# Shell Scripting

MacAdmins Conference 2013

http://tinyurl.com/psumacscripting13

# Jay Hoff

- ITS/CLC
  - Mac and Linux Group
    - Systems Administrator
  - jeh26@psu.edu
  - @jayhoff

## Rusty Myers

- ITS/CLC
  - Mac & Linux Group
    - System Admin
- rzm102@psu.edu
- @thespider

## What's BASH?

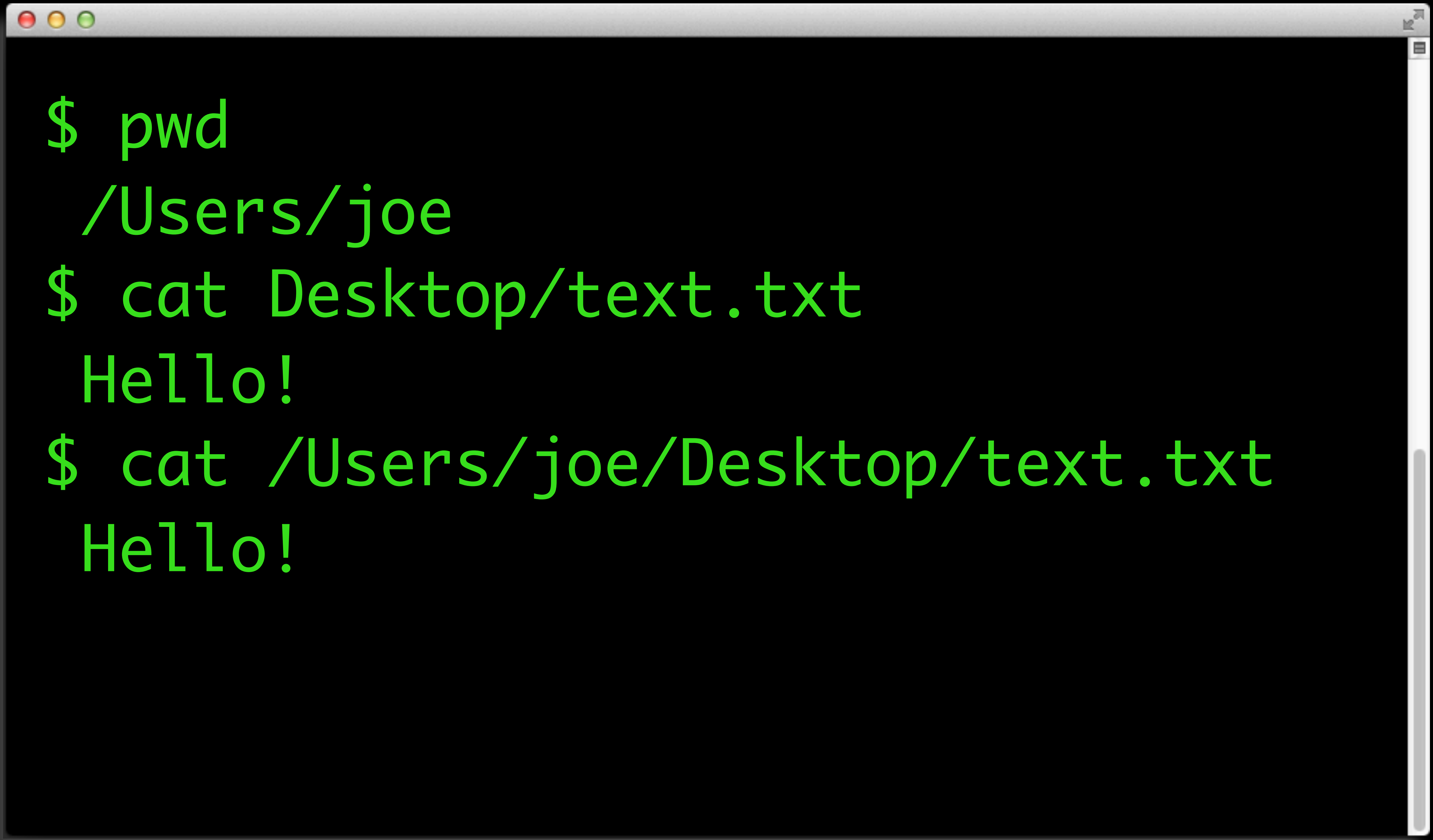
- Bourne Again Shell (bash)
- Command Interpreter
- Binary at /bin/bash
- Responsible for spawning sub-shells

## What’s BASH?

- Bourne Again Shell (bash)
- Brian Fox
  - Programmed BASH
  - beta 1989
- Replaced Bourne Shell (sh)

# Paths

- Relative
  - From current location to file
- Absolute
  - From hard drive root to file



```
$ pwd
 /Users/joe
$ cat Desktop/text.txt
 Hello!
$ cat /Users/joe/Desktop/text.txt
 Hello!
```

## Basic Commands

- man
- apropos
- which
- cat
- echo
- grep
- sleep
- clear
- read
- ls
- chmod
- tr

## OS X Commands

- sw_vers
- system_profiler
- systemsetup
- networksetup
- diskutil
- open

# MOAR Commands

- dscl
- installer
- defaults
- PlistBuddy
- osascrupt
- softwareupdate
- pkgutil
- pkgbuild
- ioreg
- bless
- lsbom
- mdfind
- plutil
- /System/Library/PrivateFrameworks/Apple80211.framework/Versions/A/Resources/airport
- launchctl
- pmset
- /System/Library/CoreServices/RemoteManagement/ARDAgent.app/Contents/Resources/kickstart
- tmutil
- type
- pwd

## Shell Variables

- What are they?
- echo $VARIABLE-NAME to show value
- run “env” to show current variables
  - Present Working Directory: $PWD
  - Current User: $USER
  - Current Shell: $SHELL
  - Search Path for commands: $PATH

## Special Chars

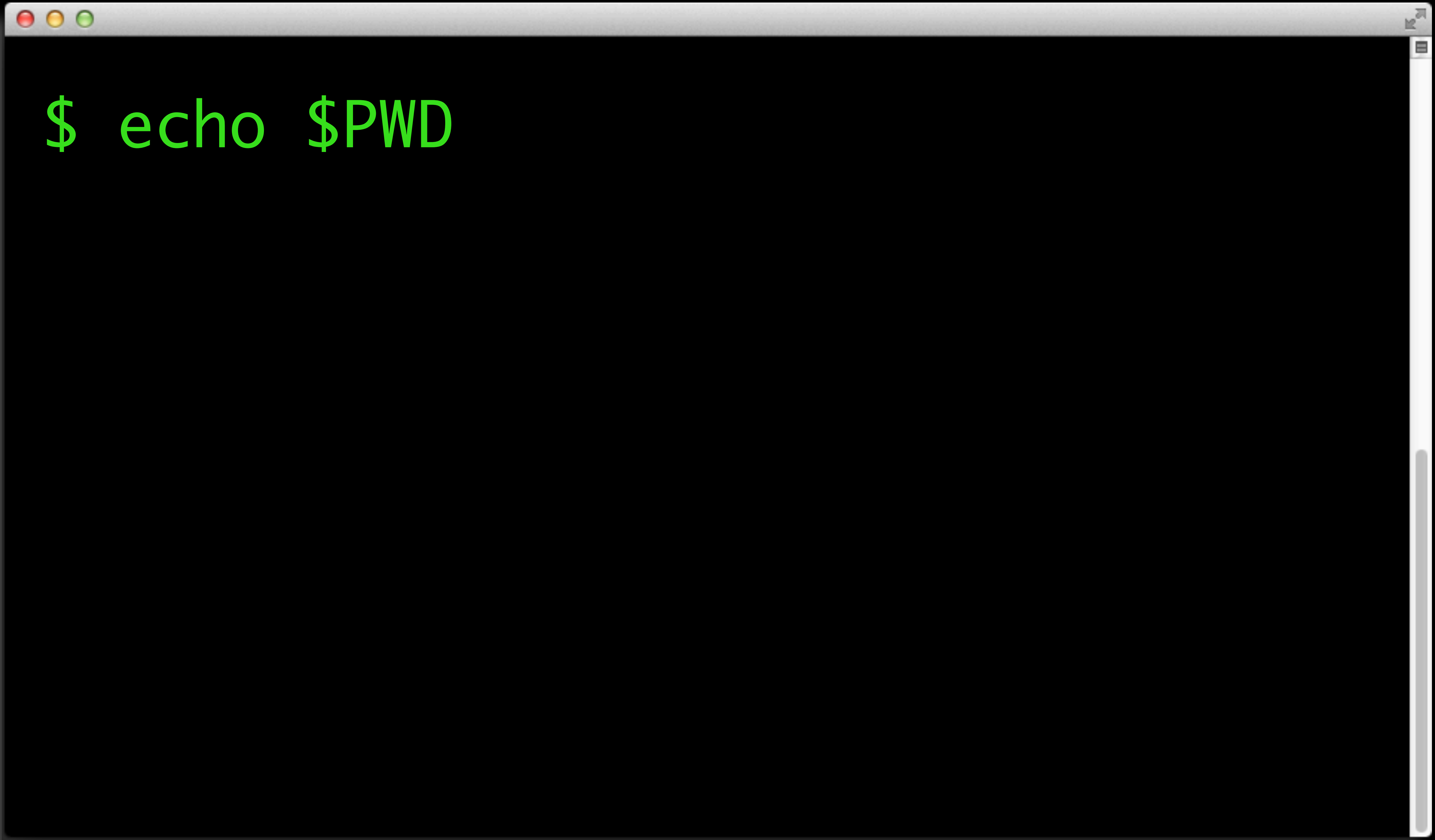
- What are they?
- Why Not?
- Gotchyas
- !&#|’”`~<>*$?\^()[]{}
  - Space

## Special Meanings

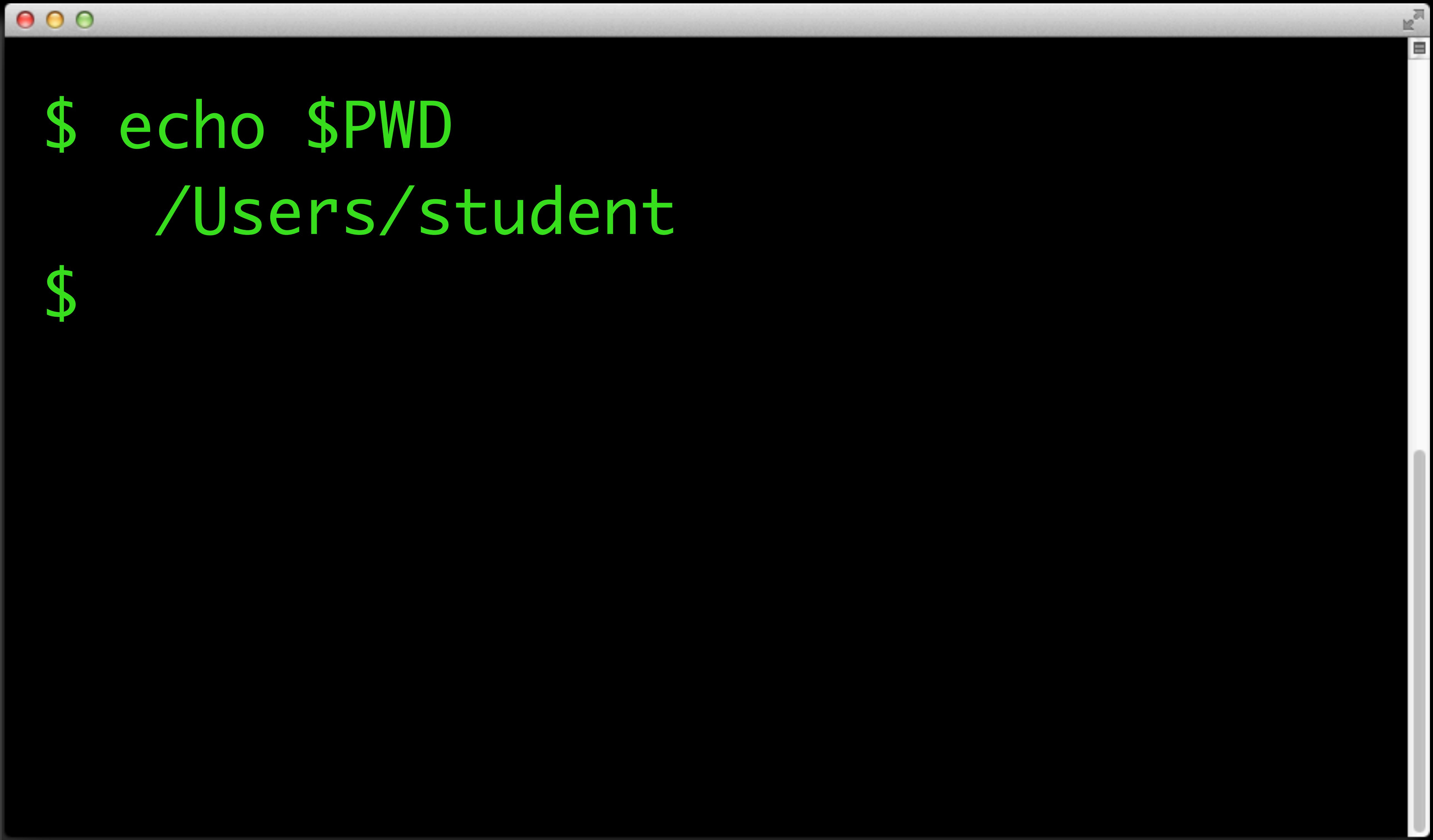
| | |
|---|---|
| Comment | # |
| Variable | $ |
| Wild Card | * |
| Current Directory | . |

