Using Bash - WK2

This week, you will learn to configure Bash Scripts to enhance and control your Linux development environment and production systems. You will also learn about Shell variables, and how to effectively use Standard In and Standard Out.

Learning Objectives

- Configure the Bash shell environment.
- Utilize shell variables to solve common Linux workflow problems.
- Explain how to use Standard In and Standard Out streams to effectively deal with data.

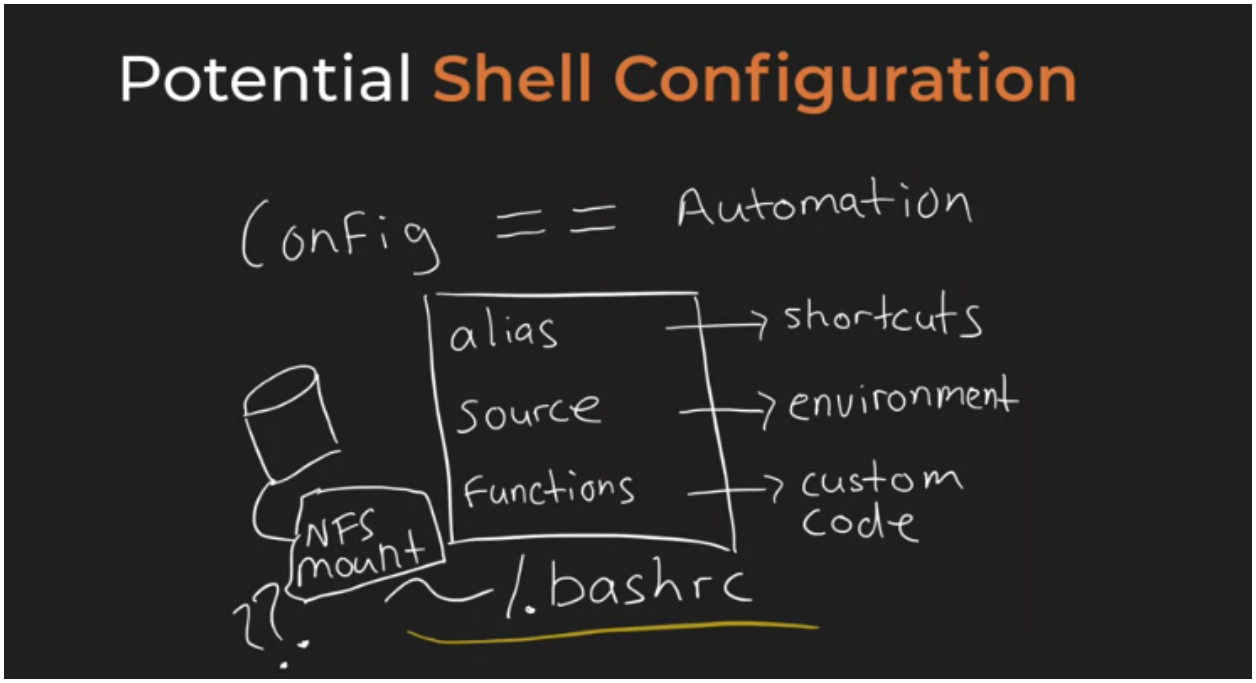
Introduction to Configuring your Bash Shell Environment

Objectives

- Let's cover the key learning objectives for this lesson.
- We're going to talk through why **shell config files** are important and what they are.
    - And some of the things you can do with them like configure your shell environment so that every time you open up a terminal, it does the correct thing.

