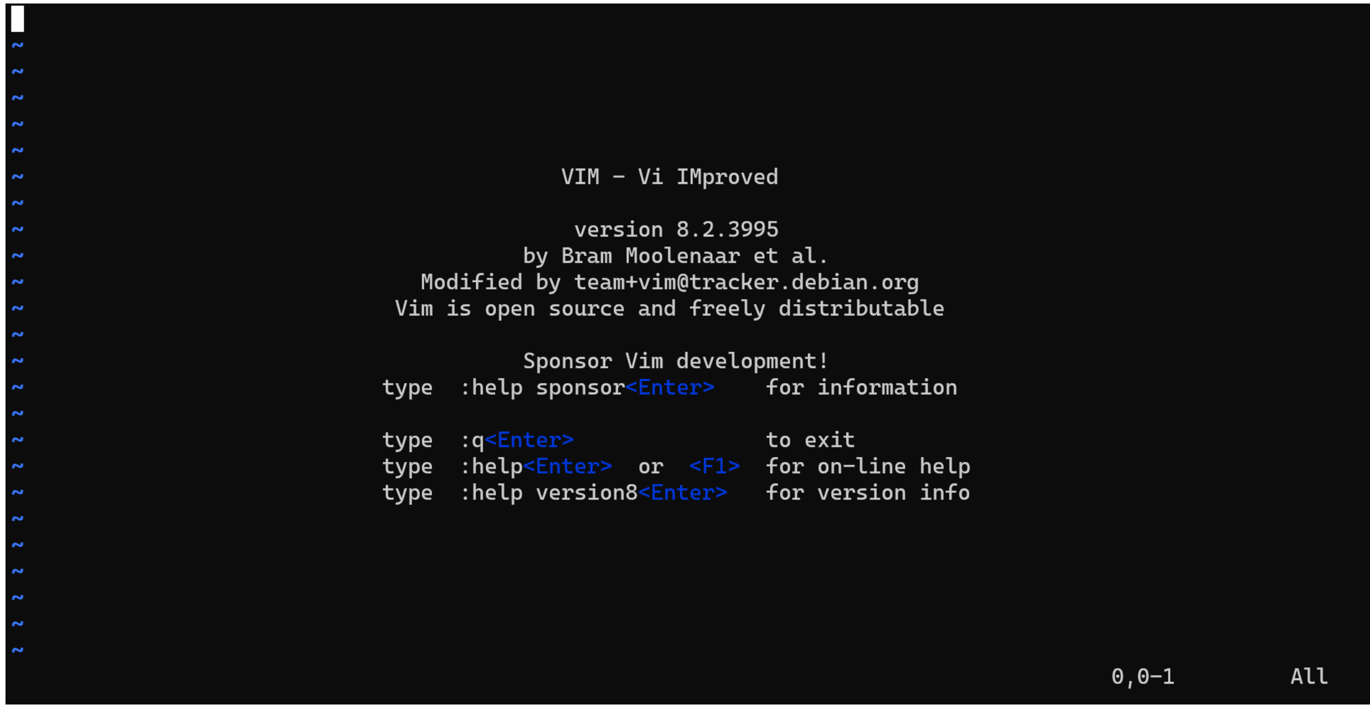
- Just start typing.
- All the functions at the bottom of the terminal are usable.
  - Find the function you want to use (the ^ in nano means the Ctrl key).
  - To activate it, type Ctrl and the letter of the function. For Example,
    - Ctrl-O will save your file
    - Ctrl-X will exit
    - Ctrl-W will search the file for a phrase/word
  - Follow the onscreen instructions to use the function.

ACM@UAB

Editing Files - VIM

What is Vim?

- A very powerful Text Editor released in 1991 (For the Amiga)
- Configurable, you can create your own keyboard shortcuts and such.
- Notorious for being impossible to exit (Spoiler: it's :q)
- Vim is also UNIX only (unless you can find a way to install Vim in Windows)



```
                    VIM - Vi IMproved

                    version 8.2.3995
                by Bram Moolenaar et al.
        Modified by team+vim@tracker.debian.org
      Vim is open source and freely distributable

                Sponsor Vim development!
   type  :help sponsor<Enter>    for information

   type  :q<Enter>               to exit
   type  :help<Enter>  or  <F1>  for on-line help
   type  :help version8<Enter>   for version info

                                                  0,0-1         All
```

Opening VIM

- Works the exact same as opening nano (but replace nano with vim).

Using VIM

- A little more complicated than Nano.
- VIM has modes, you cannot edit a file unless you are in “insert” mode.
  - The Default Mode, “normal” is where you type editor commands (like :q)
- The options that were visible at the bottom in nano are not visible in Vim
  - You will have to refer to the manual pages to learn how to do special tasks like cut, paste, etc.
  - Saving is done with :w
  - Quitting is done with :q
    - Force quit with :q!
  - You can chain these together to do both (:wq)

I went crazy trying to remember the movie name, it’s “Arrival” ☺

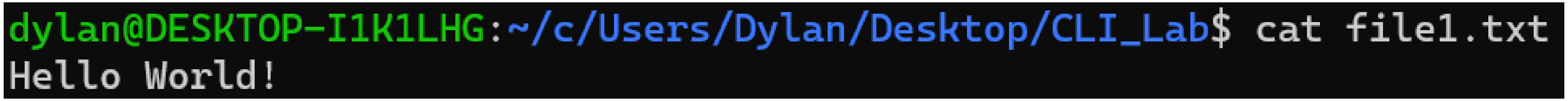
ACM@UAB

Printing Files

cat

- Prints the content of a File to the terminal
- Usage: `cat <filename/path>`

- Example:



```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab$ cat file1.txt
Hello World!
```

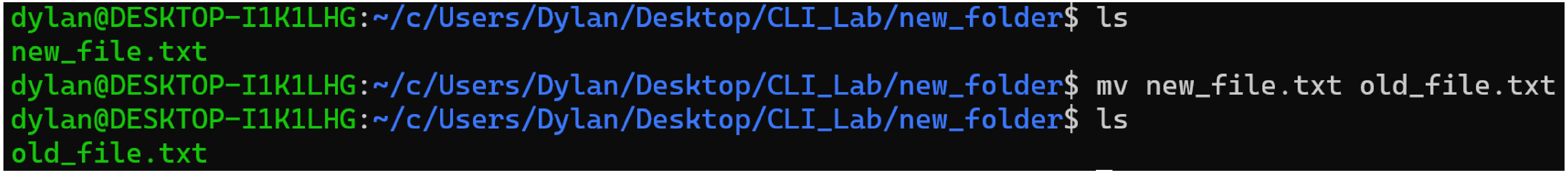
ACM@UAB

Moving / Renaming Files & Folders

mv

- This will move files and folders to the path provided
  - It is also used for renaming files and folders.
- Usage: `mv <input_file> <output_file>`

- Example:



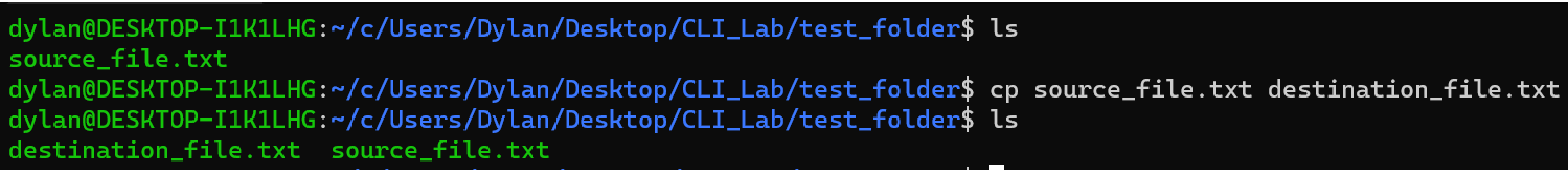
```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ls
new_file.txt
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ mv new_file.txt old_file.txt
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ls
old_file.txt
```

ACM@UAB

Copying Files

cp

- Copies a file
- Usage: `cp <input_file> <destination_file>`
  - Can also copy directories with –r
- Example:



```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/test_folder$ ls
source_file.txt
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/test_folder$ cp source_file.txt destination_file.txt
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/test_folder$ ls
destination_file.txt  source_file.txt
```

ACM@UAB

Removing Files & Folders

Disclaimer

- When using these commands to remove files & folders…

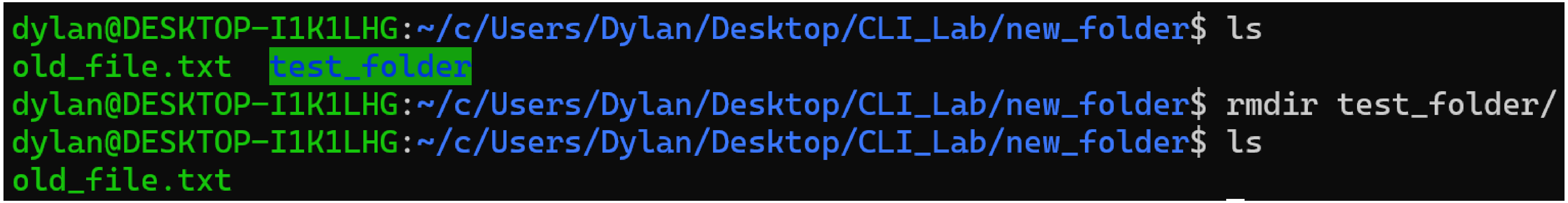
<u>There Is No Recycle Bin!</u>

- Files will be deleted **Permanently** using these commands.
  - Make extra sure that you are deleting the correct File or Folder.

rmdir

- Removes a folder using the provided path.
  - Note: Folder *Must* Be empty, otherwise rmdir will error out.
- Usage: `rmdir <foldername/path>`

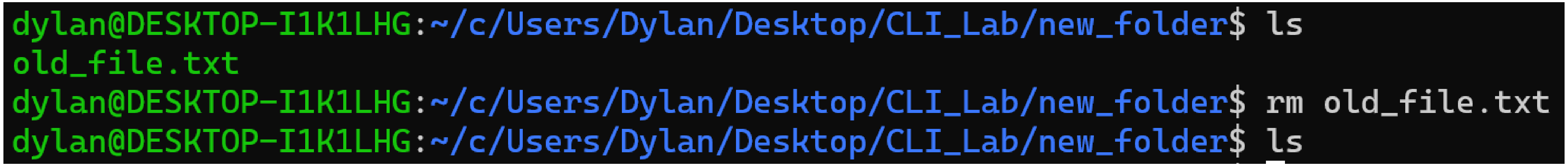
- Example:



```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ls
old_file.txt  test_folder
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ rmdir test_folder/
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ls
old_file.txt
```

rm

- Removes a file using the provided path.
- Usage: `rm <filename/path>`
  - Optionally, you can add –rf to remove folders with files inside.
  - Usage: `rm –rf <foldername/path>`
- Example:



```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ls
old_file.txt
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ rm old_file.txt
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ls
```

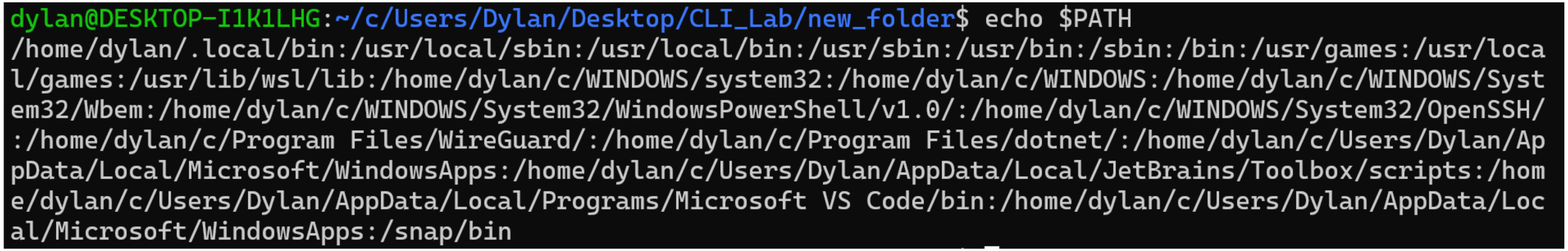
ACM@UAB

Running Programs

./ Syntax

- Basically, ./ means to run a script that is the folder you're currently in
  - . = "here", from the cd slide
  - So, ./script means the script in the current folder.
- Why have the ./? Why not just the name of the script
  - If we just say script, it will try finding script in the system path
  - The path is a list of folders where applications should be found if called.
    - For example, nano is in the system path. That's why no matter where we are in the system, nano will always reference the same application.

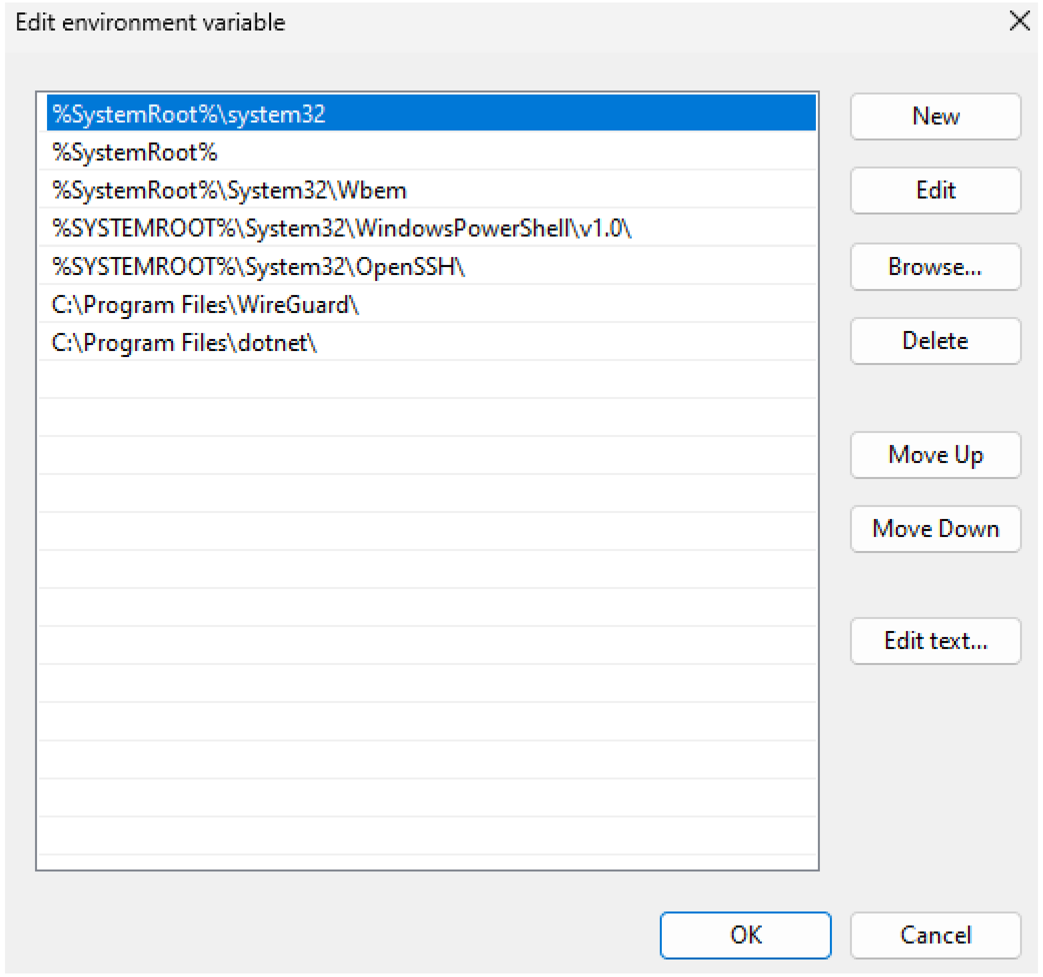
Path Examples



```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ echo $PATH
/home/dylan/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loca
l/games:/usr/lib/wsl/lib:/home/dylan/c/WINDOWS/system32:/home/dylan/c/WINDOWS:/home/dylan/c/WINDOWS/Syst
em32/Wbem:/home/dylan/c/WINDOWS/System32/WindowsPowerShell/v1.0/:/home/dylan/c/WINDOWS/System32/OpenSSH/
:/home/dylan/c/Program Files/WireGuard/:/home/dylan/c/Program Files/dotnet/:/home/dylan/c/Users/Dylan/Ap
pData/Local/Microsoft/WindowsApps:/home/dylan/c/Users/Dylan/AppData/Local/JetBrains/Toolbox/scripts:/hom
e/dylan/c/Users/Dylan/AppData/Local/Programs/Microsoft VS Code/bin:/home/dylan/c/Users/Dylan/AppData/Loc
al/Microsoft/WindowsApps:/snap/bin
```