# Quoting

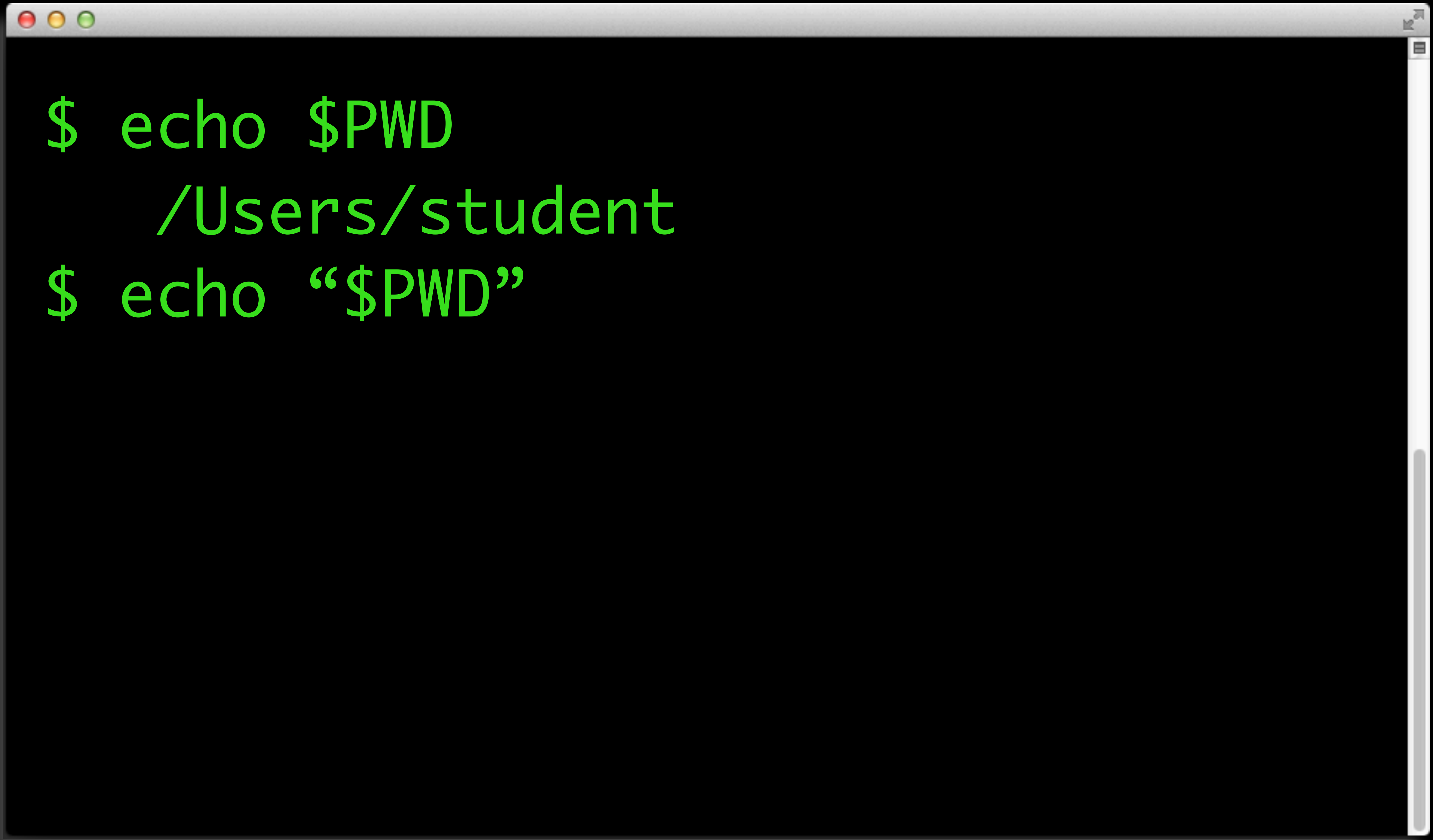
| | |
|---|---|
| Escape Next Char | \ |
| Double Quotes except $, `, \ | “” |
| Single Quotes | ‘ ’ |



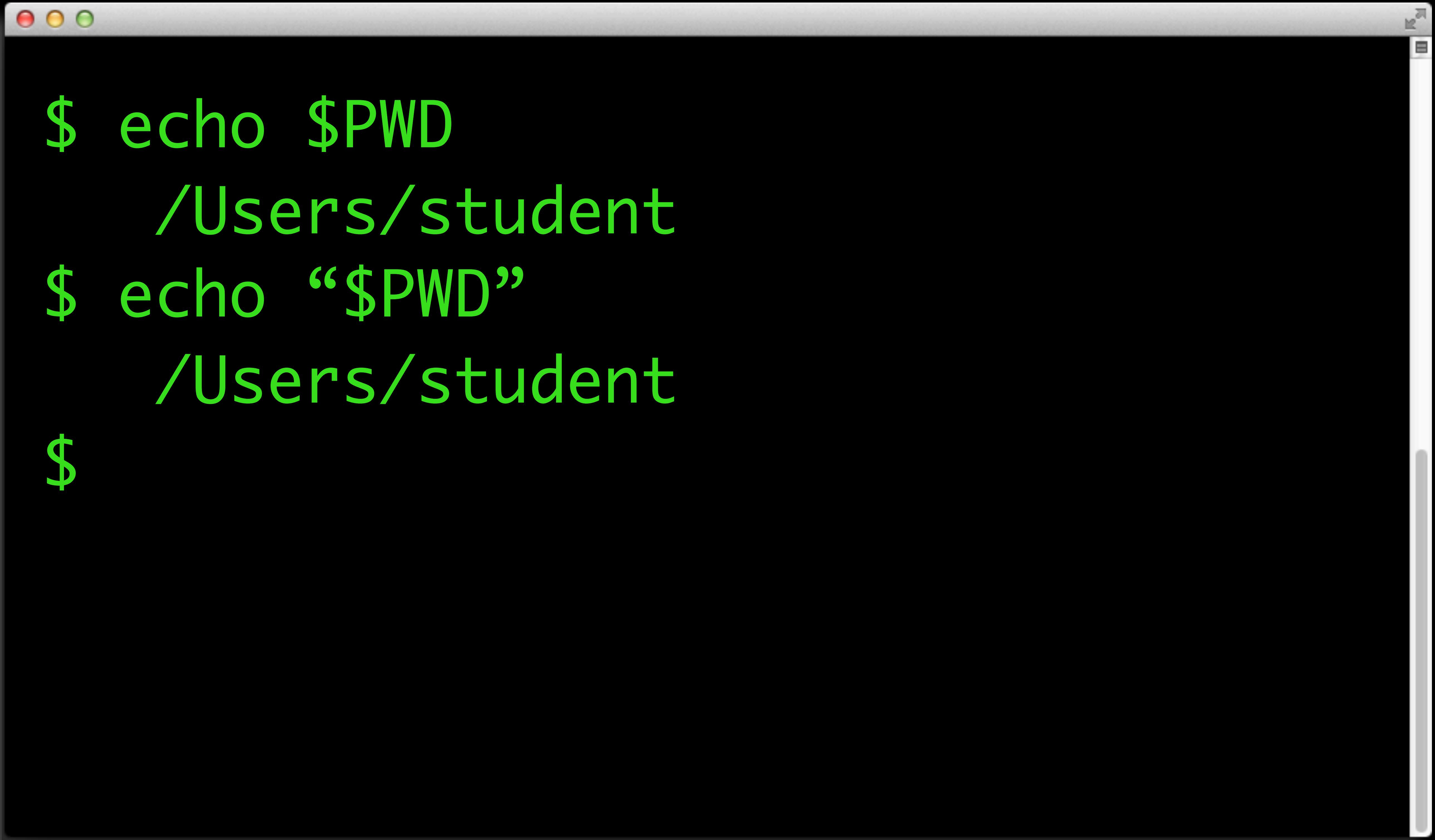
```
$ echo $PWD
```



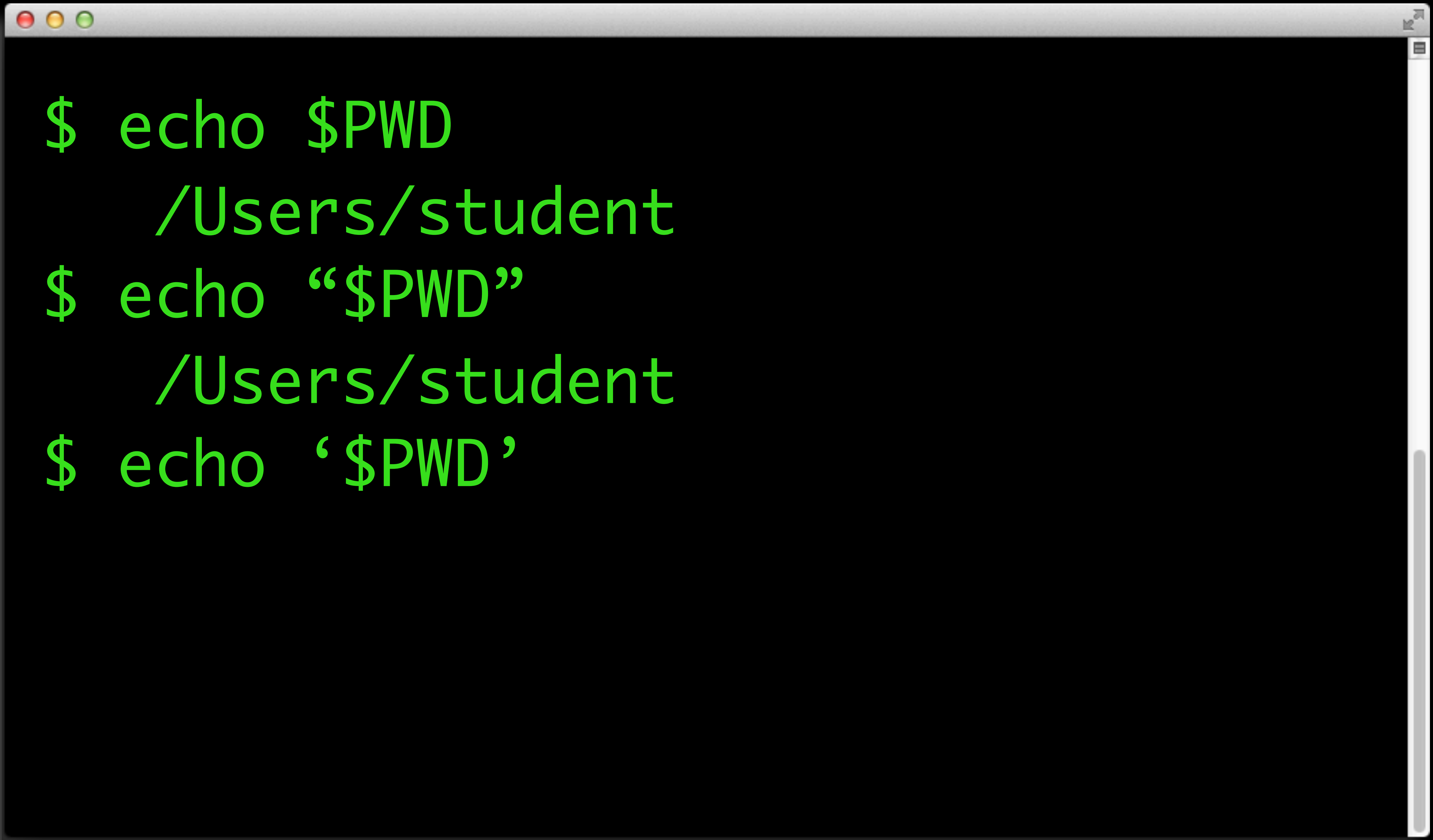
```
$ echo $PWD
   /Users/student
$
```



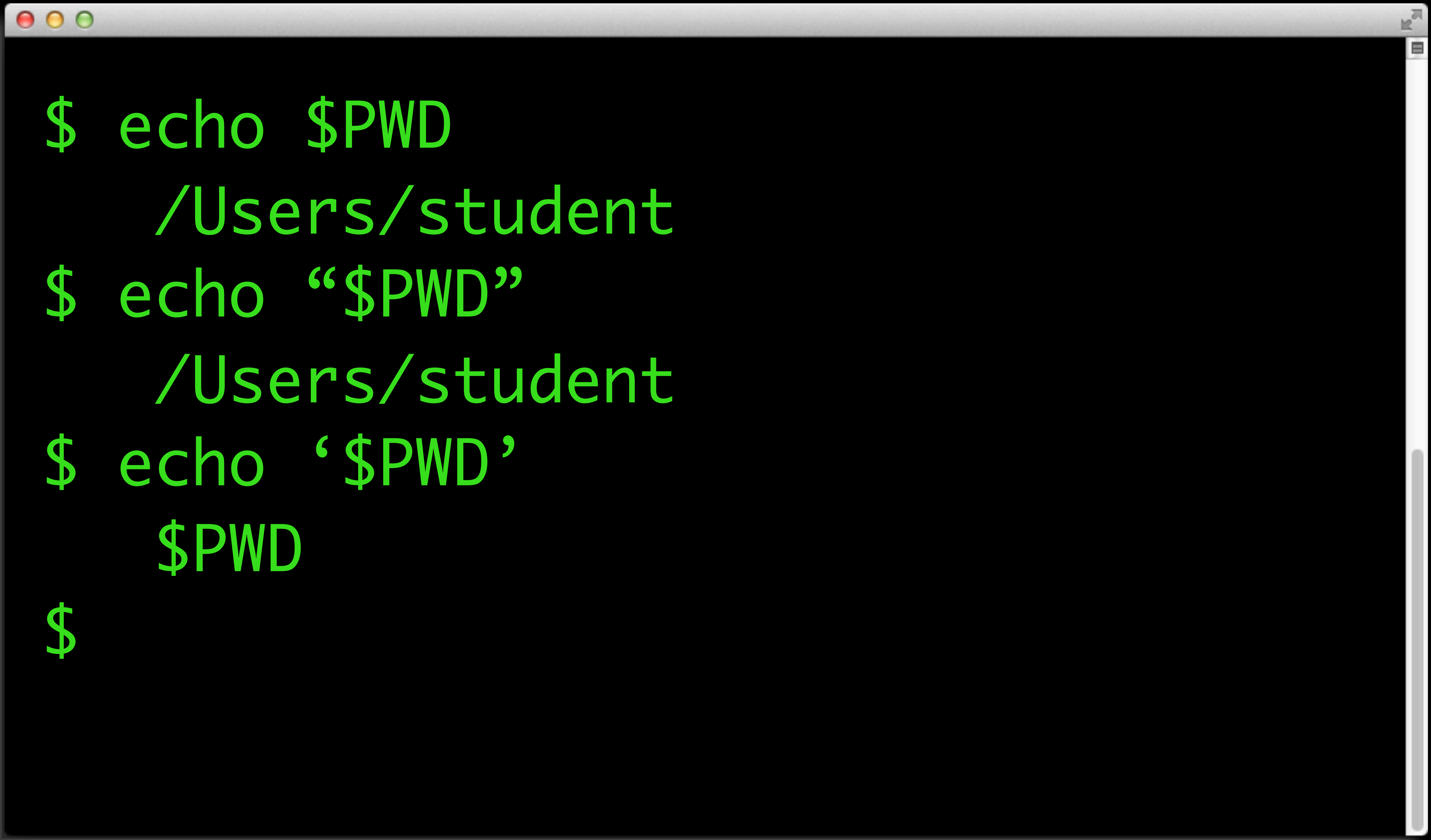
```
$ echo $PWD
   /Users/student
$ echo “$PWD”
```