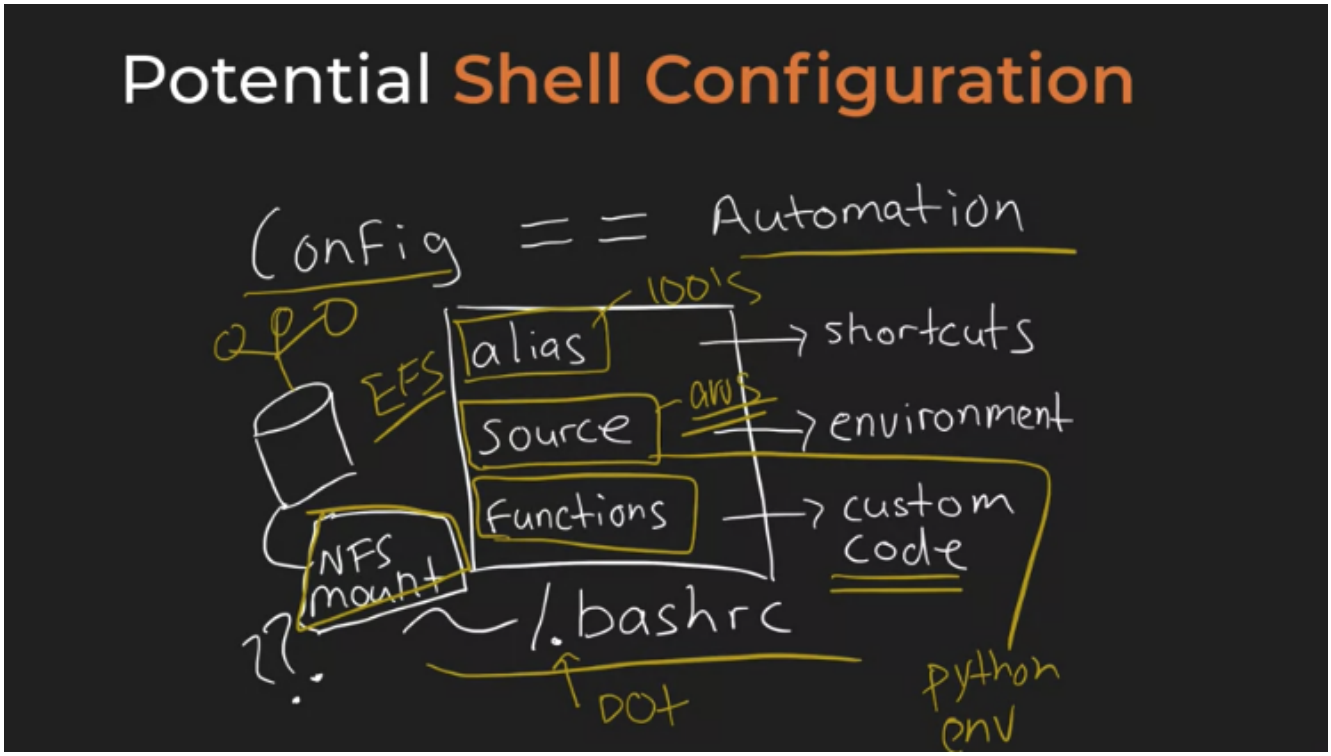
- We'll also talk about **bashrc**, which is really the most common file that you'll change when you're editing your environment.
    - It's available on pretty much every system that you would log into in the Cloud, and also in potentially a Docker environment were virtual machine.
- Finally, we'll talk about some of the really evolving technologies like.cshrc files used through third-party libraries like, oh-my-zsh, this is really an emerging trend for developers who want to fully customize your workflow and enhance it in a way that really suits their needs.

What are Shell Configuration Files? (Continuation from Above)



