# Shell(bash) Scripting

2014 MACADMINS CONFERENCE AT PENN STATE

Feedback: http://j.mp/psumac13

## Jay Hoff

- ITS/IDS
  - Identity Services
    - Systems Administrator
  - jeh26@psu.edu
  - @jayhoff

# Rusty Myers

- ITS/CLC
  - Mac and Linux Group
    - Systems Administrator
  - rzm102@psu.edu
  - @thespider

## Agenda

- Welcome
- Part 1 - Terminal Use
- Part 2 - Basic Shell
- Part 3 - Basic Bash
- Part 4 - Advanced Advanced

## Jump In

https://github.com/rustymyers/ShellScriptingPSUMAC2014

"Download Zip"

TextWrangler

http://www.barebones.com/products/textwrangler/download.html

## Welcome to bash

# What's bash?

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

## What's BASH?

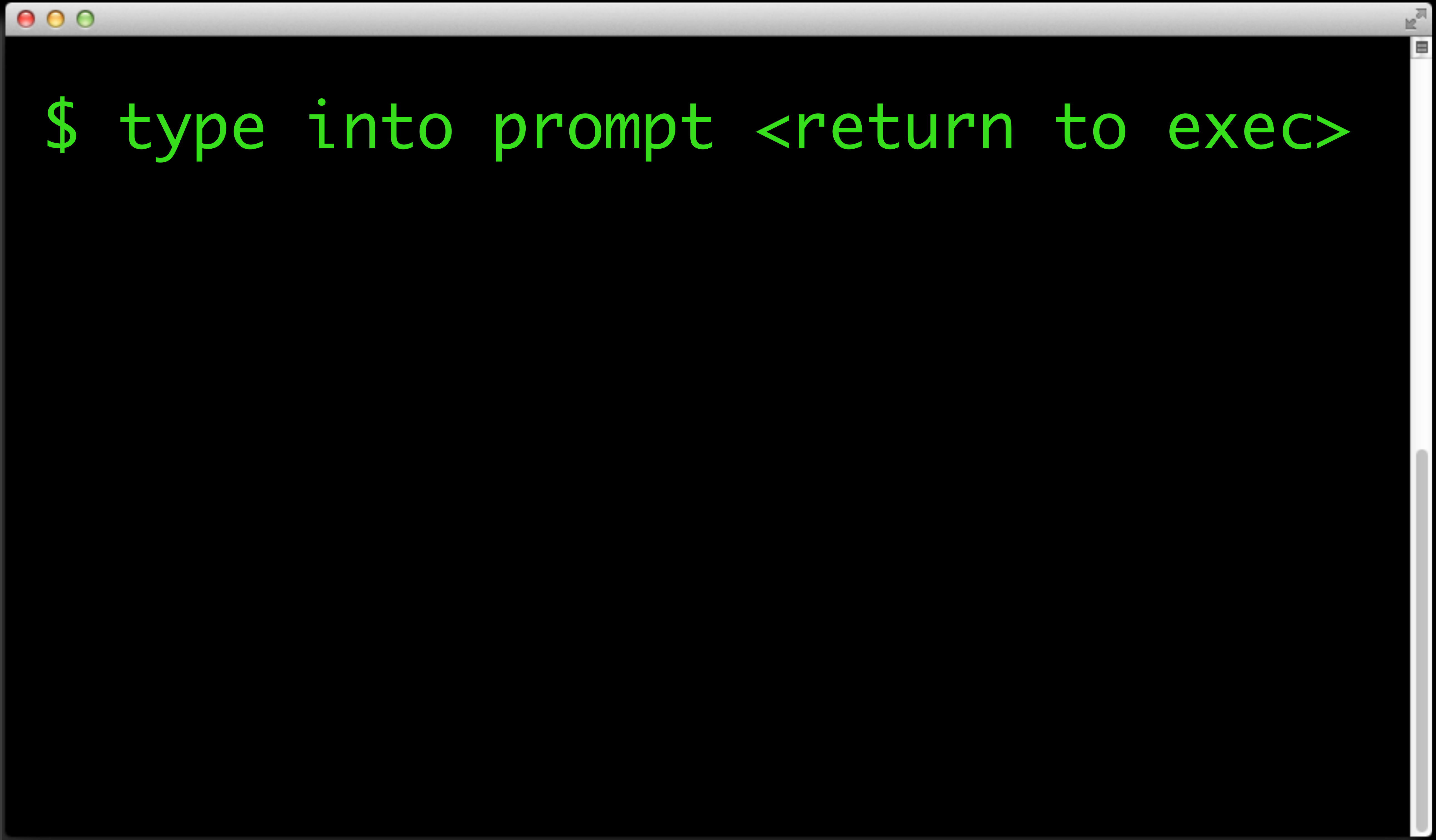
- Bourne Again Shell (bash)
- Brian Fox
  - Programmed BASH
  - beta 1989
- Updated Bourne Shell (sh)

## Part 1

- Terminal.app
- Paths
- Basic Script
- Permissions
- Basic Commands

## Terminal.app

- Open /Applications/Utilities/
- Spotlight search for “term”



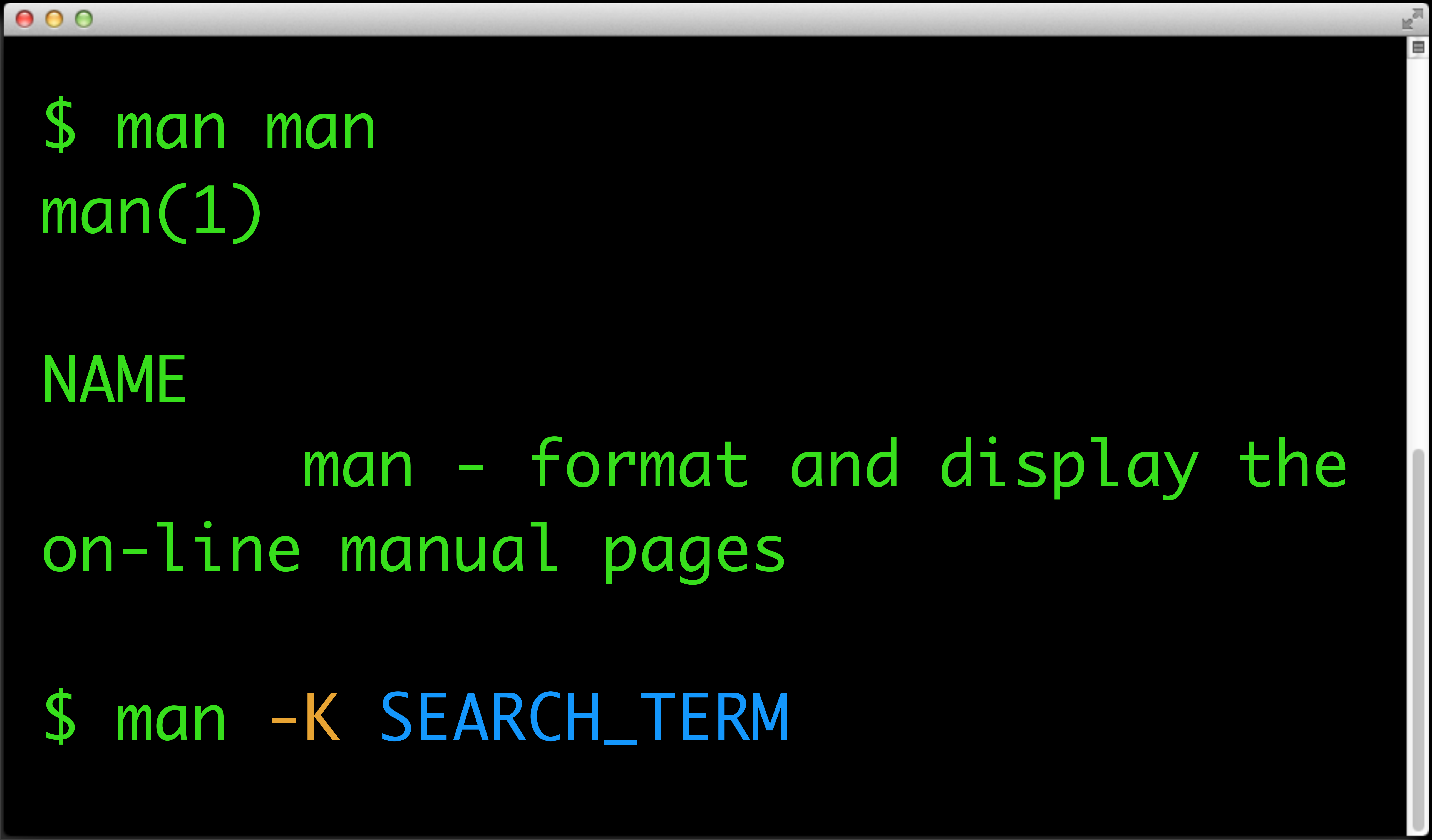
```
$ type into prompt <return to exec>
```

## Interrupts

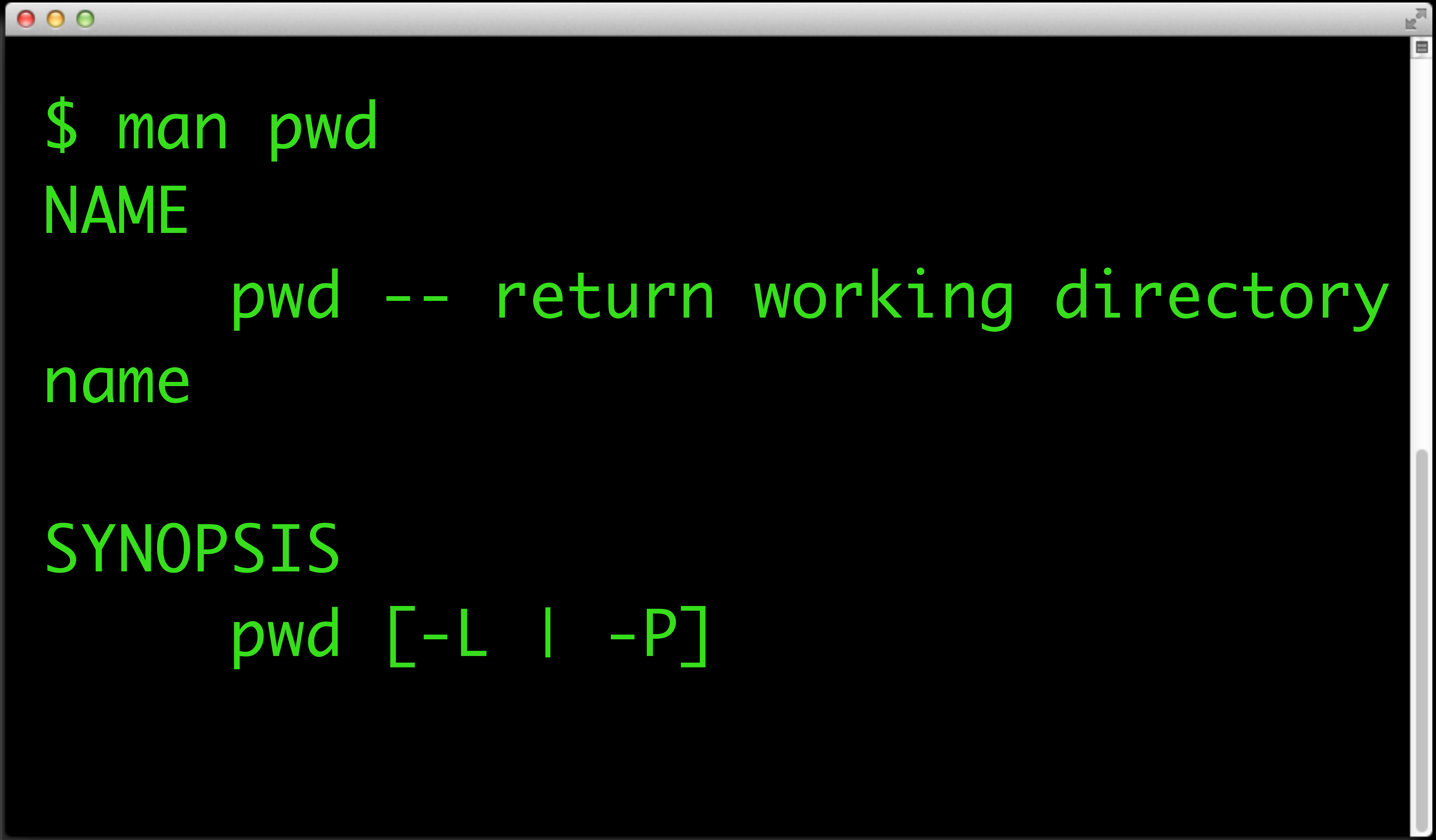
- Ctrl-C = Interrupt/Kill
- Ctrl-D = Close Shell

# Command Basics

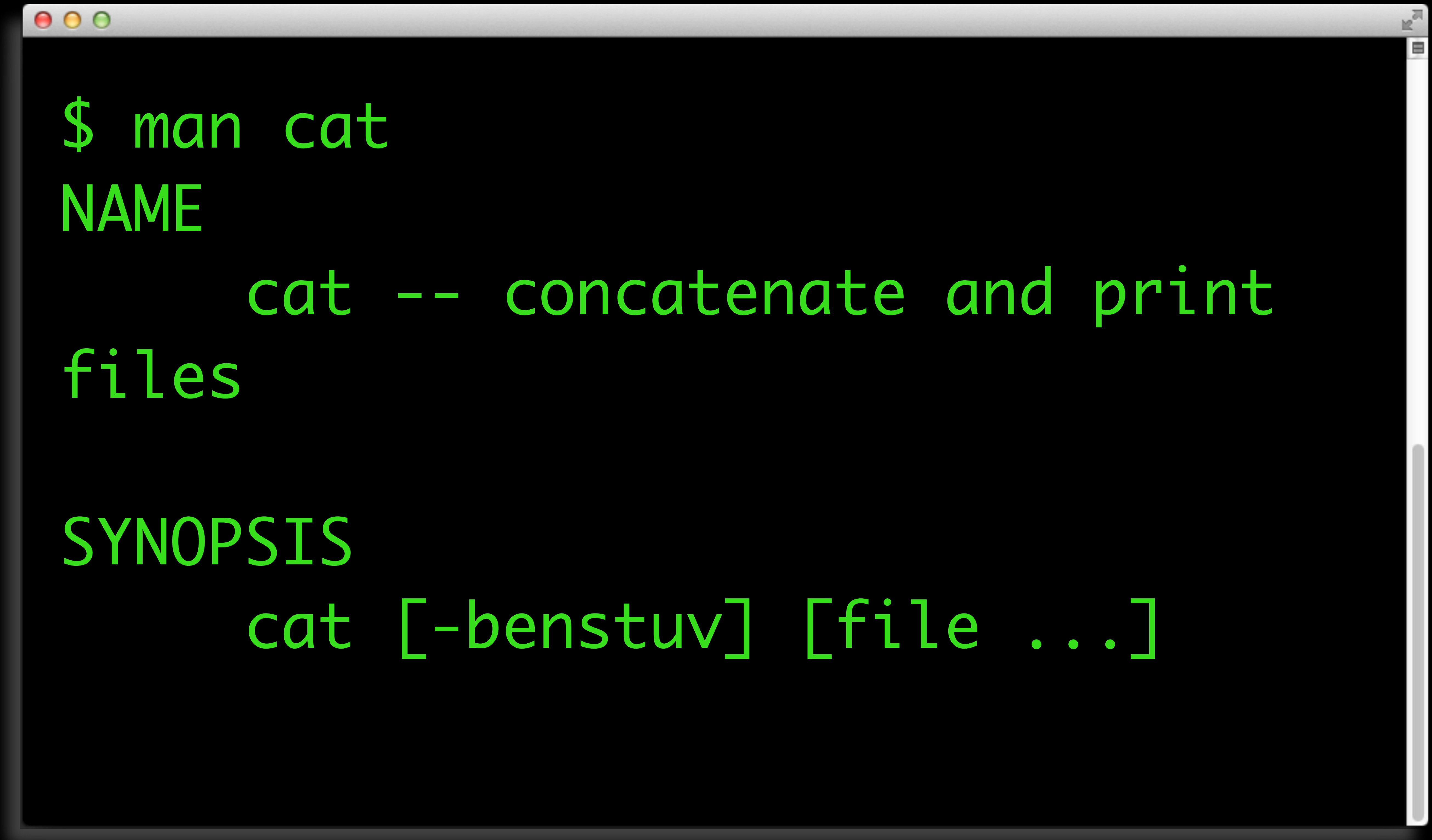
- Programs with Specific Purpose
- Simple commands have no arguments
- `command argument1 argumentN`
- Some Commands use flags
  - `command -flag arguments`



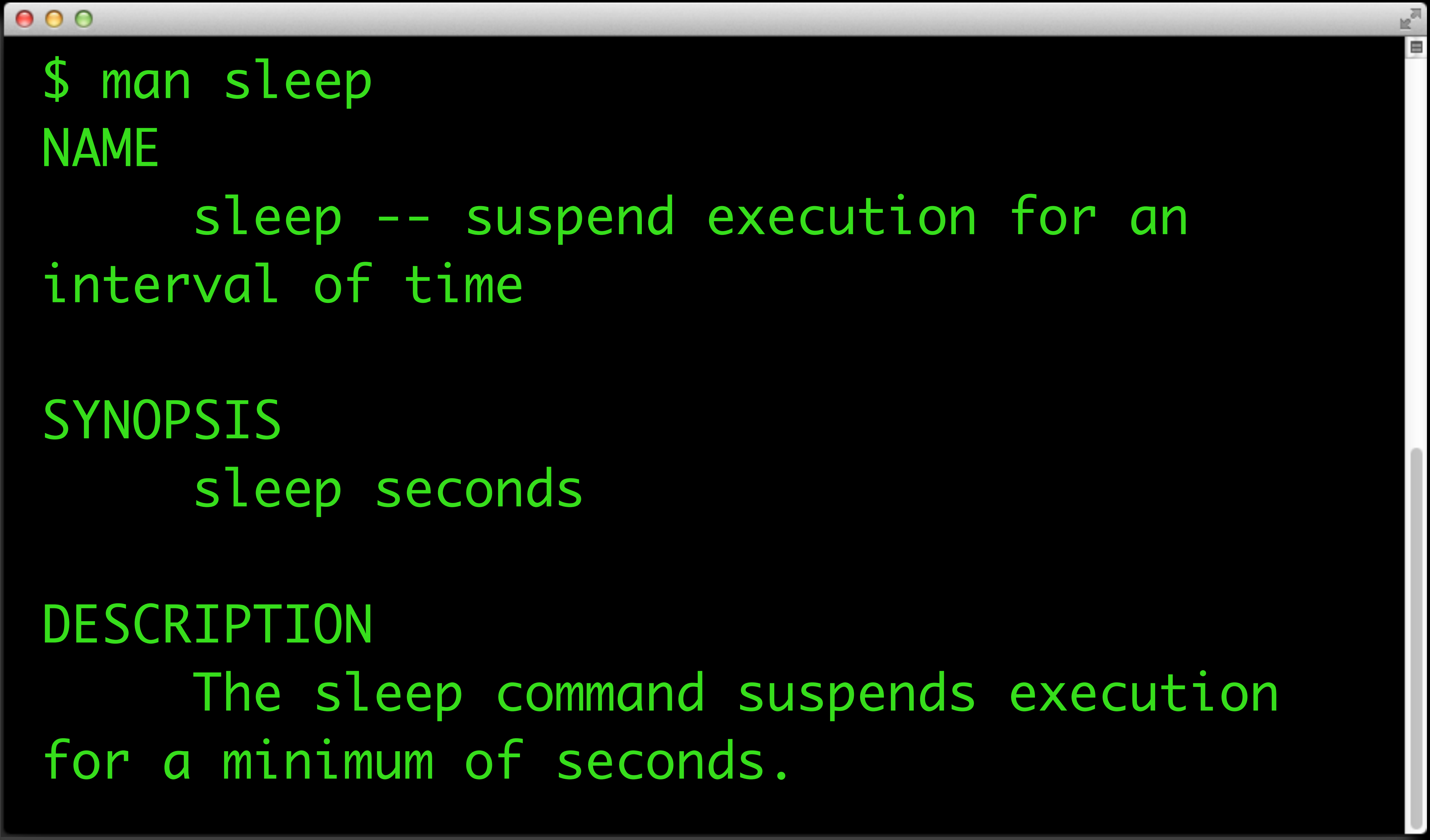
```
$ man man
man(1)

NAME
       man - format and display the
on-line manual pages

$ man -K SEARCH_TERM
```