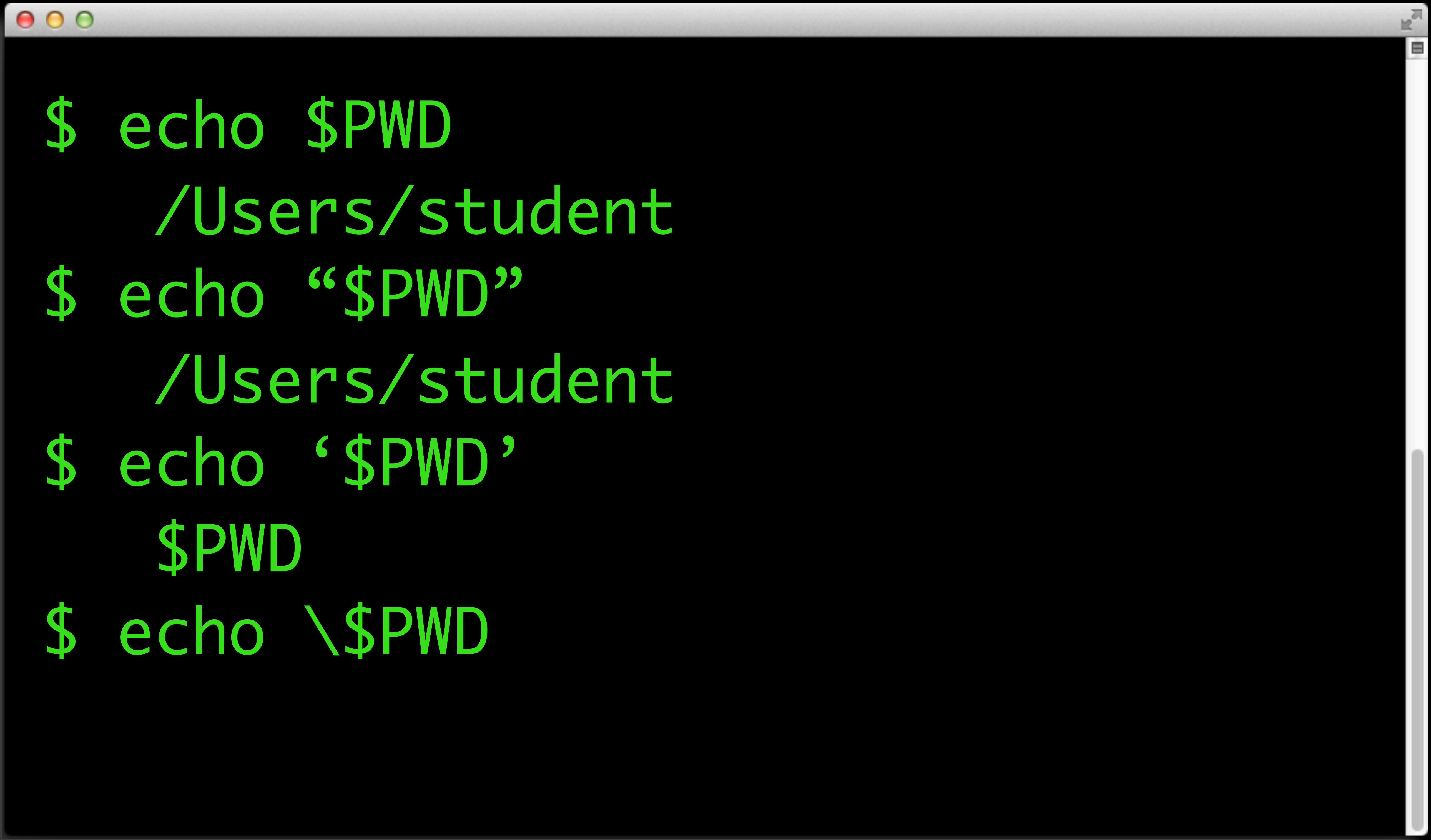
```
$ echo $PWD
   /Users/student
$ echo “$PWD”
   /Users/student
$
```



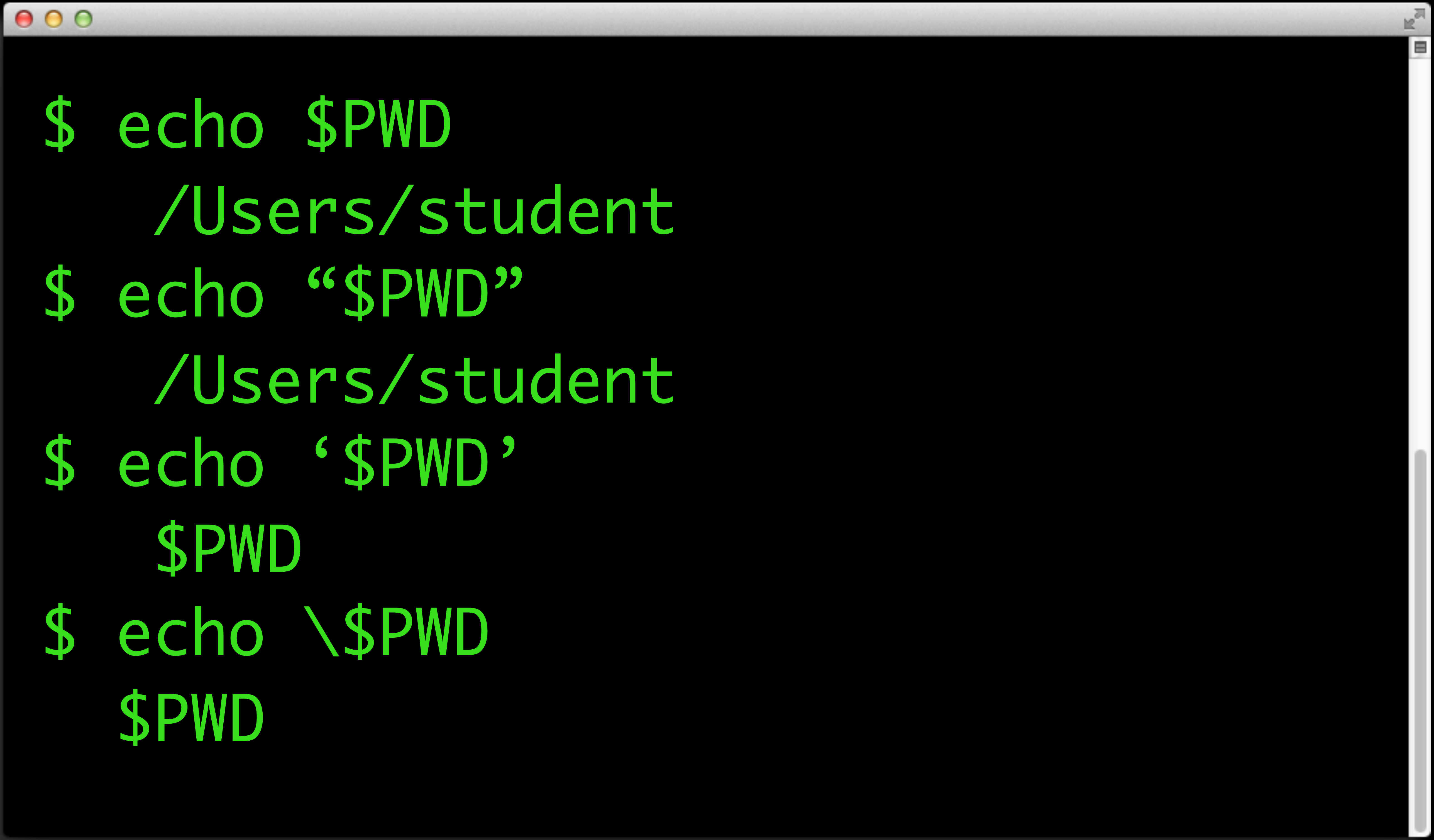
```
$ echo $PWD
   /Users/student
$ echo “$PWD”
   /Users/student
$ echo ‘$PWD’
```



```
$ echo $PWD
   /Users/student
$ echo “$PWD”
   /Users/student
$ echo ‘$PWD’
   $PWD
$
```



```
$ echo $PWD
   /Users/student
$ echo “$PWD”
   /Users/student
$ echo ‘$PWD’
   $PWD
$ echo \$PWD
```



```
$ echo $PWD
   /Users/student
$ echo “$PWD”
   /Users/student
$ echo ‘$PWD’
   $PWD
$ echo \$PWD
  $PWD
```

## What’s a Shell Script?

- Interpreted Language
- Not Compiled
- Languages
  - Bash, PHP, Python, Perl, Ruby

## Multiple Commands

- Commands In a Text Document
- Designed To Repeat a Process
- Multiple Commands Combined

## Why Create It?

- Automate Repetitive Tasks
- Eliminate Errors/Standardize
- Delegate To Others
- Self Documenting
- Saves Time

## Script Editors

- GUI
  - TextMate
  - BBEdit
  - TextWrangler
- CLI
  - vi
  - emacs
  - pico/nano

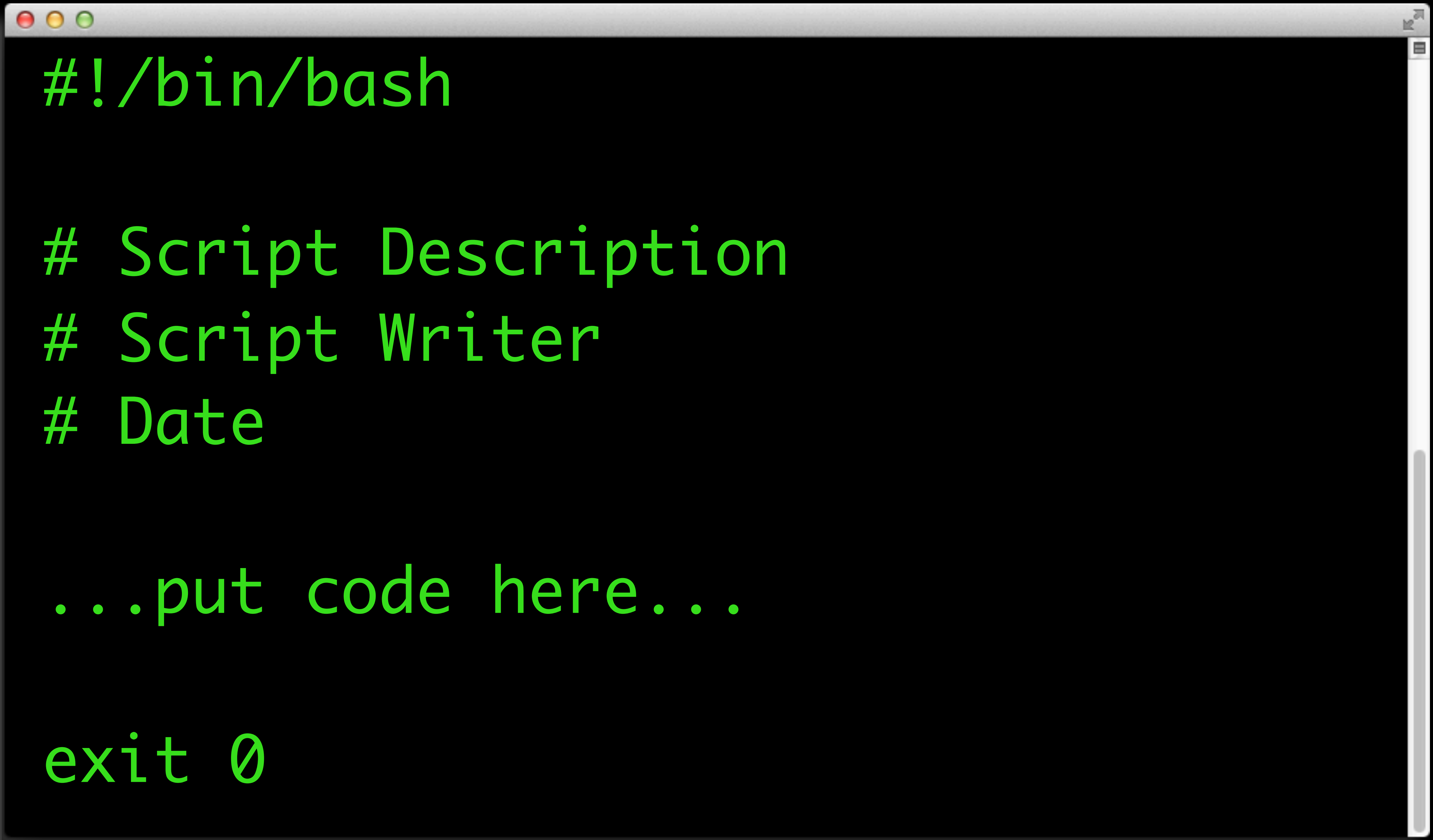
## Script Format

## Script Name

- BASH doesn’t care about extensions
- Ending with .sh
- Starting with . hides file
- Avoid spaces/special characters

## First Line

- Tells bash what interpreter to use
- sometimes called sha-bang
- #!path-to-interpreter
  - #!/bin/bash
  - #!/usr/bin/perl



```
#!/bin/bash

# Script Description
# Script Writer
# Date

...put code here...

exit 0
```