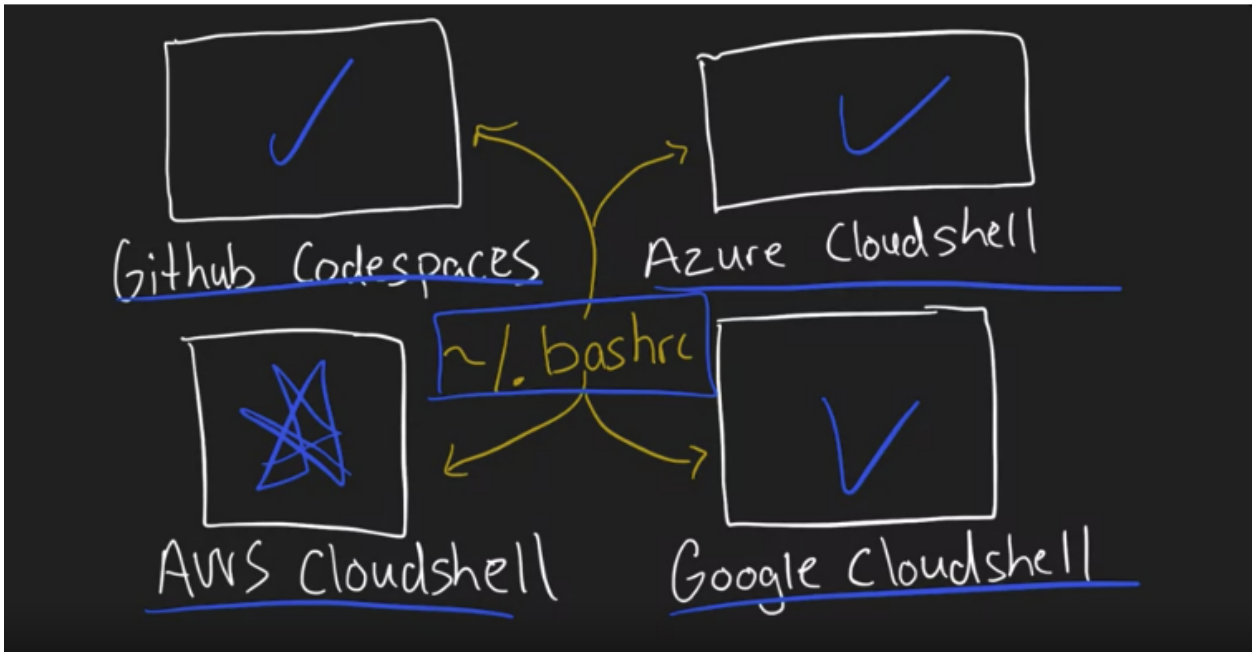
- A **config** file is automation. That's the simplest way to explain it.

Image Summarized
- Basis of Automation



Configuring .bashrc

- Versatility and pervasiveness on cloud terminal/technologies



Sample AWS CloudShell File Location - bashrc

```
aws  Services  Search for services, features, blogs, docs, and more [Option+S]  N. Virginia
AWS CloudShell
us-east-1 ×  us-east-1 ×
# .bashrc
[cloudshell-user@ip-10-0-147-197 ~]$ echo $SHELL
/bin/bash
[cloudshell-user@ip-10-0-147-197 ~]$ ls ~la ~/
ls: cannot access ~la: No such file or directory
/home/cloudshell-user/:
[cloudshell-user@ip-10-0-147-197 ~]$ ls -la ~/
total 36
drwxr-xr-x 3 cloudshell-user cloudshell-user 4096 Nov 20 22:55 .
drwxr-xr-x 4 root            root            4096 Nov 20 22:06 ..
-rw------- 1 cloudshell-user cloudshell-user   33 Nov 20 22:55 .bash_history
-rw-r--r-- 1 cloudshell-user cloudshell-user   18 Nov 20 22:06 .bash_logout
-rw-r--r-- 1 cloudshell-user cloudshell-user  193 Nov 20 22:06 .bash_profile
-rw-r--r-- 1 cloudshell-user cloudshell-user  316 Nov 20 22:55 .bashrc
drwxr-xr-x 3 cloudshell-user cloudshell-user 4096 Nov 20 22:06 .config
-rw------- 1 cloudshell-user cloudshell-user 1099 Nov 20 22:55 .viminfo
-rw-r--r-- 1 cloudshell-user cloudshell-user  777 Nov 20 22:06 .zshrc
[cloudshell-user@ip-10-0-147-197 ~]$ vim ~/.bash
```

- Creating alias commands for terminal traversing and other shortcuts / commands
- VIM file below and new alias creation

```
AWS CloudShell
us-east-1 ×  us-east-1 ×
# .bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
complete -C '/usr/local/bin/aws_completer' aws
export AWS_EXECUTION_ENV=CloudShell

## Alias
alias srcCode="cd ~/src"
~
~
~
~
```

- Video has good reminders on how to source and use common automation tools for source API Keys and Python Virtual Environments

Configuring .zshrc and Third-Party Tool oh-my-zsh (This is Yours!)

