Sysadmin DeCal Lab 2 - Core Shell & Shell Scripting Facilitator: Joey Li, Aaron Zheng **Toggle Dark Theme** 11 min read TABLE OF CONTENTS 1. Setting up 1 Setting up 2. A quick intro to vim 2 A quick intro to vim 1. Why vim? a Why vim? 2. Hello World b Hello World 3. The vim Modes c The vim Modes 4. Questions 3. A quick intro to tmux(Optional) a Normal mode: 1. Why tmux? b Insert mode: 2. Getting Started: c Visual mode: 3. Questions (Optional) d Questions 4. Scripting 3 A quick intro to tmux(Optional) 5. Why Scripting? a Why tmux? 6. Scripting Lab Assignment b Getting Started: 7. Skeleton Code c Questions (Optional) 8. Some tips to make things easier 4 Scripting 9. Submitting the lab 5 Why Scripting? 10. Additional Resources 6 Scripting Lab Assignment Skeleton Code 8 Some tips to make things easier 9 Submitting the lab 10 Additional Resources Welcome to Lab 2! In this lab you will be learning how to work productively in a shell and use that to write your first shell script. Remember to submit your answers in the Gradescope assignment! Don't forget to use Google and man when stuck. The resources linked at the bottom may be helpful as well. "I SPEND A LOT OF TIME ON THIS TASK.
I SHOULD WRITE A PROGRAM AUTOMATING IT!" THEORY: WRITING-CODE FREE TIME WORK AUTOMATION TAKES OVER WORK ON-ORIGINAL TASK TIME REALITY: ONGOING DEVELOPMENT DEBUGGING WRITING CODE RETHINKING NO TIME FOR WORK ORIGINAL TASK ANYMORE TIME Setting up This lab can be completed either on tsunami by logging in using ssh or on the VM you set up in the previous lab. This lab requires a bash shell, vim, and tmux. If you do not have tmux or vim installed: sudo apt install vim tmux A quick intro to vim vim is a very widely used text editor. It's well known for its customizability and plethora of keybinds. While it may be somewhat unintuitive to use at first (since a lot of common keybinds for things like copy-paste, saving, or exiting don't do what you think they will), it's well worth learning about, and you'll certainly come across it all the time when working in the shell! Why vim ? It's a descendant of vi, which was written in Berkeley by Bill Joy, who went on to found Sun Microsystems. Sometimes you will be suddenly thrown into vim via merging git conflicts or other programs. It's included in practically every UNIX environment. You can be very productive when familiar with it. Hello World To get started with learning vim, run the command vimtutor. This will walk you through the below material in an interactive manner! You aren't required to finish the entire tutorial, but we encourage you to at least complete lesson 1. You can then use this section as reference if you forget anything. The vim Modes Vim is a modal text editor, meaning that you can change editing modes in order to do different things. There are 3 primarily used modes: **Normal**, **Insert**, and **Visual** mode. NORMAL MODE: Used for moving around and performing actions hjkl to move left, up, down, and right (arrow keys work too but hjkl is usually faster to use!) G to move to end of file, gg to move to beginning i to enter insert mode (a, o also change mode in different ways) dd to cut a line yy to copy a line p to paste / to search u to undo Type in commands with: Save with :w Exit with :q Explore more commands online! Here's a cool cheat sheet to get you started. **INSERT MODE:** 6. index.js + (~/s/vim) - VIM (Vim) 28 * @method create 29 * @param {String} name an identifier that can used for retrieval later 30 */ 31 module.exports.create = create; 32 33 /** * Get a single instance by name. This is useful if you have your 35 * build scripts in separate files * @method get * @param {String} name * @returns {Object|Boolean} 39 */ 40 module.exports.get = function (name) { instance = gSingle(name) if (instance) { 43 return instance; 44 45 throw new Error("An instance with the name `%s` was not found.".replace("%s", name)); -- INSERT -ig Used for editing text like a usual editor Arrow keys to move Esc to exit to normal mode (lots of people bind it to Caps Lock) VISUAL MODE: Enter with v from normal mode Used to select text visually Modify selection with normal mode movement commands Use o to move the cursor to the other side of the selection Yanking, deleting, and pasting use y , d , p (sound familiar?) A key feature of vim is chaining together commands. Normal mode is essentially a massive amount of shortcuts that you can combine to quickly navigate and edit a file. Want to move down 3 lines? You know that j means move down 1 line, so you can use 3j to move down 3. d is for deletion and w is to jump to the next word, so what does dw do? Questions Try playing around with lab2.md while looking up some new commands. Use wget to download it! 1 How would you delete the previous 10 lines? 