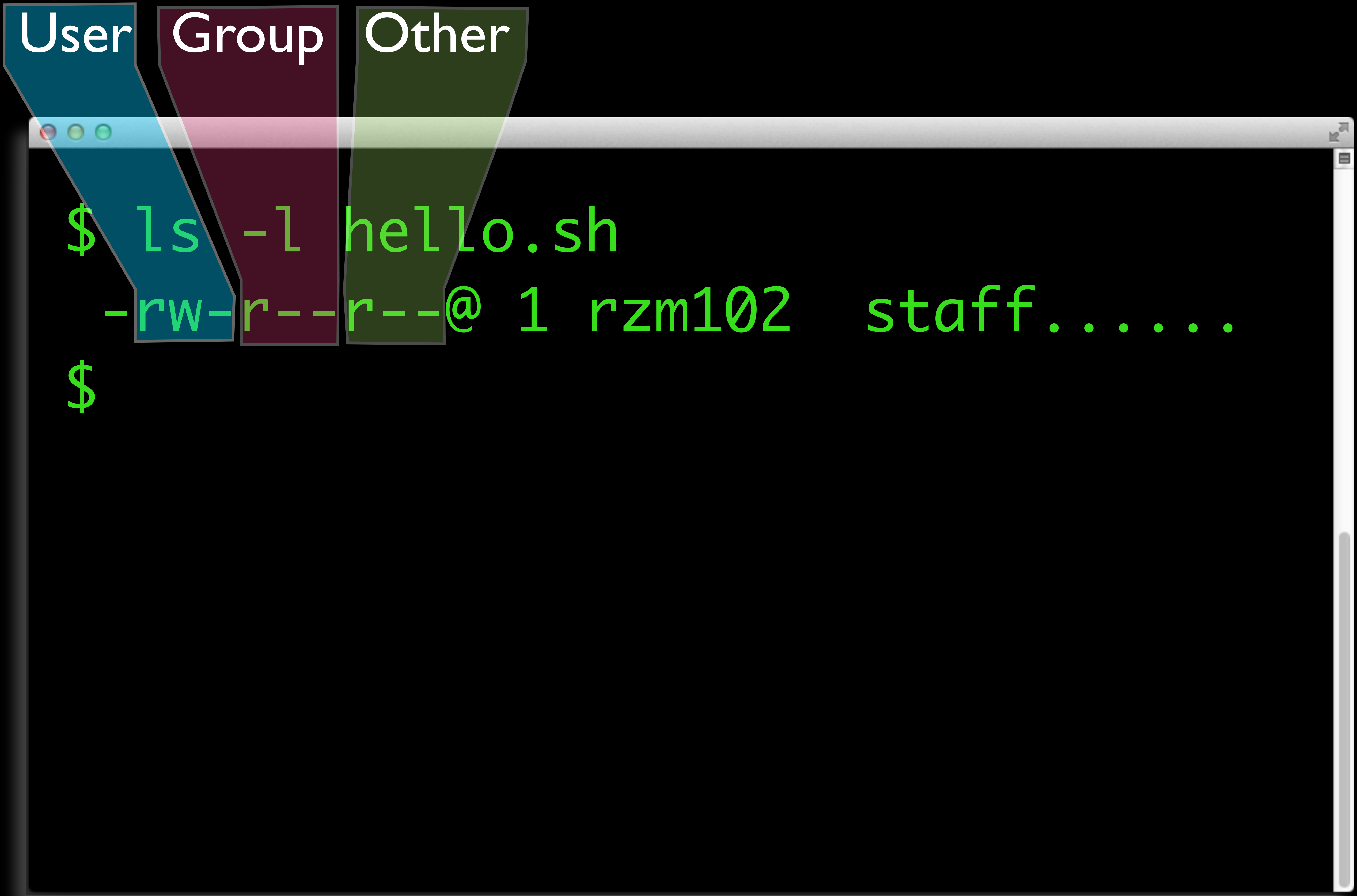
## Hello.sh

```
#!/bin/bash

# Script will say Hello
# Written by Jay & Rusty
# 05/01/2013

# echo hello MacAdmins to console
echo “hello MacAdmins”

exit 0
```

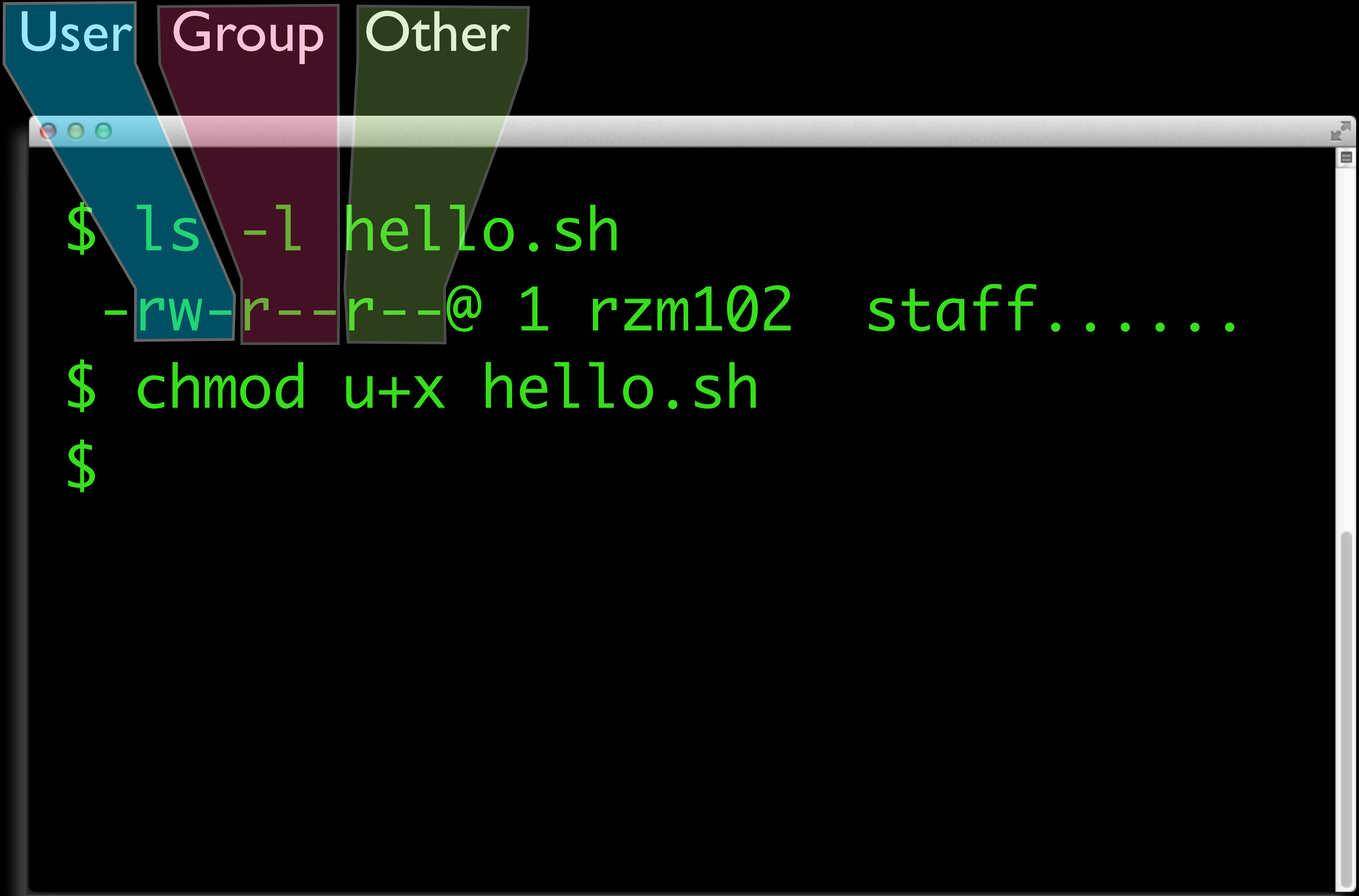
## Execute Bit!

Permissions in a nutshell

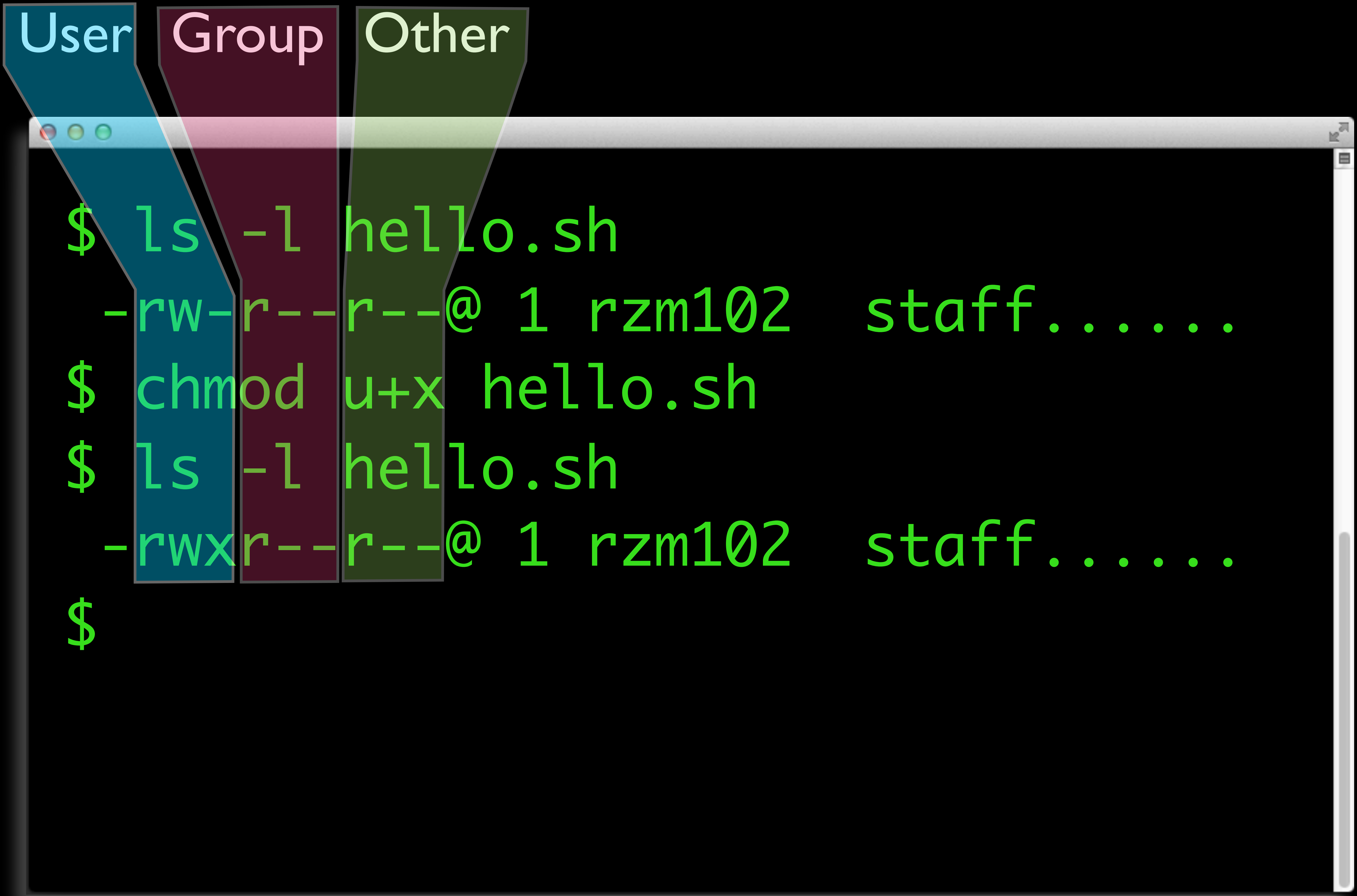
- 3 Fields: (u)ser, (g)roup, (o)ther
- 3 Bits/Field: (r)ead, (w)rite, e(x)ecute
- Execute by default not set
- List the permissions: ls -l
- Change permissions:
  chmod field+-bit(s) scriptname



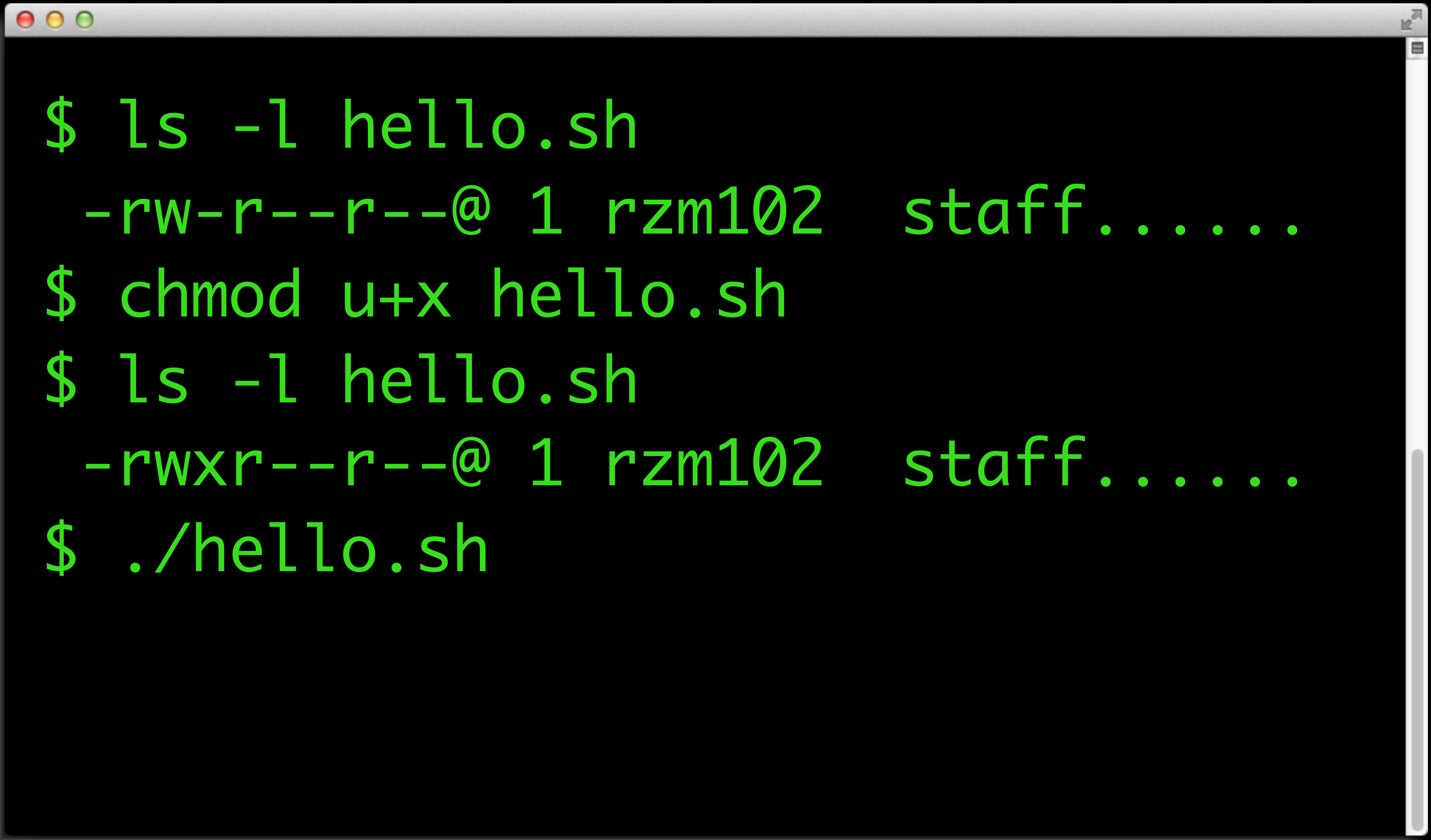
```
$ ls -l hello.sh
 -rw-r--r--@ 1 rzm102  staff......
$
```