- Here is a quick look at your $SHELL variable

```
(base) ➜  .oh-my-zsh git:(master) echo $SHELL
/bin/zsh
(base) ➜  .oh-my-zsh git:(master) ls -la
total 48
drwxr-xr-x   15 craigrupp  staff   480 Jan  4  2017 .
drwxr-xr-x+  95 craigrupp  staff  3040 Sep 14 12:22 ..
drwxr-xr-x   15 craigrupp  staff   480 Sep 14 12:22 .git
-rw-r--r--    1 craigrupp  staff   105 Dec 21  2016 .gitignore
-rw-r--r--    1 craigrupp  staff  1178 Jan  4  2017 LICENSE.txt
-rw-r--r--    1 craigrupp  staff  8933 Dec 21  2016 README.md
drwxr-xr-x    3 craigrupp  staff    96 Dec 21  2016 cache
drwxr-xr-x    4 craigrupp  staff   128 Dec 21  2016 custom
drwxr-xr-x   20 craigrupp  staff   640 Jan  4  2017 lib
drwxr-xr-x    4 craigrupp  staff   128 May 17  2017 log
-rw-r--r--    1 craigrupp  staff  3406 Dec 21  2016 oh-my-zsh.sh
drwxr-xr-x  233 craigrupp  staff  7456 Dec 21  2016 plugins
drwxr-xr-x    3 craigrupp  staff    96 Dec 21  2016 templates
drwxr-xr-x  142 craigrupp  staff  4544 Feb 16  2017 themes
drwxr-xr-x    8 craigrupp  staff   256 Dec 21  2016 tools
```

- All types of plug-ins!

- Look at the vim zshrc (config file)

```
(base) ➜  / vim ~/.zshrc
```

Configuring the Bash Shell : Lab w/VIM

Part 1: Open ~/.bashrc and edit it

1. Edit the Bash configuration file**~/.bashrc**by using Vim, **vim ~/.bashrc**or opening the Visual Studio Code Editor
2. Add the following statements at the end of the file:

export API="API-Key-Goes-Here" echo "This is an example of using a variable at shell launch: " $API

1. Test this out by sourcing the new config: **source ~/.bashrc**. What did you see?

Part 2: Open a second shell and interact with the new environment

1. Open a new shell. What did you see?
2. Echo the variable $API **echo $API**
3. What do you see?

Part 3: Create an alias in ~/.bashrc and use it

1. Edit the Bash configuration file: **~/.bashrc**by using Vim, **vim ~/.bashrc**or opening the Visual Studio Code Editor
2. Add the following statements at the end of the file:

alias root="cd /"

1. Test this out by sourcing the new config: **source ~/.bashrc**. What did you see when you type in **alias**?
2. What do you see when you type in **root**? Why could this be helpful to you personally?

- While in command mode you can simply enter “o” (without the quotes) to open up a new line and add text
  - https://sites.radford.edu/~mhtay/CPSC120/VIM_Editor_Commands.htm
- After making addition for Part 1 we can then hit Esc to give a command to write file to disk and quit the vim editor
  - `:wq Write file to disk and quit the editor`

Part 1 : Shell Output

```
coder@c5736c757da2:~/project/Coursera-DE-C2-configure-shell$ vim ~/.bashrc
coder@c5736c757da2:~/project/Coursera-DE-C2-configure-shell$ vim ~/.bashrc
coder@c5736c757da2:~/project/Coursera-DE-C2-configure-shell$ vim ~/.bashrc
coder@c5736c757da2:~/project/Coursera-DE-C2-configure-shell$ vim ~/.bashrc
coder@c5736c757da2:~/project/Coursera-DE-C2-configure-shell$ vim ~/.bashrc
coder@c5736c757da2:~/project/Coursera-DE-C2-configure-shell$ source
~/.bashrc
This is an emaple of using a variable at shell launch:  transferArsenalApi
```

- After having made the addition to the .bashrc file we ran the source item for Part 1 to validate the echo statement with the variable output … which it did!