2 How would you jump back to the shell without exiting vim? 3 How would you edit a new file alongside another file? 4 How would you indent a block of text? Tell us about one other cool vim feature you found out about that isn't mentioned in this lab! A quick intro to tmux(Optional) While we reccomend that you complete this section of the lab, it's completely optional and will not affect your lab grade. Feel free to skip ahead to the Scripting Section. Tmux Tips # New random direction: ((n=\$RANDOMS\$S-1)) ((n=(\$m1||\$m=0)?\$1:\$1=\$n)) ((n=(\$m<0)?3:\$mM4)) tput cup Sy Sx echo *\033[1:35(c)a\2c2\x945(v[\$15n]\c:* ((\$tv\$r)) && tput clear && t=0 || ((t++)) || =\$n Hostname: vpn3-144170.near.uiuc.edu Distro: OS X 10.9.1 Kernel: Darwin Uptime: 20:27 Shell: /bin/zsh Terminal: screen >> tweet "I'm Tired." cores dev efi_changed Why tmux? · You can open multiple windows when sshed into a machine. You can go compile and run programs while editing them. · You can logout and ssh back in without having to reopen all your files. **Getting Started:** Start a session with tmux. Detach from a session with Ctrl-b d (press d after releasing Ctrl-b) Split into 2 panes with Ctrl-b % (vertical) or Ctrl-b " (horizontal) Swap current pane with Ctrl-b o Find more information about tmux online. You might find [this cheat sheet] [Tmux cheat sheet] helpful! **Questions (Optional)** 1 Make a new tmux session. Using tmux shortcuts, try to make your session have a similar layout to the one below, and upload a screenshot of it to Gradescope! bencuan@bigrip:~\$ neofetch bencuan@bigrip:~\$ Hello World! bencuan@bigrip OS: Debian GNU/Linux 9.12 (stretch Model: Standard PC (i440FX + PIIX, Kernel: 4.9.0-12-amd64 Uptime: 97 days, 9 hours, 14 minut Packages: 985 Shell: bash 4.4.12 CPU: Intel Xeon E312xx (Sandy Brid GPU: Cirrus Logic GD 5446 Memory: 3015MB / 3955MB `Y\$\$ `Y\$\$b. "Y\$b.___ bencuan@bigrip:~\$ Tasks: 273, 1808 thr; 4 running 58.0%] Load average: **15.75 7.94** 6.05 Uptime: 97 days, 09:15:20 ||||2.94G/3.86G] SHR S CPU% MEM% 15771 root 901M 458M 22160 S 13.5 11.6 38h26:10 kube-apiserver --advertise-address=172.17.0.4 0 3467M 89672 1440 S 12.9 2.2 16643 bencuan 118h java -jar server.jar 20 **117h** java -jar server.jar 0 3467M 89672 1440 S 12.9 2.2 16666 bencuan 20 0 627M 362M 22932 S 12.3 9.2 6h19:41 kube-apiserver --advertise-address=172.17.0.2 11291 root 16147 root 0 1456M 62848 6160 S 9.7 1.6 **37h**45:31 /var/lib/minikube/binaries/v1.18.2/kubelet --au 13605 ocfdeploy 20 0 144M 35744 8380 S 8.4 0.9 30:17.72 /usr/bin/cdi-operator /bin/bash 26071 root 0 1439M 58612 10600 S 7.7 1.4 **18h**05:02 /var/lib/minikube/binaries/v1.18.2/kubelet --au 5810 ocfdeploy 20 0 144M 34216 8972 S 7.1 0.8 30:28.32 /usr/bin/cdi-operator /bin/bash 0 1638M 43056 2672 S 5.8 1.1 9h43:08 /usr/bin/dockerd -H tcp://0.0.0.0:2376 -H unix 25454 root 9092 bencuan 0 17420 4536 2492 R 5.2 0.1 0:08.65 htop 788 root 14317 root F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7Nice -F8Nice +F9Kill F10Quit "bigrip" 19:54 23-Jun-20 Some things to note: The top left panel is resized. By how much, it doesn't matter. The top right panel is named "Hello World". (You can see this name displayed on the bottom left.) · You don't need to run any of the commands I did, but they do look pretty cool :) Try to figure out what command the bottom panel is running, and what it does! Don't worry about copying the layout exactly. The purpose of this exercise is simply to help you get comfortable making custom layouts in tmux. 1 If you haven't already, detach from your current tmux session using Ctrl+b d . Now, what command would you type to attach back to it? 2 What command will delete your session?

```
3 What command will create a new session?
Scripting
Why Scripting?
Many of the tasks that someone would like to perform on a computer are regular, require repetition, or
are menial or tedious to do by hand. Shell scripting allows one to interact programmatically with a
shell to do certain tasks. For example, the command for scanning log files in the previous topic guide
could be automated to be performed on a schedule by means of a shell script. bash scripts are an
incredibly powerful tool for sysadmins to automate tasks that are otherwise difficult to remember or
long-running.
In cases where shell syntax is inappropriate for the task at hand, one can instead call into programs
written in other languages, such as Python, which can read from stdin, process data, and write to
stdout.
Before you jump into the assignment we reccomend reading through our Scripting Reference which
provides a introduction to Bash and Python(optional part in this lab) scripting in Linux. Feel free to
come back to this reference as you work through the assignment.
Scripting Lab Assignment
You'll be completing a classic first shell scripting assignment: make a phonebook.
Write a shell script phonebook which has the following behavior:
• _/phonebook new <name> <number> adds an entry to the phonebook. Don't worry about
```