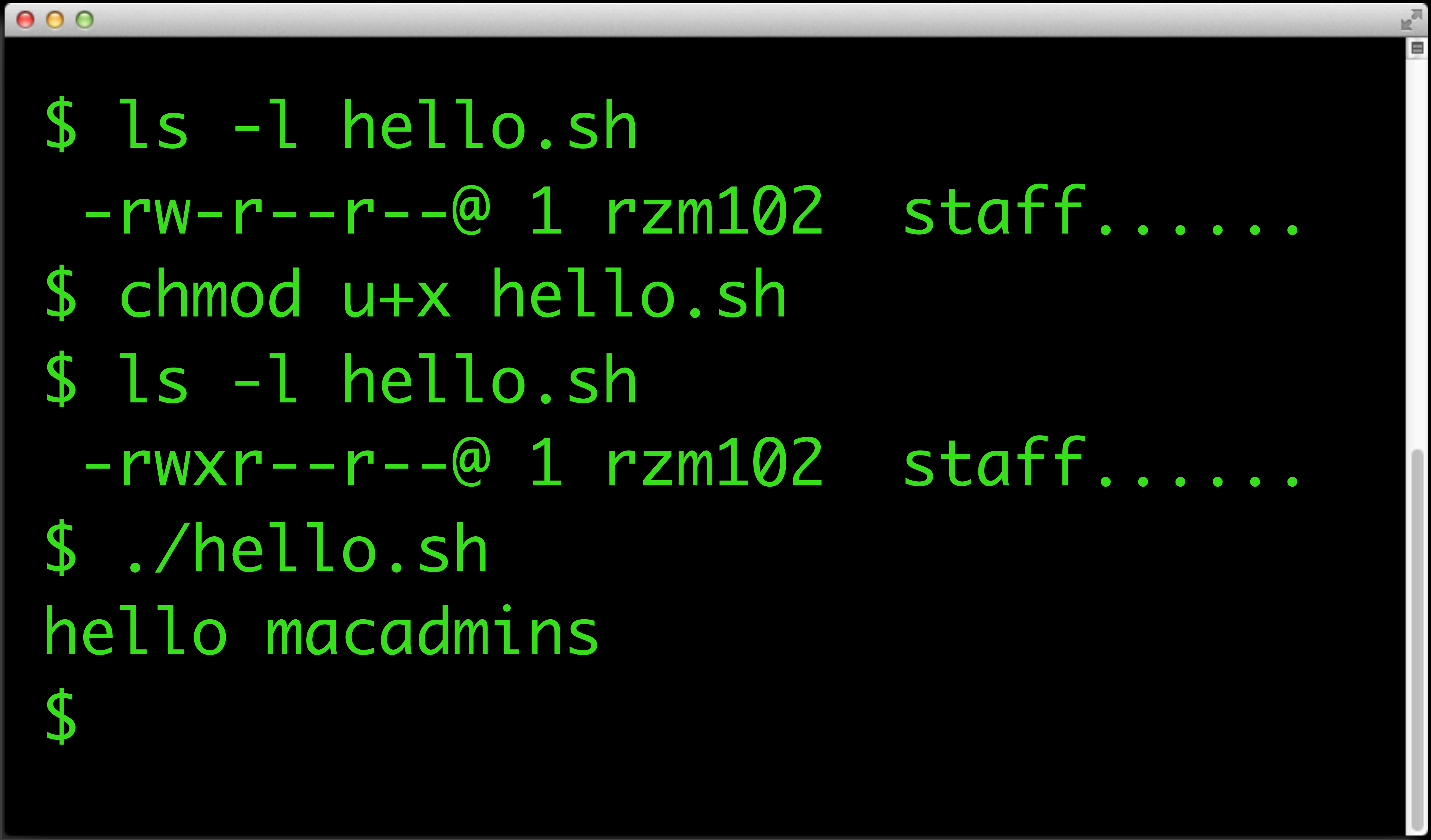
```
$ ls -l hello.sh
 -rw-r--r--@ 1 rzm102  staff......
$ chmod u+x hello.sh
$
```



```
$ ls -l hello.sh
 -rw-r--r--@ 1 rzm102  staff......
$ chmod u+x hello.sh
$ ls -l hello.sh
 -rwxr--r--@ 1 rzm102  staff......
$
```



```
$ ls -l hello.sh
 -rw-r--r--@ 1 rzm102  staff......
$ chmod u+x hello.sh
$ ls -l hello.sh
 -rwxr--r--@ 1 rzm102  staff......
$ ./hello.sh
```



```
$ ls -l hello.sh
 -rw-r--r--@ 1 rzm102  staff......
$ chmod u+x hello.sh
$ ls -l hello.sh
 -rwxr--r--@ 1 rzm102  staff......
$ ./hello.sh
hello macadmins
$
```

## Exercises

# Terminal Exercise

- sw_vers
- system_profiler
- systemsetup
- networksetup
- diskutil
- open

## Exercise 1

## echo Command

- Outputs string to stdout
- Double Quotes around string
- Add echos for
  - debugging
  - information

## Network Setup

- Output IP Address of
  - Wi-Fi and Ethernet ports
  - Output Ethernet 2 port
- man networksetup for usage

## Exercise 2

## System Profile & System Setup

- Print System Hardware Data
- Print Computer Name

## Exercise 3

## Software Version & Disk Utility

- Print OS X Product Name, Product Version, Build Version
- Print Hard Drive(s) Size, Available Space, Type
- Don’t Forget About CoreStorage!

## Exercise 4

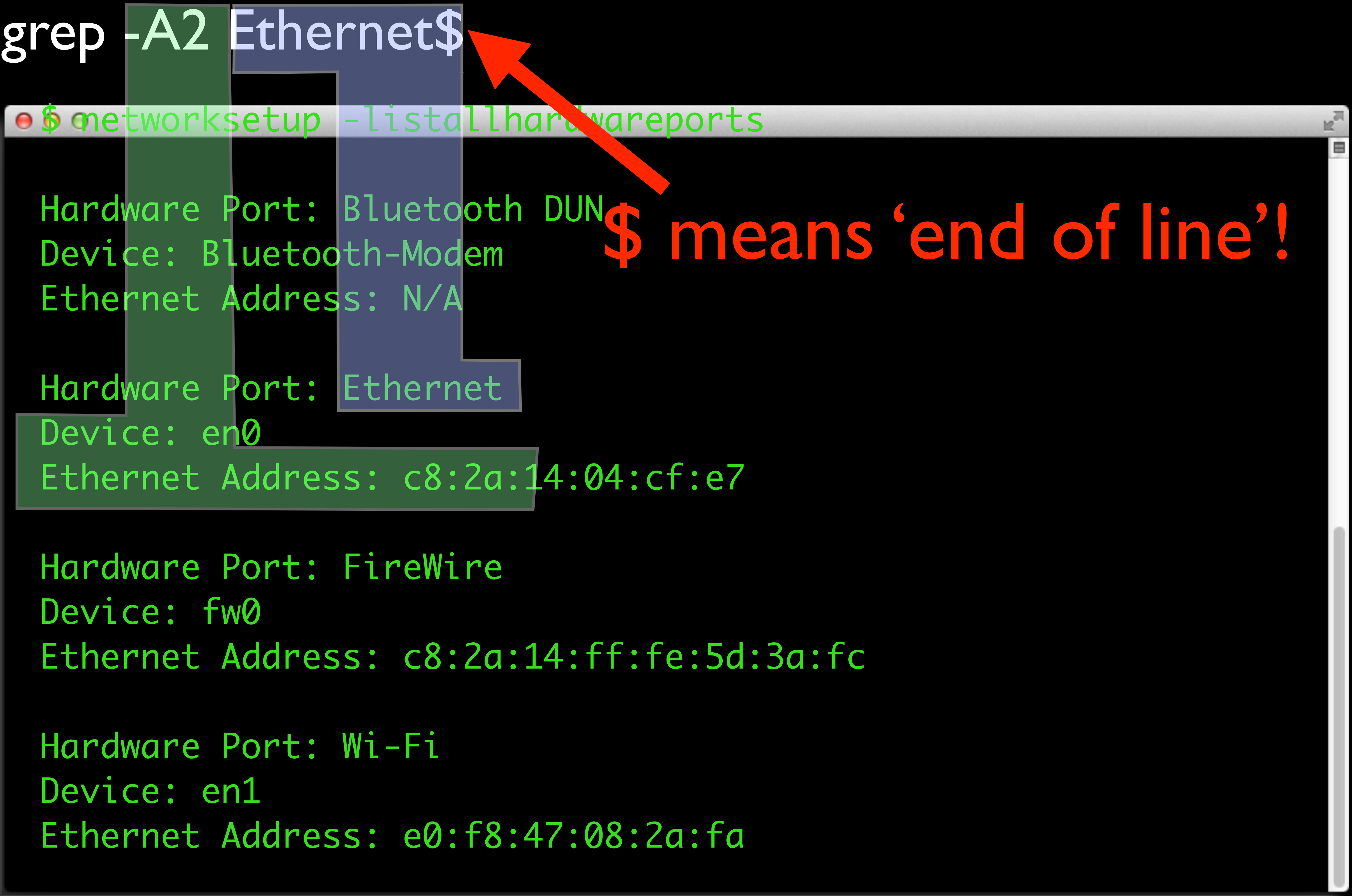
## Basic RegEx

- Beginning of Line: ^
- End of Line: $
- All Chars,  Any Amount: *
- All Chars, Single Char: .

## Grep

- Search & Match Patterns
- Prints Match to stdout
- Ignore Case: -i
- Print 5 Lines After Match: -A5
- Print 5 Lines Before Match: -B5

## grep -A2 Ethernet$

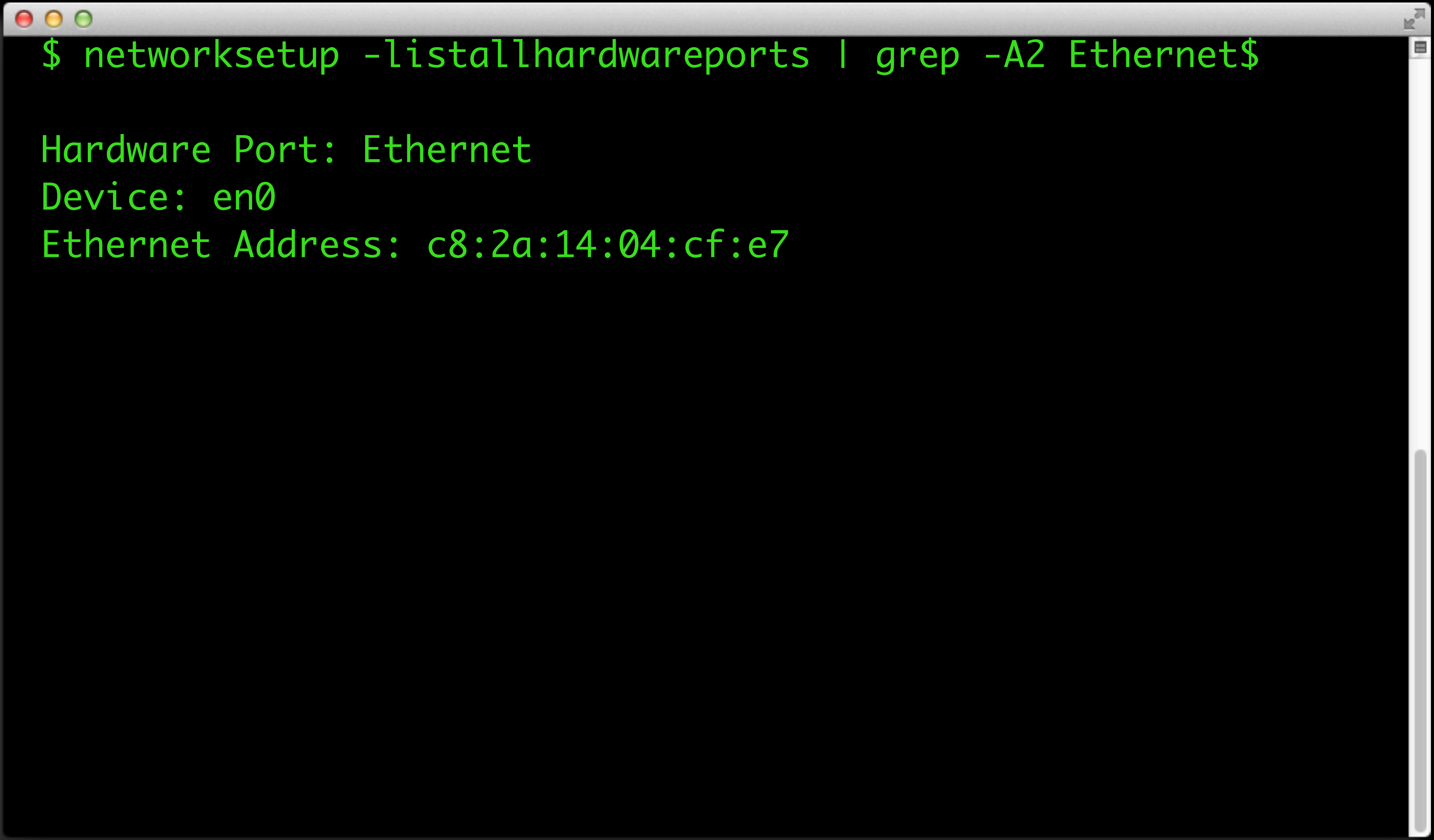


```
$ networksetup -listallhardwareports

Hardware Port: Bluetooth DUN
Device: Bluetooth-Modem
Ethernet Address: N/A

Hardware Port: Ethernet
Device: en0
Ethernet Address: c8:2a:14:04:cf:e7

Hardware Port: FireWire
Device: fw0
Ethernet Address: c8:2a:14:ff:fe:5d:3a:fc

Hardware Port: Wi-Fi
Device: en1
Ethernet Address: e0:f8:47:08:2a:fa
```