Part 2 : Open New Shell

```
This is an emaple of using a variable at shell launch:  transferArsenalApi
coder@c5736c757da2:~/project/Coursera-DE-C2-configure-shell$
```

```
coder@c5736c757da2:~/project/Coursera-DE-C2-configure-shell$ echo $API
transferArsenalApi
```

Part 3 : Create Alias and Source New Config

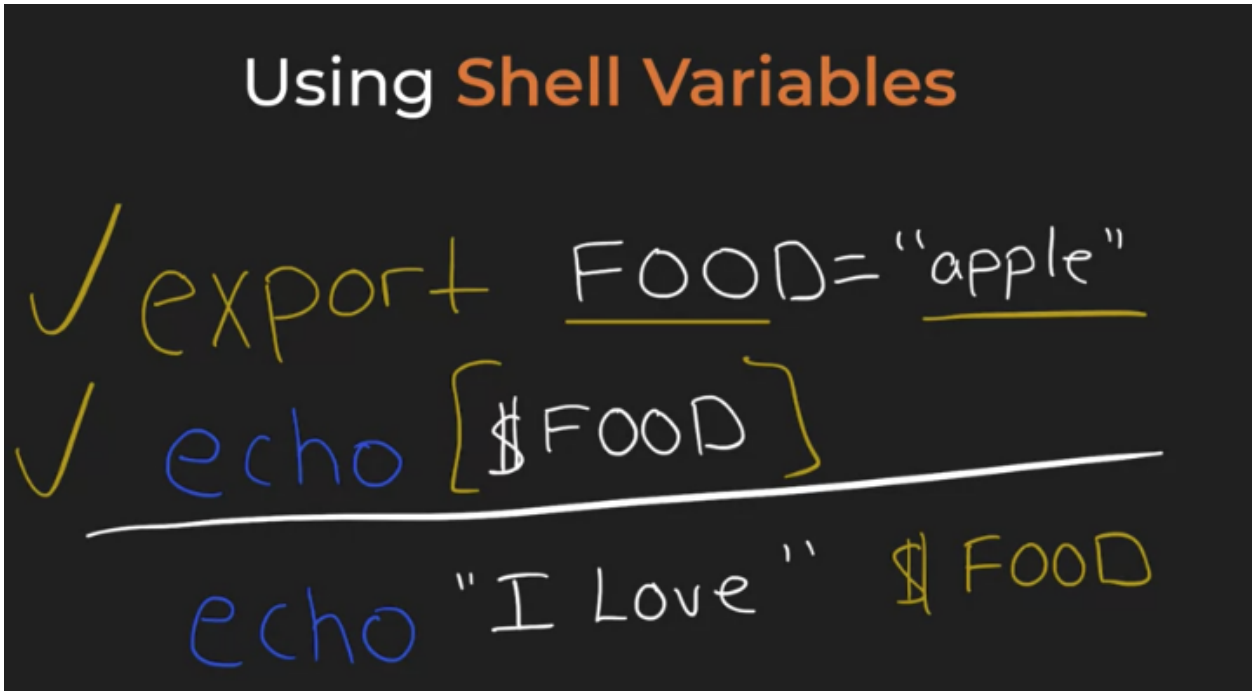
```
coder@c5736c757da2:~/project/Coursera-DE-C2-configure-shell$ vim ~/.bashrc
coder@c5736c757da2:~/project/Coursera-DE-C2-configure-shell$ source
~/.bashr
bash: /home/coder/.bashr: No such file or directory
coder@c5736c757da2:~/project/Coursera-DE-C2-configure-shell$ source
~/.bashrc
This is an emaple of using a variable at shell launch:  transferArsenalApi
coder@c5736c757da2:~/project/Coursera-DE-C2-configure-shell$ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal ||
echo error)" "$(history|tail -n1|sed -e
'\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
alias root='cd /'
coder@c5736c757da2:~/project/Coursera-DE-C2-configure-shell$ root
```

Introduction to Working with Shell Variables

What are Shell Variables?

- Really it's a way of storing data that you use in another script or another process.
- They come up a lot when you're building things with APIs, and the key reason here is that you want to store those special keys somewhere where they won't accidentally be checked into your repository.
- So a lot of times what you'll do is you'll use a GIT Ignore file which tells your source control system to ignore certain files.
    - And inside potentially you'll say I want you to ignore a file called .env.
    - Inside that file you'll have API keys for the cloud provider, third party services you're using, maybe authentication credentials for your database.
    - And all that gets stored inside these files that are local just to your machine.
- So then when you work with a developer that's on your team, you just tell them you need to create your own env file and put the things that you need inside that env file.
- And then you source that env file when you work on that particular project.
    - And you can even go further and actually put that in as part of your let's say .bash or C file so that every time you change into a certain directory it will source the environment variables just for that particular project.