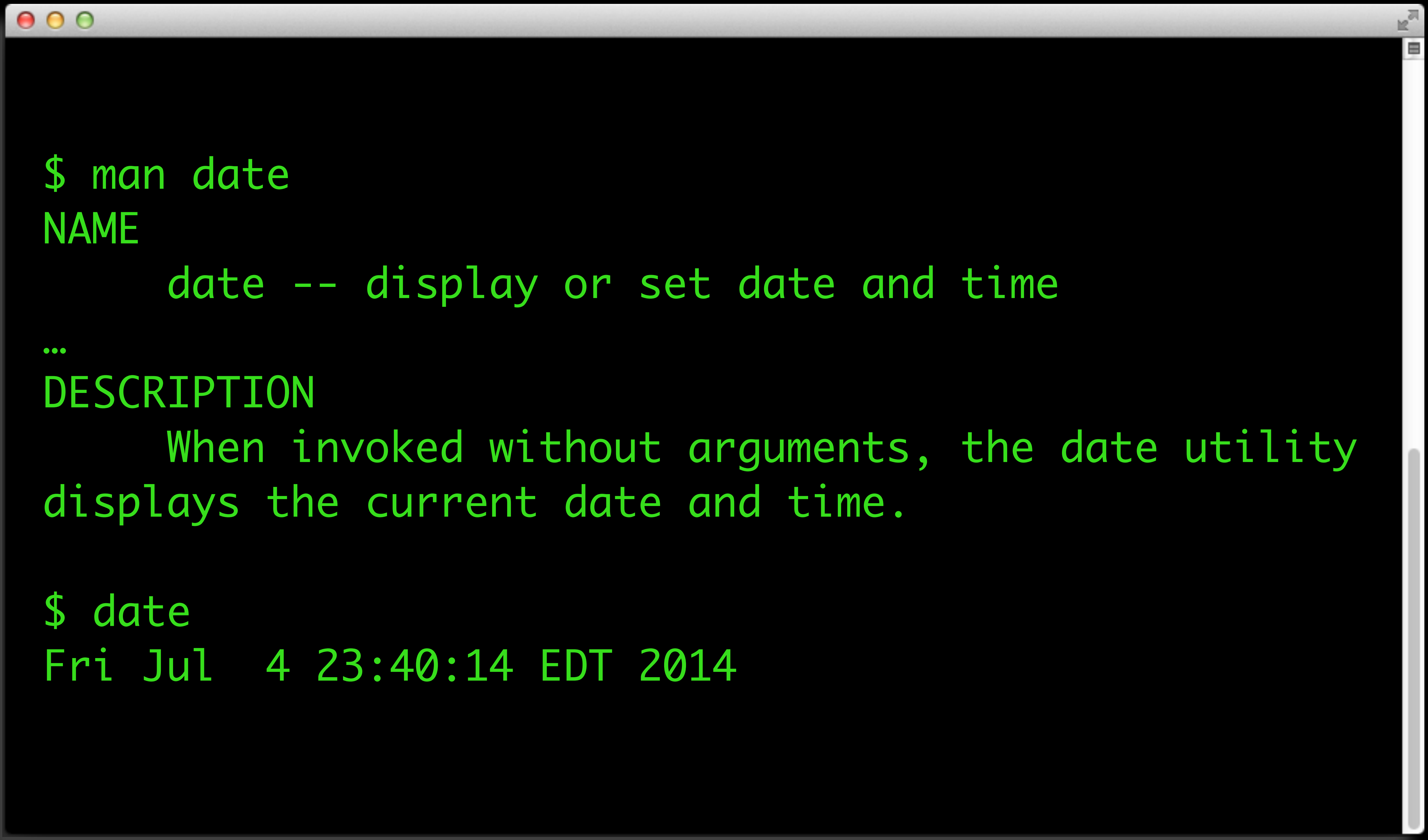
```
$ man pwd
NAME
     pwd -- return working directory
name

SYNOPSIS
     pwd [-L | -P]
```



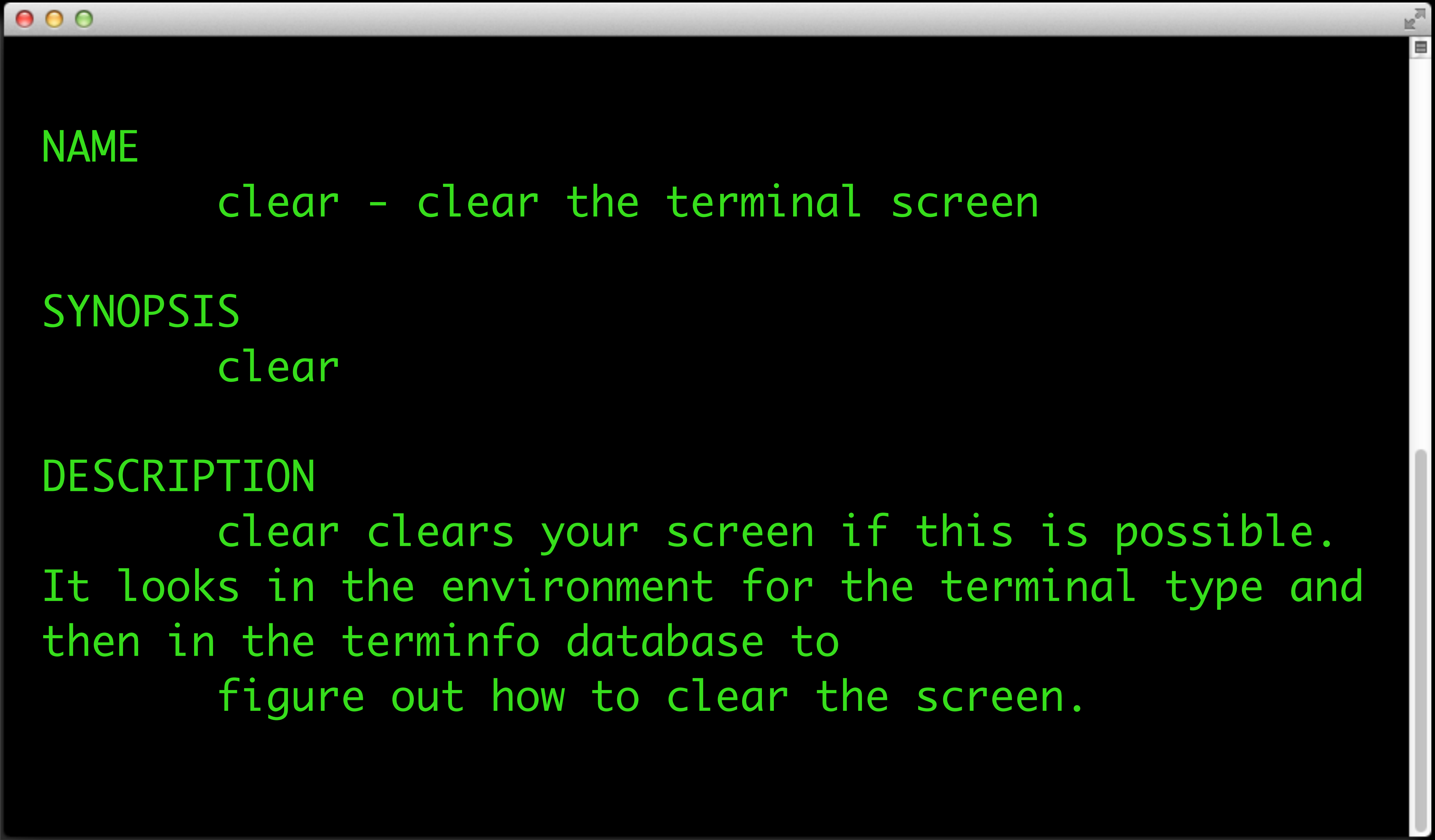
```
$ man cat
NAME
     cat -- concatenate and print
files

SYNOPSIS
     cat [-benstuv] [file ...]
```



```
$ man sleep
NAME
     sleep -- suspend execution for an
interval of time

SYNOPSIS
     sleep seconds

DESCRIPTION
     The sleep command suspends execution
for a minimum of seconds.
```

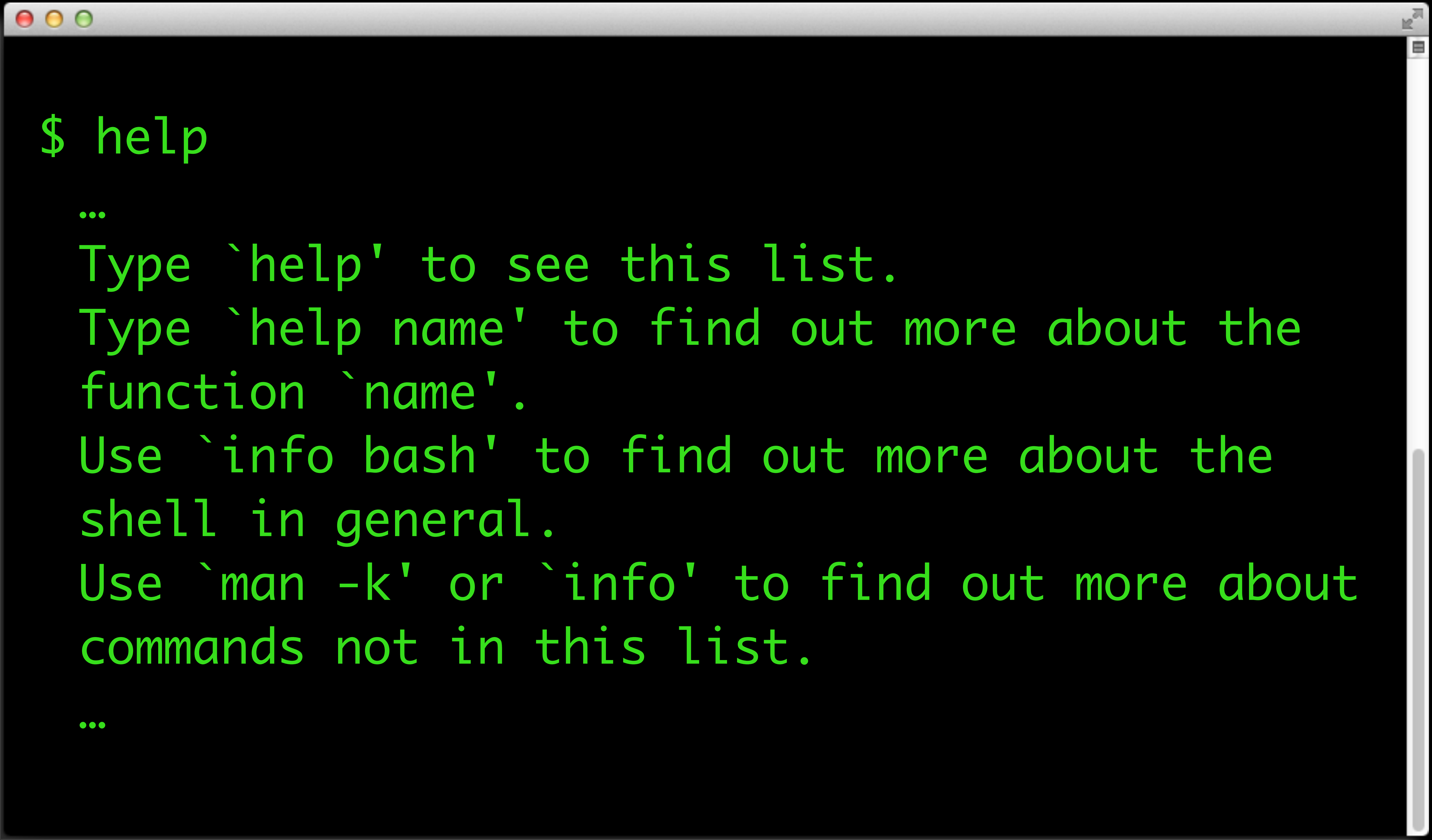


```
$ man date
NAME
     date -- display or set date and time
…
DESCRIPTION
     When invoked without arguments, the date utility
displays the current date and time.

$ date
Fri Jul  4 23:40:14 EDT 2014
```

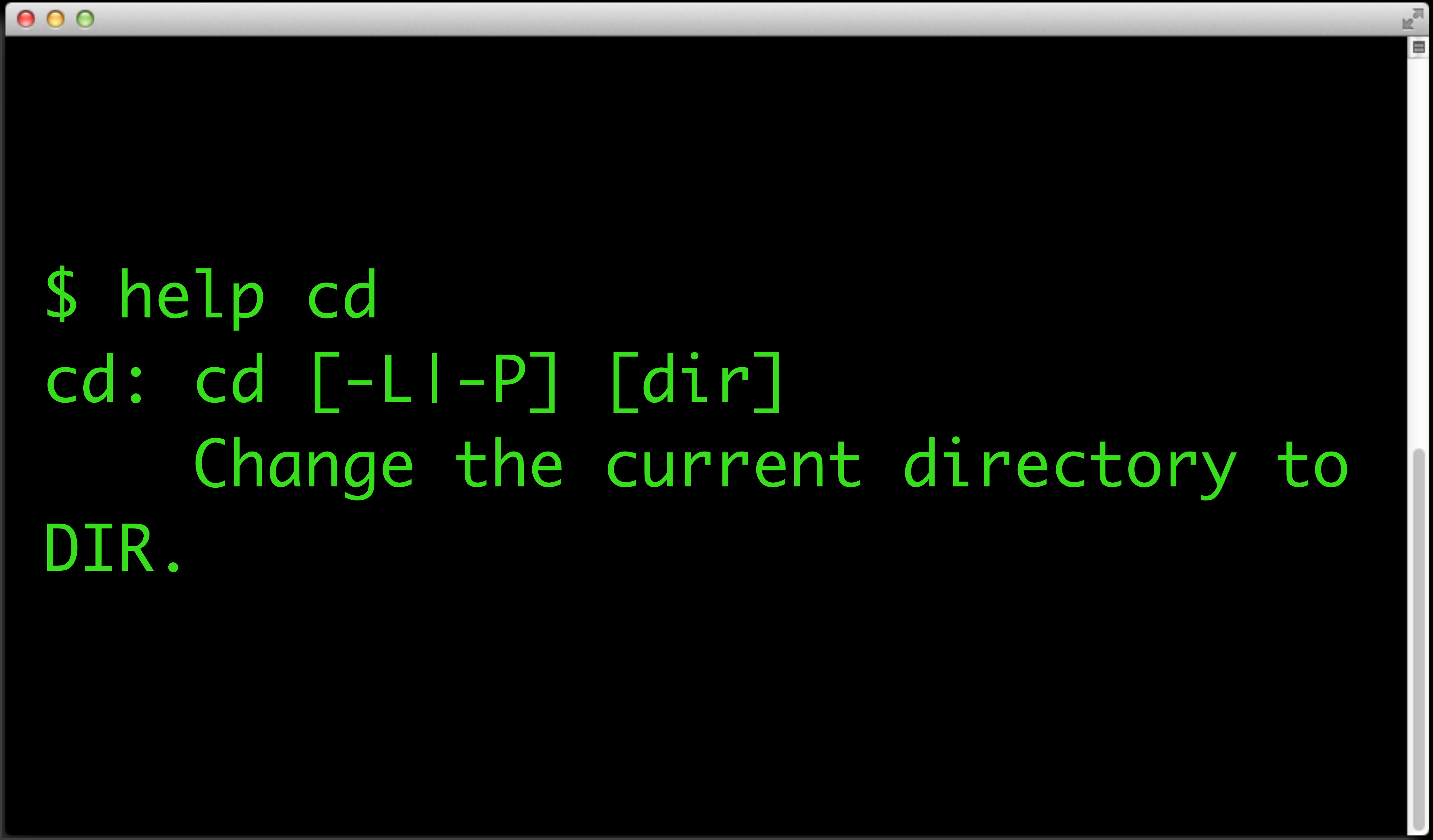


```
NAME
       clear - clear the terminal screen

SYNOPSIS
       clear

DESCRIPTION
       clear clears your screen if this is possible.
It looks in the environment for the terminal type and
then in the terminfo database to
       figure out how to clear the screen.
```