duplicates (always add a new entry, even if the name is the same).

phonebook has no entries, display phonebook is empty

./phonebook clear deletes the entire phonebook.

\$./phonebook new "Linus Torvalds" 101-110-0111

\$./phonebook new "Tux Penguin" 555-666-7777

\$./phonebook new "Linus Torvalds" 222-222-2222

name is not in the phonebook.

For example,

\$./phonebook list

\$./phonebook list

OPTIONAL BEHAVIOR

ALTERNATIVE BEHAVIOR

\$./phonebook clear

Skeleton Code

\$ cat foobar.txt

\$ cat foobar.txt

\$ cat foobar.txt

\$ echo "hello" > foobar.txt

of a line). For example:

\$ cat foobar.txt

\$ cat foobar.txt

\$ cat foobar.txt

\$./argscript.sh foo bar

jello world

echo "\$1"

echo "\$2"

end of file

foo

bar

structure

!/usr/bin/python

hello from python

Submitting the lab

for some of the questions.

Additional Resources

contents of fileman.py

\$./argscript.py foo bar

foo

bar

\$ sed -i "s/h/j/g" foobar.txt

hello 123

jello 123

hello, reader

foobar

foobar

file.

foobar

bash.

101-110-1010

222-222-2222

Linus Torvalds 101-110-1010

Linus Torvalds 101-110-1010

Linus Torvalds 222-222-2222

\$./phonebook lookup "Linus Torvalds"

task, you can submit your work on Gradescope.

