Command Line + BASH Scripting

Nel Escher
Agenda

• Command line
  • Working with absolute and relative paths
  • Running programs
  • Redirecting program output

• Scripting
  • Automate sequences of commands!
Say Goodbye to Your Precious Windowed GUI
Open up the Command Line Interface

• On Mac – Terminal
• On PC – Ubuntu
Mac Users!!

- Terminal --> Preferences --> Pick a color scheme that speaks to you
The Shell

• You type commands into the shell
• Operating system performs those commands

• The jobs of a shell
  • Spawn (launch) new programs
  • Handle input and output to programs
  • Kill and clean up old programs
Navigating the file system
Absolute paths

• A path that specifies the location of a file or directory from the root directory (/)

• To write an absolute pathname:
  • Start at the root directory (/) and work down.
  • Write a slash (/) after every directory name

/Users/root/Desktop/
/Users/root/Documents/DataCamp/
/Users/root/Documents/DataCamp/shell_slides.pdf
Relative paths

- Relative path is defined as the path related to the present working directly. It starts at your current directory and never starts with a / .

Documents/
Documents/DataCamp/
• if we are looking for photos then absolute path for it will be provided as `/home/jono/photos` but assuming that we are already present in jono directory then the relative path for the same can be written as simple `photos`.

https://www.geeksforgeeks.org/absolute-relative-pathnames-unix/
pwd

- **Print Working Directory**
- Prints the absolute path of the working directory, starting from the root
cd

• Change directory
• cd directory_name/
  • Change directory “down” a level to a folder inside working directory
• cd ..
  • Change directory “up” a level to the folder that contains the working directory
• if we are already present in jono directory, then after issuing the command we will be in the work directory

```
cd work/
```
It’s a similar idea to a GUI folder interface
My working directory is Documents/
By double clicking, I’ll change directory (cd) to Data\ Camp/
• if we are already present in jono directory, then after issuing the command we will be in the home directory

```
    cd ..
```
My working directory is `Data\Camp/`
By clicking the back button, I’ll change directory to the Documents/ folder (like `cd ..`
Commands

- **pwd**
  - Print working directory

- **ls**
  - List files and directories

- **cd**
  - Change directory

- **cat [filename]**
  - E.g. `cat clue1.txt`
  - Print the contents of the file
Scavenger Hunt

- `cat [filename]`
  - E.g. `cat clue1.txt`
  - Print the contents of the file
- `ls`
  - List files and directories inside working directory
- `cd directory_name/`
  - Change directory “down” a level to a folder inside working directory
- `cd ..`
  - Change directory “up” a level to the folder that contains the working directory
- `pwd`
  - Print working directory
  - Use this if you get lost!

Hot tip! Use the tab button to autocomplete file + folder names
The Python Program: The Interactive Interpreter

• cd into the python/ folder
• Start up the python program by running the python command
  $ python
• You can try running lines of python code in this interactive interpreter
  >>> (10 * "dog")
• When you want to go back to the command line
  >>> exit()
Running Python Files

• Format of the command:
  $ python <filename>.py

• Run the python program and pass it the file hello.py
  $ python hello.py

Try it!
• Run the python program and pass it the file hello_lots.py
Passing Command Line Arguments to Python Files

• For some programs, you can change behavior by providing additional arguments
• Run the python program and pass it the file hello_name.py and a string
  
  $ python hello_name.py nel

Try it!

• Run the python program and pass it the file hello_name.py and the name of your dearest pal
Passing Relative Paths as Command Line Arguments to Python Files

Make use of relative paths if you wish to pass in a file that is in a different directory!

$ python cleaner.py data/dracula.txt

OR

$ cd data/
$ python ../cleaner.py dracula.txt
File Redirection

• **Operators**
  < send file as input
  > send output to file (create/overwrite)

• **Try it!**
  $ python hello_lots.py > hello_lots_out.txt
  $ cat hello_lots_out.txt

Run the python program, pass it the file hello_name.py and your name, and save the output in a file hello_to_me.txt
Putting it together

$ python3 cleaner.py data/dracula.txt > intermediate/cleaner_dracula.out
$ cat intermediate/cleaner_dracula.out

Try it out! Can you clean up huckleberry.txt and save the cleaner version as cleaner_huckleberry.out in the intermediate/ folder?
What are some other cool programs that can be run at the command line?