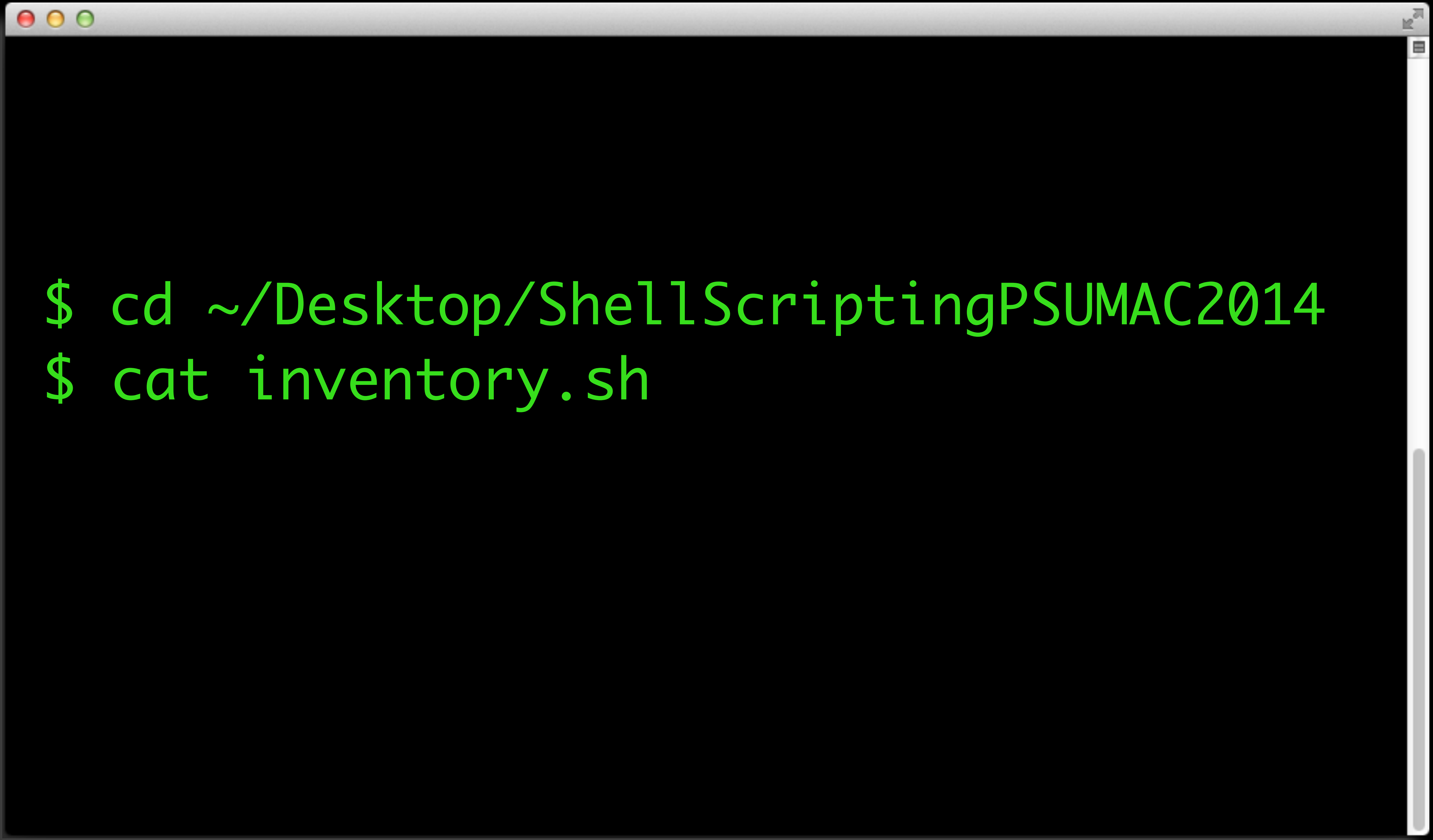
Feedback: http://j.mp/psumac13



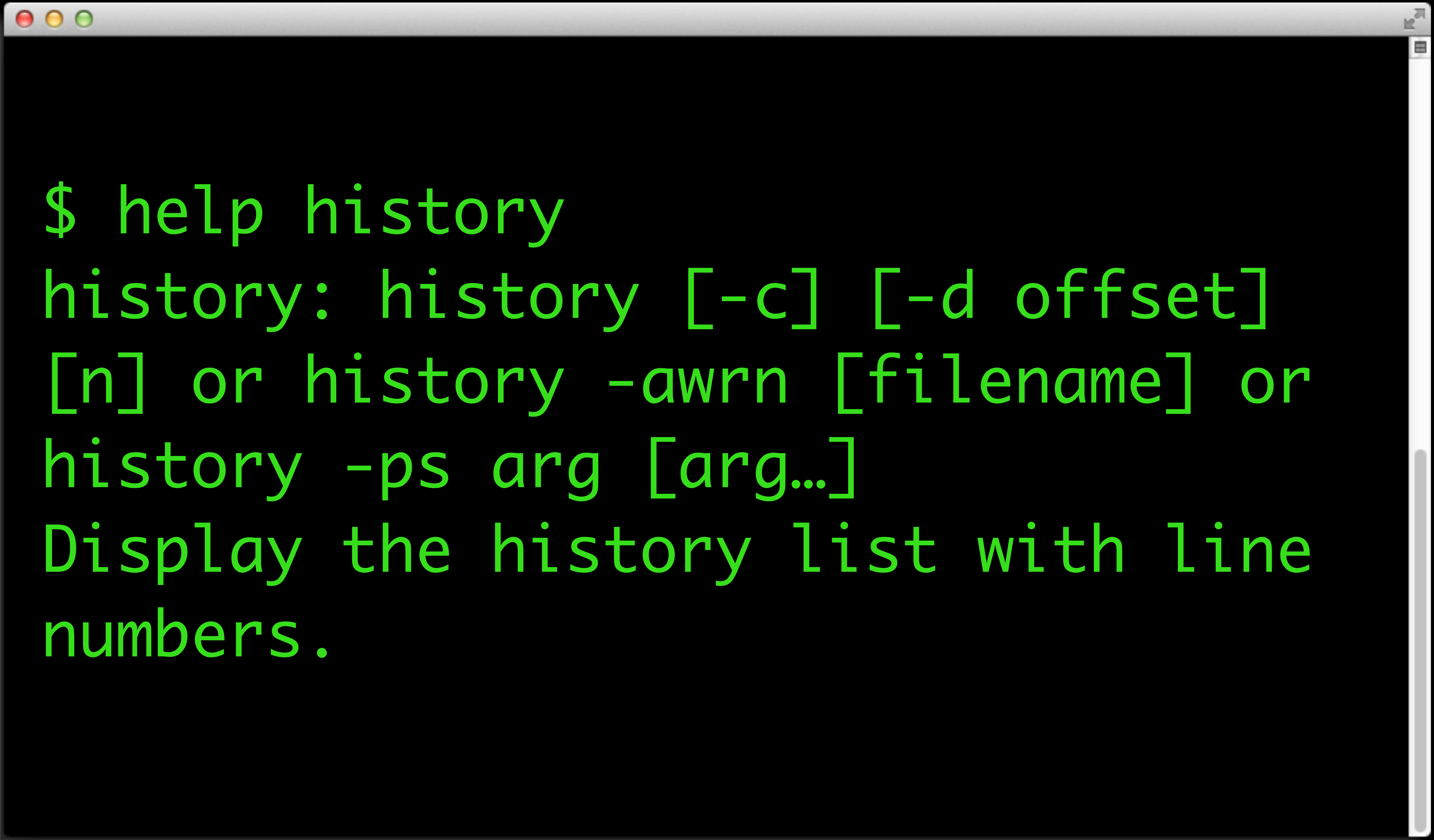
```
$ help
  …
  Type `help' to see this list.
  Type `help name' to find out more about the
  function `name'.
  Use `info bash' to find out more about the
  shell in general.
  Use `man -k' or `info' to find out more about
  commands not in this list.
  …
```



```
$ help cd
cd: cd [-L|-P] [dir]
    Change the current directory to
DIR.
```



```
$ cd ~/Desktop/ShellScriptingPSUMAC2014
$ cat inventory.sh
```



```
$ help history
history: history [-c] [-d offset]
[n] or history -awrn [filename] or
history -ps arg [arg…]
Display the history list with line
numbers.
```

## history

- history = Show previous commands
- !! = Run previous command
- !n = Run previous command #n

## Terminal Tricks

- Up/Down Arrows
  - Cycle previous commands
- TAB = Autocomplete!

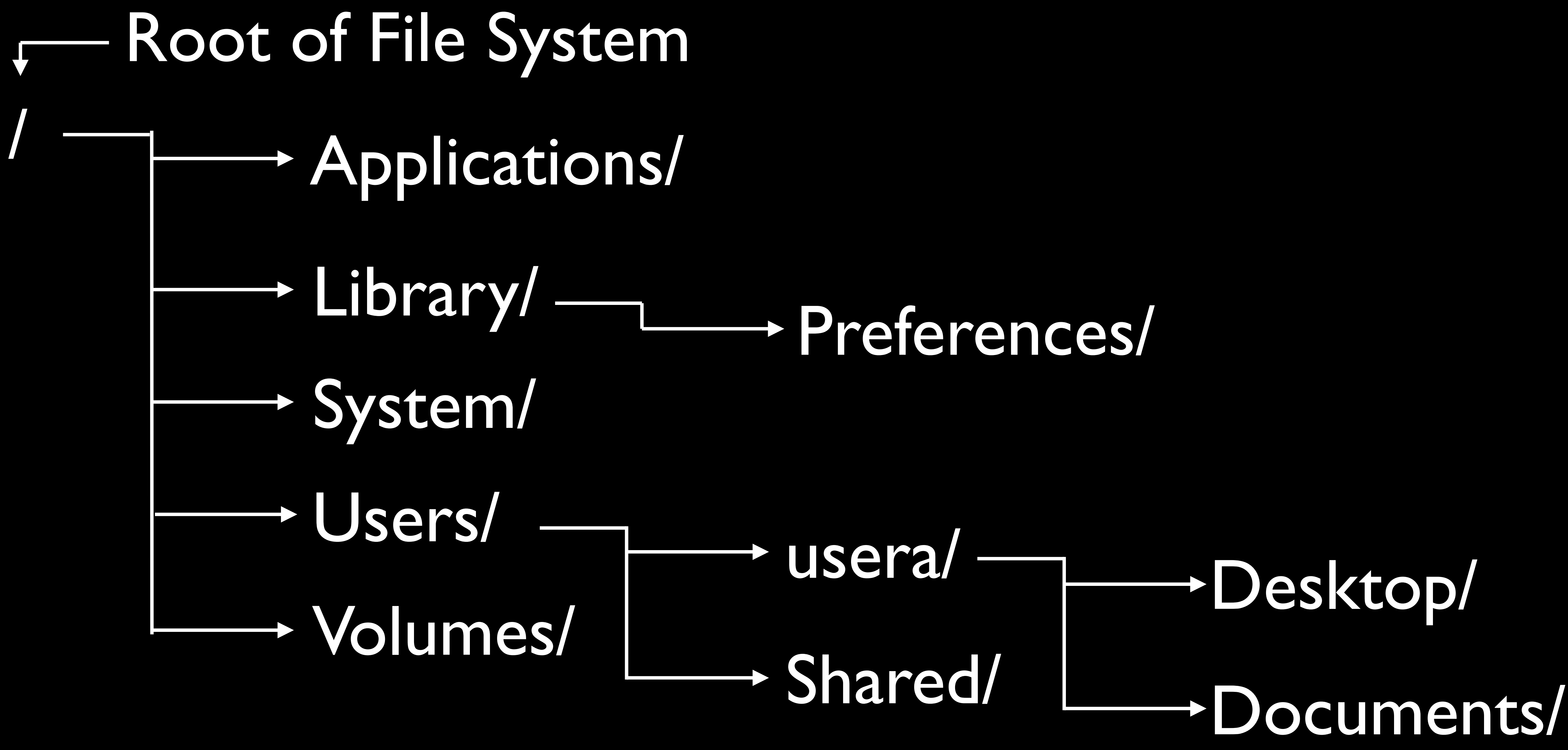
# Try It!

- Open Terminal.app
- man, pwd, cat, sleep, date, help, cd, history, !!, clear
- Up arrow through History
- Move to End of Line
- Clear Screen

## Paths

- Relative
  - From current location to file
- Absolute
  - From file system root to file

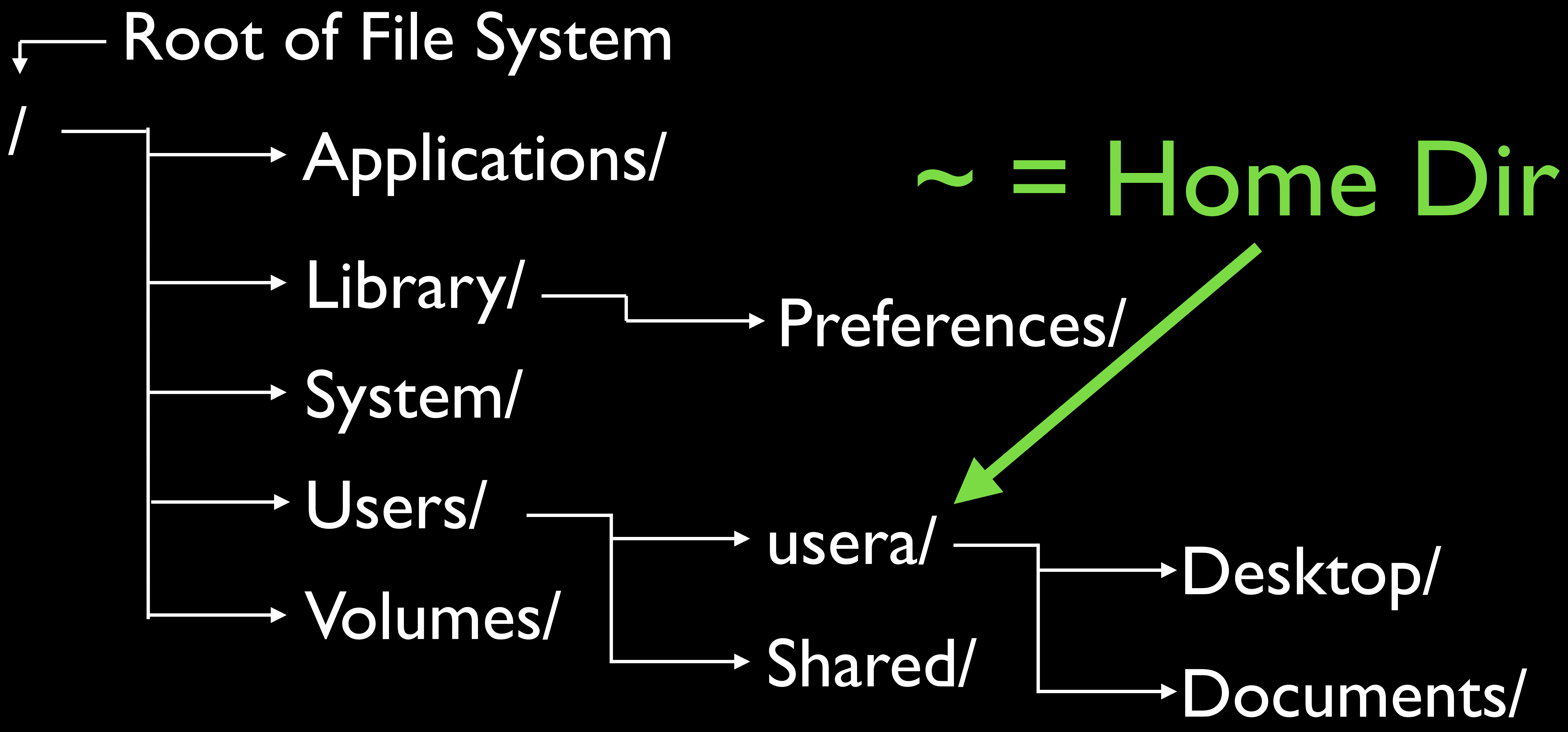
## OS X

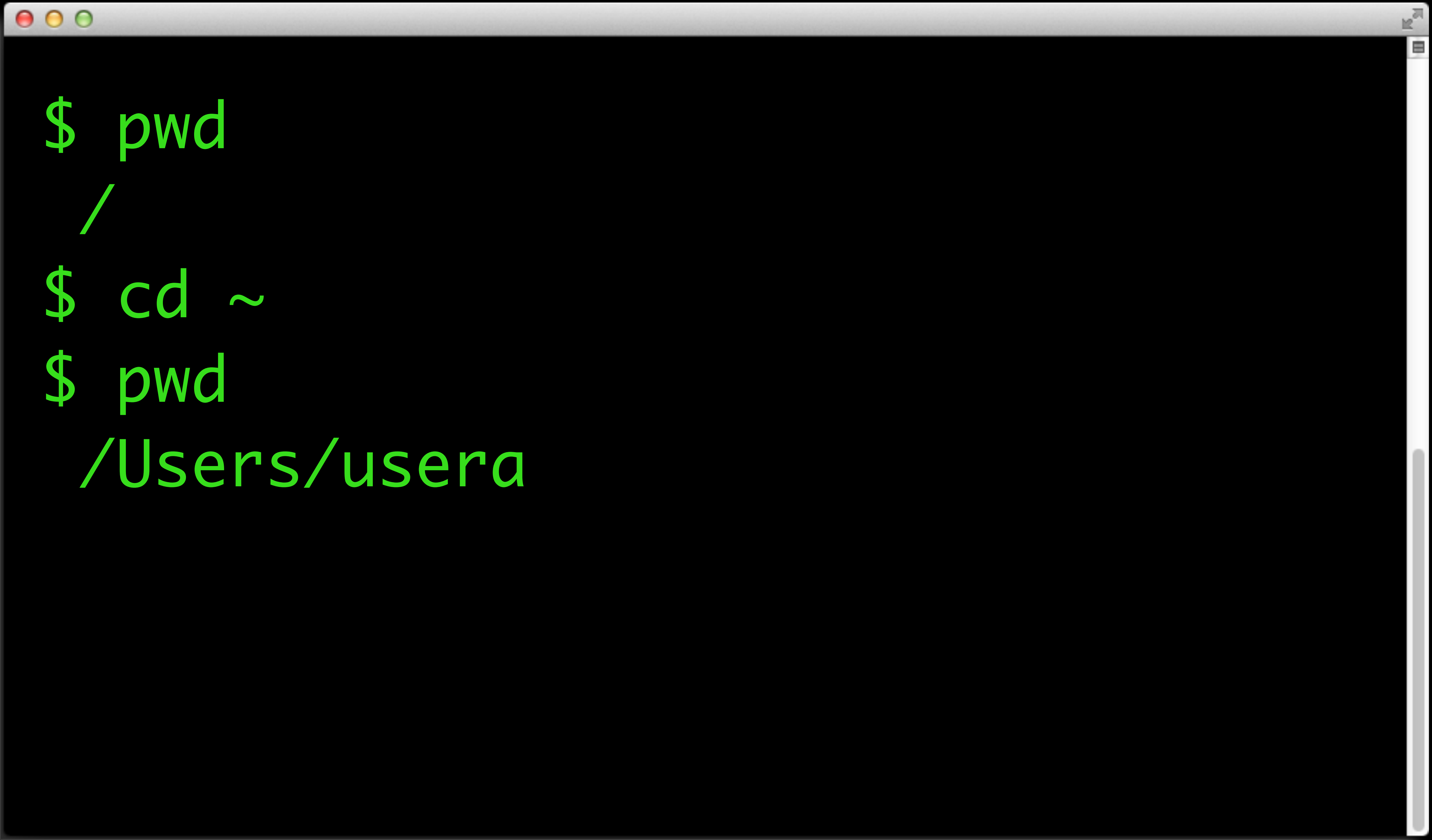

Root of File System
/
Applications/
Library/
Preferences/
System/
Users/
usera/
Desktop/
Volumes/
Shared/
Documents/