Export Command BASH



Coursera-DE-C2-shell-variables - Lab (Source Shell Variables)

Goal: Learn to use shell variables and source them

Let's practice using and sourcing shell variables

Part 1: Sourcing a script full of shell variables

1. Source `projectAlias.sh` by running the command `source projectAlias.sh`
2. View the current aliases by running the command `alias`
3. What do you see?

Part 2: Use the aliases

1. Use the `fooTOP` alias by typing it in the terminal?
2. What do you see?
3. Now use the `barTOP` alias by typing it in the terminal?
4. Did it work? If not explain why and figure out a way you could fix it?

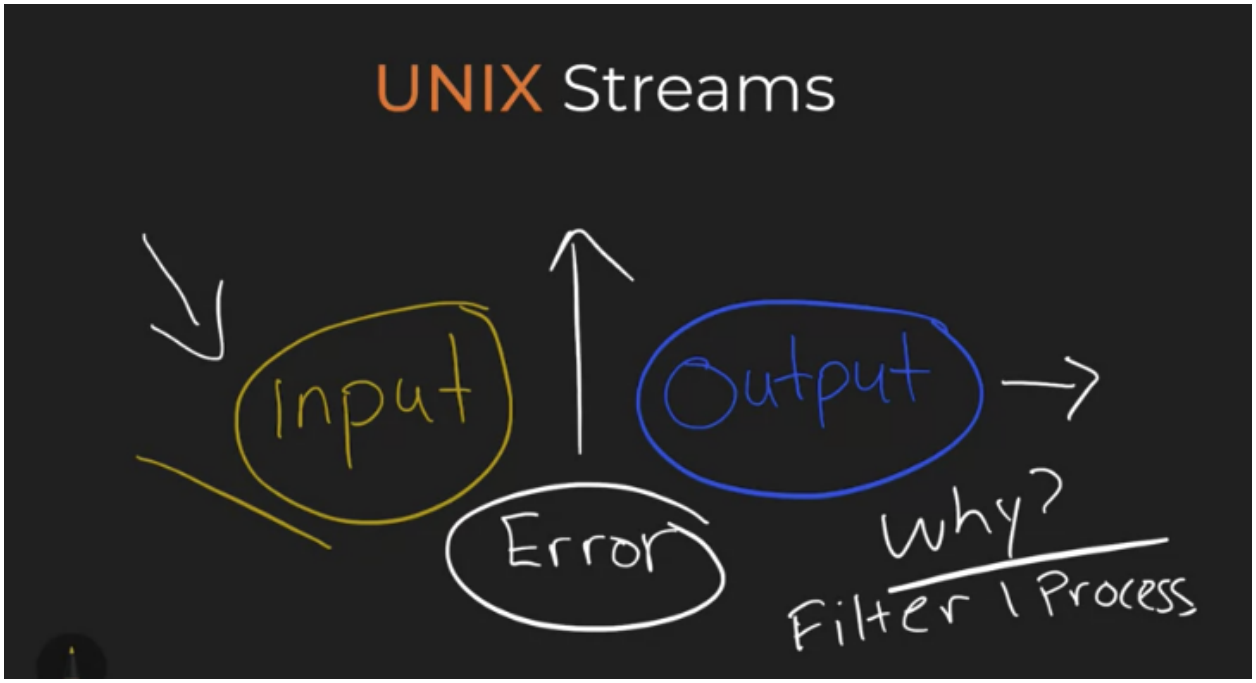
Part 3: echo sourced variables

1. Let's use the sourced variables in a statement: `echo "I enjoy eating an $fruit after a tasty $meal for dinner"`
2. What do you see?
3. Use the shell variables you sourced earlier to print a statement in Bash you created yourself.

```
oder@477546b7611f:~/project/Coursera-DE-C2-shell-variables$ pwd
/home/coder/project/Coursera-DE-C2-shell-variables
coder@477546b7611f:~/project/Coursera-DE-C2-shell-variables$ ls
bar  foo  LICENSE  projectAlias.sh  README.md
coder@477546b7611f:~/project/Coursera-DE-C2-shell-variables$ source
projectAlias.sh
coder@477546b7611f:~/project/Coursera-DE-C2-shell-variables$ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal ||
echo error)" "$(history|tail -n1|sed -e
'\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'
alias barTOP='cd bar'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias fooTOP='cd foo'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
coder@477546b7611f:~/project/Coursera-DE-C2-shell-variables$ fooTOP
coder@477546b7611f:~/project/Coursera-DE-C2-shell-variables/foo$ barTOP
bash: cd: bar: No such file or directory
coder@477546b7611f:~/project/Coursera-DE-C2-shell-variables/foo$ cd ..
coder@477546b7611f:~/project/Coursera-DE-C2-shell-variables$ barTOP
coder@477546b7611f:~/project/Coursera-DE-C2-shell-variables/bar$ echo "I
enjoy eating an $fruit after a tast $meal for dinner"
I enjoy eating an Apple after a tast Salad for dinner
coder@477546b7611f:~/project/Coursera-DE-C2-shell-variables/bar$ echo "I
was raised in the state of Washington, know for their delicious $fruit"
I was raised in the state of Washington, know for their delicious Apple