Example of a UNIX-like path

Path Examples



```
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\WireGuard\
C:\Program Files\dotnet\
```

Example of a Windows-like Path

Running an installed application

- Just type the name of the application and any arguments required.
    - Example: `nano file.txt`

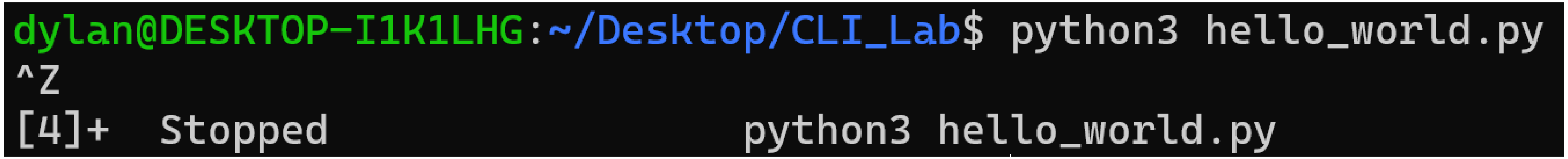
ACM@UAB

Creating Background Process

Ctrl-Z

- Suspends (stops) a foreground process and turns it into a job.
- Usage: Just type Ctrl-Z

- Example:

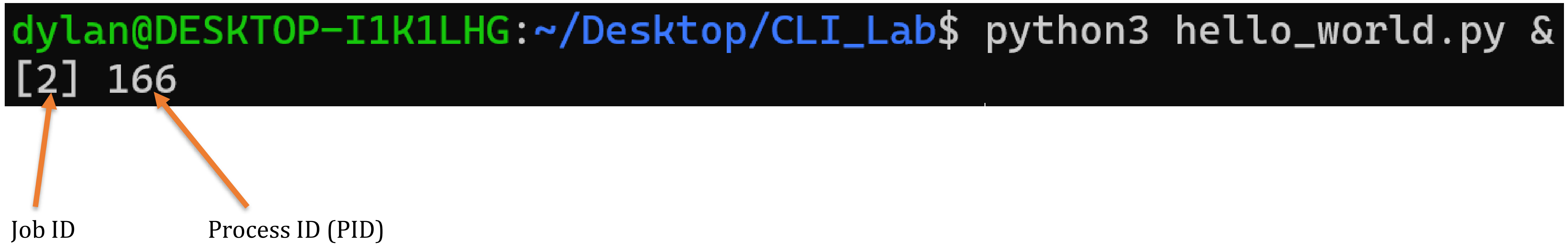


```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ python3 hello_world.py
^Z
[4]+  Stopped                 python3 hello_world.py
```

&

- Adding an ampersand (&) to the end of a command starts that command in the background (as a job).
- Usage: `<some_command> &`

- Example:

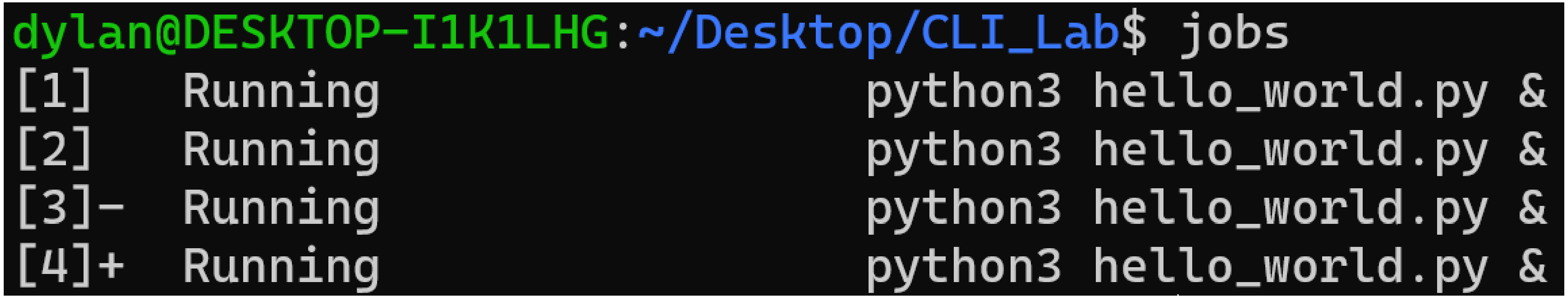


```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ python3 hello_world.py &
[2] 166
```

jobs

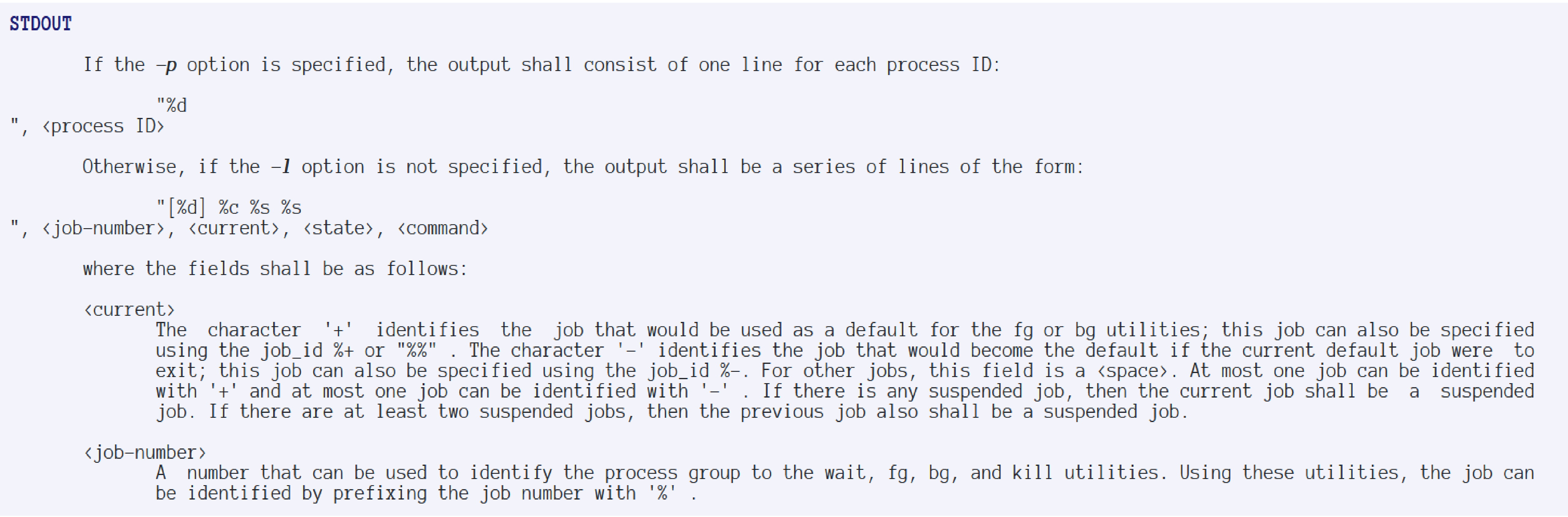
- Shows all background jobs
- Usage: `jobs`

- Example:



```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ jobs
[1]   Running                 python3 hello_world.py &
[2]   Running                 python3 hello_world.py &
[3]-  Running                 python3 hello_world.py &
[4]+  Running                 python3 hello_world.py &
```

jobs man page

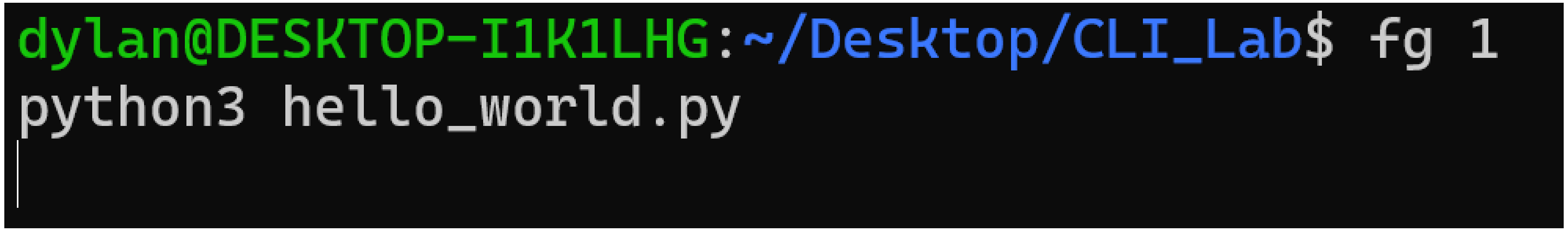


```
STDOUT

       If the -p option is specified, the output shall consist of one line for each process ID:

              "%d
", <process ID>

       Otherwise, if the -l option is not specified, the output shall be a series of lines of the form:

              "[%d] %c %s %s
", <job-number>, <current>, <state>, <command>

       where the fields shall be as follows:

       <current>
              The  character  '+'  identifies  the  job that would be used as a default for the fg or bg utilities; this job can also be specified
              using the job_id %+ or "%%" . The character '-' identifies the job that would become the default if the current default job were  to
              exit; this job can also be specified using the job_id %-. For other jobs, this field is a <space>. At most one job can be identified
              with '+' and at most one job can be identified with '-' . If there is any suspended job, then the current job shall be  a  suspended
              job. If there are at least two suspended jobs, then the previous job also shall be a suspended job.

       <job-number>
              A  number that can be used to identify the process group to the wait, fg, bg, and kill utilities. Using these utilities, the job can
              be identified by prefixing the job number with '%' .
```

fg

- Brings a background process into the foreground
- Usage: `fg <job_id>`

- Example:

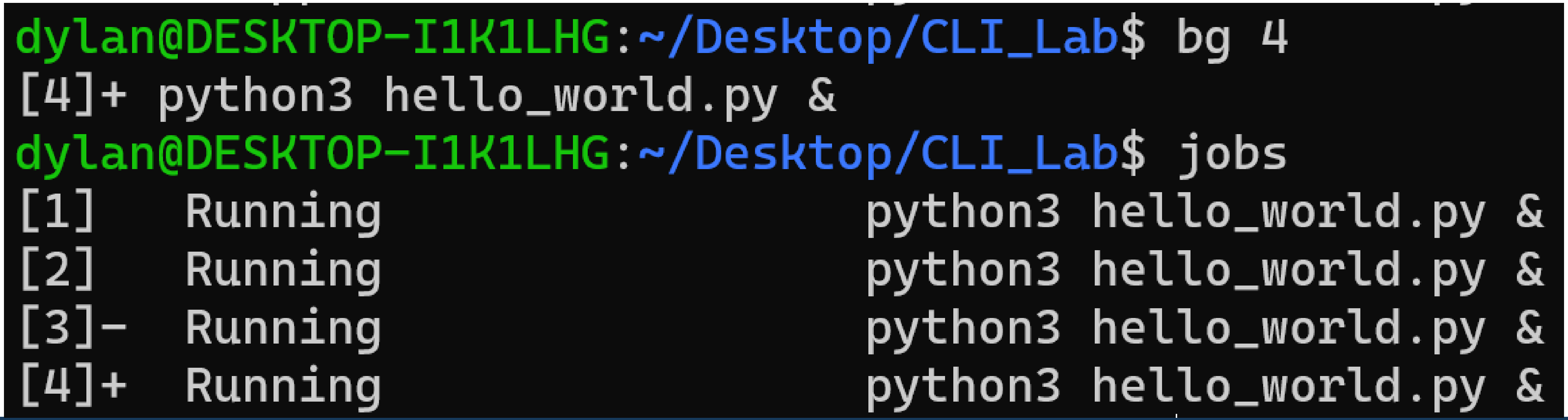


```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ fg 1
python3 hello_world.py
```

bg

- Sends a separate running process into background (or restarts stopped job)
- Usage: `bg <PID_or_Job_ID>`

- Example:



```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ bg 4
[4]+ python3 hello_world.py &
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ jobs
[1]   Running                 python3 hello_world.py &
[2]   Running                 python3 hello_world.py &
[3]-  Running                 python3 hello_world.py &
[4]+  Running                 python3 hello_world.py &
```

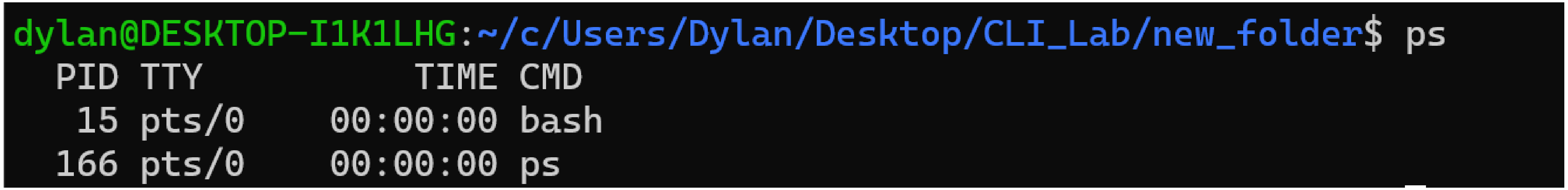
ACM@UAB

Task Manager in the Terminal

ps

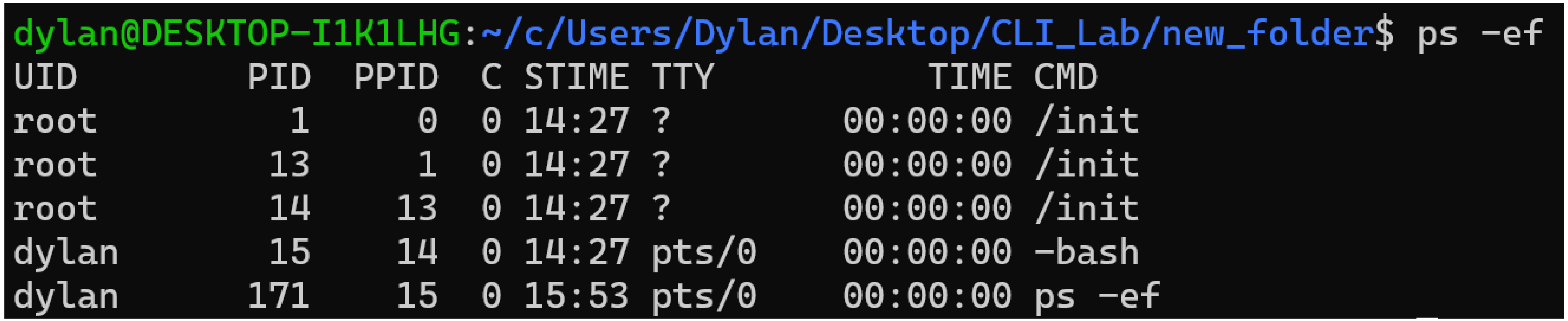
- Lists running processes under your username (your processes)
- Usage: `ps`
  - Adding a `-ef` lists *every* running process.