## Path Shortcuts

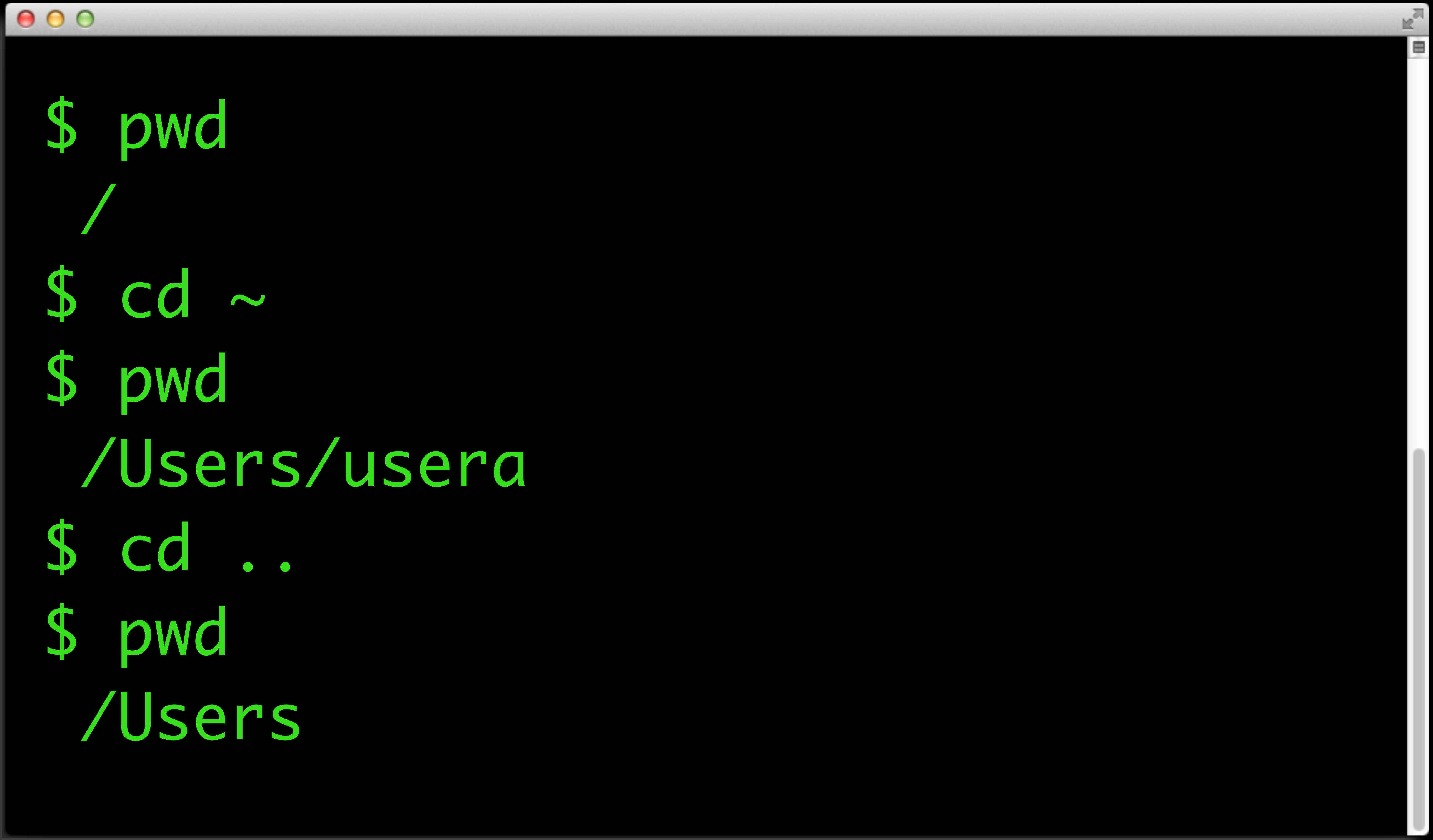
- ~ = User's Home
- . = Current Directory
- .. = One Directory Up

# Log in as usera

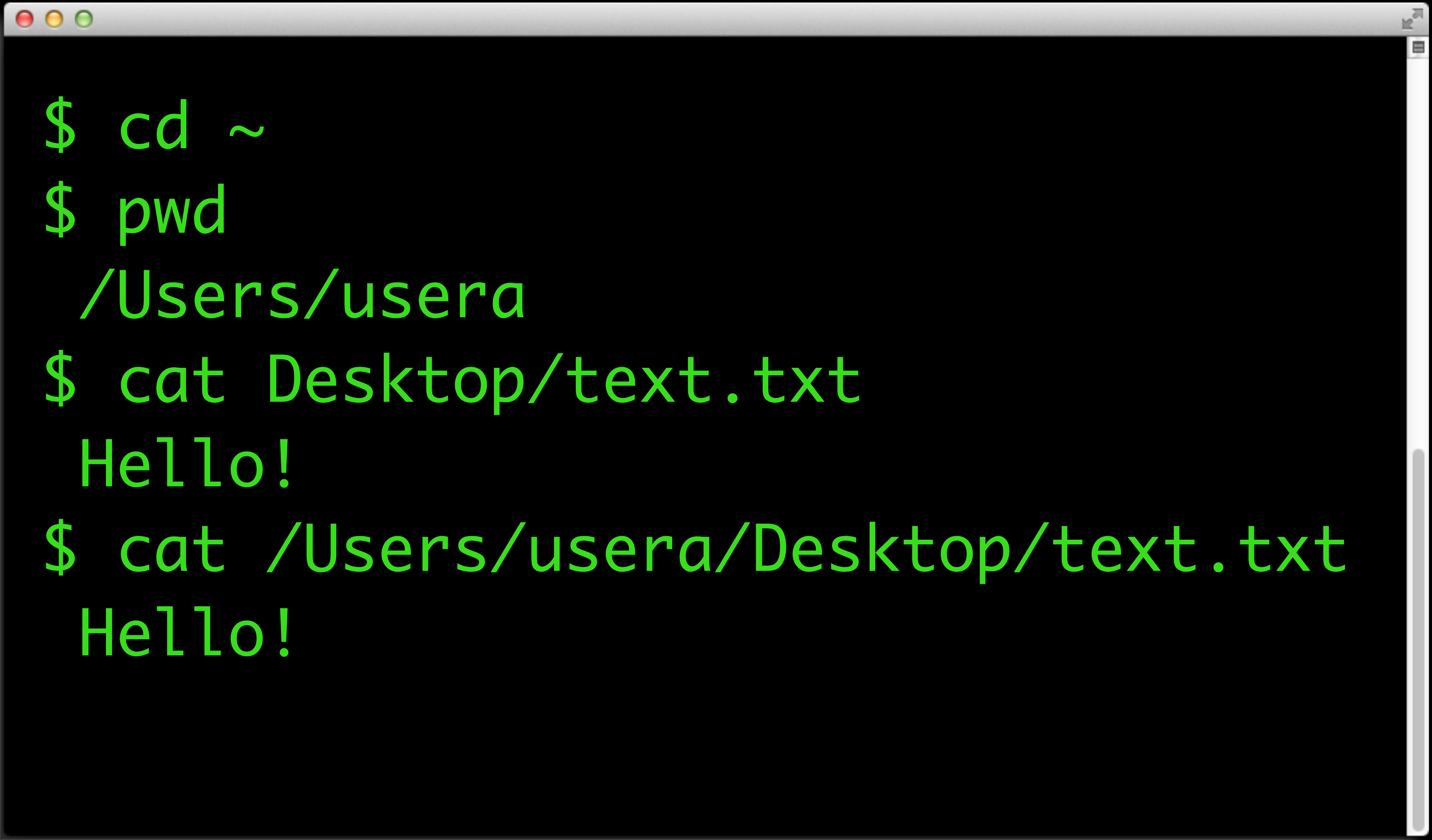




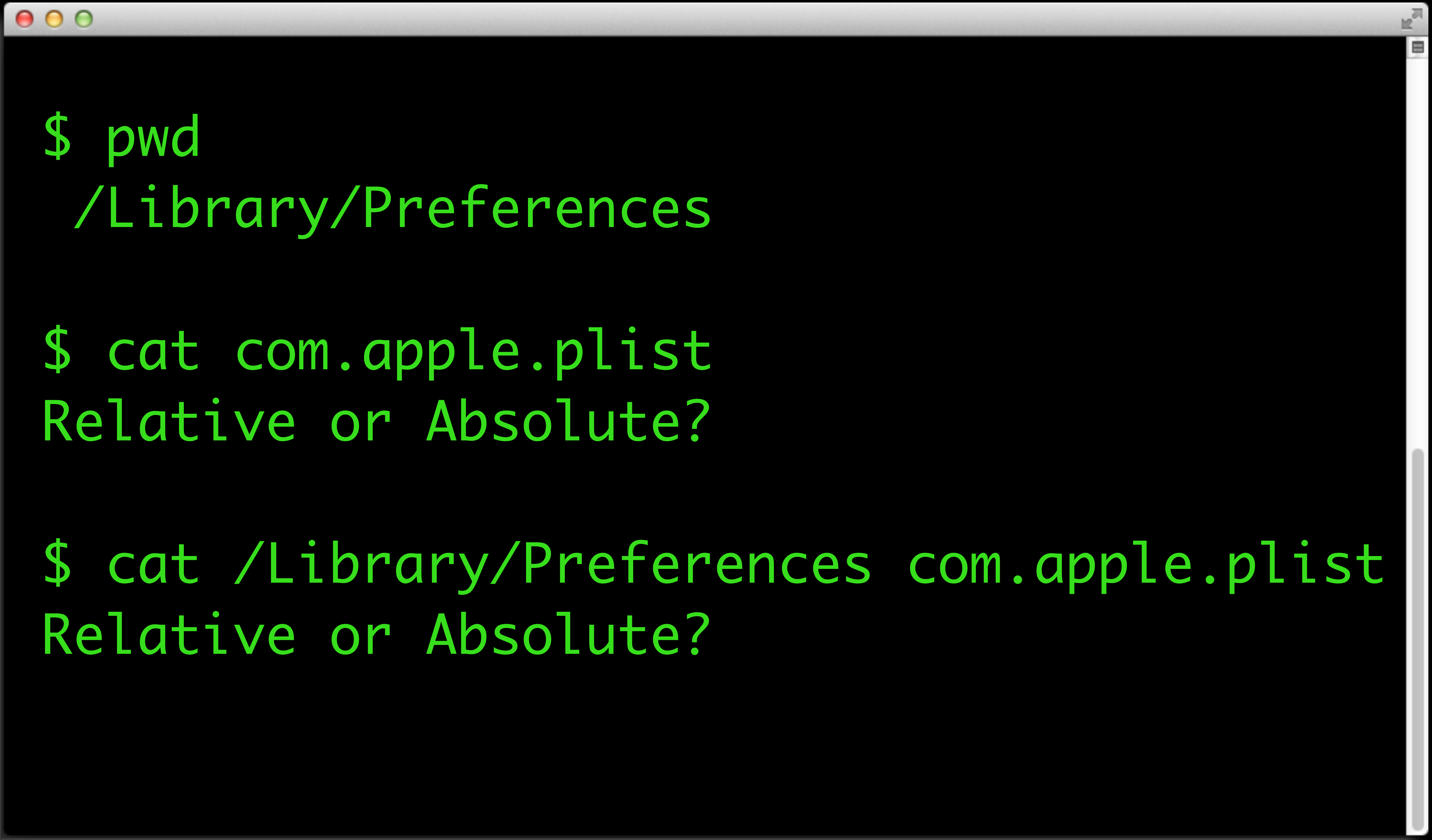
```
$ pwd
 /
$ cd ~
$ pwd
 /Users/usera
```



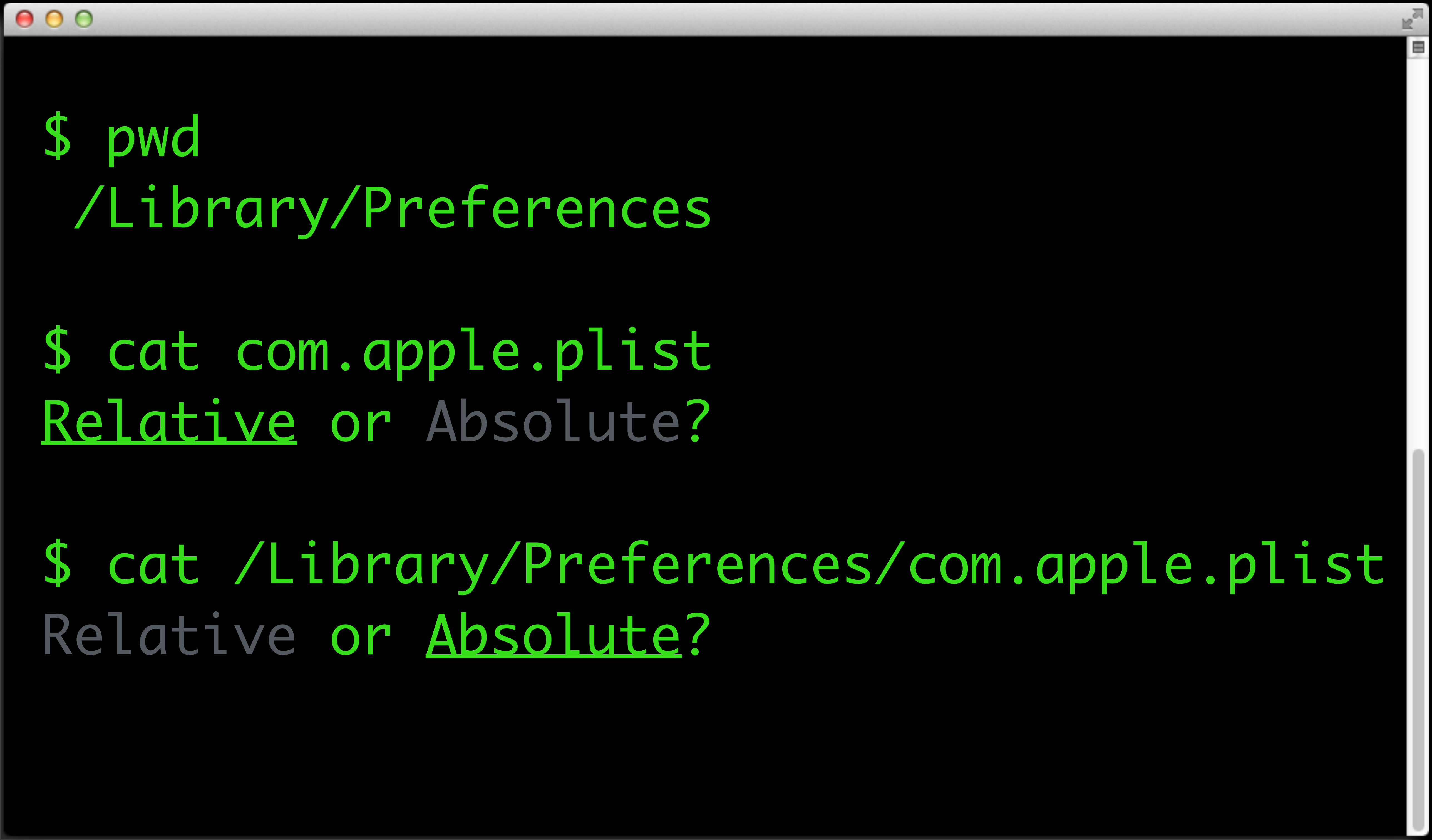
```
$ pwd
 /
$ cd ~
$ pwd
 /Users/usera
$ cd ..
$ pwd
 /Users
```



```
$ cd ~
$ pwd
 /Users/usera
$ cat Desktop/text.txt
 Hello!
$ cat /Users/usera/Desktop/text.txt
 Hello!
```



```
$ pwd
 /Library/Preferences

$ cat com.apple.plist
Relative or Absolute?

$ cat /Library/Preferences com.apple.plist
Relative or Absolute?
```

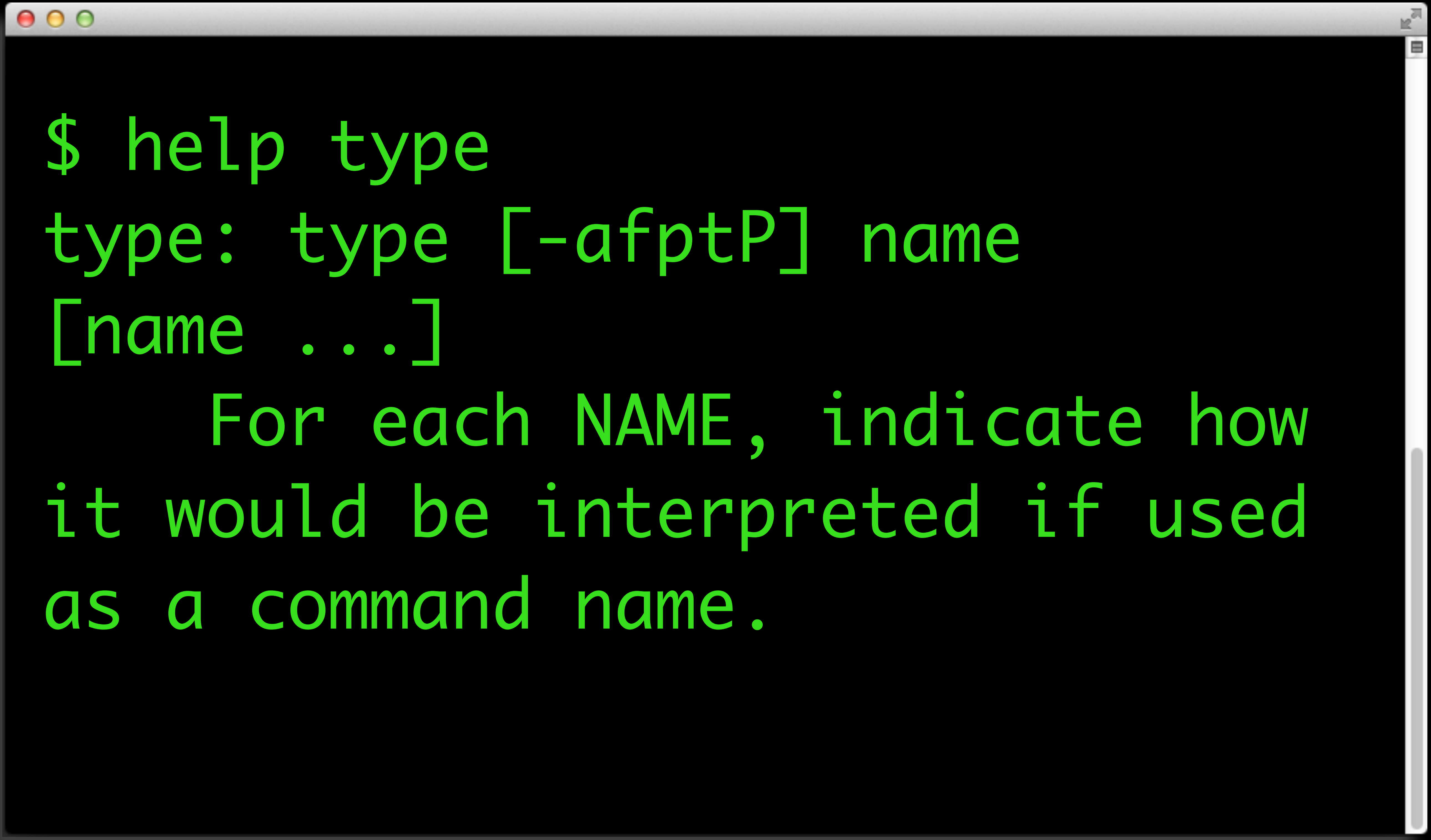


```
$ pwd
 /Library/Preferences

$ cat com.apple.plist
Relative or Absolute?

$ cat /Library/Preferences/com.apple.plist
Relative or Absolute?
```