Some tips to make things easier

\$ echo "hello, reader" >> foobar.txt

argument to the end of the second argument.

Tux Penguin 555-666-7777

./phonebook list displays every entry in the phonebook (in no particular order). If the

./phonebook remove <name> deletes all entries associated with that name. Do nothing if that

./phonebook lookup <name> displays all phone number(s) associated with that name. You can

NOTE: You can print the name as well as the number for each line. For an additional challenge,

try printing all phone numbers without their names. (See the example below for more details)

assume all phone numbers are in the form ddd-ddd-dddd where d is a digit from 0-9.

\$./phonebook lookup "Linus Torvalds" Linus Torvalds 101-110-1010 Linus Torvalds 222-222-2222 \$./phonebook remove "Linus Torvalds" \$./phonebook list Tux Penguin 555-666-7777

\$./phonebook list phonebook is empty If you run into an edge case that isn't described here, you can handle it however you wish (or don't handle it at all). You can assume all inputs are in the correct format.

To help you in this task, skeleton code for this lab can be found here. Once you are done with this

As an optional (but recommended) assignment: Try implementing the same Phonebook behavior,

but in python! This will highlight some of the strengths and weaknesses between the two languages.

If you're already familiar with python, you may find it helpful to do this before implementing it in

bash has an append operator >> which, as you might guess, appends the data from its first

bash also has a redirect operator > , which takes the output of one command and outputs it to a

• In bash, changing lines can be done through the sed command. If you wish to do so, the format

sed -i "s/<old>/<new>/g" ./filename may be helpful in this lab (e.g. for deleting a line or part

Remember that you can simply write to and read from a file to persist data.

\$ echo "hello 123" > foobar.txt # writes hello to foobar.txt

You can also use regex: learn more at regex101.com

\$ sed -i "s/[0-9]\{3\}/world/g" foobar.txt

\$ cat foobar.txt hello \$ > foobar.txt \$ cat foobar.txt

parameters: #!/bin/bash # contents of argscript.sh

• In bash , single quotes '' preserve the literal value of the characters they enclose. Double

quotes "" preserve the literal value of all characters except for \$, backticks `, and the

Recall that bash exposes its command line arguments through the \$<integer> positional

```
backslash \\ . The most important implication of this is that double quotes allow for variable
  interpolation, while single quotes do not. You can think of single quotes and the stronger "escape
  everything" syntax while double quotes are the more lax "escape most things" syntax.
 $ echo '$LANG'
 $LANG
 $ echo "$LANG"
 en_US.UTF-8
• In python, you can interact with command-line arguments through the syslargv list
 # !/usr/bin/python
 # contents of argscript.py
 import sys
 print(sys.argv[1])
 print(sys.argv[2])
```

with open('./newfile.txt', 'w') as f: f.write("hello from python\n") # end of file \$ python fileman.py \$ cat newfile.txt

• If you are getting permission denied issues, you will probably need to make your phonebook.sh

Once you're done remember to submit your answers to Gradescope. There are multiple valid answers

For the scripting assignment, you will need to upload a file containing your script. If you edit and test

your script in your VM using vim or another shell editor, you can copy the file to your local machine

python lets you manipulate files with the open function, commonly used with the with control

```
using scp or simply copy-paste the text into a local file.
Don't be stressed about getting something correct; just have fun exploring. We'll release the answers
after the lab is due!
```

NVIDIA

executable: chmod +x phonebook.sh

Keybindings Learning vim progressively Tmux cheat sheet

Thanks to Nvidia for sponsoring GPUs to the OCF.

linode Huge thanks to <u>Linode</u> for sponsoring the equipment used to record digital lectures for the Decal Copyright © 2017-2022 Open Computing Facility and eXperimental Computing Facility This website and its course materials are licensed under the terms of the CC BY-NC-SA 4.0 License. Source Code available on GitHub