- Example:



```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ps
  PID TTY          TIME CMD
   15 pts/0    00:00:00 bash
  166 pts/0    00:00:00 ps
```

Sample ps -ef

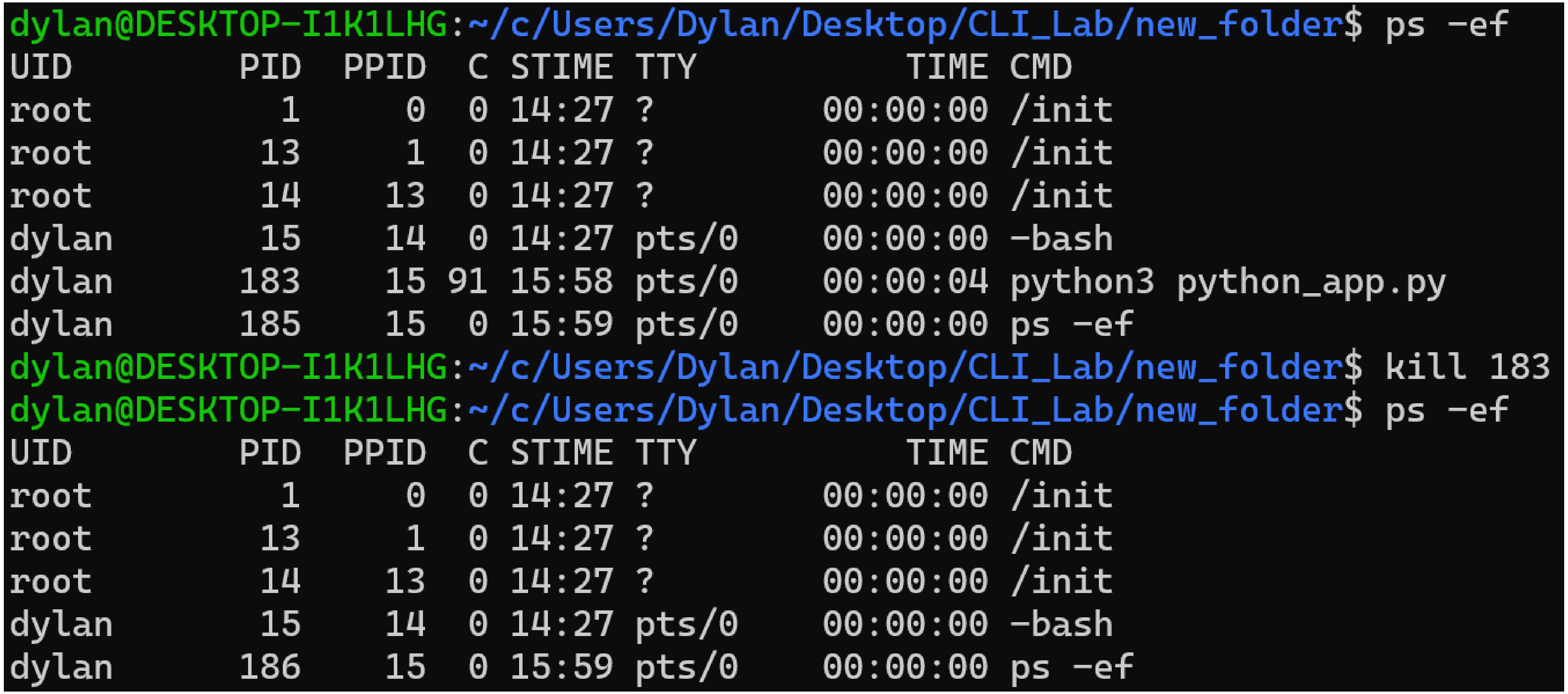


```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ps -ef
UID         PID   PPID  C STIME TTY          TIME CMD
root          1      0  0 14:27 ?        00:00:00 /init
root         13      1  0 14:27 ?        00:00:00 /init
root         14     13  0 14:27 ?        00:00:00 /init
dylan        15     14  0 14:27 pts/0    00:00:00 -bash
dylan       171     15  0 15:53 pts/0    00:00:00 ps -ef
```

kill

- Kills a process using the provided Process ID (PID)
- Usage: `kill <pid>`

kill (example)

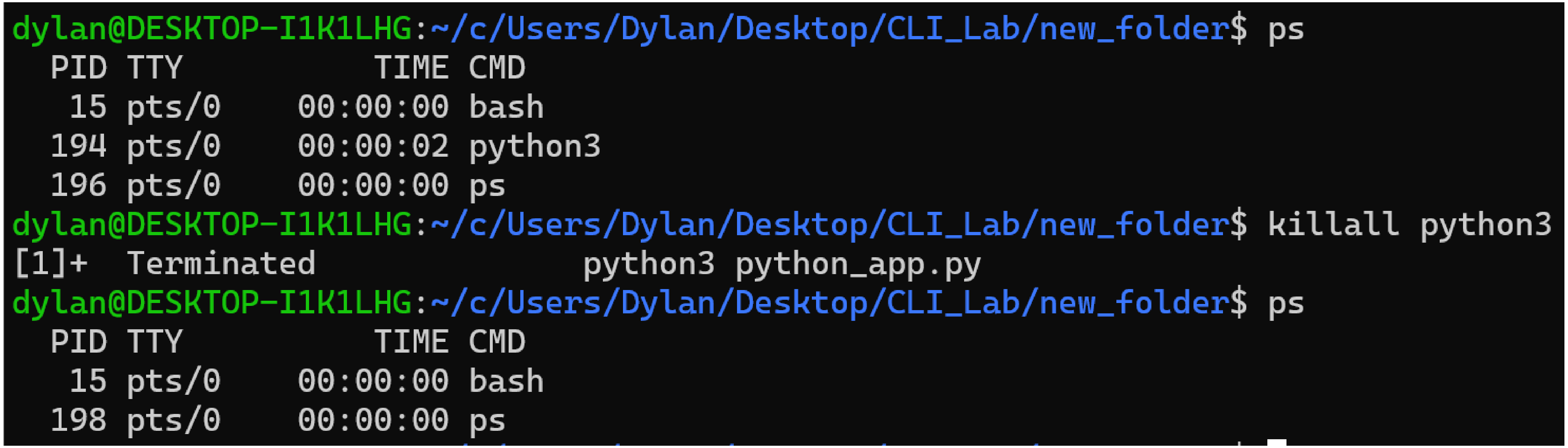


```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 14:27 ?        00:00:00 /init
root          13       1  0 14:27 ?        00:00:00 /init
root          14      13  0 14:27 ?        00:00:00 /init
dylan         15      14  0 14:27 pts/0    00:00:00 -bash
dylan        183      15 91 15:58 pts/0    00:00:04 python3 python_app.py
dylan        185      15  0 15:59 pts/0    00:00:00 ps -ef
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ kill 183
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 14:27 ?        00:00:00 /init
root          13       1  0 14:27 ?        00:00:00 /init
root          14      13  0 14:27 ?        00:00:00 /init
dylan         15      14  0 14:27 pts/0    00:00:00 -bash
dylan        186      15  0 15:59 pts/0    00:00:00 ps -ef
```

killall (UNIX only)

- Kills all processes that match the provided name
- Usage: `killall <process_name>`

killall (Example)

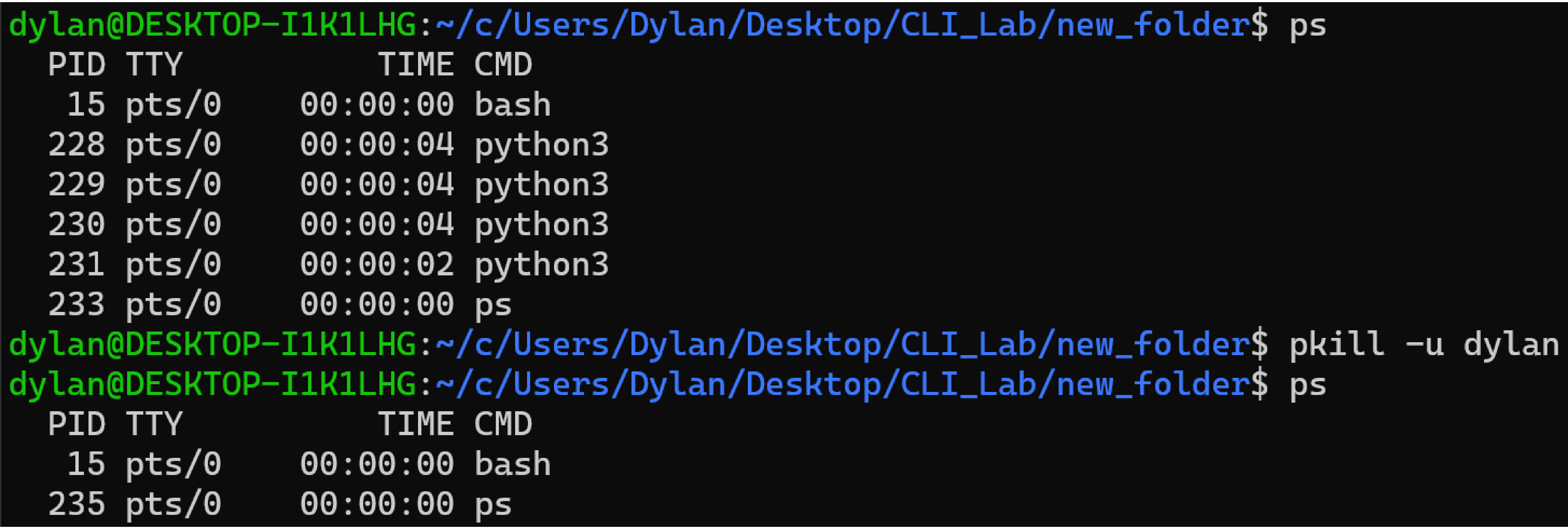


```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ps
  PID TTY          TIME CMD
   15 pts/0    00:00:00 bash
  194 pts/0    00:00:02 python3
  196 pts/0    00:00:00 ps
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ killall python3
[1]+  Terminated              python3 python_app.py
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ps
  PID TTY          TIME CMD
   15 pts/0    00:00:00 bash
  198 pts/0    00:00:00 ps
```

Honorable Mention: pkill –u (UNIX only)

- Kills all running processes running under a username
- Usage: `pkill –u <username>`

pkill –u Example



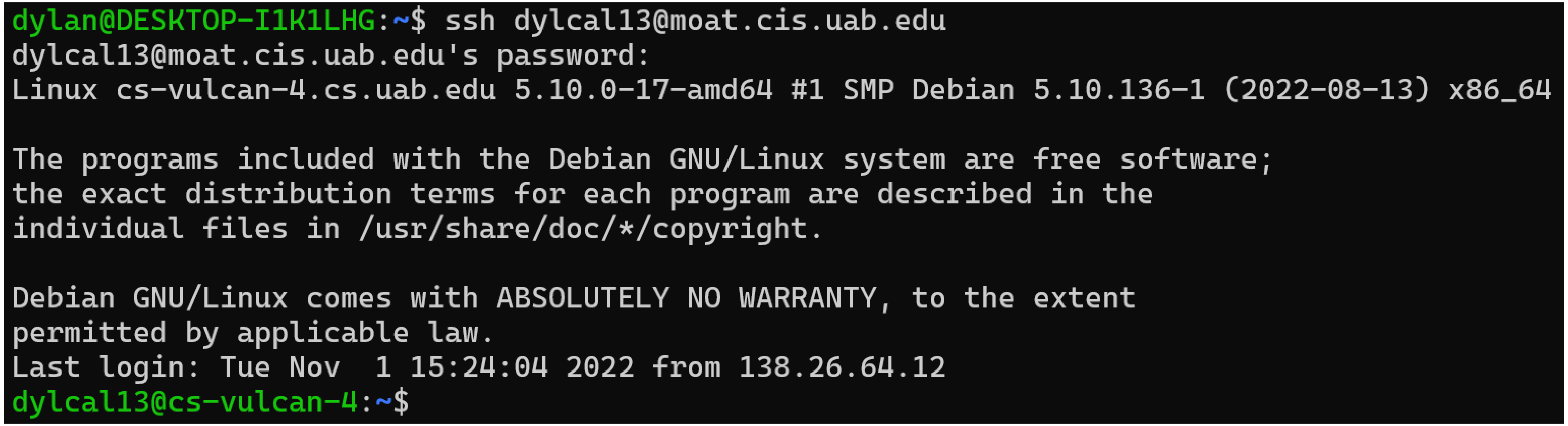
```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ps
  PID TTY          TIME CMD
   15 pts/0    00:00:00 bash
  228 pts/0    00:00:04 python3
  229 pts/0    00:00:04 python3
  230 pts/0    00:00:04 python3
  231 pts/0    00:00:02 python3
  233 pts/0    00:00:00 ps
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ pkill -u dylan
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab/new_folder$ ps
  PID TTY          TIME CMD
   15 pts/0    00:00:00 bash
  235 pts/0    00:00:00 ps
```

ACM@UAB

Connecting to Remote Servers

ssh

- Connects your terminal to an ssh server.
- Usage: `ssh <username>@<hostname>`
- Example:



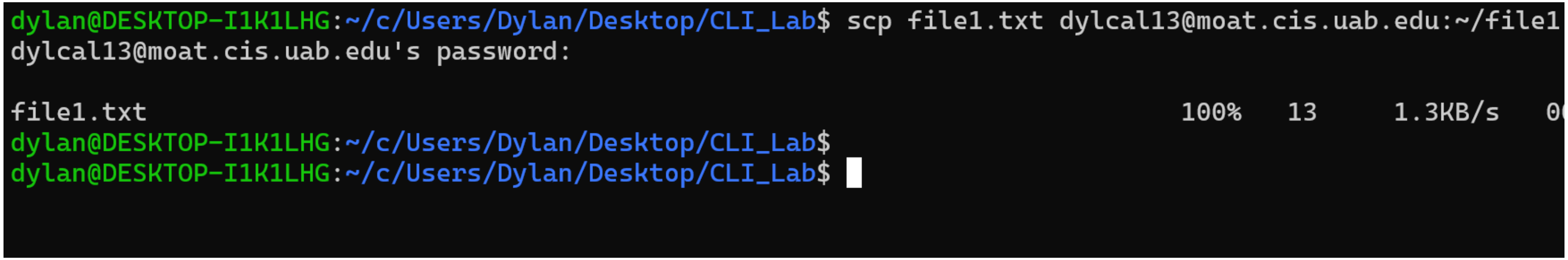
```
dylan@DESKTOP-I1K1LHG:~$ ssh dylcal13@moat.cis.uab.edu
dylcal13@moat.cis.uab.edu's password:
Linux cs-vulcan-4.cs.uab.edu 5.10.0-17-amd64 #1 SMP Debian 5.10.136-1 (2022-08-13) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Nov  1 15:24:04 2022 from 138.26.64.12
dylcal13@cs-vulcan-4:~$
```

scp

- Used to download and upload files to a remote server over ssh.
- Usage:
  - `scp <local_path> <username>@<hostname>:<remote_path>` (If uploading)
  - `scp <username>@<hostname>:<remote_path> <local_path>` (If downloading)

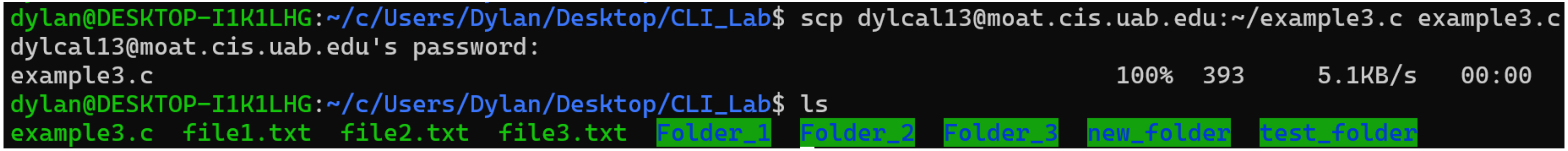
scp Examples

- Uploading:



```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab$ scp file1.txt dylcal13@moat.cis.uab.edu:~/file1
dylcal13@moat.cis.uab.edu's password:

file1.txt                                                                100%   13      1.3KB/s    0
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab$
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab$
```

- Downloading:



```
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab$ scp dylcal13@moat.cis.uab.edu:~/example3.c example3.c
dylcal13@moat.cis.uab.edu's password:
example3.c                                                          100%  393     5.1KB/s   00:00
dylan@DESKTOP-I1K1LHG:~/c/Users/Dylan/Desktop/CLI_Lab$ ls
example3.c  file1.txt  file2.txt  file3.txt  Folder_1  Folder_2  Folder_3  new_folder  test_folder
```

Honorable Mention: sftp

- scp kinda sucks, some people (like myself) prefer sftp.
- It’s like an interactive scp, I’m not going to cover it
  - Google It.

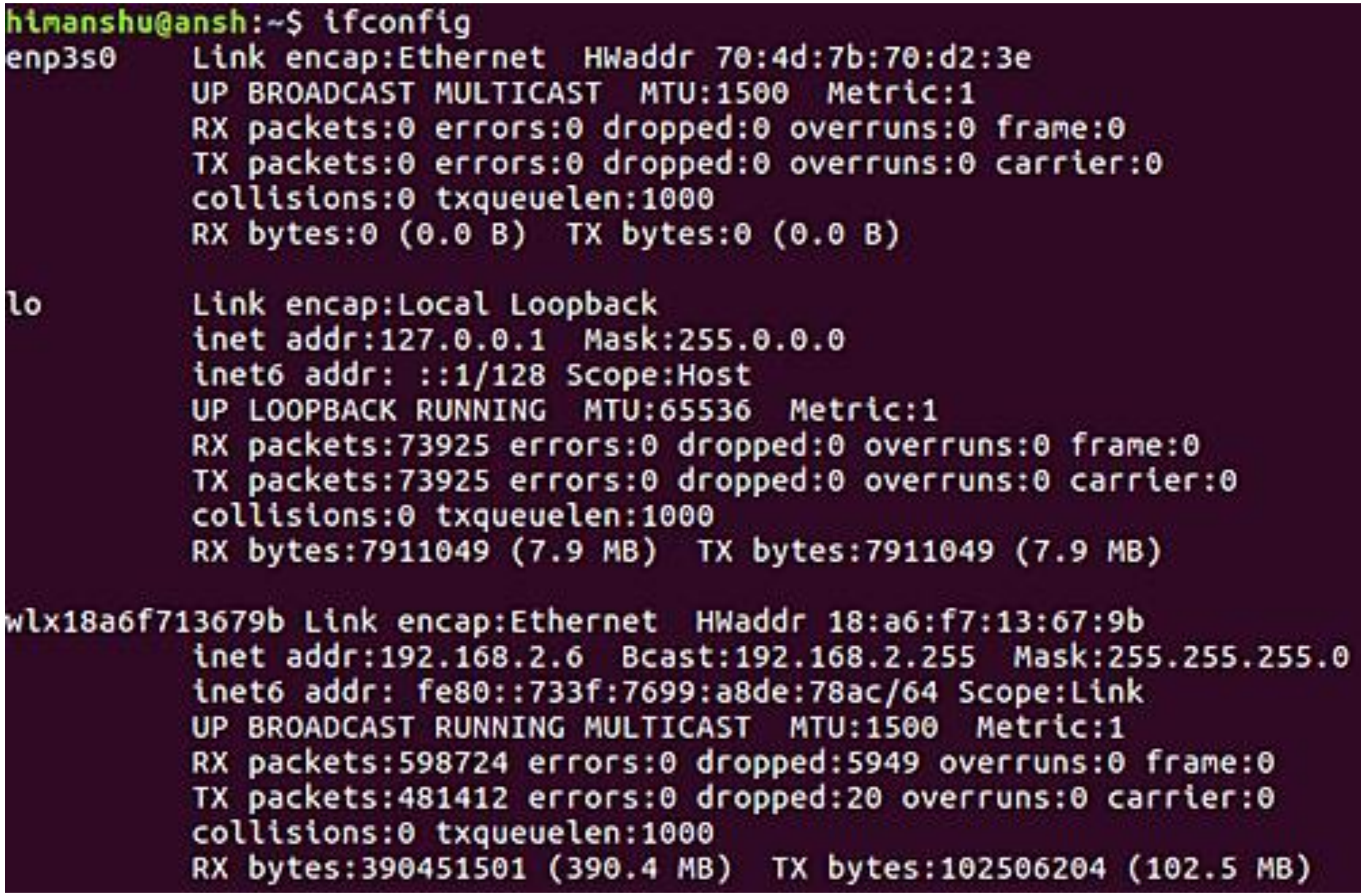
ACM@UAB

Network Info

ifconfig (UNIX only)

- Displays all active network adapters and their details
- Usage: `ifconfig`

Ifconfig Example

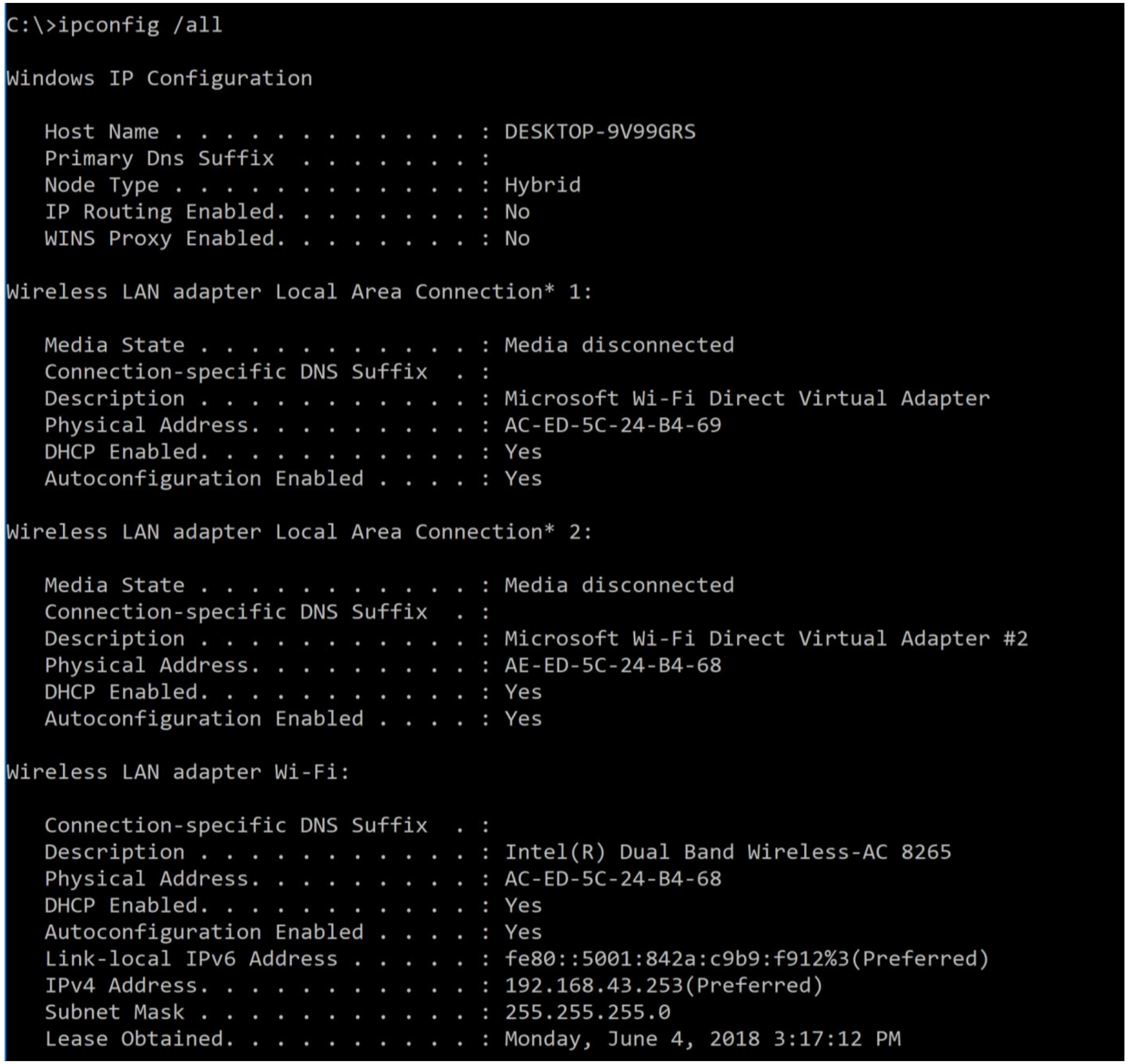


```
himanshu@ansh:~$ ifconfig
enp3s0    Link encap:Ethernet  HWaddr 70:4d:7b:70:d2:3e
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:73925 errors:0 dropped:0 overruns:0 frame:0
          TX packets:73925 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7911049 (7.9 MB)  TX bytes:7911049 (7.9 MB)

wlx18a6f713679b Link encap:Ethernet  HWaddr 18:a6:f7:13:67:9b
          inet addr:192.168.2.6  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::733f:7699:a8de:78ac/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:598724 errors:0 dropped:5949 overruns:0 frame:0
          TX packets:481412 errors:0 dropped:20 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:390451501 (390.4 MB)  TX bytes:102506204 (102.5 MB)
```

ipconfig (Windows Only)

- Displays all active network adapters and their details
- Usage: `ipconfig`

Ipconfig Example

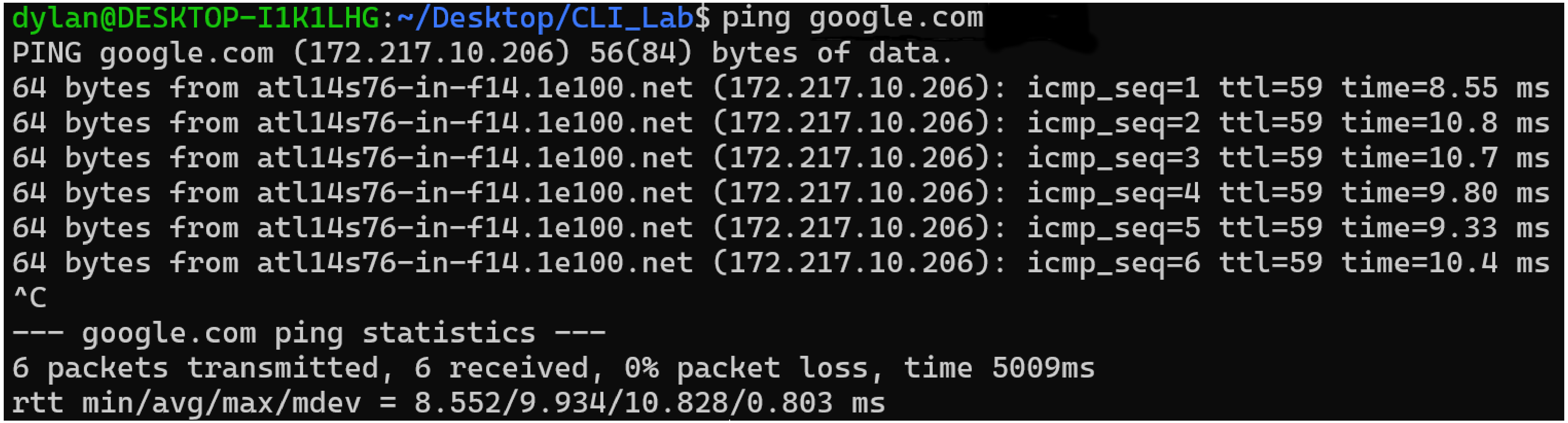


```
C:\>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : DESKTOP-9V99GRS
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Microsoft Wi-Fi Direct Virtual Adapter
   Physical Address. . . . . . . . . : AC-ED-5C-24-B4-69
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #2
   Physical Address. . . . . . . . . : AE-ED-5C-24-B4-68
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) Dual Band Wireless-AC 8265
   Physical Address. . . . . . . . . : AC-ED-5C-24-B4-68
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::5001:842a:c9b9:f912%3(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.43.253(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Monday, June 4, 2018 3:17:12 PM
```

https://www.certblaster.com/network-plus-n10-007-examnotes-for-sub-objective-1-4-part-1-of-2/output-of-the-ipconfig-all-command/

ping

- Pings a server and displays relevant connection information
- Usage: `ping <address>`
- Example:



```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ ping google.com [redacted]
PING google.com (172.217.10.206) 56(84) bytes of data.
64 bytes from atl14s76-in-f14.1e100.net (172.217.10.206): icmp_seq=1 ttl=59 time=8.55 ms
64 bytes from atl14s76-in-f14.1e100.net (172.217.10.206): icmp_seq=2 ttl=59 time=10.8 ms
64 bytes from atl14s76-in-f14.1e100.net (172.217.10.206): icmp_seq=3 ttl=59 time=10.7 ms
64 bytes from atl14s76-in-f14.1e100.net (172.217.10.206): icmp_seq=4 ttl=59 time=9.80 ms
64 bytes from atl14s76-in-f14.1e100.net (172.217.10.206): icmp_seq=5 ttl=59 time=9.33 ms
64 bytes from atl14s76-in-f14.1e100.net (172.217.10.206): icmp_seq=6 ttl=59 time=10.4 ms
^C
--- google.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5009ms
rtt min/avg/max/mdev = 8.552/9.934/10.828/0.803 ms
```