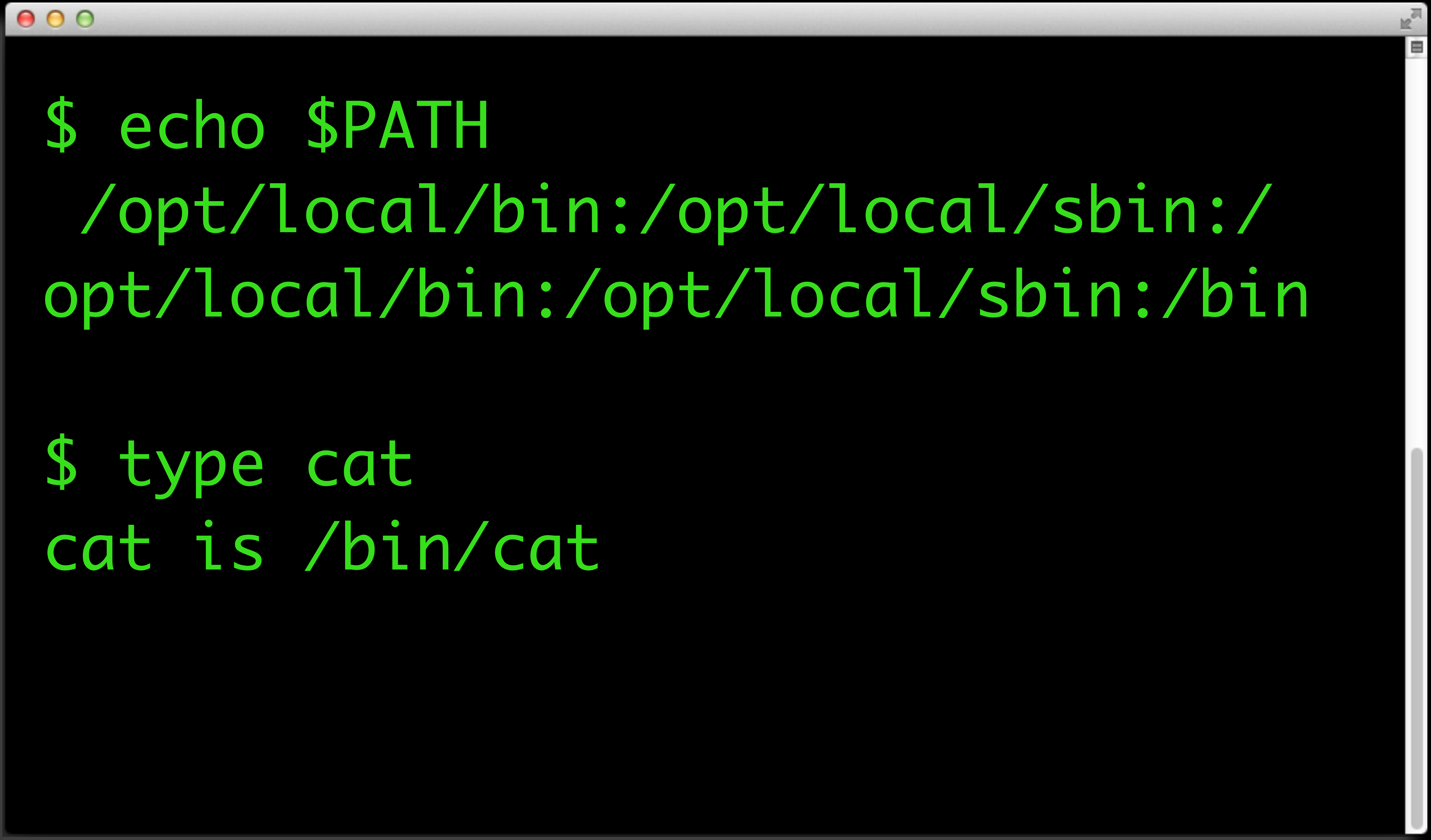
## $PATH

- Global Shell Variable
- Paths Searched when Executing Commands
- Separated by ":"
- Show path to programs:
  - `type program_name`

- Global Shell Variable
- Paths Searched when Executing Commands
- Separated by “:"
  - use “type” command to locate



```
$ help type
type: type [-afptP] name
[name ...]
    For each NAME, indicate how
it would be interpreted if used
as a command name.
```



```
$ echo $PATH
 /opt/local/bin:/opt/local/sbin:/
opt/local/bin:/opt/local/sbin:/bin

$ type cat
cat is /bin/cat
```

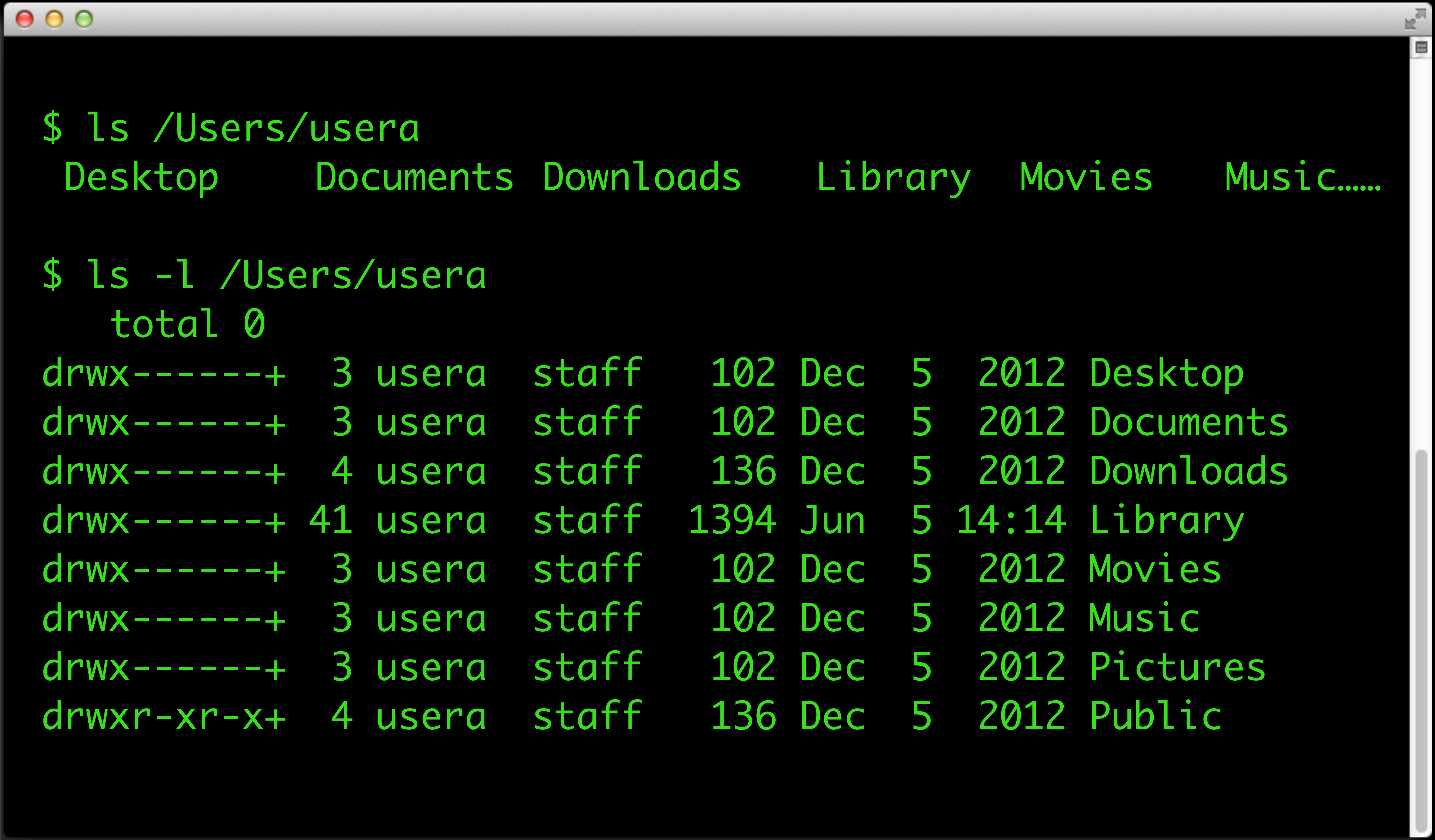
## ls

- List Directory Contents
- `ls -l` = Long Listing
- `ls -a` = Show Hidden Files
- `ls -R` = List Recursively

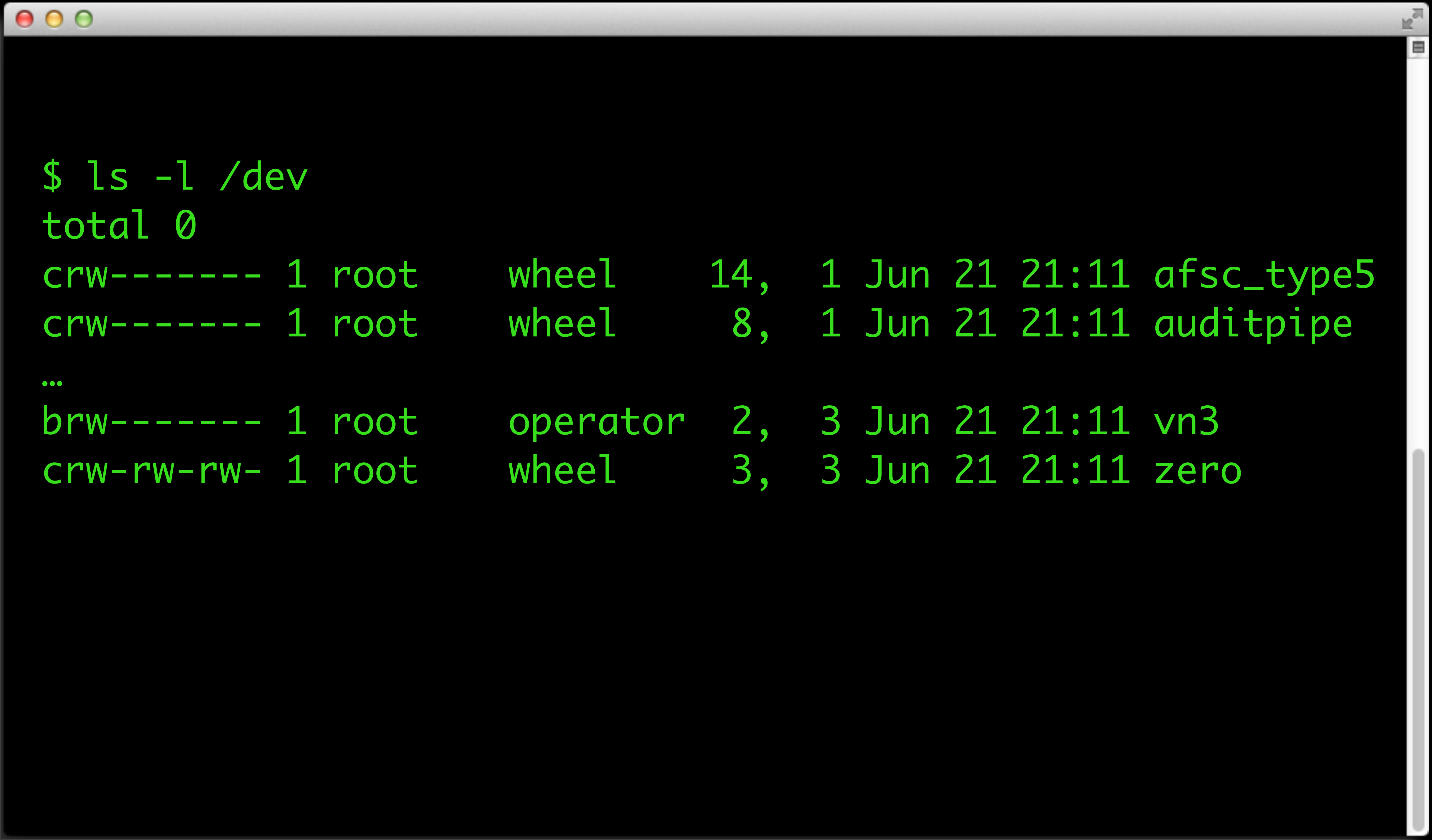
# ls -l

- Long Listing shows file type

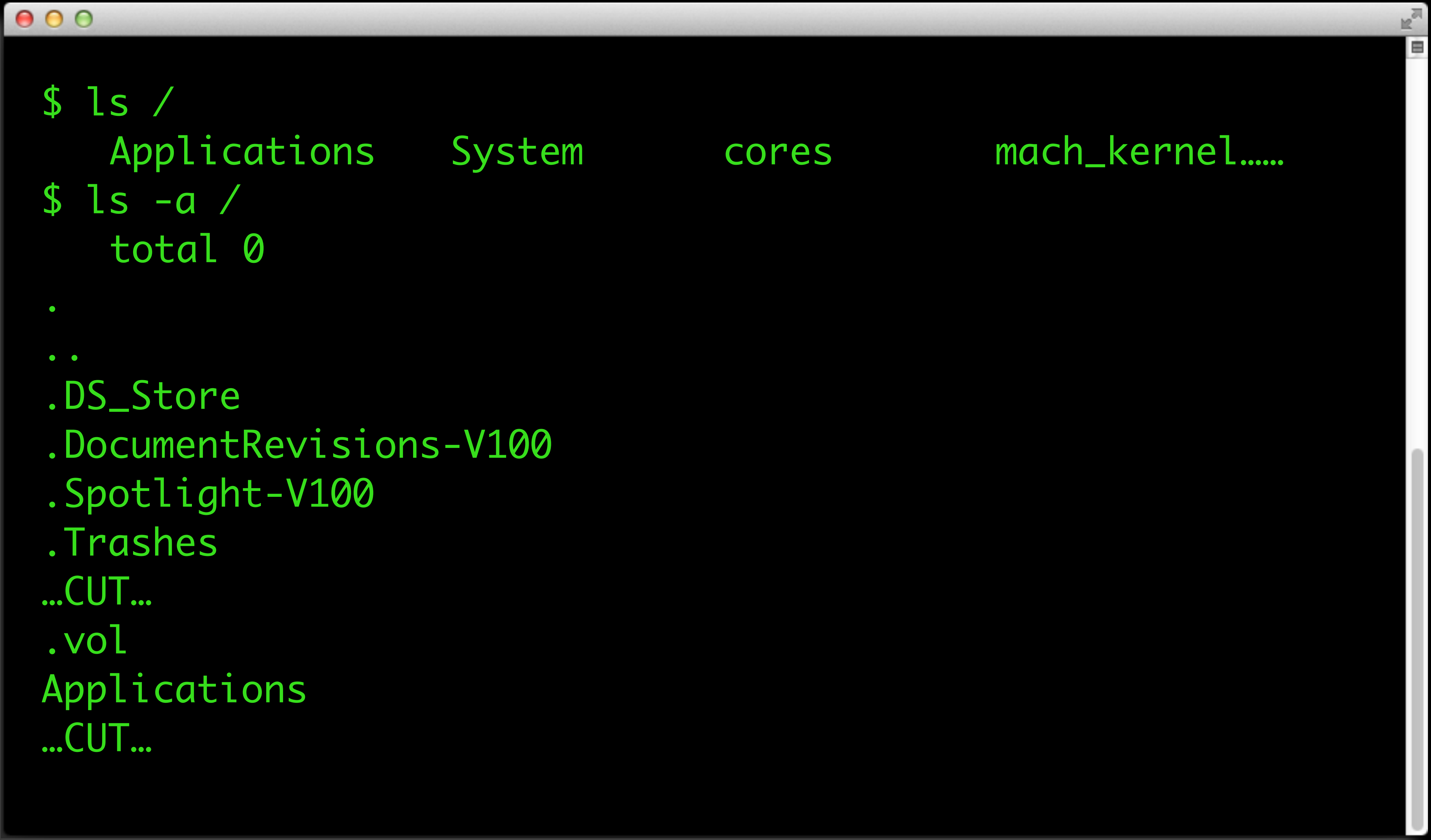
- directories = d
- file = -
- soft (symbolic) link = l
- hard link = -
- block device = b
- character device = c



```
$ ls /Users/usera
 Desktop     Documents Downloads   Library  Movies    Music……

$ ls -l /Users/usera
   total 0
drwx------+  3 usera  staff   102 Dec  5  2012 Desktop
drwx------+  3 usera  staff   102 Dec  5  2012 Documents
drwx------+  4 usera  staff   136 Dec  5  2012 Downloads
drwx------+ 41 usera  staff  1394 Jun  5 14:14 Library
drwx------+  3 usera  staff   102 Dec  5  2012 Movies
drwx------+  3 usera  staff   102 Dec  5  2012 Music
drwx------+  3 usera  staff   102 Dec  5  2012 Pictures
drwxr-xr-x+  4 usera  staff   136 Dec  5  2012 Public
```