## Piping

## Pipe

- A pipe is: |
- Pass output of left side to right side
- String multiple commands together



```
$ networksetup -listallhardwareports | grep -A2 Ethernet$

Hardware Port: Ethernet
Device: en0
Ethernet Address: c8:2a:14:04:cf:e7
```

## Grep It!

- Grep output of `networksetup -listallhardwareports`
- for:
  - Ethernet
  - Wi-Fi
- Return 3 lines from each

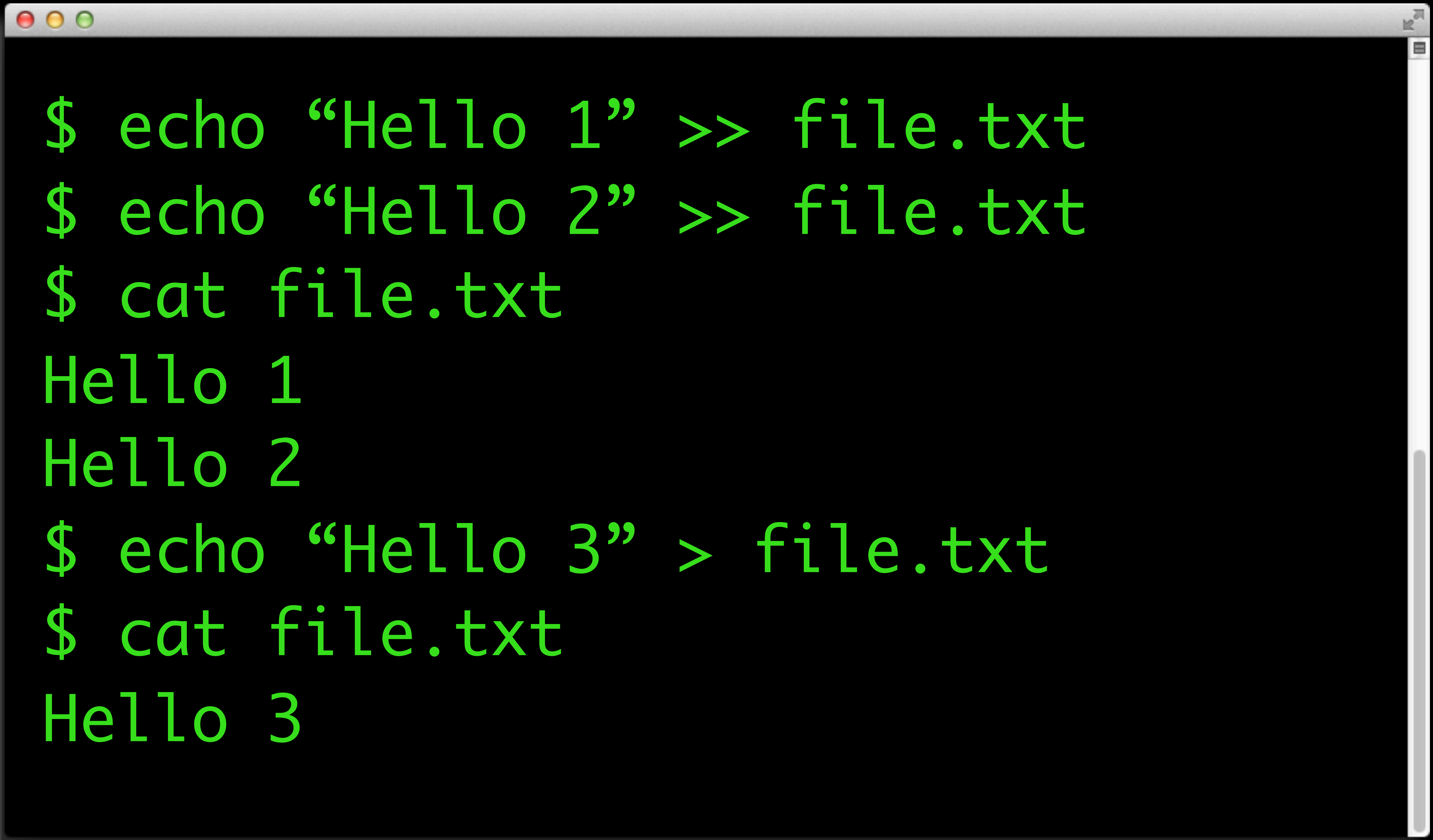
## Exercise 5

## Variables

- Set with ‘=’
  - VAR=10
- Precede Variable With ‘$’ After Value Has Been Set
  - echo ‘$VAR’
  - Prints “10”

## Redirection

- Overwrite File: >
- Append to File: >>
- Pipe Text Between Programs: |



```
$ echo “Hello 1” >> file.txt
$ echo “Hello 2” >> file.txt
$ cat file.txt
Hello 1
Hello 2
$ echo “Hello 3” > file.txt
$ cat file.txt
Hello 3
```

## tr

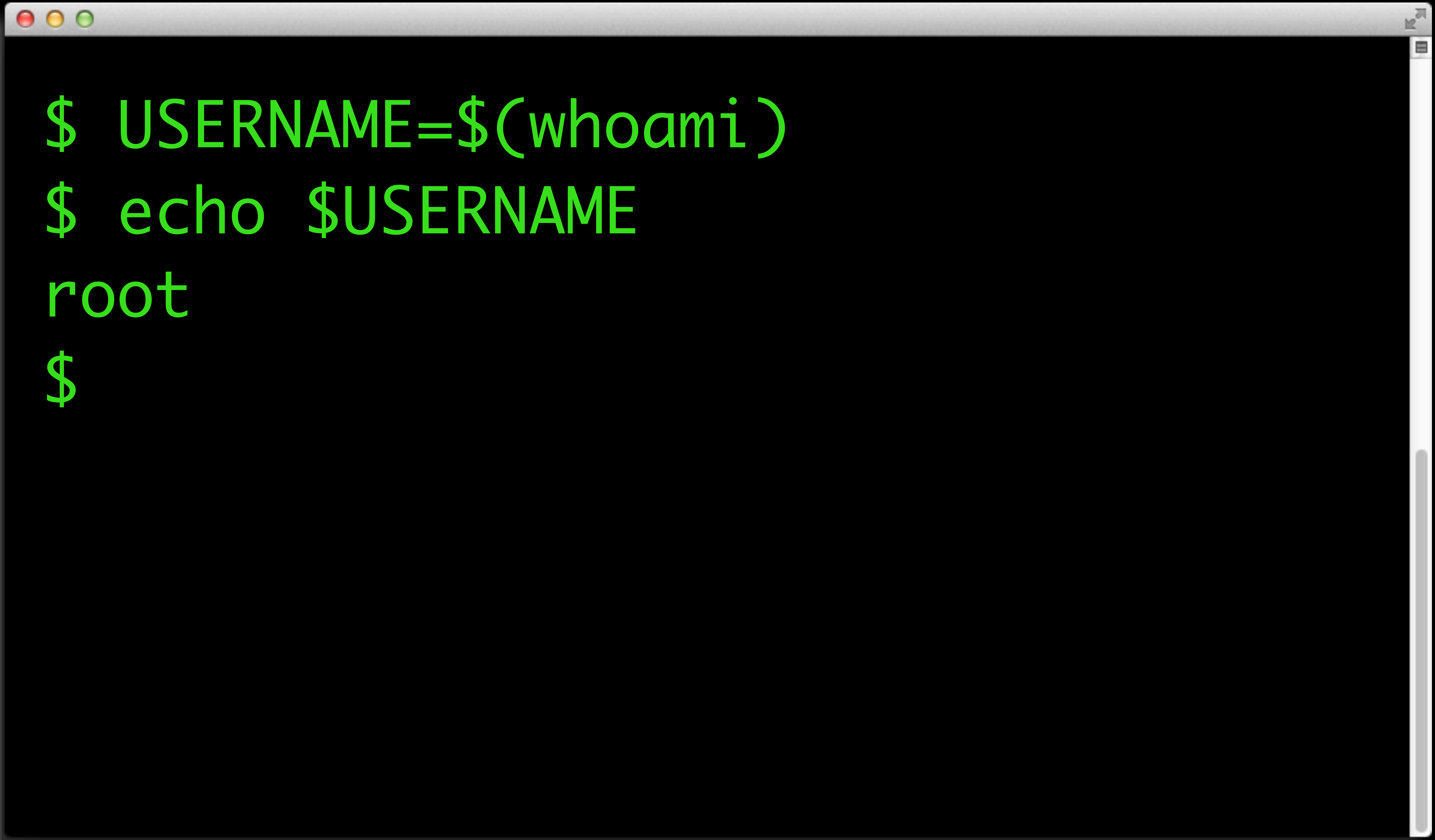
- Delete Pattern: tr -d “pattern”
- Serial Number:

```
system_profiler SPHardwareDataType |
 grep "Serial Number" |
 tr -d "Serial Number (system): "
```

## Command Substitution

- Execute this command, use it’s output
- Use in variables

```
variable=$(command here)
```



```
$ USERNAME=$(whoami)
$ echo $USERNAME
root
$
```

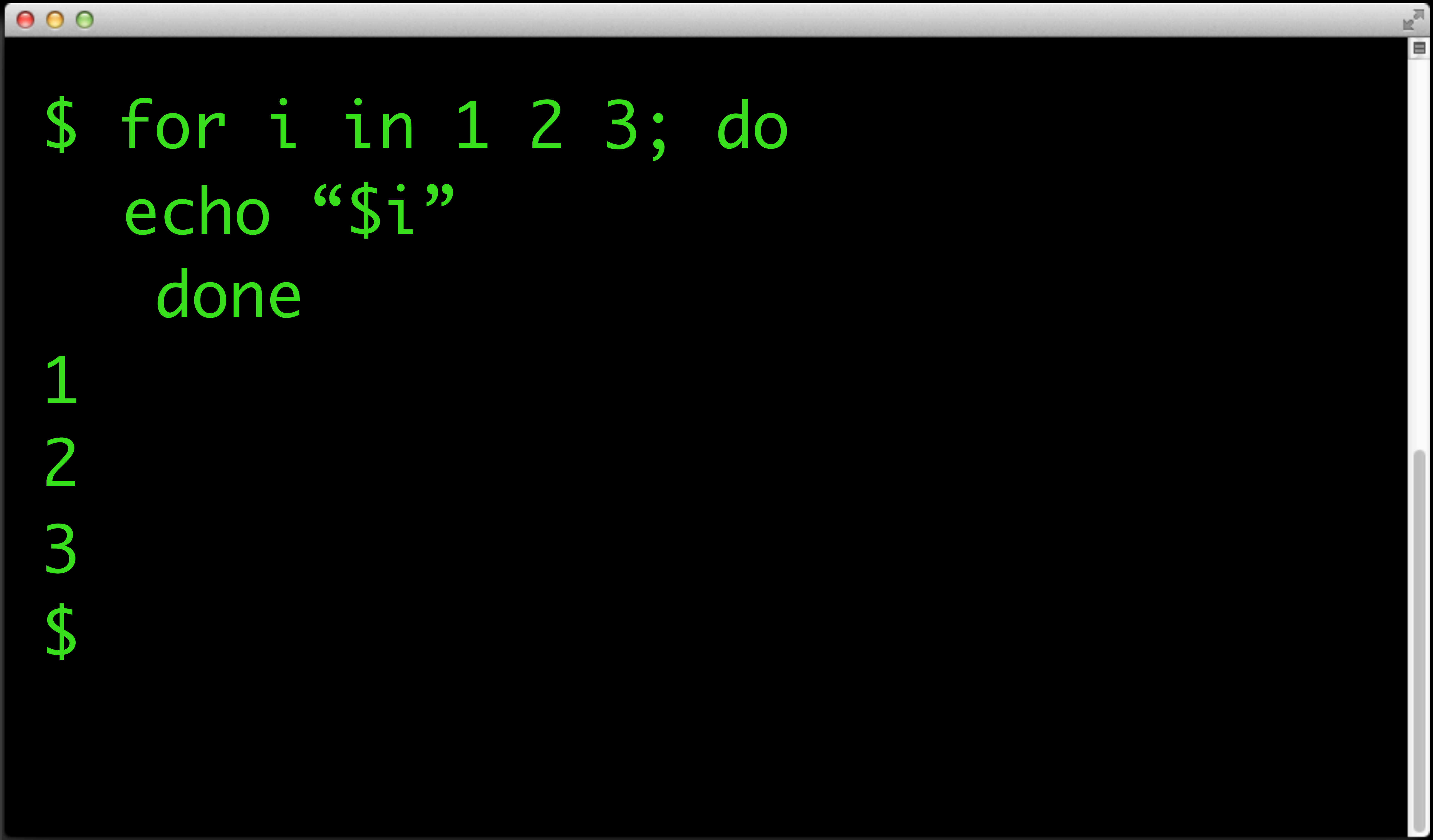
# Grep & Redirect!