coder@477546b7611f:~/project/Coursera-DE-C2-shell-variables/bar$
```

Introduction to Standard Streams

- Unix streams are a way of capturing data into an input and filtering it, for example, this is really common in [inaudible] science you'll filter something via SQL query or you use pandas and you go through and take some subset of the data so that you can process it later down the stream. Similarly, when you have output, you'll take that data and maybe you'll pipe it to some other location.



Bash Streams Lab

Let's practice working with Bash Streams

<u>Part 1: Create a new file and write to it via stdout</u>
Run the following command in your Bash terminal: echo "fruit" > meal.txt

Read the output of the file using the cat command: cat meal.txt

Explain what happened?

Part 2: Append new data to the file by using stdout

Run the following command in your Bash terminal: echo "chocolate" >> meal.txt

Read the output of the file using the cat command: cat meal.txt

Explain what happened?

How could you write more data to the file?

Part 3: Redirect standard input

Use the tr command to substitute characters by reading standard input: tr fruit steak < meal.txt

What output do you see?

Could you improve this result? How?

Part 4: Throwaway stdout to /dev/null

Send the output of standard out to /dev/null via cat meal.txt > /dev/null

Explain what happened?

Why would you want to send the output of a command to/dev/null?

Coursera-DE-C2-Standard-Streams

Standard Streams Lab for Coursera

Goal: Learn to work with streams

Let's practice working with Bash Streams

Part 1: Create a new file and write to it via stdout

1. Run the following command in your Bash terminal: `echo "fruit" > meal.txt`
2. Read the output of the file using the cat command: `cat meal.txt`
3. Explain what happened?

Part 2: Append new data to the file by using stdout

1. Run the following command in your Bash terminal: `echo "chocolate" >> meal.txt`
2. Read the output of the file using the cat command: `cat meal.txt`
3. Explain what happened?
4. How could you write more data to the file?

Part 3: Redirect standard input

1. Use the `tr` command to substitute characters by reading standard input: `tr fruit steak < meal.txt`
2. What output do you see?
3. Could you improve this result? How?