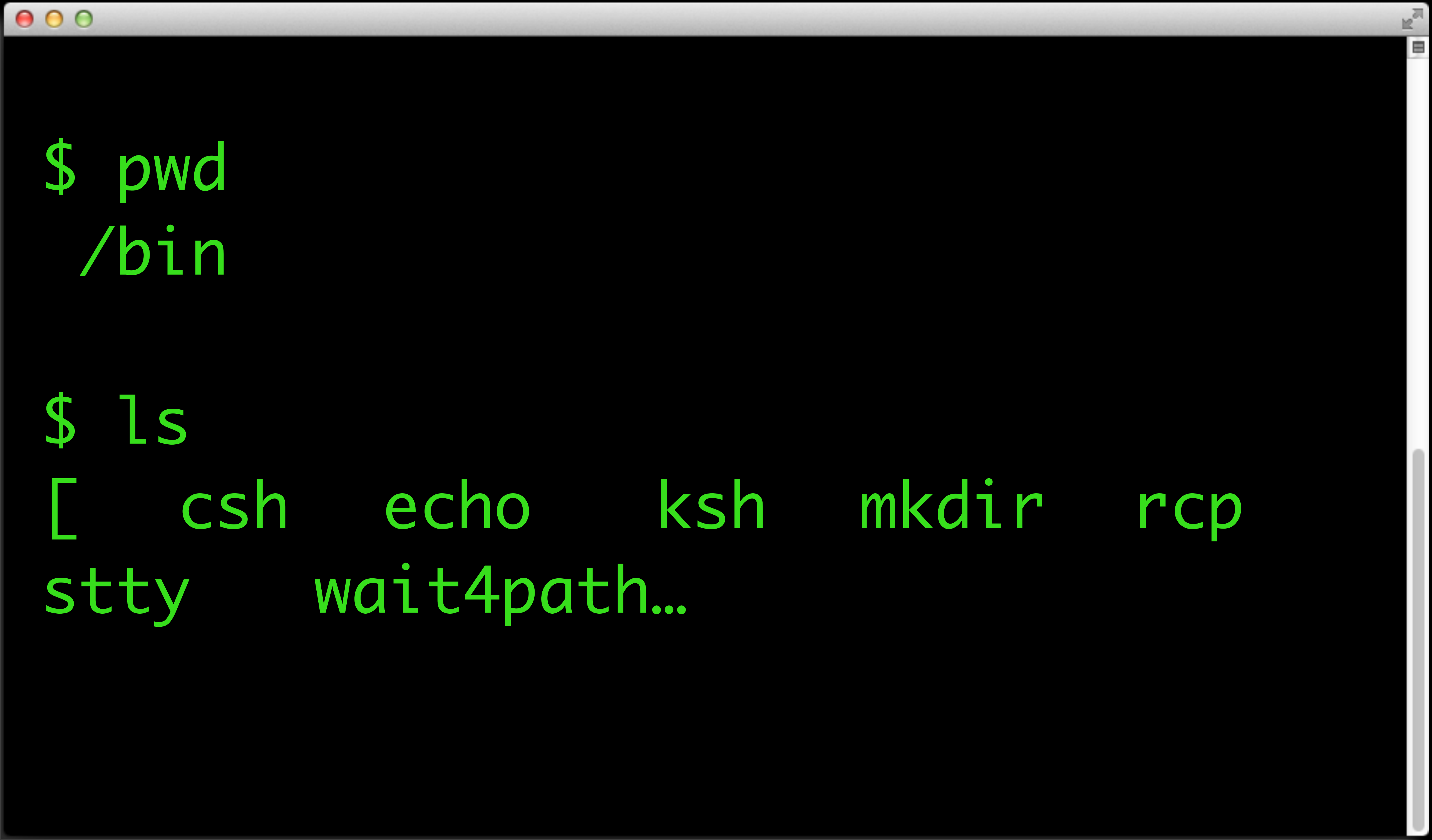
```
$ ls -l /dev
total 0
crw------- 1 root    wheel     14,   1 Jun 21 21:11 afsc_type5
crw------- 1 root    wheel      8,   1 Jun 21 21:11 auditpipe
…
brw------- 1 root    operator  2,   3 Jun 21 21:11 vn3
crw-rw-rw- 1 root    wheel      3,   3 Jun 21 21:11 zero
```



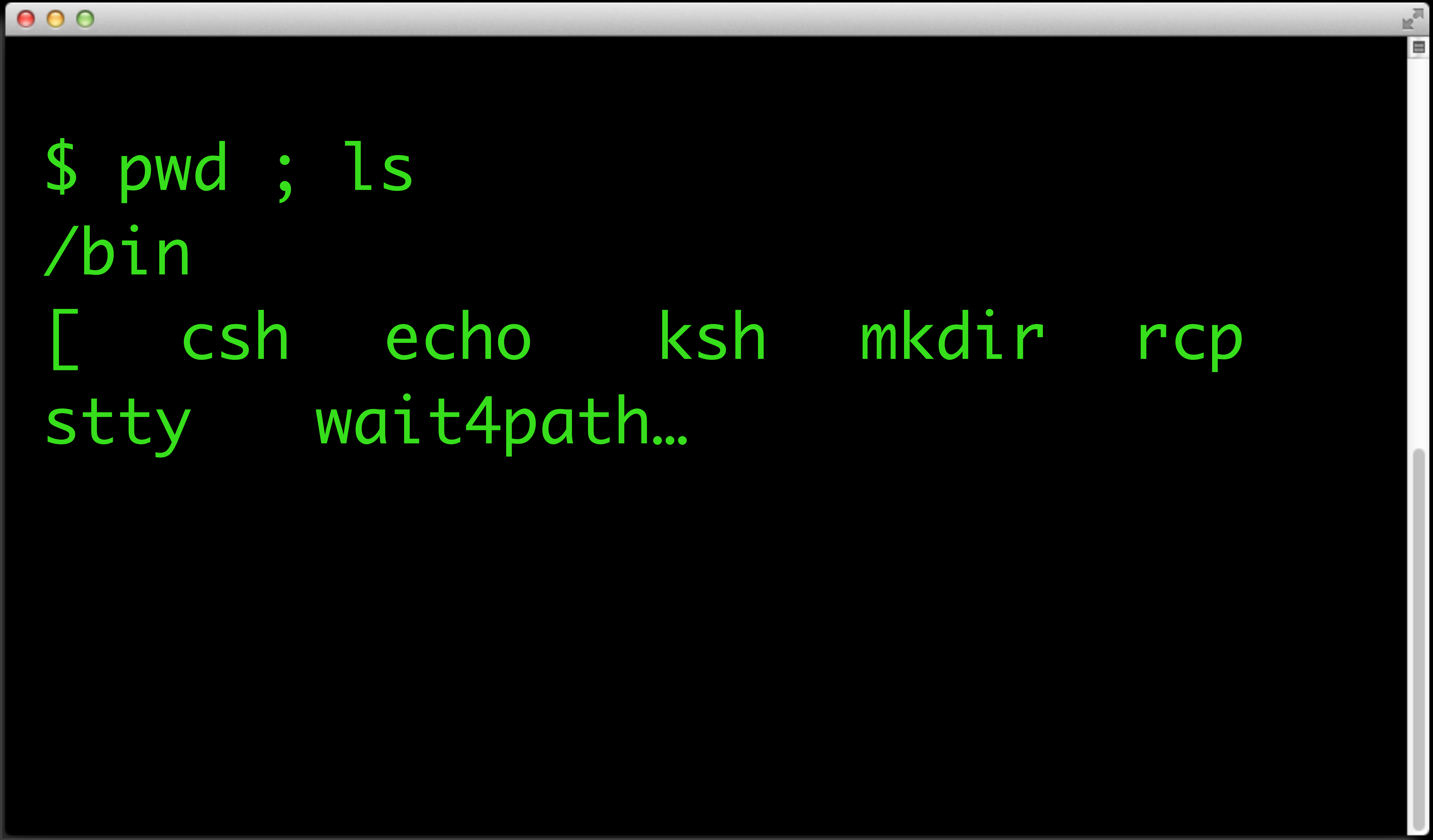
```
$ ls /
   Applications   System      cores       mach_kernel……
$ ls -a /
   total 0
.
..
.DS_Store
.DocumentRevisions-V100
.Spotlight-V100
.Trashes
…CUT…
.vol
Applications
…CUT…
```

## Linear Execution

- Commands run one at a time
- Separate commands with ;
- Run left to right



```
$ pwd
 /bin

$ ls
[   csh   echo    ksh   mkdir   rcp
stty    wait4path...
```



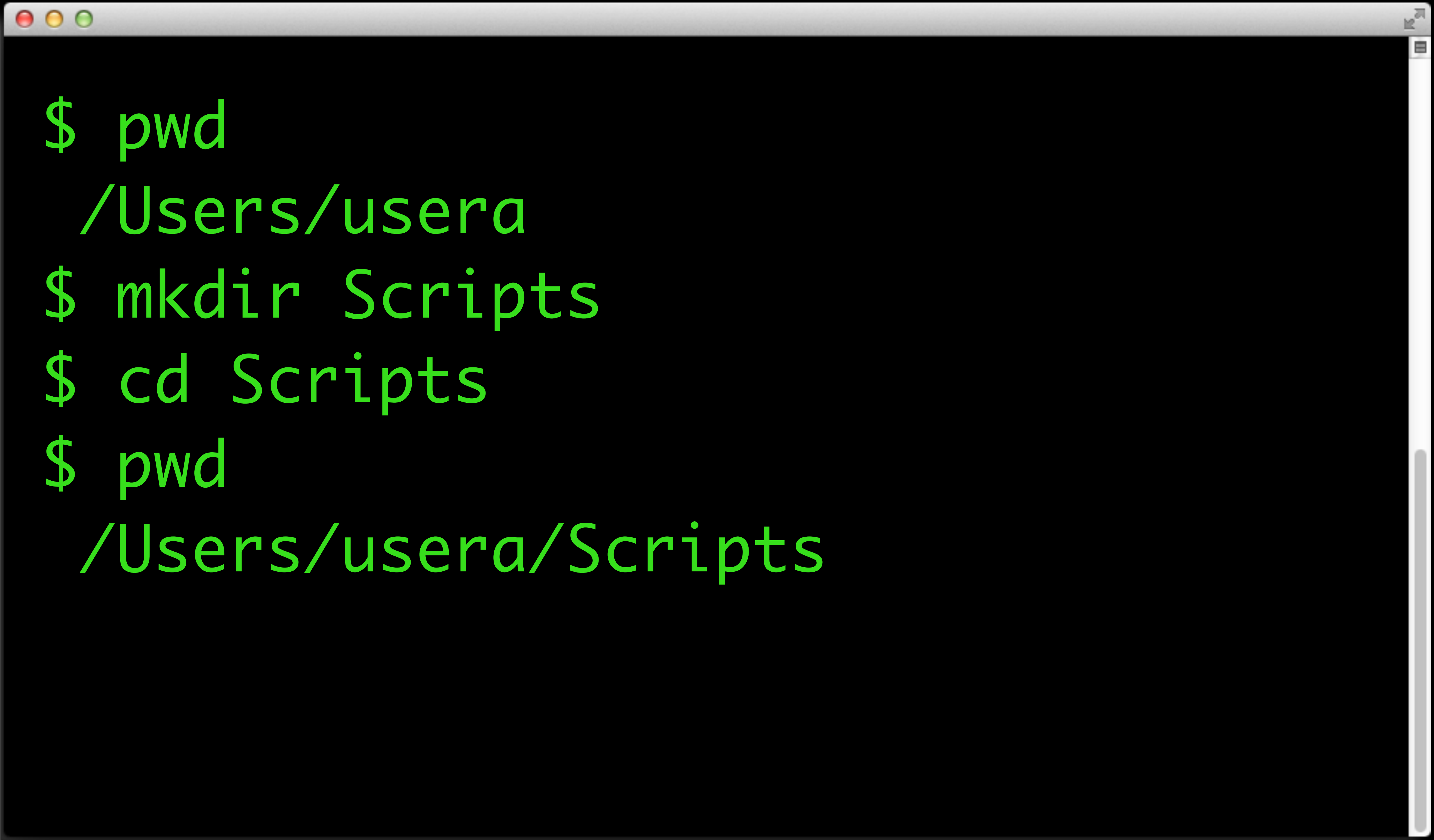
```
$ pwd ; ls
/bin
[    csh    echo    ksh    mkdir    rcp
stty    wait4path…
```

## Try It!

- Open terminal.app
- Try Basic commands (cd, ls, cat, pwd)
    - find a directory
    - list the files
    - cat a file
- Do them all in a row with one <enter>!

## Making Folders

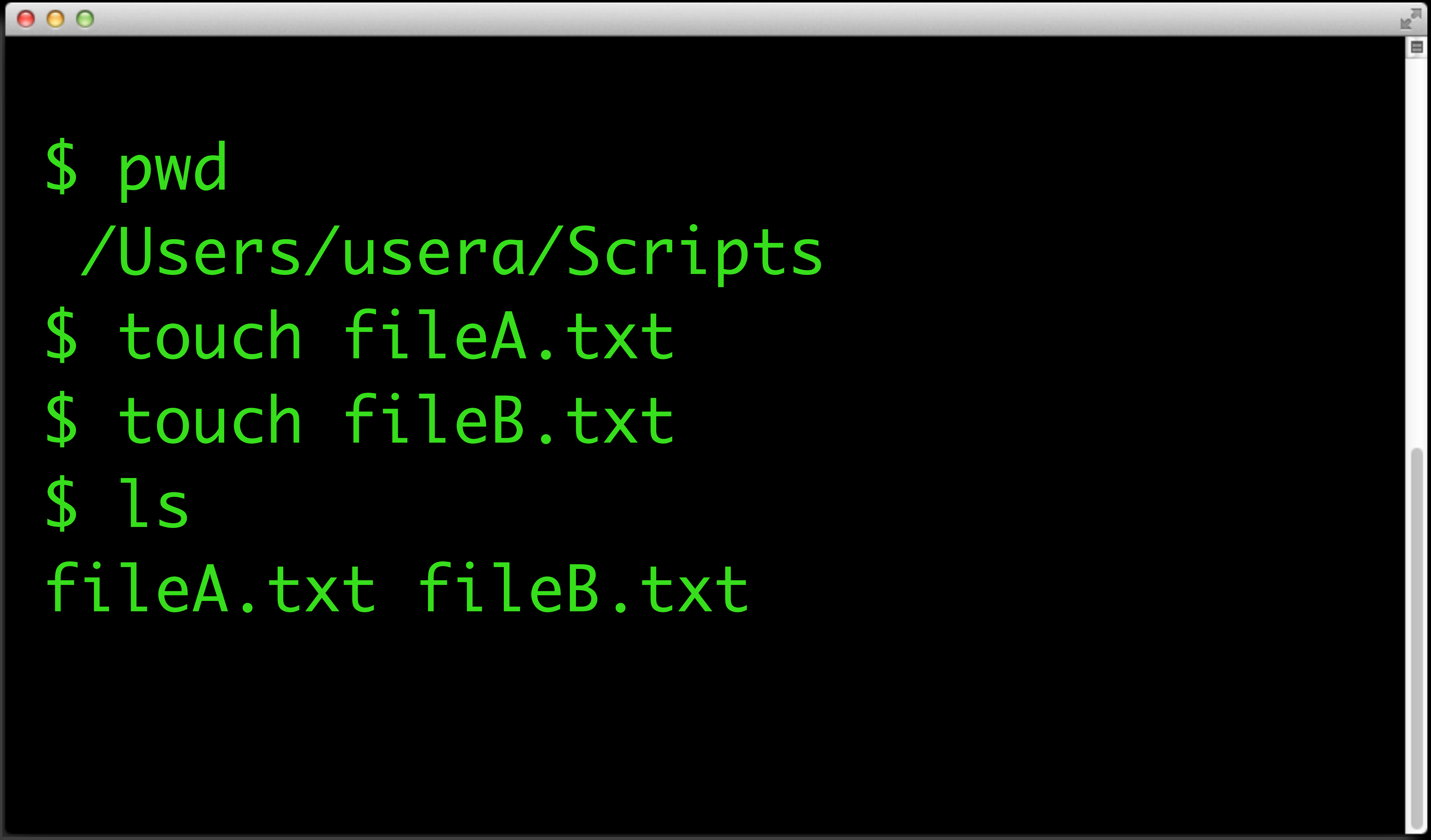
- `mkdir foldername`
- `mkdir -p /path/to/newfolder`



```
$ pwd
 /Users/usera
$ mkdir Scripts
$ cd Scripts
$ pwd
 /Users/usera/Scripts
```

## Making Files

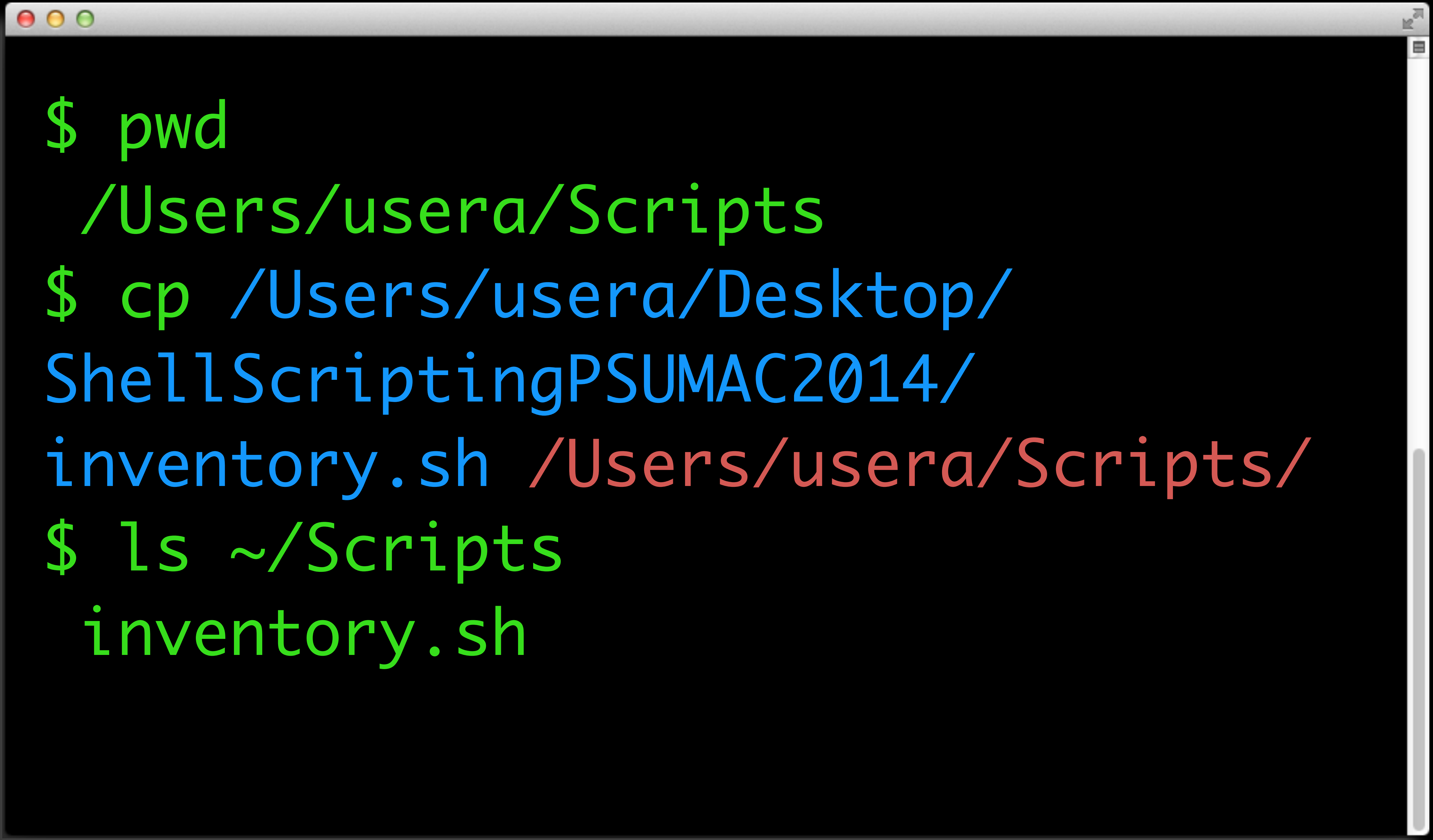
- Create blank files
- Update modification dates
- `touch filename`



```
$ pwd
 /Users/usera/Scripts
$ touch fileA.txt
$ touch fileB.txt
$ ls
fileA.txt fileB.txt
```