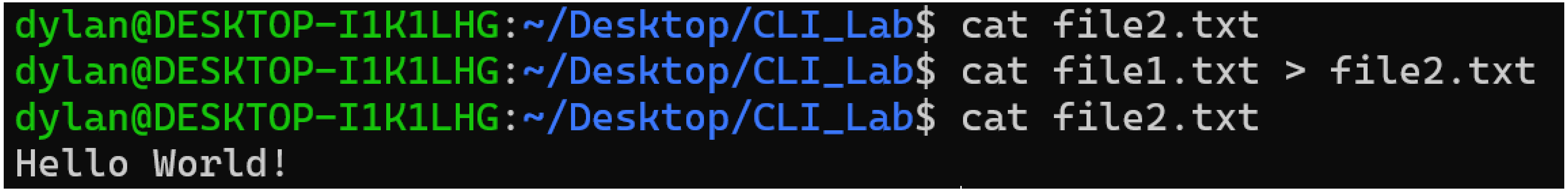
ACM@UAB

I/O Redirection

>

- Redirects the output of a command and overwrites the contents of a file with that output.
- Usage: `[some_command] > [some_file]`

- Example:

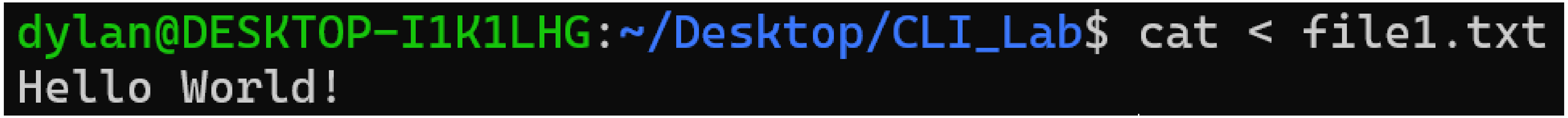


```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ cat file2.txt
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ cat file1.txt > file2.txt
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ cat file2.txt
Hello World!
```

<

- Used to redirect file contents into commands.
- Usage: `[some_command] < [some_file]`

- Example:

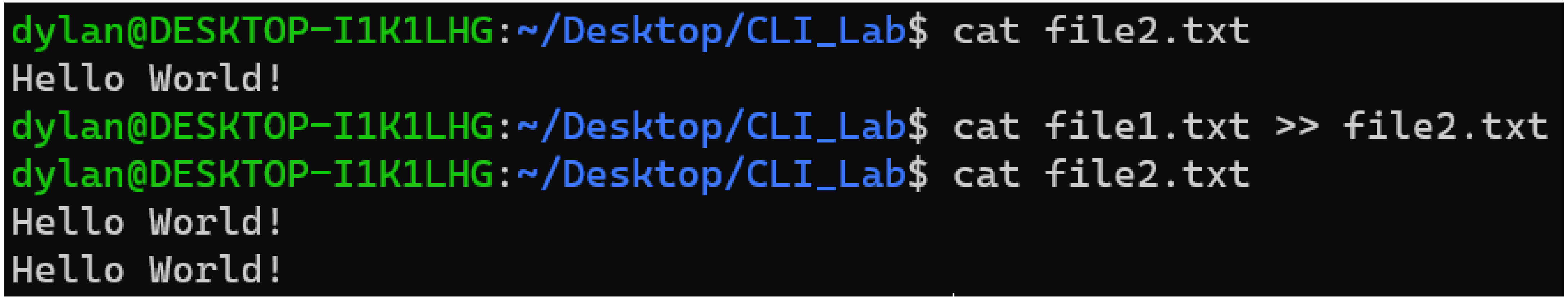


```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ cat < file1.txt
Hello World!
```

>>

- Redirects the output of a command and appends to the contents of a file with that output.
- Usage: `[some_command] >> [some_file]`

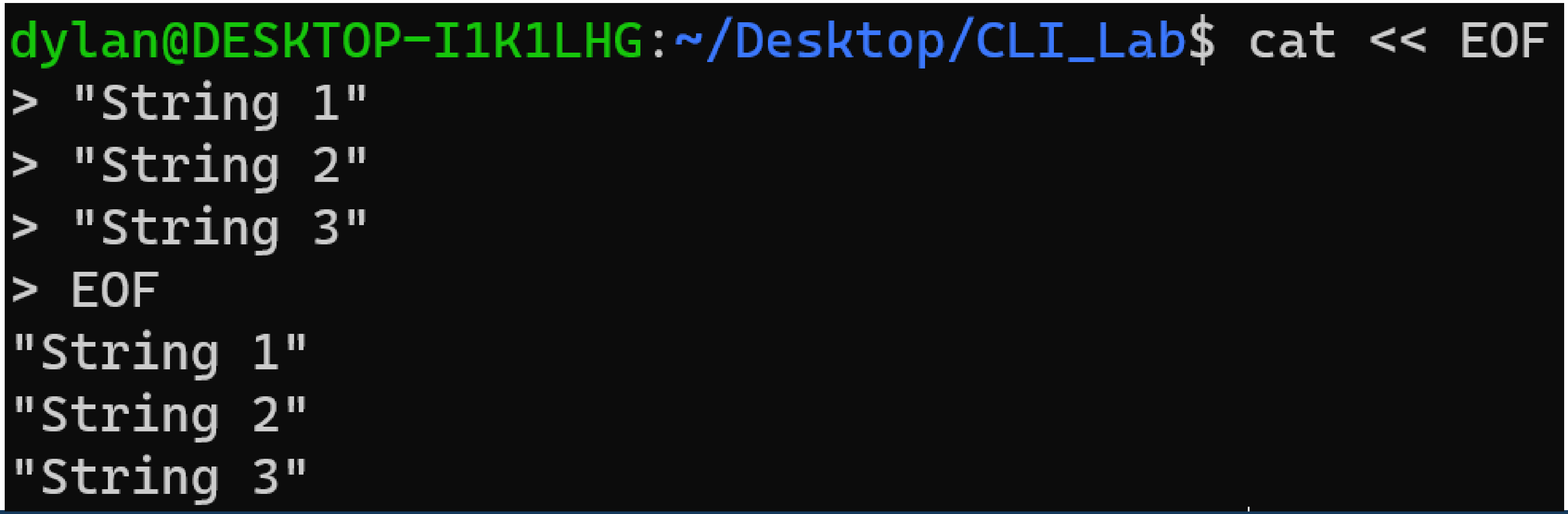
- Example:



```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ cat file2.txt
Hello World!
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ cat file1.txt >> file2.txt
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ cat file2.txt
Hello World!
Hello World!
```

<<

- Used to redirect inputs to a command until a delimiter is hit.
  - Being honest, rarely used. Wizards might know when you should use it.
- Usage: `[some_command] << [some_file]`
- Example:



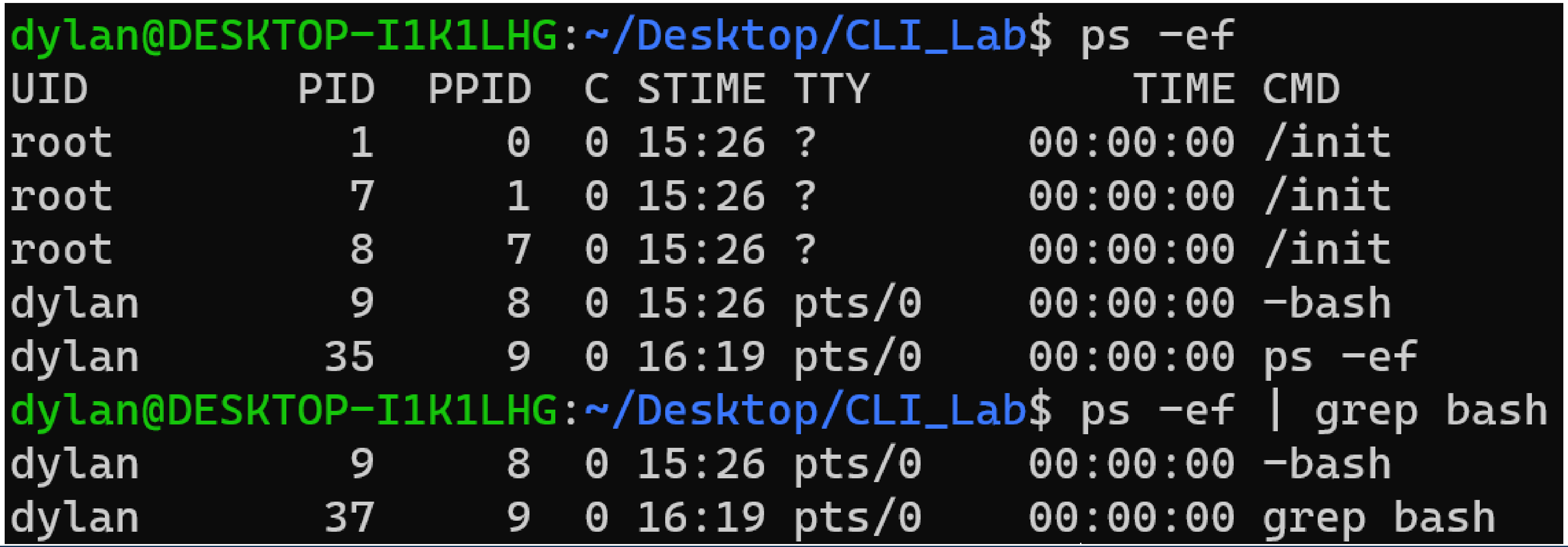
```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ cat << EOF
> "String 1"
> "String 2"
> "String 3"
> EOF
"String 1"
"String 2"
"String 3"
```

ACM@UAB

I/O Piping

| (This is not an "i", it's that character next to backslash (\))

- Sends the output of one command as the input of the next command
  - Allows us to chain commands together
- Usage: `<some_command> | <some_other_command>`
- Example:



```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 15:26 ?        00:00:00 /init
root         7     1  0 15:26 ?        00:00:00 /init
root         8     7  0 15:26 ?        00:00:00 /init
dylan        9     8  0 15:26 pts/0    00:00:00 -bash
dylan       35     9  0 16:19 pts/0    00:00:00 ps -ef
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ ps -ef | grep bash
dylan        9     8  0 15:26 pts/0    00:00:00 -bash
dylan       37     9  0 16:19 pts/0    00:00:00 grep bash
```

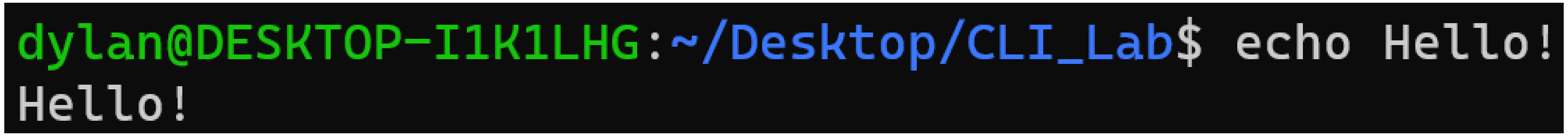
ACM@UAB

Miscellaneous Commands

echo

- Echos whatever is used as input.
- Usage: `echo <input>`

- Example:

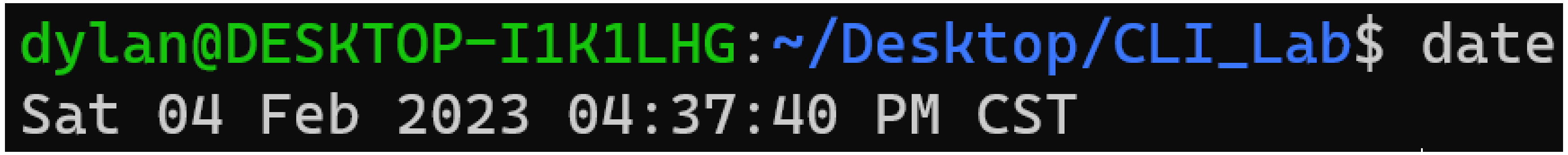


```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ echo Hello!
Hello!
```

date

- Displays the date and time.
- Usage: `date`

- Example:

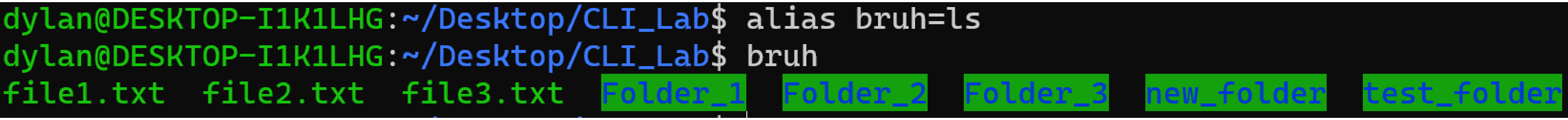


```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ date
Sat 04 Feb 2023 04:37:40 PM CST
```

alias

- Creates a new word to reference a command.
  - Note: if you type alias with no arguments, prints all aliases and their values.
- Usage: `alias <alias_name>=<command_to_alias>`

- Example:



```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ alias bruh=ls
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ bruh
file1.txt  file2.txt  file3.txt  Folder_1  Folder_2  Folder_3  new_folder  test_folder
```