- Grep system_profiler for Memory, Model Identifier, Processor Type, Processor Speed
- Set Variable For Filename
- Output all text to file
- Name file username-serial#

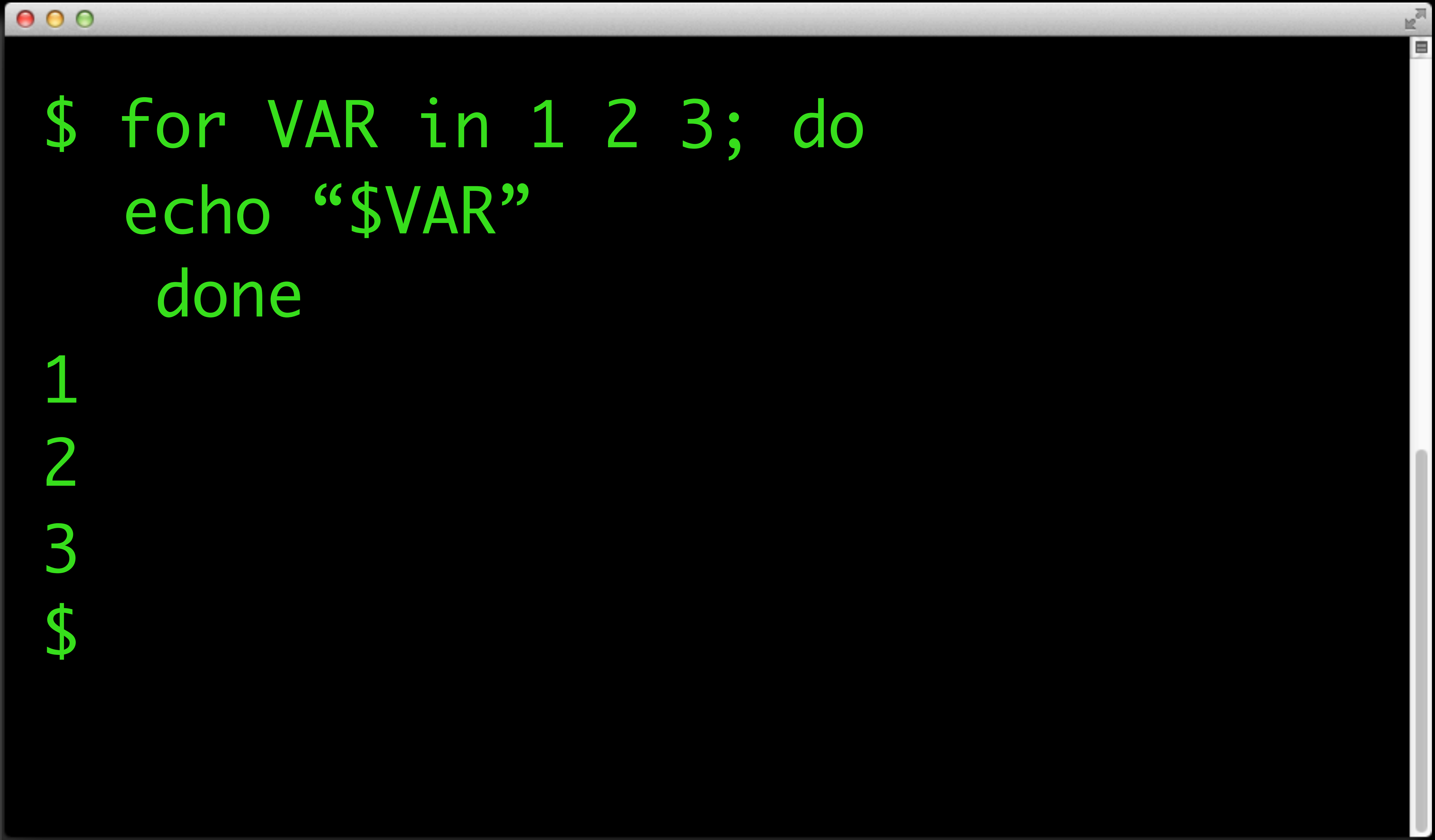
## Exercise 6

## For Loops

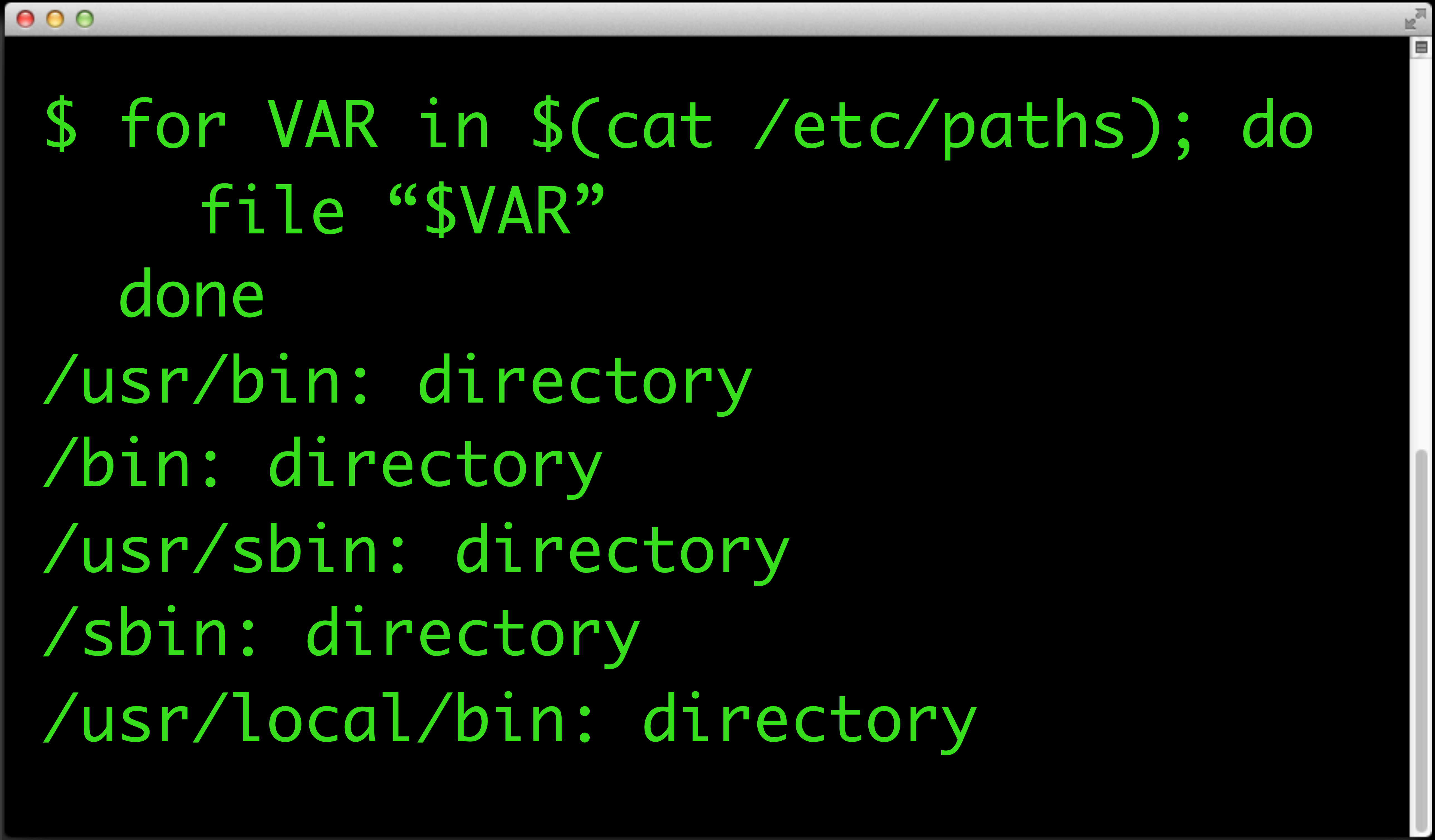
- Repeat Commands
- Pass Arguments for each loop from:
  - A command output
  - A list of text



```
$ for i in 1 2 3; do
  echo “$i”
   done
1
2
3
$
```



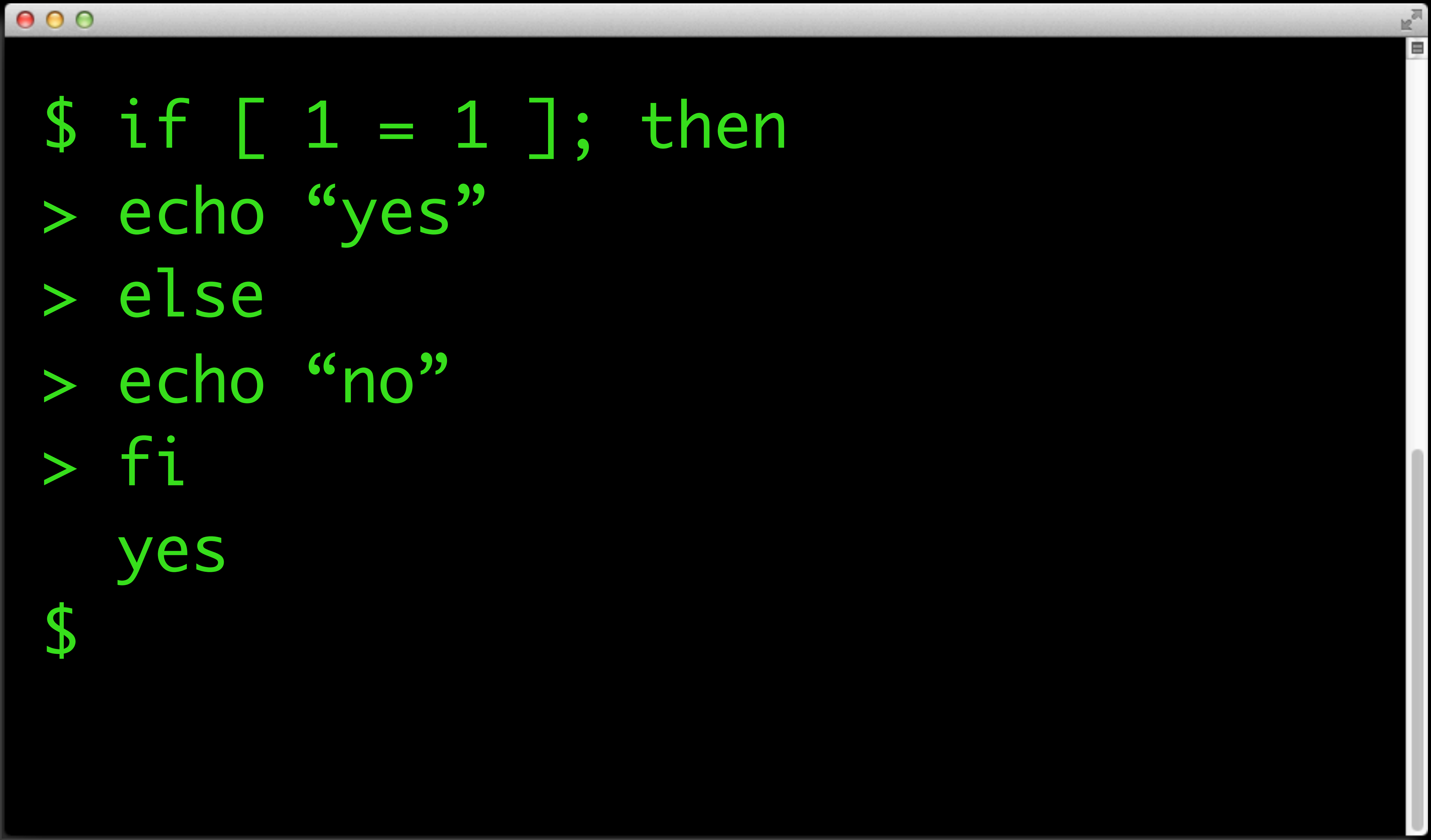
```
$ for VAR in 1 2 3; do
   echo “$VAR”
    done
1
2
3
$
```



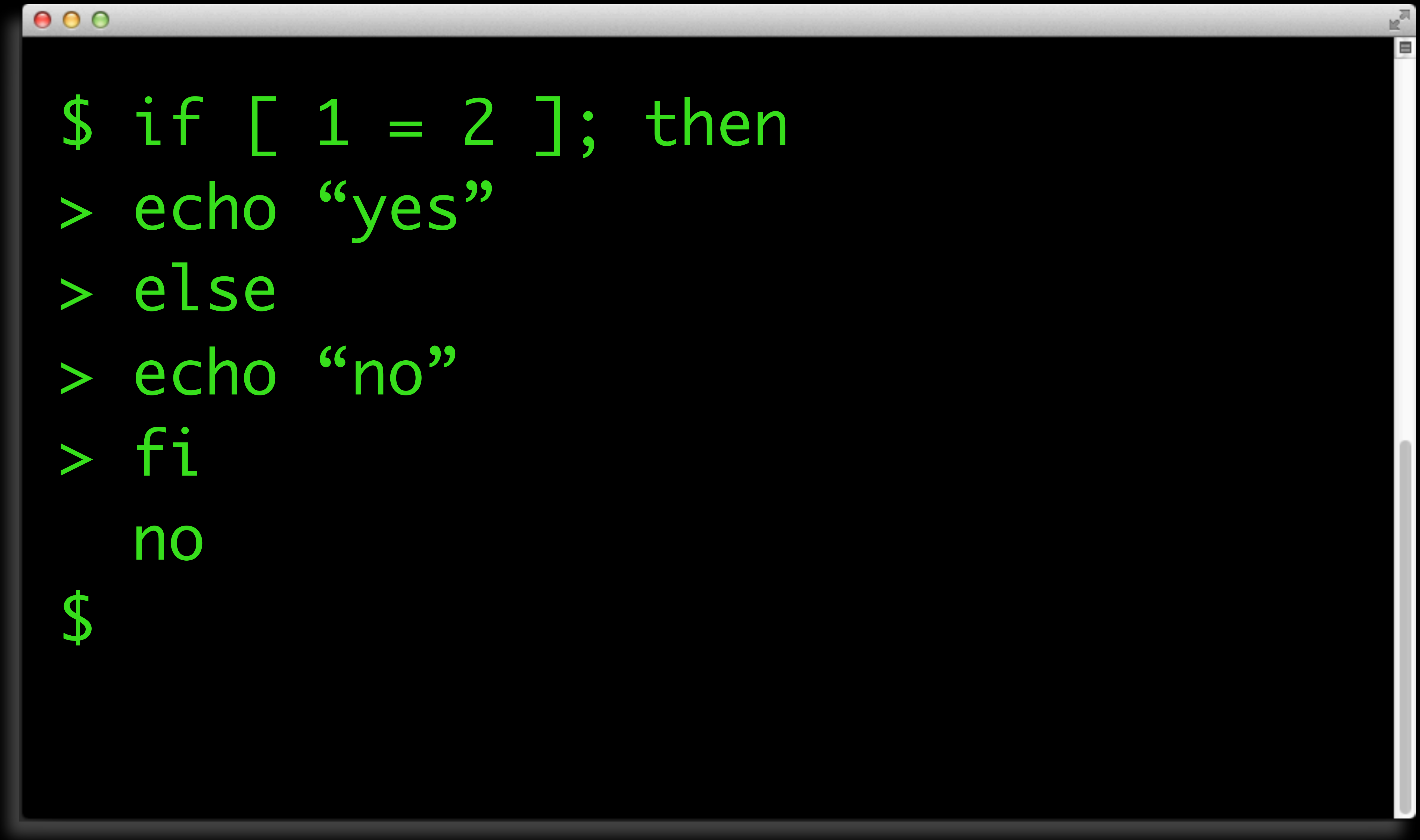
```
$ for VAR in $(cat /etc/paths); do
    file “$VAR”
  done
/usr/bin: directory
/bin: directory
/usr/sbin: directory
/sbin: directory
/usr/local/bin: directory
```