# Making Copies

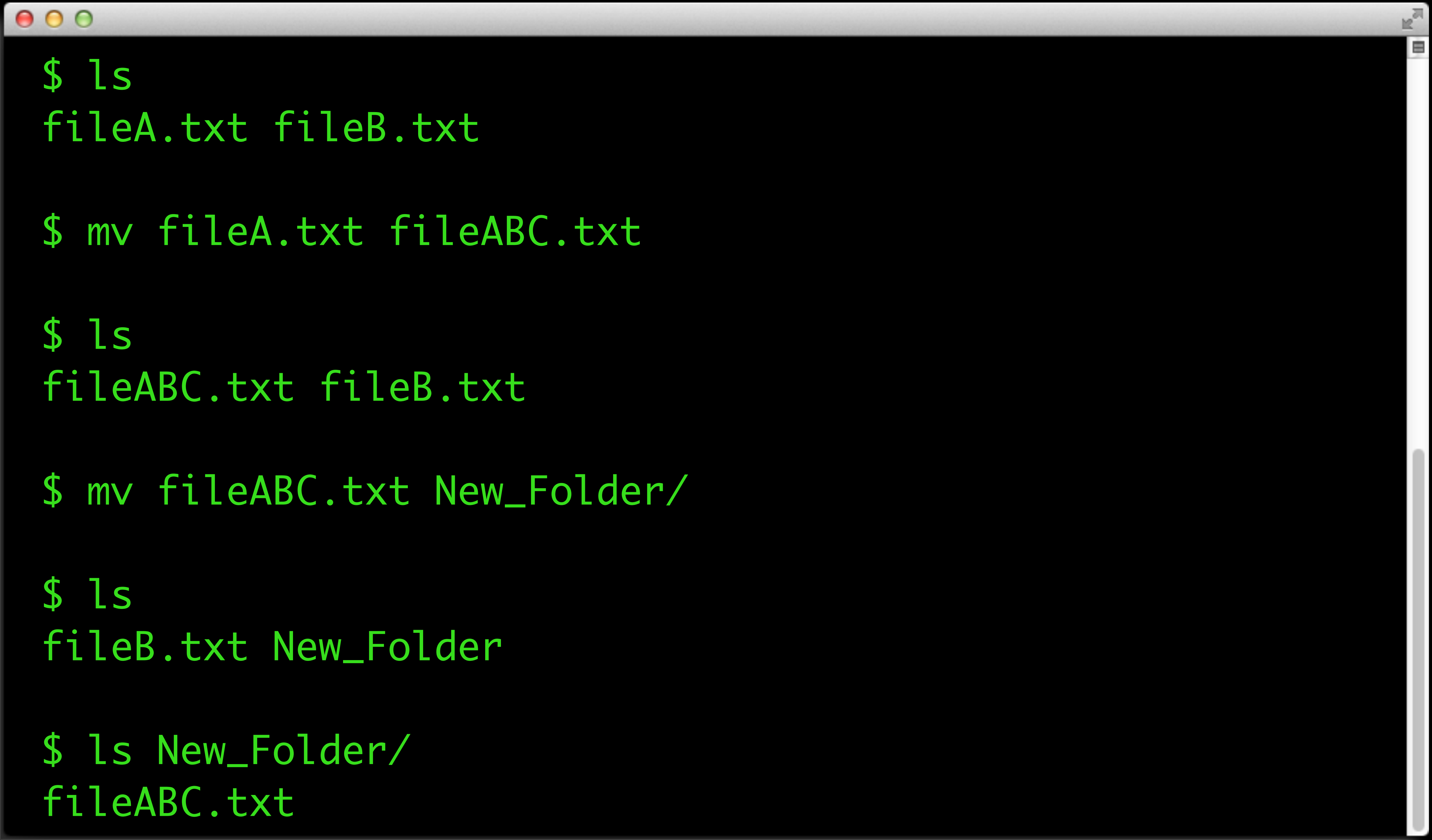
- Copy File / Folders
- `cp original newfile`
- `cp -R` = Copy Recursively



```
$ pwd
 /Users/usera/Scripts
$ cp /Users/usera/Desktop/
ShellScriptingPSUMAC2014/
inventory.sh /Users/usera/Scripts/
$ ls ~/Scripts
 inventory.sh
```

## Moving (renaming) Files

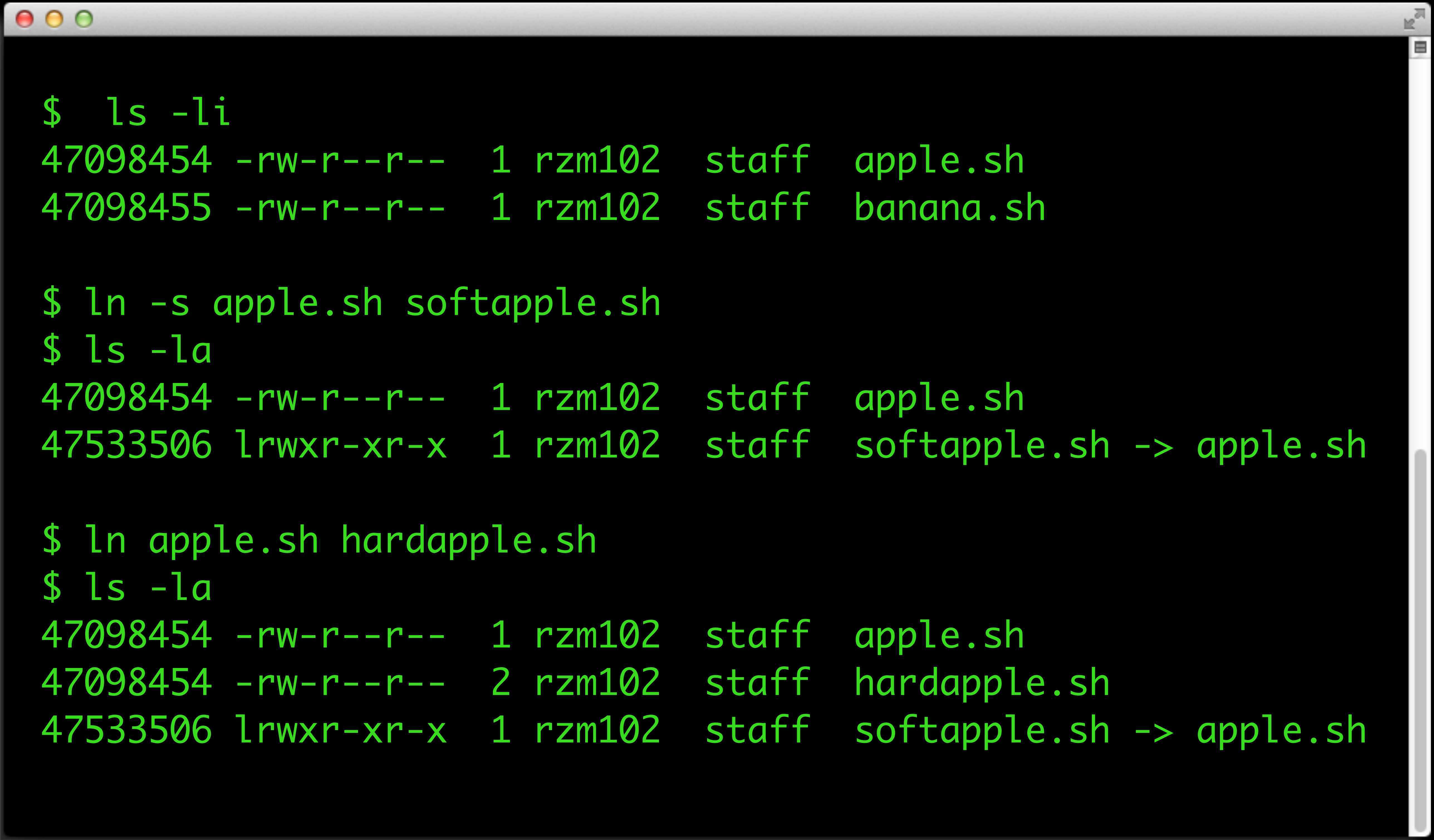
- Move File / Folders
- `mv original newfile`
- `mv original /new/path/`



```
$ ls
fileA.txt fileB.txt

$ mv fileA.txt fileABC.txt

$ ls
fileABC.txt fileB.txt

$ mv fileABC.txt New_Folder/

$ ls
fileB.txt New_Folder

$ ls New_Folder/
fileABC.txt
```

## Making Links

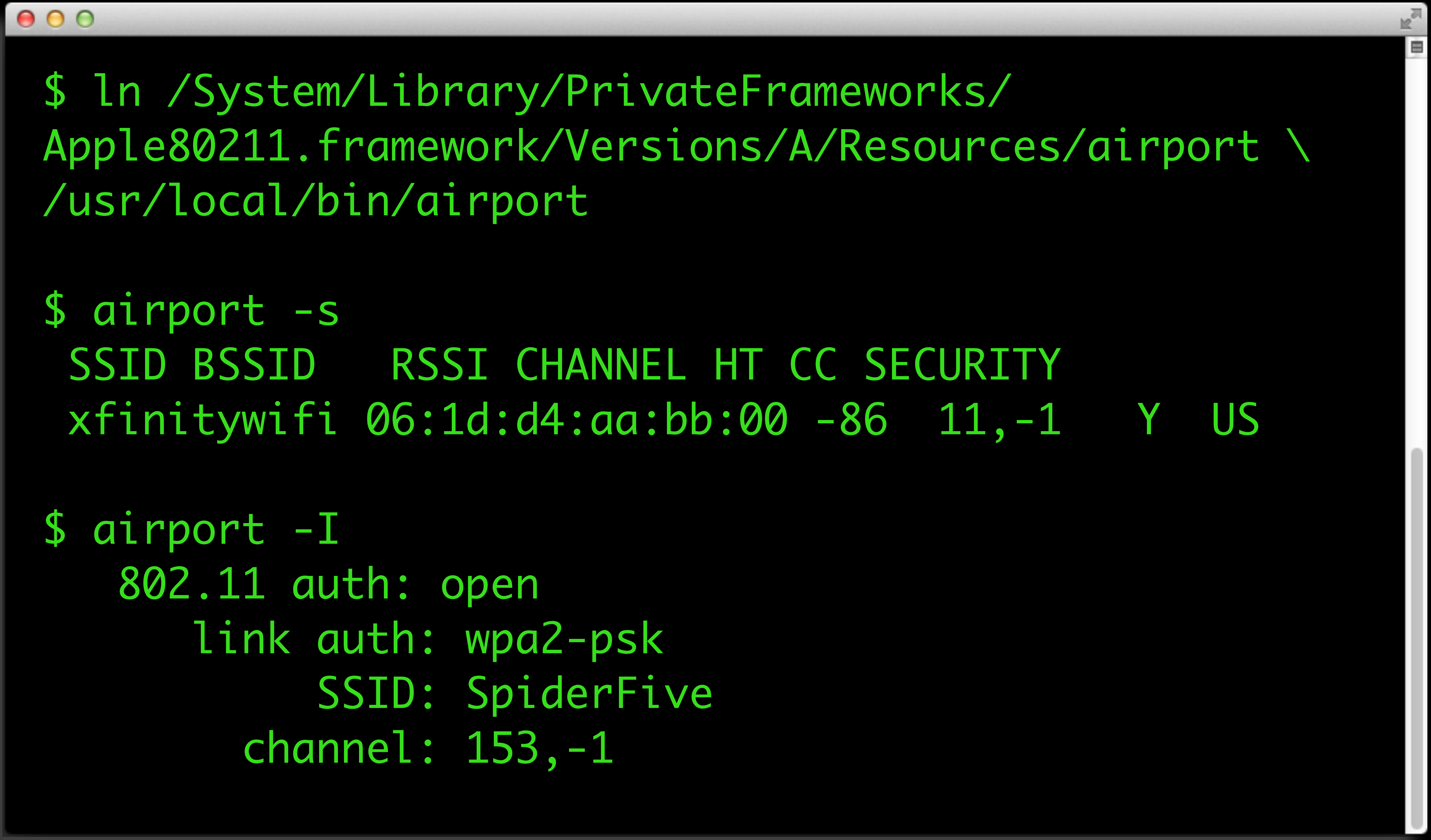
- Hard Link:
  Can’t Span FileSystems, Direct Pointer to inode
  - `ln original hardlink`
- Sym Links:
  Can span volumes, Points to Original
  - `ln -s original symlink`



```
$  ls -li
47098454 -rw-r--r--  1 rzm102  staff  apple.sh
47098455 -rw-r--r--  1 rzm102  staff  banana.sh

$ ln -s apple.sh softapple.sh
$ ls -la
47098454 -rw-r--r--  1 rzm102  staff  apple.sh
47533506 lrwxr-xr-x  1 rzm102  staff  softapple.sh -> apple.sh

$ ln apple.sh hardapple.sh
$ ls -la
47098454 -rw-r--r--  1 rzm102  staff  apple.sh
47098454 -rw-r--r--  2 rzm102  staff  hardapple.sh
47533506 lrwxr-xr-x  1 rzm102  staff  softapple.sh -> apple.sh
```

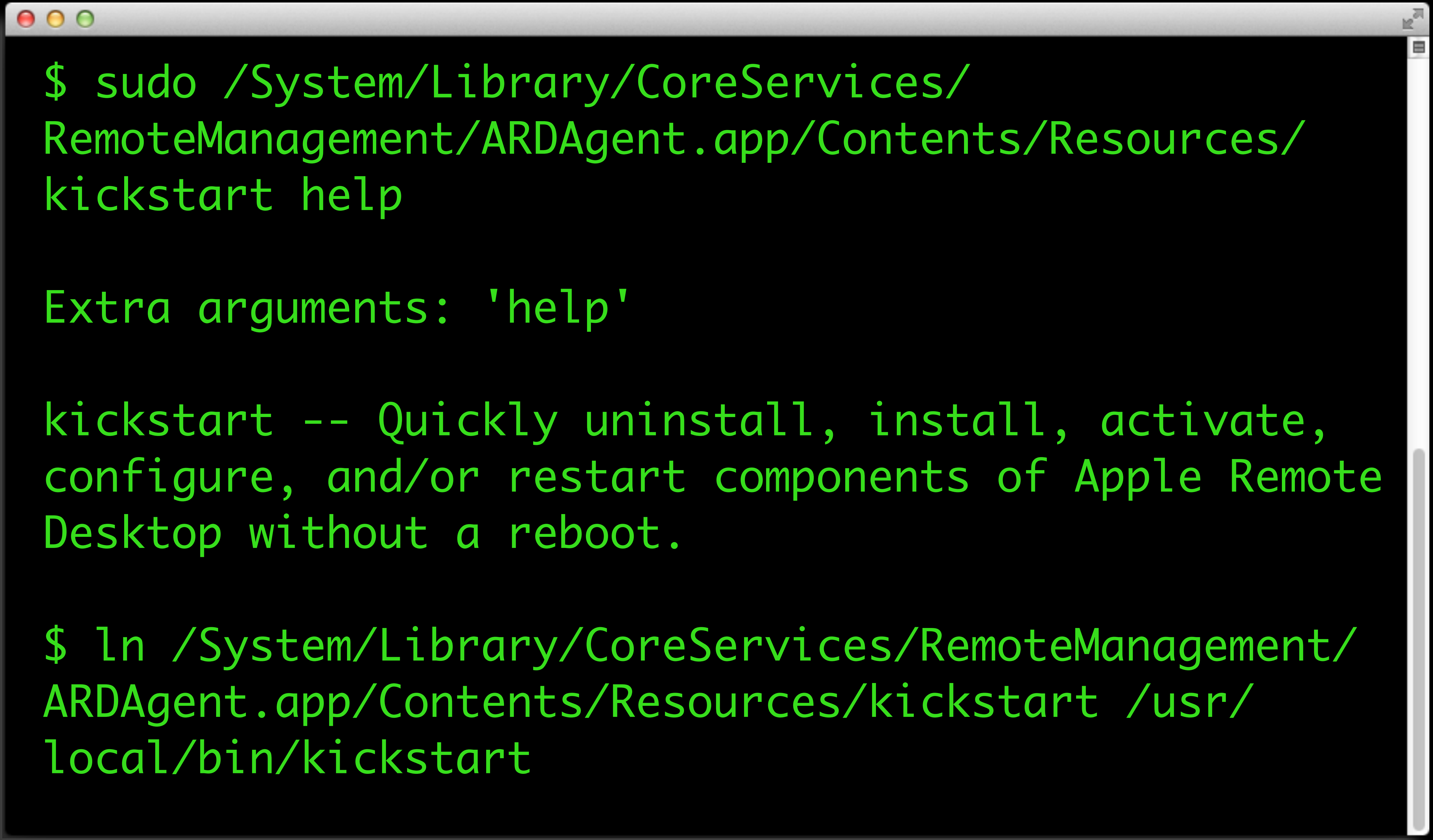
# airport

- /System/Library/PrivateFrameworks/Apple80211.framework/Versions/A/Resources/airport
- Display Wireless Information
- Scan for Networks


```
$ ln /System/Library/PrivateFrameworks/
Apple80211.framework/Versions/A/Resources/airport \
/usr/local/bin/airport

$ airport -s
 SSID BSSID   RSSI CHANNEL HT CC SECURITY
 xfinitywifi 06:1d:d4:aa:bb:00 -86  11,-1   Y  US

$ airport -I
   802.11 auth: open
      link auth: wpa2-psk
           SSID: SpiderFive
        channel: 153,-1
```