• git
  • Version control!
  • Good for collaborating on coding projects
• vi
  • Text editor you can use inside the shell
• diff
  • Compare two different files and get the lines where they are different

Programs you write yourself!
Flags

• Sometimes you can change how a program or command works by including flags

$ ls
native packages props repCache systemDialogs weka.log

$ ls -a
.
  native props systemDialogs wekaMetaStore ..
packages repCache weka.log
How do I know what I can do with a program?

• **man**
  - Manual
  - Has documentation for programs
  
  $ man python

• **help**
  - Provides help for bash built-in commands
  
  $ help cd
What about scripting?

• Surprise! You've been scripting this whole time!

• Typing commands into the bash shell and running a bash script are the same

  $ cat test.sh
  
  python hello.py > hello.txt
  
  cat hello.txt

  $ chmod +x test.sh # makes your file an executable

  $ ./test.sh
How to write a bash script?

• Try things out in the terminal
• Copy things that work into a file ($ history)
• Run that file
• Repeat
Bash

• Bash is old...
• But useful, especially for really short things
• But has ugly and finicky syntax
• But running programs is really easy
• (it's what it was built for after all)
• `g++ -O3 -m32 thread.o libinterrupt.a test1.cpp -ldl -o test1`
• `./test1`
Scripting

• First line of scripts:
  
  ```bash
  #!/bin/bash
  ```

• Special variables
  • $0 current script
  • $n script args 1, 2, 3...

• Other variables, math, if/then, etc. are available
Let’s run a script!

Make sure yr working directory is the python/ folder

$ chmod +x bin/hello.sh
$ ./bin/hello.sh

Try it out! Try to run the script located at bin/excessive_greetings.sh
Let’s run a cooler script!

Make sure yr working directory is the python/ folder

This script takes two arguments

$ chmod +x bin/hello_cooler.sh
$ ./bin/hello_cooler.sh nel hi_to_nel.txt

Try it! Run the script with your own name and filename. Use cat to verify file contents

Then, open up the hello_cooler.sh file in a text editor (Sublime, Atom, Notepad, etc.) and take a look at the syntax
Scripting exercise – the main idea

• We will be making a script that runs a series of python commands
• Given a book that has chapters, we will count up how many times each word appears in each chapter

input

CHAPTER I.

YOU don't know about me without you have read a book by the name of The Adventures of Tom Sawyer; but that ain't no matter. [...]
Scripting exercise – the python files

cleaner.py
• Takes in a text file
• Outputs that text file in all lowercase and common punctuation removed

chapter_word.py
• Takes in a text file that contains chapters
• Outputs each word in the text file along with the chapter in which it appears (a key/value pair)

key_val_total.py
• Takes in a key value pair
• Prints that key value pair and how many times that key value pair

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER I.</td>
<td>chapter i</td>
</tr>
<tr>
<td>YOU don't know</td>
<td>you dont</td>
</tr>
<tr>
<td>chapter i you</td>
<td>i you</td>
</tr>
<tr>
<td>you dont know</td>
<td>i dont</td>
</tr>
<tr>
<td>i know</td>
<td>i know 3</td>
</tr>
<tr>
<td>i you</td>
<td>i you 9</td>
</tr>
<tr>
<td>i dont 4</td>
<td></td>
</tr>
<tr>
<td>i know</td>
<td></td>
</tr>
</tbody>
</table>
Now you make a script!

• Your bash script will take two arguments – the file you want to process and the location of the final output
  • Reference the first argument to the scripts using $1
  • Reference the second argument to the scripts using $2
• Tip: try running these three python files on the command line before sticking them in your script
  • (Follow the comments in process_book.sh for implementation details)

Example runs:
$ ./bin/process_book.sh data/huckleberry.txt output/huckleberry.out
$ ./bin/process_book.sh data/dracula.txt output/dracula.out
Check out that sweet sweet data

```python
$ python
>>> import pandas
>>> data = pandas.read_csv('output/huckleberry.out', sep=" ", header=None, names=['chapter', 'word', 'count'])
```