Part 4: Throwaway stdout to /dev/null

1. Send the output of standard out to /dev/null via `cat meal.txt > /dev/null`
2. Explain what happened?
3. Why would you want to send the output of a command to `/dev/null`?

```
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ pwd
/home/coder/project/Coursera-DE-C2-Standard-Streams
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ ls
LICENSE  README.md
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ echo "fruit"
> meal.txt
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ cat meal.txt
fruit
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ echo
"chocolate" >> meal.txt
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ less <
meal.txt
bash: less: command not found
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ less meal.txt
bash: less: command not found
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ cat meal.txt
```

```
fruit
chocolate
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ tr fruit
steak < meal.txt
steak
chocolake
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ cat meal.txt
fruit
chocolate
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ tr fruit
steak << meal.txt
> ^C
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ cat meal.txt
fruit
chocolate
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ tr fruit
steak < meal.txt > meal.txt
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ cat meal.txt
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ echo
"fruit\nchocoloate" > meal.txt
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ cat meal.txt
fruit\nchocoloate
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ printf
'fruit\nchocloate' > meal.txt
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ cat meal.txt
fruit
chocloatecoder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ cat
meal.txt
fruit
chocloatecoder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$
printf 'fruit\nchocolate\n' > meal.txt
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ cat meal.txt
fruit
chocolate
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ cat meal.txt
> /dev/null
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ pwd
/home/coder/project/Coursera-DE-C2-Standard-Streams
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ ls
LICENSE  meal.txt  README.md
coder@72501476eee5:~/project/Coursera-DE-C2-Standard-Streams$ cd /dev/
coder@72501476eee5:/dev$ ls
core  fd  full  mqueue  null  ptmx  pts  random  shm  stderr  stdin  stdout
```

```
tty  urandom  zero
coder@72501476eee5:/dev$ cd null
bash: cd: null: Not a directory
coder@72501476eee5:/dev$
```

Using Bash Section Quiz

Congratulations! You passed!

Grade received 90%	Latest Submission Grade 90%	To pass 80% or higher

Go to next item

1. Explain the purpose of the `~/.bashrc` file.

1 / 1 point

- ○ It is a Bash login shell that only runs at the start of a new login shell.
- ○ It is a Bash logout shell that only runs when a shell exits.
- ◉ It is a Bash shell script that runs whenever Bash is started interactively.

Correct
You got it! This script is used to configure your shell environment for interactive use.

2. What is the purpose of the `~/.zshrc` file?

1 / 1 point

- ○ It is a ZSH login shell that only runs at the start of a new login shell.
- ◉ It is a ZSH shell script that runs whenever Bash is started interactively.
- ○ It is a ZSH logout shell that only runs when a shell exits.

Correct
You got it! This script is used to configure your shell environment for interactive use.

3. What shell example creates a shell variable to the current shell only.

1 / 1 point

- ( ) `export FRUIT="cherry"`
- (•) `FRUIT="cherry"`
- ( ) `echo $FRUIT`

Correct
You got it! Setting a variable makes the

4. What would be the output of this command?

1 / 1 point

`FRUIT="Cherries"; echo $FRUIT are tasty`

- ( ) `echo Cherries are tasty`
- ( ) `$FRUIT are tasty`
- (•) `Cherries are tasty`

Correct
You got it! The shell variable value prints out as part of the echo command.

5. When you run the command `alias` from a terminal what will you see?

1 / 1 point

- (•) You will see output like the following:

```
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
```

- ( ) It will display the help menu for a command.
- ( ) It will print the name of the shell currently in use.

Correct

6. What is a good example of what appears in standard out?

1 / 1 point

- ○ Errors from the improper execution of a shell command.
- ◉ The content of a file.
- ○ It is always blank.

Correct
You got it! The content of a file is often sent to stdout, such as in the example of `less file.txt`

7. What is the output of this command?

1 / 1 point

```
ls fakefile.txt &>/dev/null
```

- ○ It will create the file since it doesn't exist.
- ○ It will display the following error.

  ```
  ls: cannot access fakefile: No such file or directory
  ```

- ◉ There is no output.

Correct
You got it!

8. What would the following command do?

0 / 1 point

```
ls fakefile 2>error.txt
```

- ○ Write stderr to `error.txt`
- ○ It deletes the output of stderr
- ◉ Write stdout to `error.txt`

Incorrect
Not quite. The `2>` command sends stderr to a location.

9. What would this command do?

1 / 1 point

```
for i in {1..10}; do echo $RANDOM >> rando.txt; done
```

- ○ It overwrites a file with a new random number 10 times.
- ○ It throws away the output of standard out.
- ◉ It appends 10 random numbers to a file.

Correct
You got it! The `>>` operator appends the output to a file. For every execution of a command using this operator, it will append new content to the file.

10. What is happening in the following shell command?

1 / 1 point

```
sort -r < /etc/passwd
```

- ○ The sort command is appending to the /etc/password file.
- ◉ The /etc/password file is reverse sorted.
- ○ The /etc/password file is overwritten with a sorted version

Correct
You got it! The `sort` command can accept input redirection and this command reverse sorts the content of the file.