## Tests

- True/False
- If condition is true
    - do something!
- else
    - do something else!



```
$ if [ 1 = 1 ]; then
> echo “yes”
> else
> echo “no”
> fi
  yes
$
```

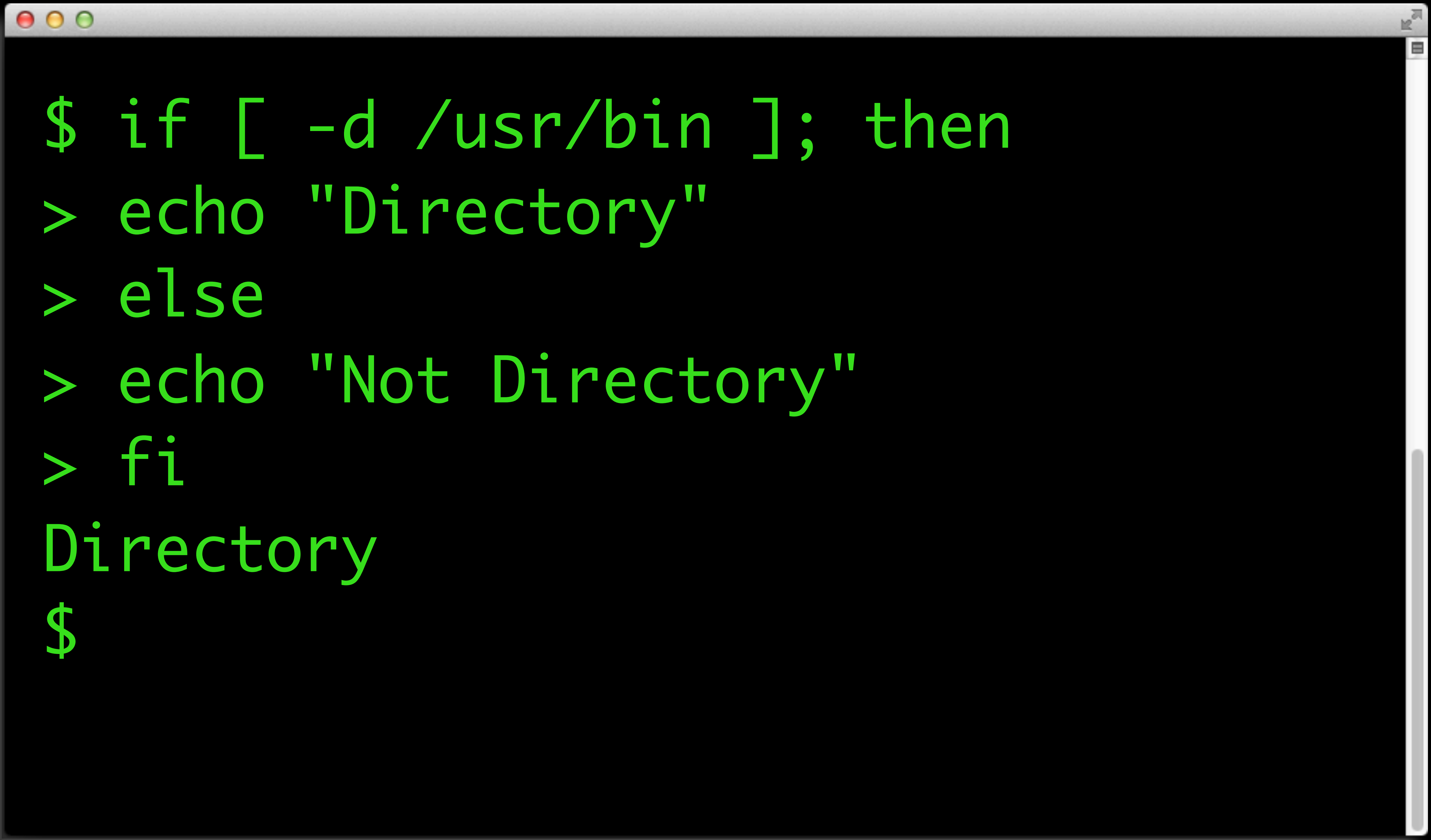


```
$ if [ 1 = 2 ]; then
> echo “yes”
> else
> echo “no”
> fi
  no
$
```

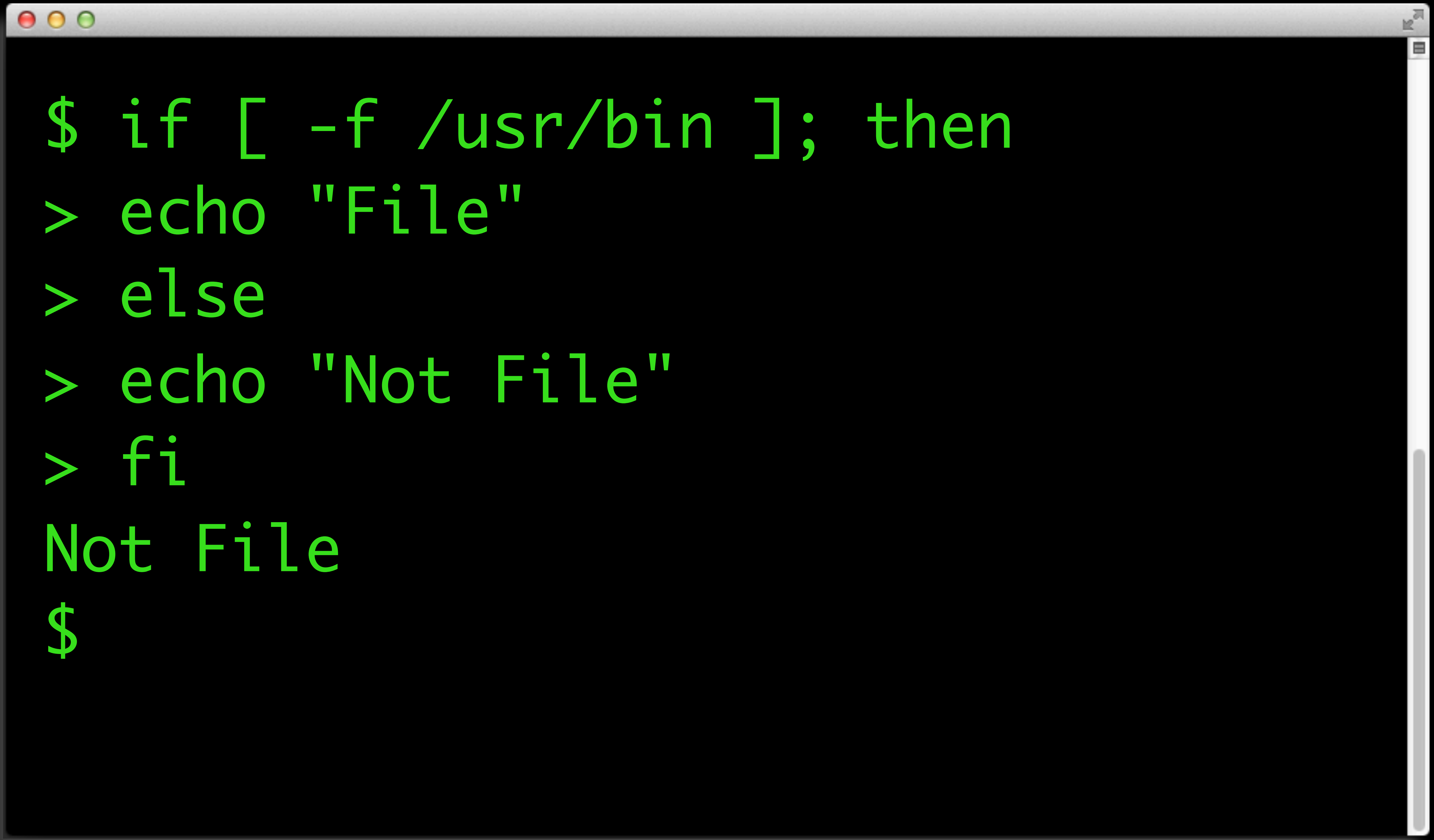
## File Tests

- File Exists:  `[ -e ./file ]`
- Not Zero Size: `[ -s ./file ]`
- Symbolic Link: `[ -h ./file ]`

http://tldp.org/LDP/abs/html/fto.html



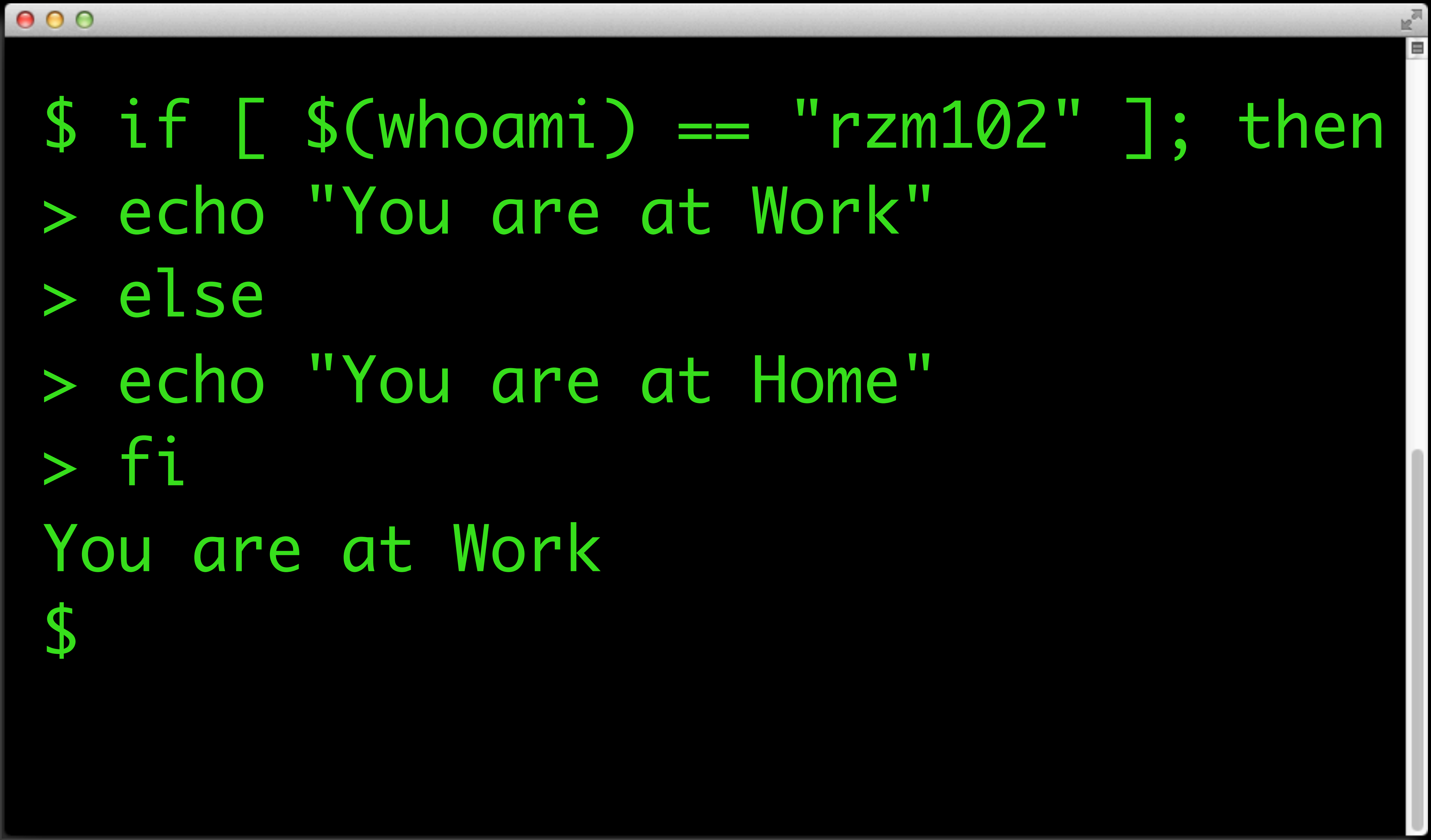
```
$ if [ -d /usr/bin ]; then
> echo "Directory"
> else
> echo "Not Directory"
> fi
Directory
$
```



```
$ if [ -f /usr/bin ]; then
> echo "File"
> else
> echo "Not File"
> fi
Not File
$
```

## String Comparison

- Is Equal To:
  `[ “$string1” == “$string2” ]`
- Is NOT Equal:
  `[ “$string1” != “$string2” ]`
- String is Null:
  `[ -n “$string1” ]`



```
$ if [ $(whoami) == "rzm102" ]; then
> echo "You are at Work"
> else
> echo "You are at Home"
> fi
You are at Work
$
```

# Read, Loop, Test, Redirect!

- Overwrite File With First Echo
- For Loop: diskutil disk0-disk4
- Ask to:  View File Output, Save File Output
- Sleep At End
- Clear Terminal Window

## Exercise 7

## Show & Tell

## Q & A

## What Now?

- Run Scripts With:
  - Apple Remote Desktop
    Open Lab, Room 109 Wed. @ 3:00pm
  - Payload Free Package
    Practical Packaging, Room 207 Wed. @ 1:30pm
  - LaunchD plist

## Thank You!

# Thank You!

Resources:

- http://tldp.org/LDP/abs/html/index.html
- http://lifehacker.com/5743814/become-a-command-line-ninja-with-these-time+saving-shortcuts
- http://mywiki.wooledge.org/BashGuide
- http://developer.apple.com/library/mac/documentation/OpenSource/Conceptual/ShellScripting/ShellScripting.pdf