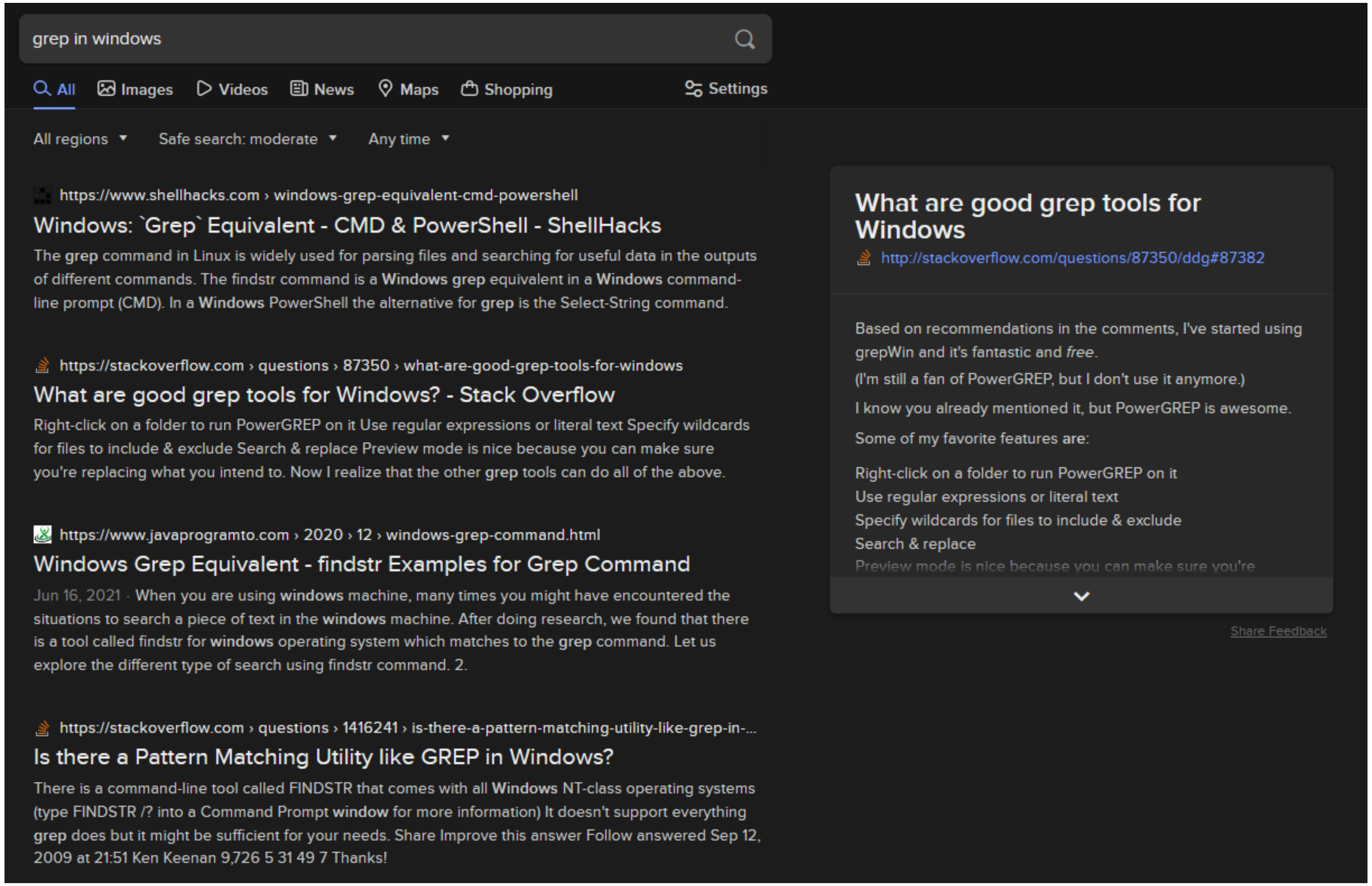
ACM@UAB

UNIX Only Section

Like I said before,

- The following commands are the UNIX only ones I talked about before.
- If you need/want to have a windows equivalent, just google what an equivalent command would be.
- UNIX commands can be used in WSL(2) on Windows.
    - To install WSL2, Virtualization must be enabled in your BIOS and in Windows.

For Example…



https://www.shellhacks.com › windows-grep-equivalent-cmd-powershell

Windows: `Grep` Equivalent - CMD & PowerShell - ShellHacks

The **grep** command in Linux is widely used for parsing files and searching for useful data in the outputs of different commands. The findstr command is a **Windows grep** equivalent in a **Windows** command-line prompt (CMD). In a **Windows** PowerShell the alternative for **grep** is the Select-String command.

https://stackoverflow.com › questions › 87350 › what-are-good-grep-tools-for-windows

What are good grep tools for Windows? - Stack Overflow

Right-click on a folder to run PowerGREP on it Use regular expressions or literal text Specify wildcards for files to include & exclude Search & replace Preview mode is nice because you can make sure you're replacing what you intend to. Now I realize that the other **grep** tools can do all of the above.

https://www.javaprogramto.com › 2020 › 12 › windows-grep-command.html

Windows Grep Equivalent - findstr Examples for Grep Command

Jun 16, 2021 · When you are using **windows** machine, many times you might have encountered the situations to search a piece of text in the **windows** machine. After doing research, we found that there is a tool called findstr for **windows** operating system which matches to the **grep** command. Let us explore the different type of search using findstr command. 2.

https://stackoverflow.com › questions › 1416241 › is-there-a-pattern-matching-utility-like-grep-in-...

Is there a Pattern Matching Utility like GREP in Windows?

There is a command-line tool called FINDSTR that comes with all **Windows** NT-class operating systems (type FINDSTR /? into a Command Prompt **window** for more information) It doesn't support everything **grep** does but it might be sufficient for your needs. Share Improve this answer Follow answered Sep 12, 2009 at 21:51 Ken Keenan 9,726 5 31 49 7 Thanks!

What are good grep tools for Windows

http://stackoverflow.com/questions/87350/ddg#87382

Based on recommendations in the comments, I've started using grepWin and it's fantastic and *free*.

(I'm still a fan of PowerGREP, but I don't use it anymore.)

I know you already mentioned it, but PowerGREP is awesome.

Some of my favorite features are:

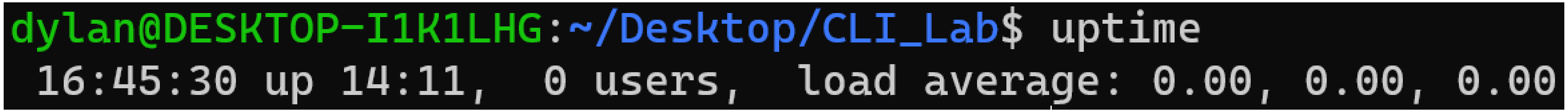
Right-click on a folder to run PowerGREP on it
Use regular expressions or literal text
Specify wildcards for files to include & exclude
Search & replace
Preview mode is nice because you can make sure you're

Share Feedback

uptime

- Displays the amount of uptime and load averages.
- Usage: `uptime`

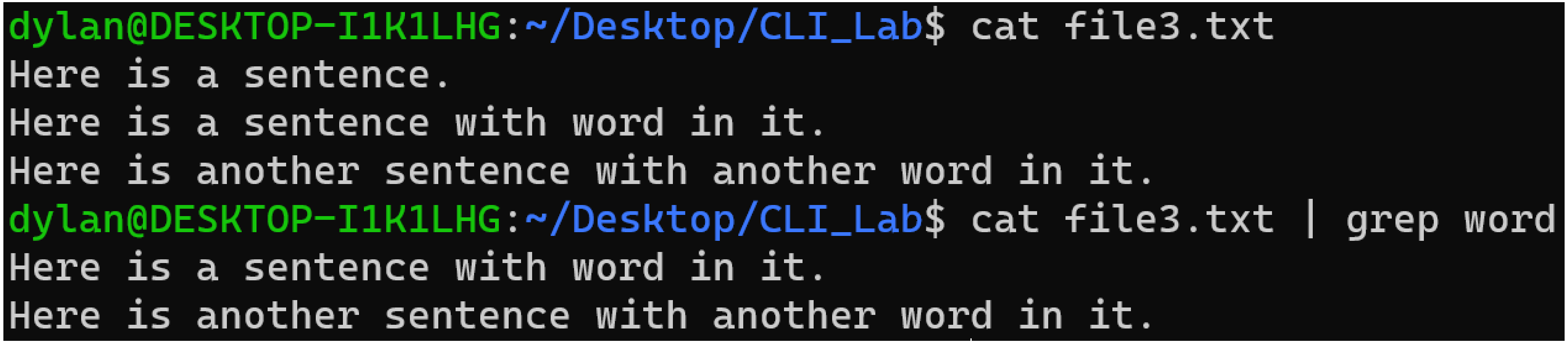
- Example:



```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ uptime
 16:45:30 up 14:11,  0 users,  load average: 0.00, 0.00, 0.00
```

grep

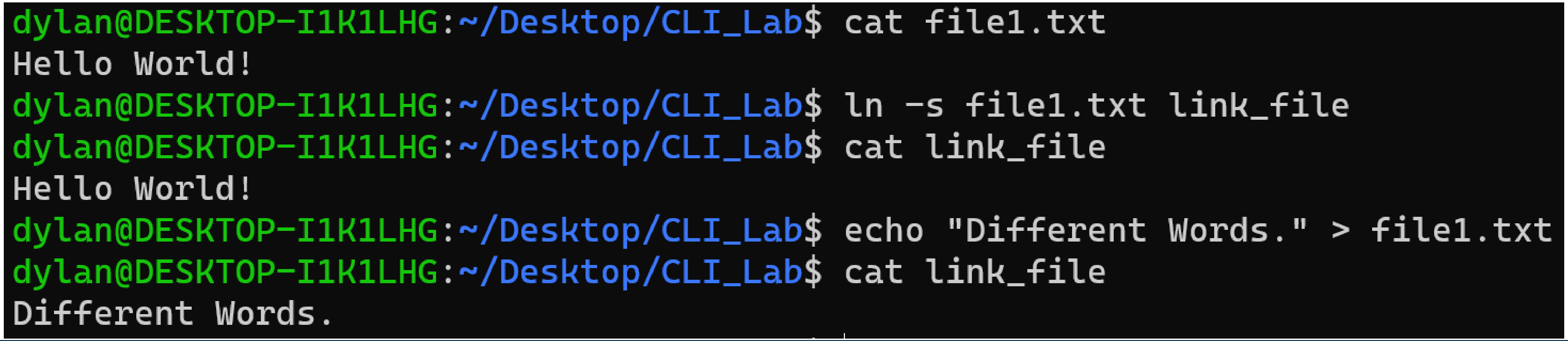
- Stands for "Get Regular Expression", it will use a text pattern to filter results.
  - Typically used in Pipes
- Usage: `grep <pattern>`
- Example:



```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ cat file3.txt
Here is a sentence.
Here is a sentence with word in it.
Here is another sentence with another word in it.
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ cat file3.txt | grep word
Here is a sentence with word in it.
Here is another sentence with another word in it.
```

ln -s

- Creates a Symbolic Link to a file (think like a shortcut)
- Unless you know why, **DO NOT FORGET THE –s!**
  - Forgetting the –s make it a *hard link*, which can and will break your filesystem.
- Usage: `ln -s <file_to_link> <link_name>`
- Example:

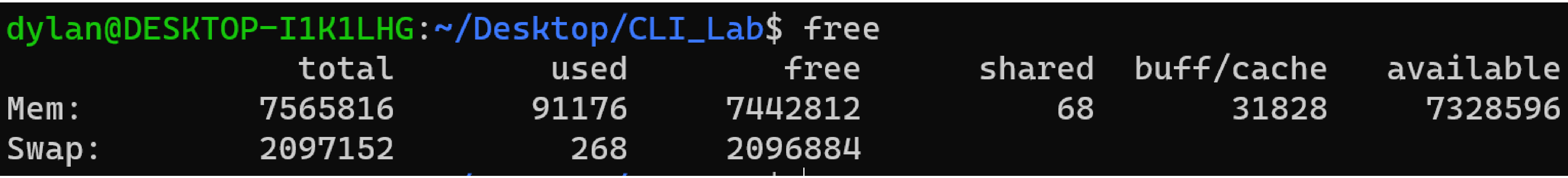


```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ cat file1.txt
Hello World!
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ ln -s file1.txt link_file
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ cat link_file
Hello World!
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ echo "Different Words." > file1.txt
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ cat link_file
Different Words.
```

free

- Shows memory information
- Usage: `free`

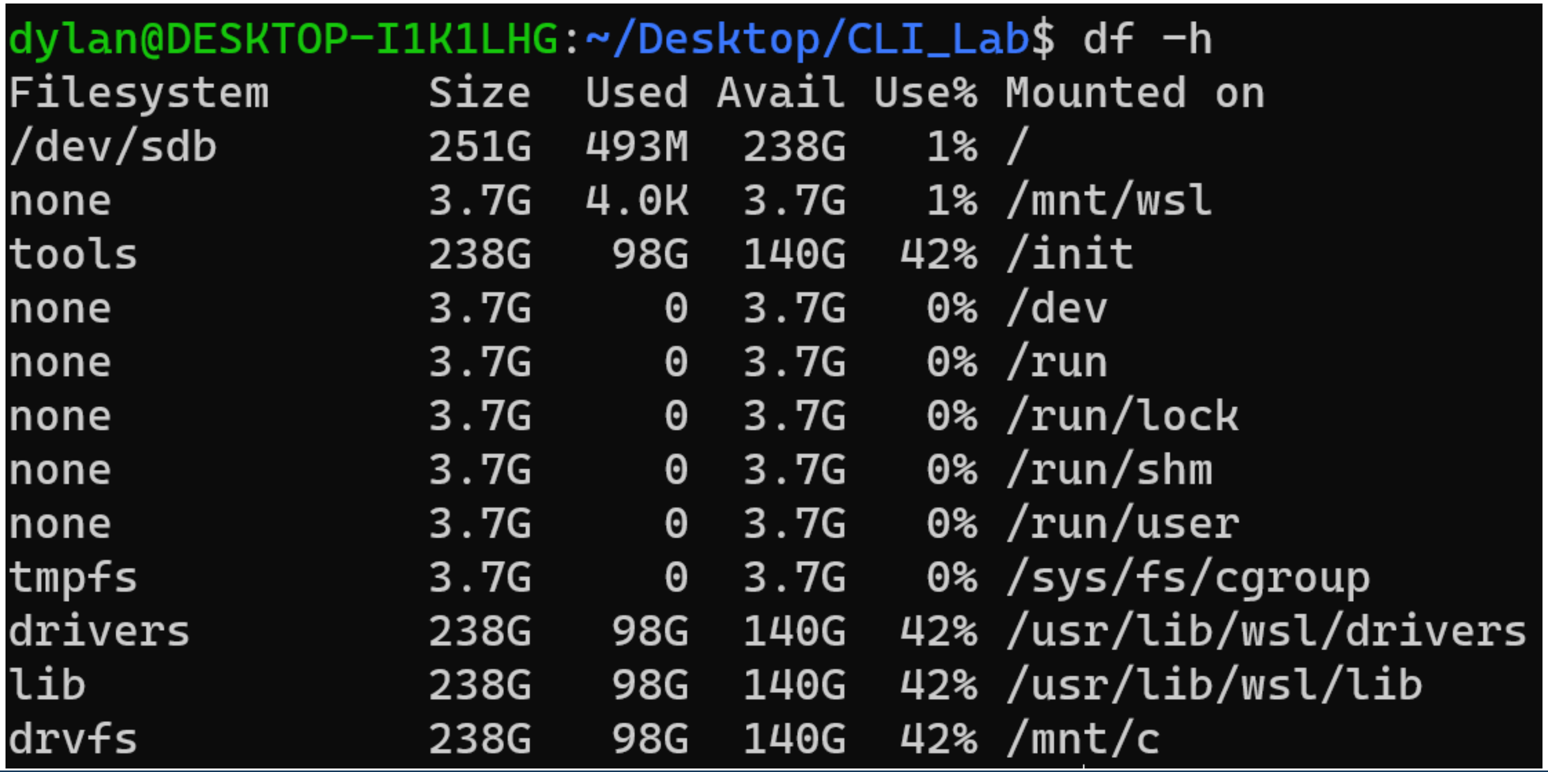
- Example:



```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ free
               total        used        free      shared  buff/cache   available
Mem:         7565816       91176     7442812          68       31828     7328596
Swap:        2097152         268     2096884
```

df -h

- Shows disk usage statistics
- Usage: df –h [option_flags] <optional_device>
- Example:

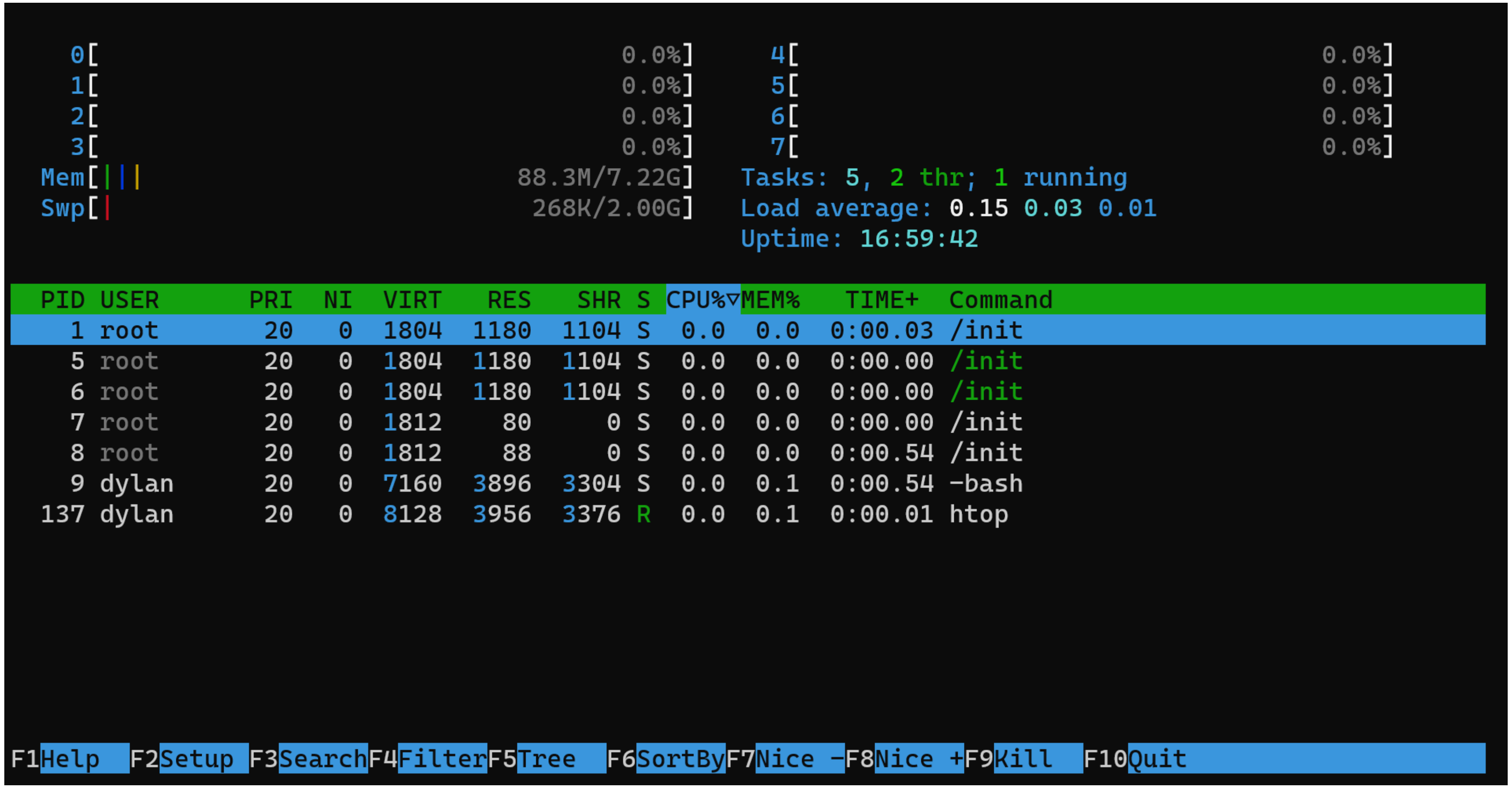


```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb        251G  493M  238G   1% /
none            3.7G  4.0K  3.7G   1% /mnt/wsl
tools           238G   98G  140G  42% /init
none            3.7G     0  3.7G   0% /dev
none            3.7G     0  3.7G   0% /run
none            3.7G     0  3.7G   0% /run/lock
none            3.7G     0  3.7G   0% /run/shm
none            3.7G     0  3.7G   0% /run/user
tmpfs           3.7G     0  3.7G   0% /sys/fs/cgroup
drivers         238G   98G  140G  42% /usr/lib/wsl/drivers
lib             238G   98G  140G  42% /usr/lib/wsl/lib
drvfs           238G   98G  140G  42% /mnt/c
```

htop

- Like free, but interactive and has live updating
    - Closest thing to a task manager in the terminal
    - Not installed on every machine by default
- Usage: `htop`

htop example

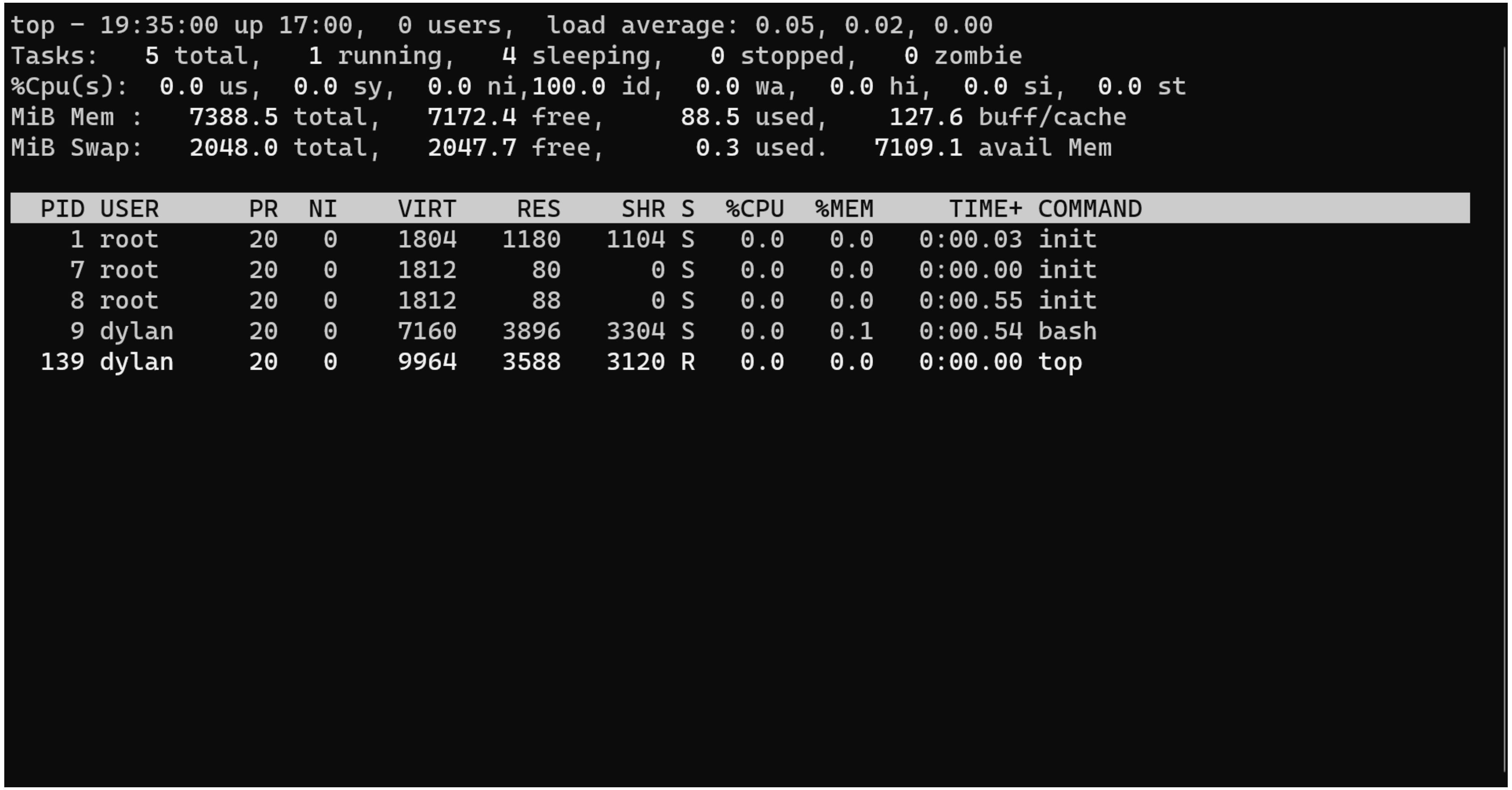


```
    0[                                     0.0%]     4[                                     0.0%]
    1[                                     0.0%]     5[                                     0.0%]
    2[                                     0.0%]     6[                                     0.0%]
    3[                                     0.0%]     7[                                     0.0%]
  Mem[|||                         88.3M/7.22G]   Tasks: 5, 2 thr; 1 running
  Swp[|                            268K/2.00G]   Load average: 0.15 0.03 0.01
                                                 Uptime: 16:59:42

  PID USER      PRI  NI  VIRT   RES   SHR S CPU%▽MEM%   TIME+  Command
    1 root       20   0  1804  1180  1104 S  0.0  0.0  0:00.03 /init
    5 root       20   0  1804  1180  1104 S  0.0  0.0  0:00.00 /init
    6 root       20   0  1804  1180  1104 S  0.0  0.0  0:00.00 /init
    7 root       20   0  1812    80     0 S  0.0  0.0  0:00.00 /init
    8 root       20   0  1812    88     0 S  0.0  0.0  0:00.54 /init
    9 dylan      20   0  7160  3896  3304 S  0.0  0.1  0:00.54 -bash
  137 dylan      20   0  8128  3956  3376 R  0.0  0.1  0:00.01 htop









F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

top

- Like htop but worse, but should be installed on all systems
- Usage: `top`

top example

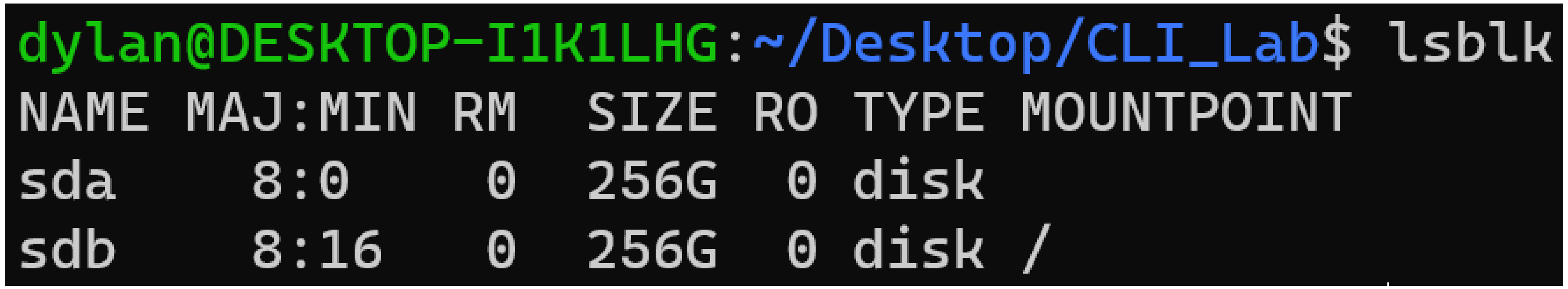


```
top - 19:35:00 up 17:00,  0 users,  load average: 0.05, 0.02, 0.00
Tasks:   5 total,   1 running,   4 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   7388.5 total,   7172.4 free,     88.5 used,    127.6 buff/cache
MiB Swap:   2048.0 total,   2047.7 free,      0.3 used.   7109.1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0    1804   1180   1104 S   0.0   0.0   0:00.03 init
    7 root      20   0    1812     80      0 S   0.0   0.0   0:00.00 init
    8 root      20   0    1812     88      0 S   0.0   0.0   0:00.55 init
    9 dylan     20   0    7160   3896   3304 S   0.0   0.1   0:00.54 bash
  139 dylan     20   0    9964   3588   3120 R   0.0   0.0   0:00.00 top
```

lsblk

- Lists physical storage devices (not including partitions like df –h)
- Usage: `lsblk`

- Example:

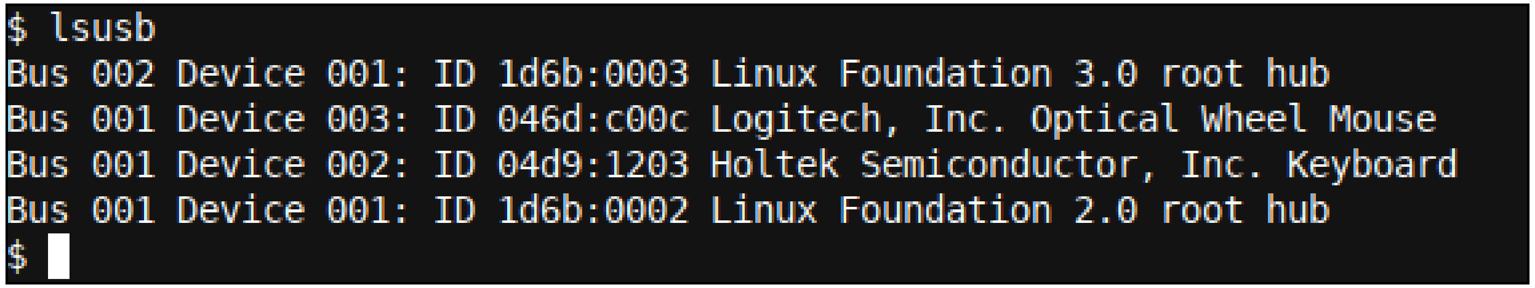


```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ lsblk
NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda    8:0    0  256G  0 disk
sdb    8:16   0  256G  0 disk /
```

lsusb

- Lists physical usb devices
- Usage: lsusb

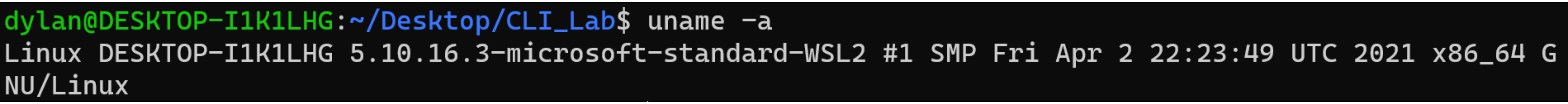
- Example:



```
$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 046d:c00c Logitech, Inc. Optical Wheel Mouse
Bus 001 Device 002: ID 04d9:1203 Holtek Semiconductor, Inc. Keyboard
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
$ 
```

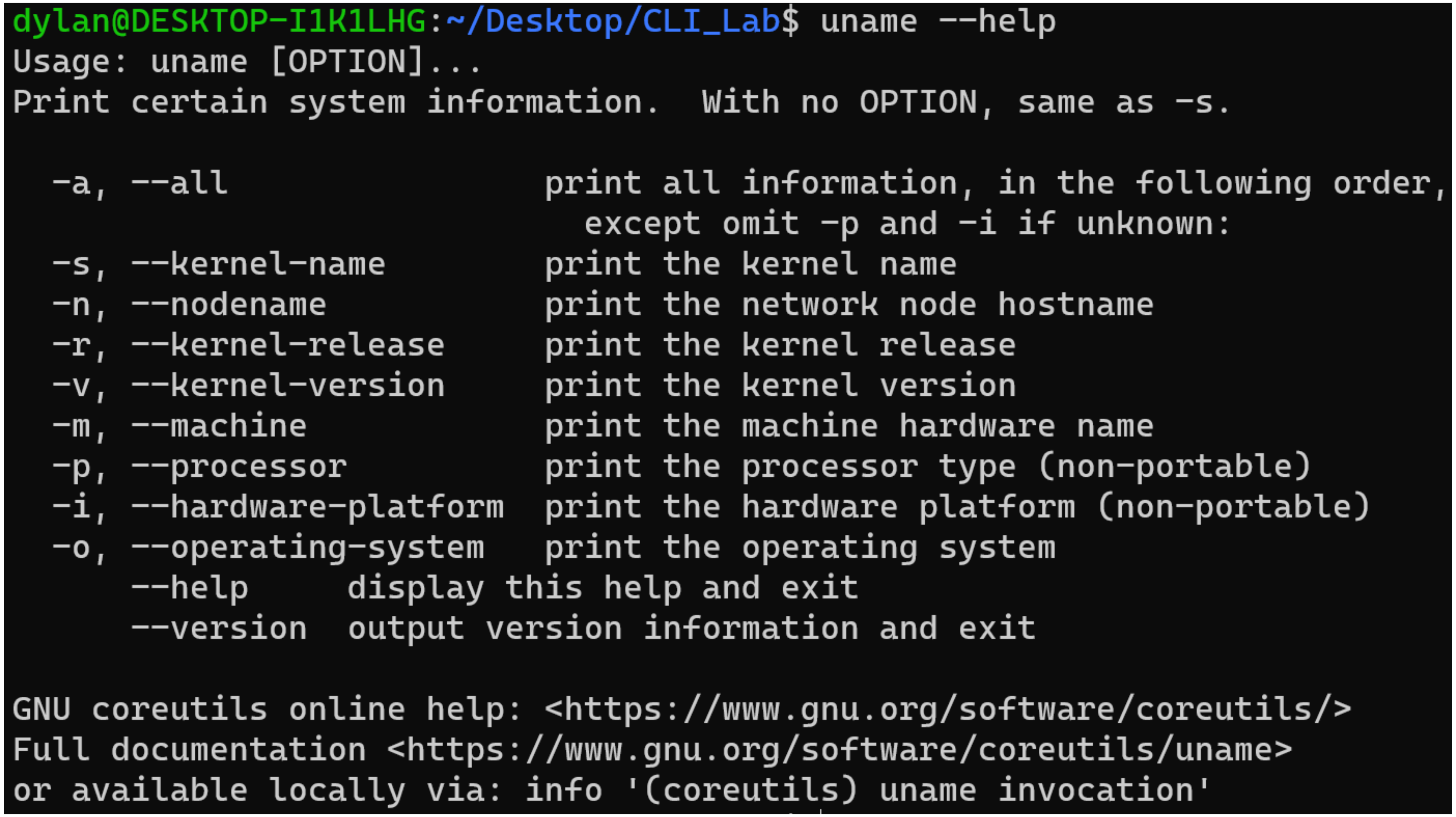
uname -a

- Prints all system information
- Usage: `uname -a`
- Example:



```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ uname -a
Linux DESKTOP-I1K1LHG 5.10.16.3-microsoft-standard-WSL2 #1 SMP Fri Apr 2 22:23:49 UTC 2021 x86_64 G
NU/Linux
```

uname flags



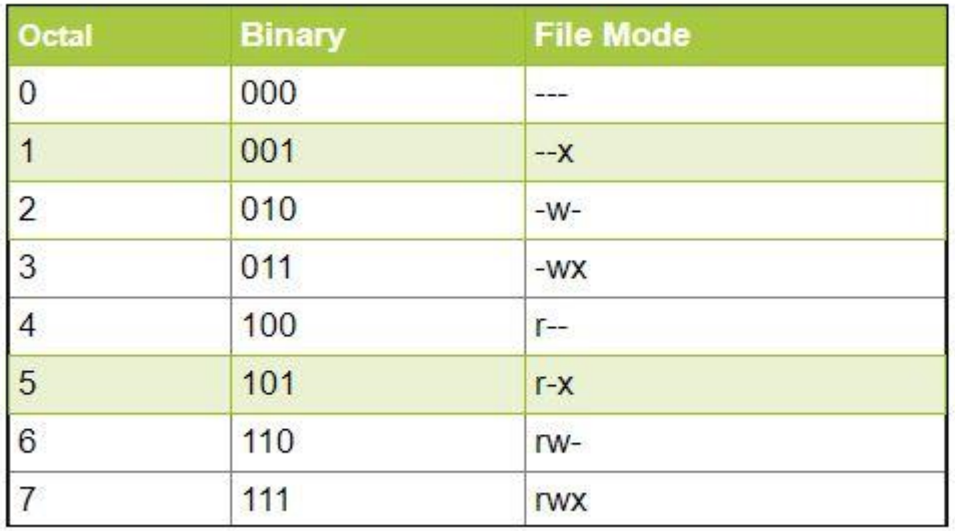
```
dylan@DESKTOP-I1K1LHG:~/Desktop/CLI_Lab$ uname --help
Usage: uname [OPTION]...
Print certain system information.  With no OPTION, same as -s.

  -a, --all                print all information, in the following order,
                             except omit -p and -i if unknown:
  -s, --kernel-name        print the kernel name
  -n, --nodename           print the network node hostname
  -r, --kernel-release     print the kernel release
  -v, --kernel-version     print the kernel version
  -m, --machine            print the machine hardware name
  -p, --processor          print the processor type (non-portable)
  -i, --hardware-platform  print the hardware platform (non-portable)
  -o, --operating-system   print the operating system
      --help     display this help and exit
      --version  output version information and exit

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation <https://www.gnu.org/software/coreutils/uname>
or available locally via: info '(coreutils) uname invocation'
```

ACM@UAB

File Ownership (UNIX Only)

Ownership Masks (755 format)

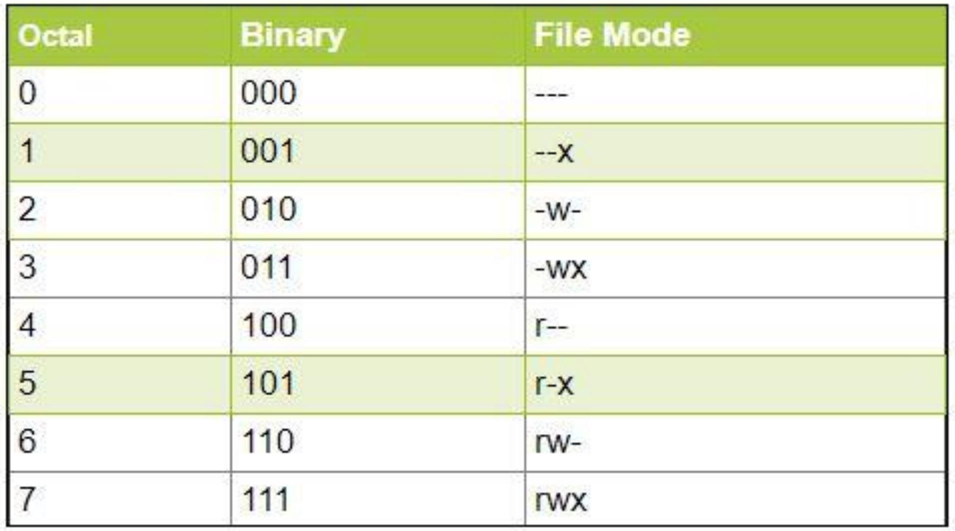
- Permissions in UNIX systems are split into 3 groups,
  - Owner
  - Group
  - Everyone
- The number that represents permissions for each group goes from 0 to 7
  - 0 = no permissions at all
  - 7 = full read/write/execute permissions



Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Ownership Masks (drwx format)

- Just like 755 format, the drwx format represents permissions for the 3 groups.
  - R = Read permissions
  - W = Write Permissions
  - X = Execute Permissions
- If there is a D at the beginning of the 10 character string, it means that file is a directory.
  - I prefer drwxrwxrwx because of this.



Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Pop Quiz

- What permissions does the owner have from the mask 755?
- What about -rw-rw-rw?

- What permissions does everyone have from the mask 777?
  - Why is this not a good idea?
  - What would this mask look like in drwx format?

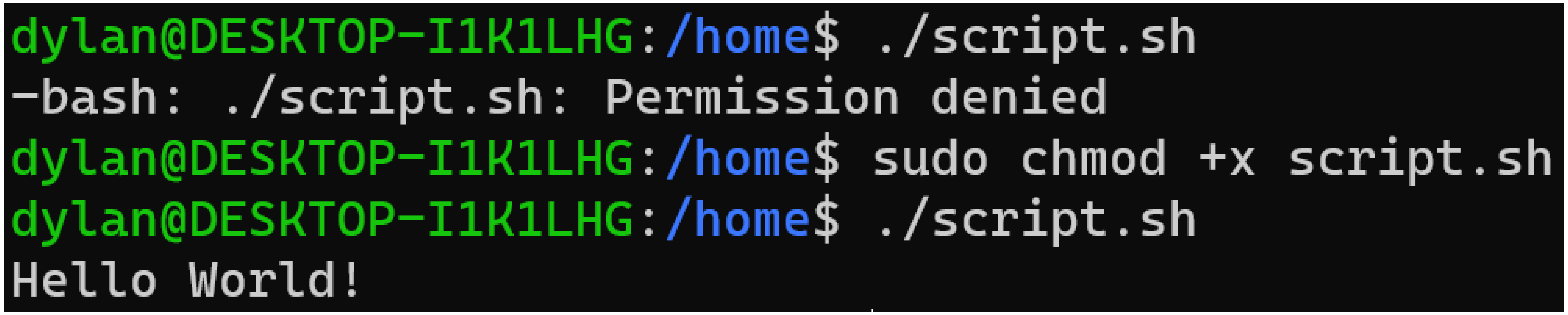
(Almost) Never use 777!

- This gives everybody full read/write/execute permissions on that file.
  - Even people on the internet can edit and access this file this way!
  - It can also break your operating system.
- It is general practice to use 755, so that only the owner of the file can execute that file.

chmod

- Used to add/remove a permission to a file
  - Generally requires admin privileges
- Usage: `chmod <+/-><permission_letter> <file>`

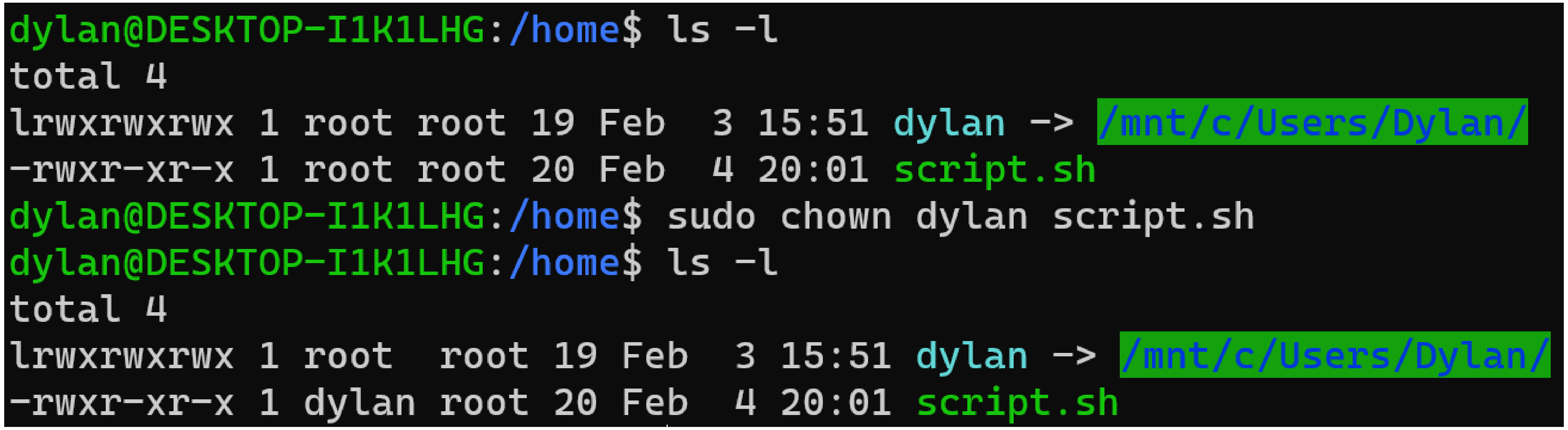
- Example:



```
dylan@DESKTOP-I1K1LHG:/home$ ./script.sh
-bash: ./script.sh: Permission denied
dylan@DESKTOP-I1K1LHG:/home$ sudo chmod +x script.sh
dylan@DESKTOP-I1K1LHG:/home$ ./script.sh
Hello World!
```

chown

- Used to change the owner of a file
  - There can only be one owner.
  - Also generally requires admin privileges.
- Usage: `chown <username> <file>`
- Example:



```
dylan@DESKTOP-I1K1LHG:/home$ ls -l
total 4
lrwxrwxrwx 1 root root 19 Feb  3 15:51 dylan -> /mnt/c/Users/Dylan/
-rwxr-xr-x 1 root root 20 Feb  4 20:01 script.sh
dylan@DESKTOP-I1K1LHG:/home$ sudo chown dylan script.sh
dylan@DESKTOP-I1K1LHG:/home$ ls -l
total 4
lrwxrwxrwx 1 root  root 19 Feb  3 15:51 dylan -> /mnt/c/Users/Dylan/
-rwxr-xr-x 1 dylan root 20 Feb  4 20:01 script.sh
```

ACM@UAB

You Made It!

I’ve only scratched the surface.

- There are tons more commands you can learn that do more.
- The more you use the command line, the more you will learn and make it easier to use.
- The best teacher will be practice.
- There are lots of free resources to learn from, such as…

Resources to learn from

- TLCL: A free textbook about the Linux Command Line
  - https://sourceforge.net/projects/linuxcommand/files/TLCL/19.01/TLCL-19.01.pdf/download
- A mostly complete list of all windows command prompt commands
  - https://www.lifewire.com/list-of-command-prompt-commands-4092302
- A-Z Linux Commands
  - https://linuxhandbook.com/a-to-z-linux-commands/
- Linux Command Cheat Sheet
  - https://phoenixnap.com/kb/linux-commands-cheat-sheet
- These Slides will be available to download to use for reference!
  - https://acmatuab.org/workshops/

ACM@UAB

Questions?