# kickstart

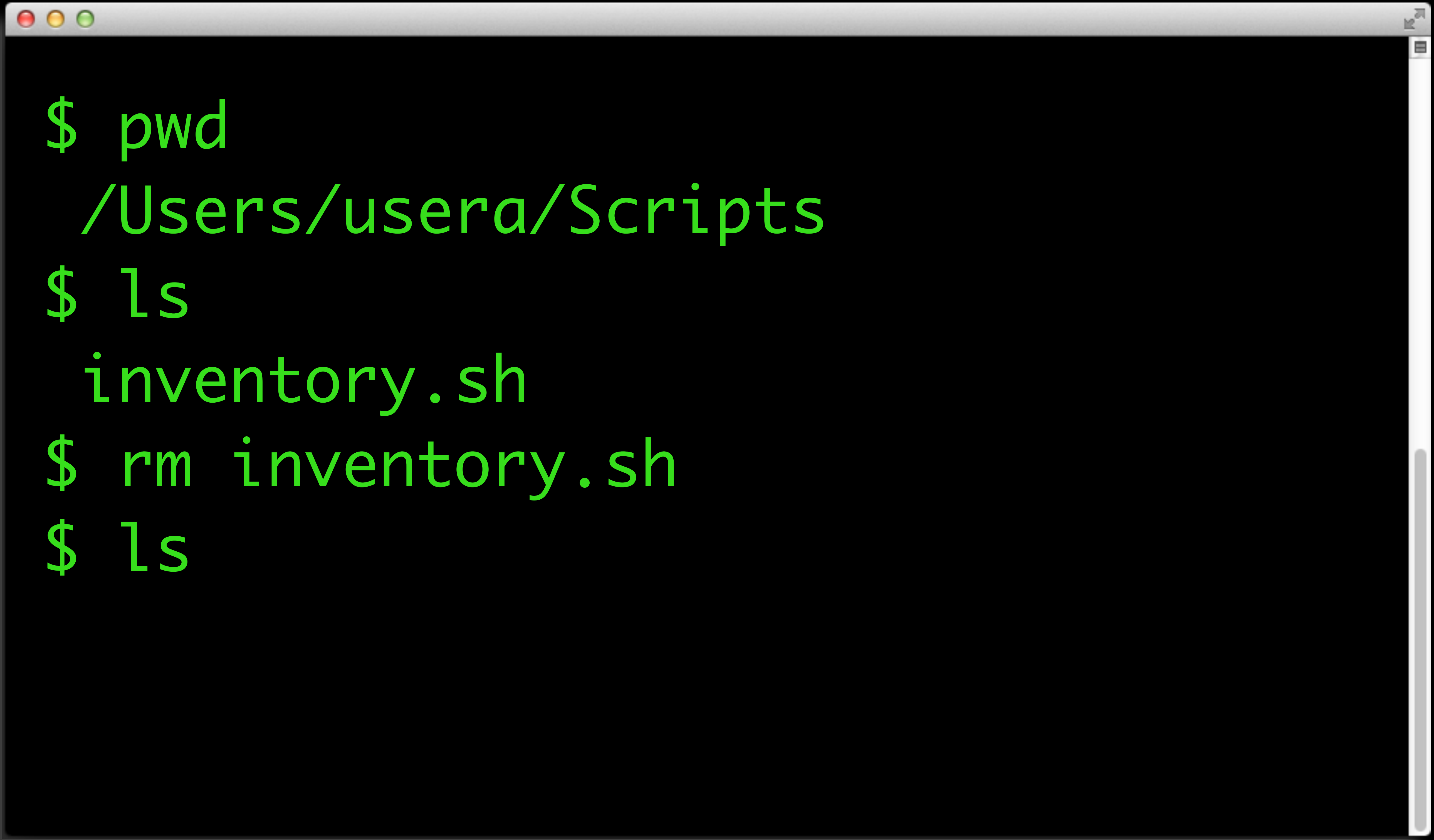
- /System/Library/CoreServices/RemoteManagement/ARDAgent.app/Contents/Resources/kickstart
- Apple Remote Desktop/VNC
- (Un)install/Activate/Configure/Restart



```
$ sudo /System/Library/CoreServices/
RemoteManagement/ARDAgent.app/Contents/Resources/
kickstart help

Extra arguments: 'help'

kickstart -- Quickly uninstall, install, activate,
configure, and/or restart components of Apple Remote
Desktop without a reboot.

$ ln /System/Library/CoreServices/RemoteManagement/
ARDAgent.app/Contents/Resources/kickstart /usr/
local/bin/kickstart
```

## Deleting Files

- `rm` - Remove file
- `rm -R /path` - Recursive Delete



```
$ pwd
 /Users/usera/Scripts
$ ls
 inventory.sh
$ rm inventory.sh
$ ls
```

## Try It!

- Move into Home Directory
- Create new Directory
- Copy file into new Directory
- Move a File
- Create a Link
- Delete a copy

## Shell Script

- Structure
- Execution

## 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

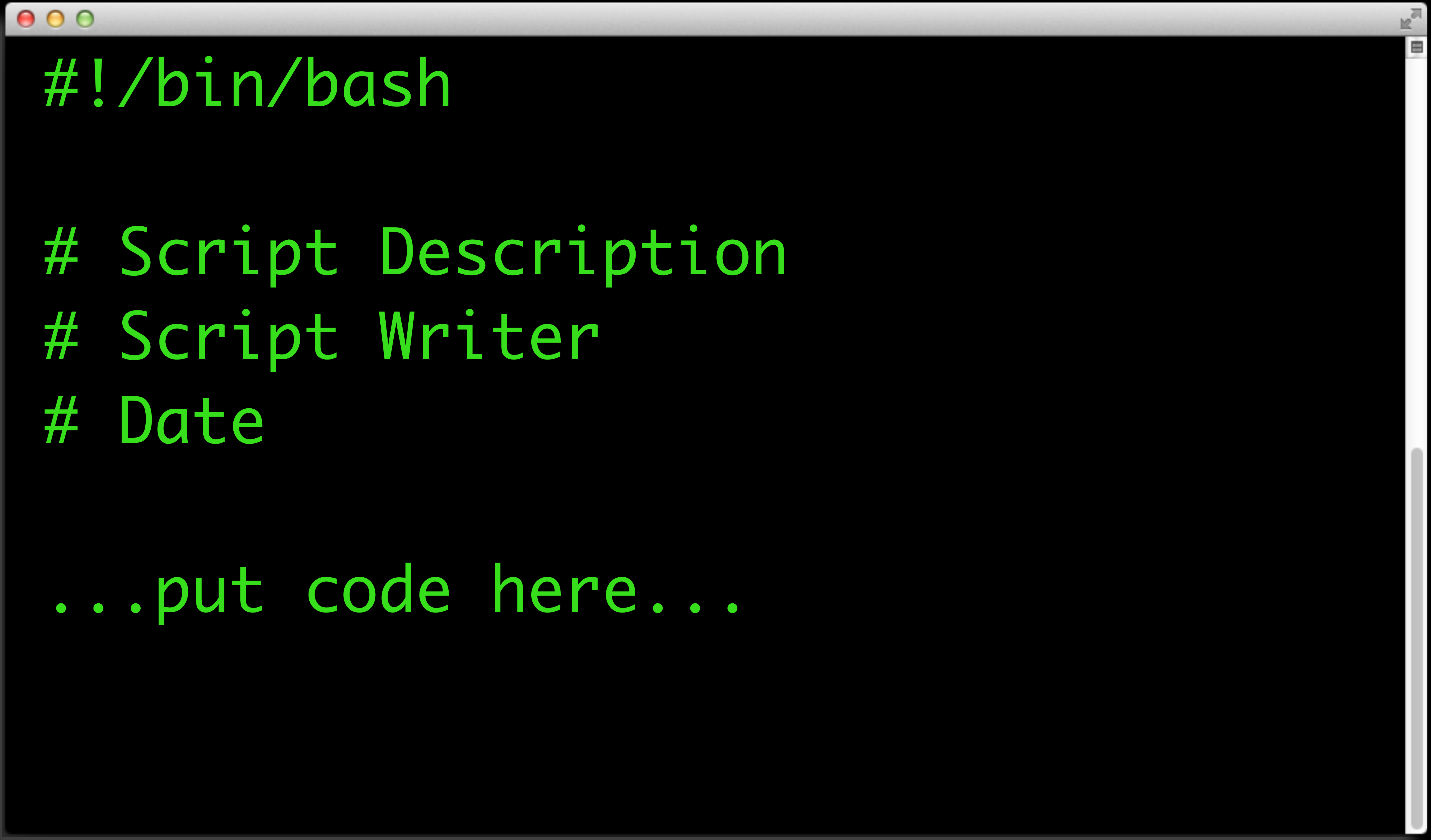
## Script Format

## Script Name

- BASH doesn't care about extensions
- Standard is ending with .sh
- Starting with . hides file
- Avoid spaces/special characters

## First Line

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



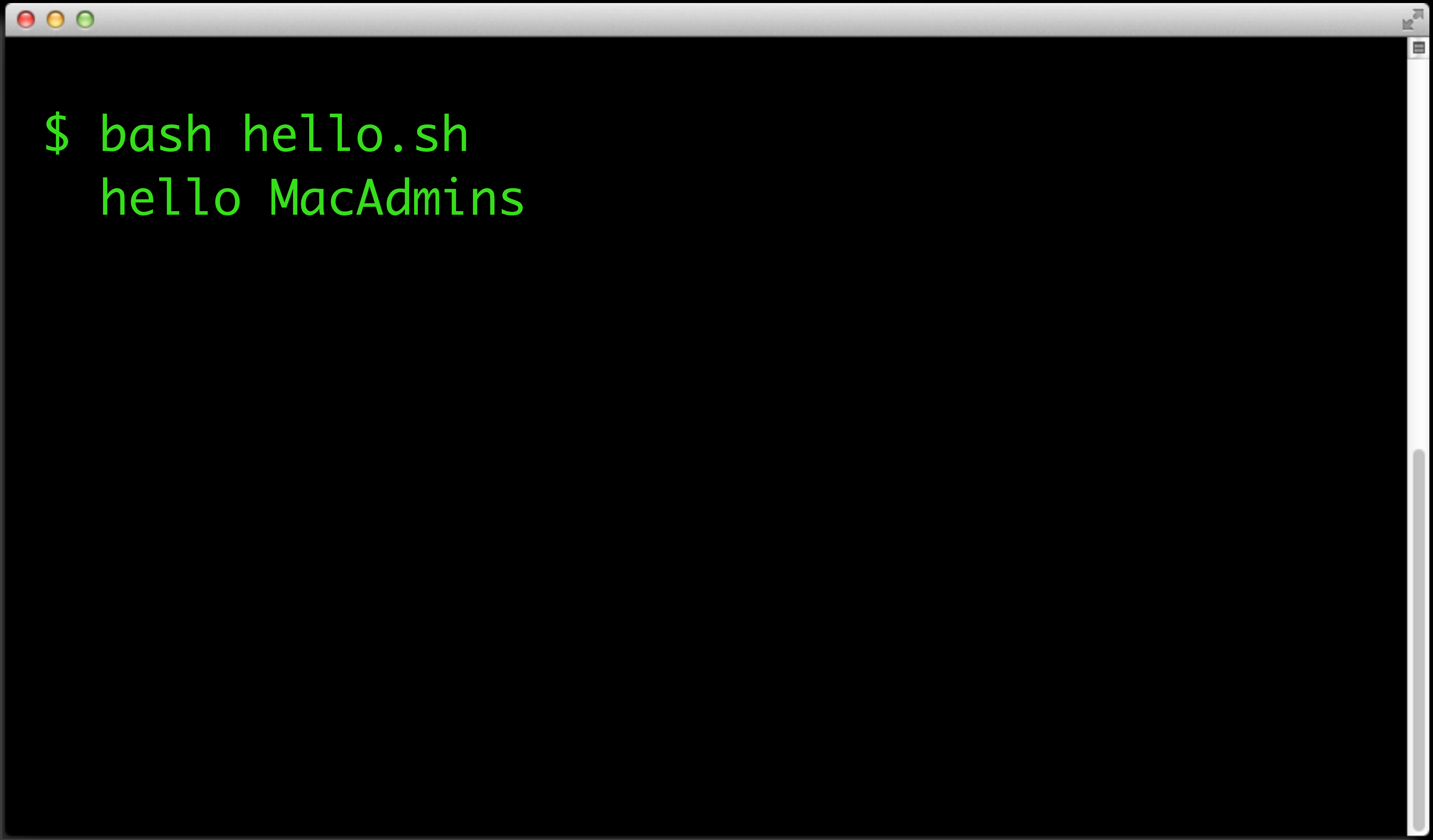
```
#!/bin/bash

# Script Description
# Script Writer
# Date

...put code here...
```

# hello.sh

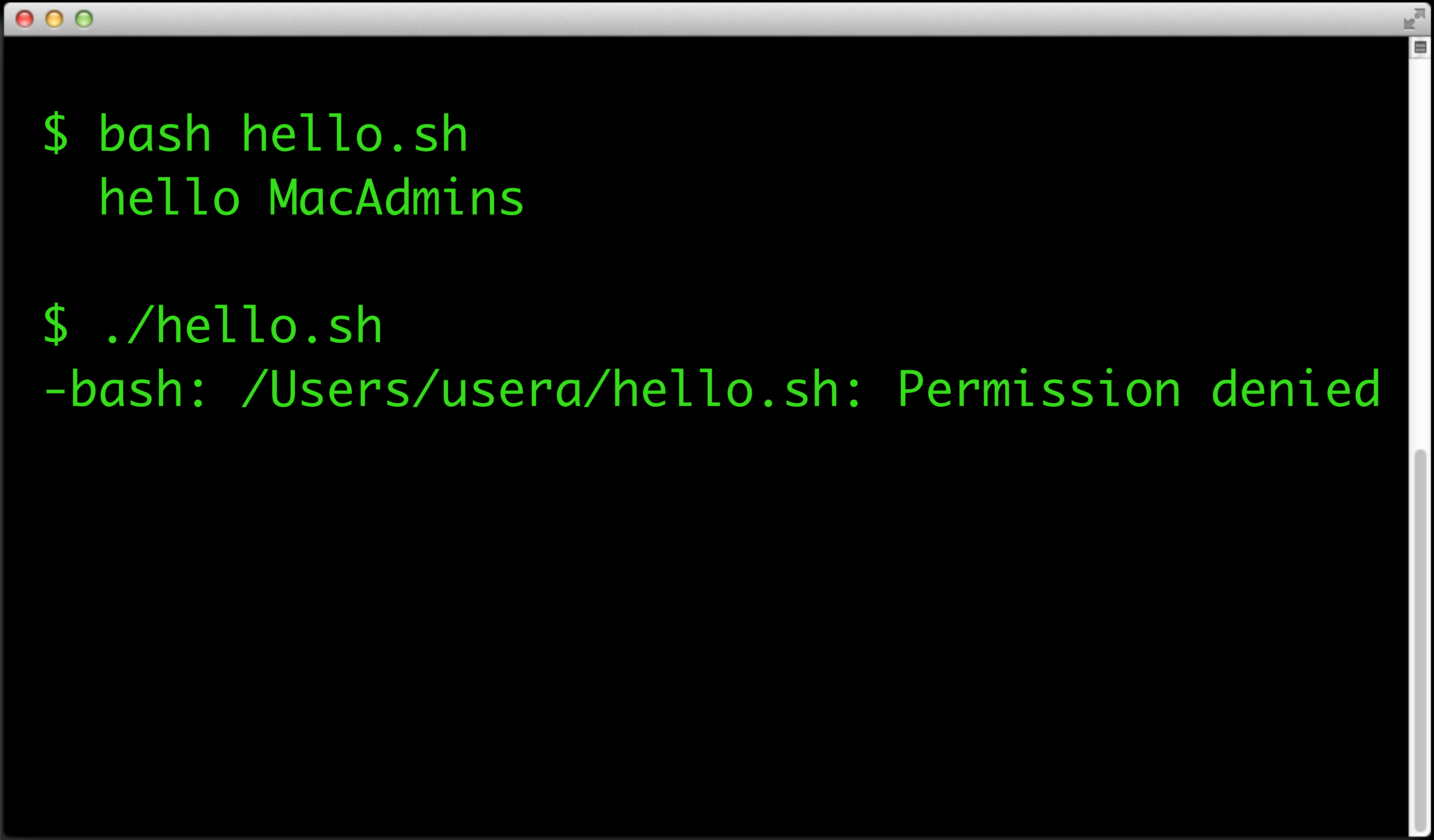
```
#!/bin/bash

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

# echo hello MacAdmins to console
echo "hello MacAdmins"
```

## echo

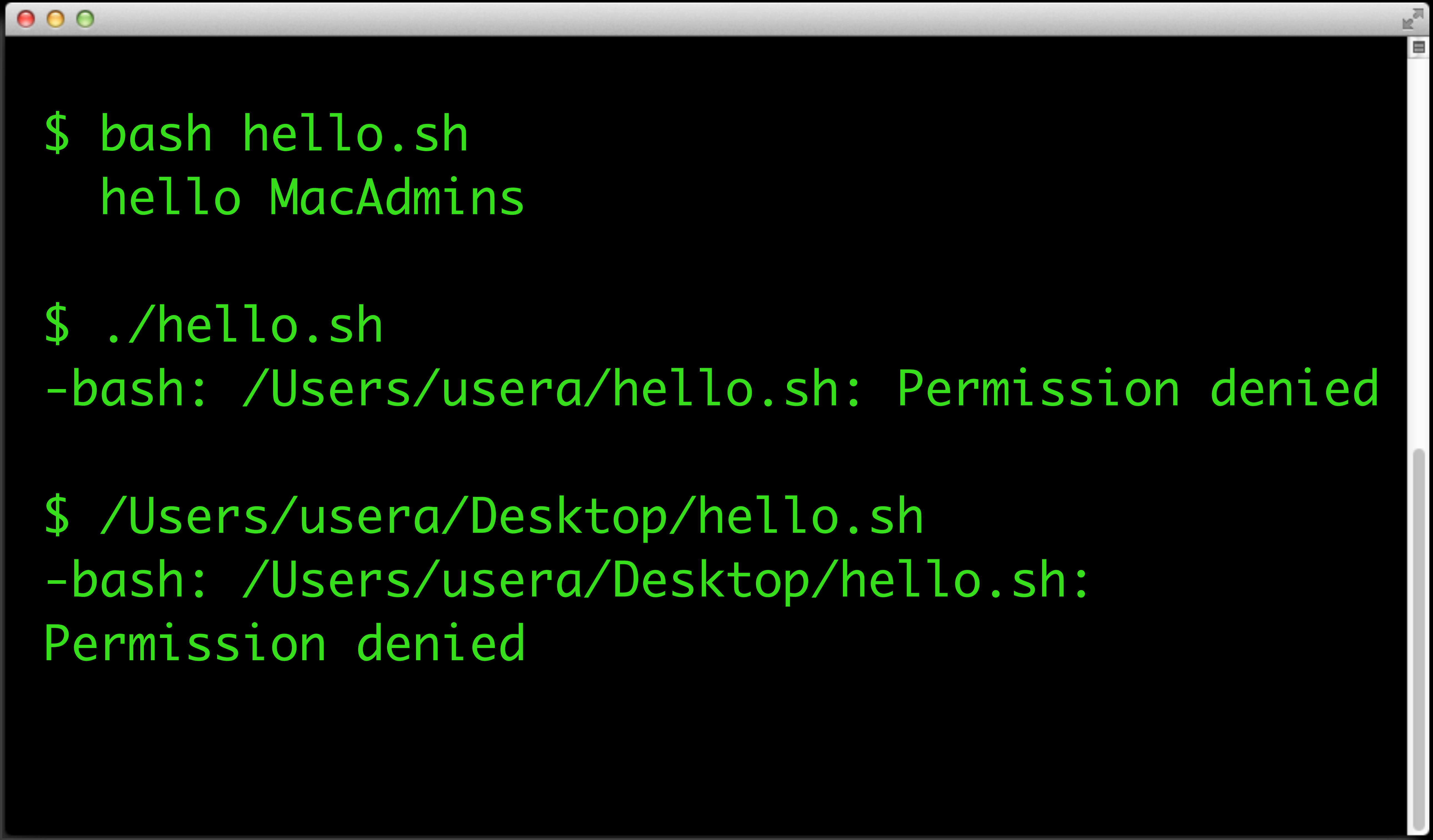
- Outputs string to stdout
- Double Quotes around string
- Add echos for
  - debugging
  - information



```
$ bash hello.sh
  hello MacAdmins
```



```
$ bash hello.sh
  hello MacAdmins

$ ./hello.sh
-bash: /Users/usera/hello.sh: Permission denied
```



```
$ bash hello.sh
  hello MacAdmins

$ ./hello.sh
-bash: /Users/usera/hello.sh: Permission denied

$ /Users/usera/Desktop/hello.sh
-bash: /Users/usera/Desktop/hello.sh:
Permission denied
```

## Permissions

- List the permissions: `ls -l`
- Change Permissions:

```
chmod field+-bit(s) filename
```

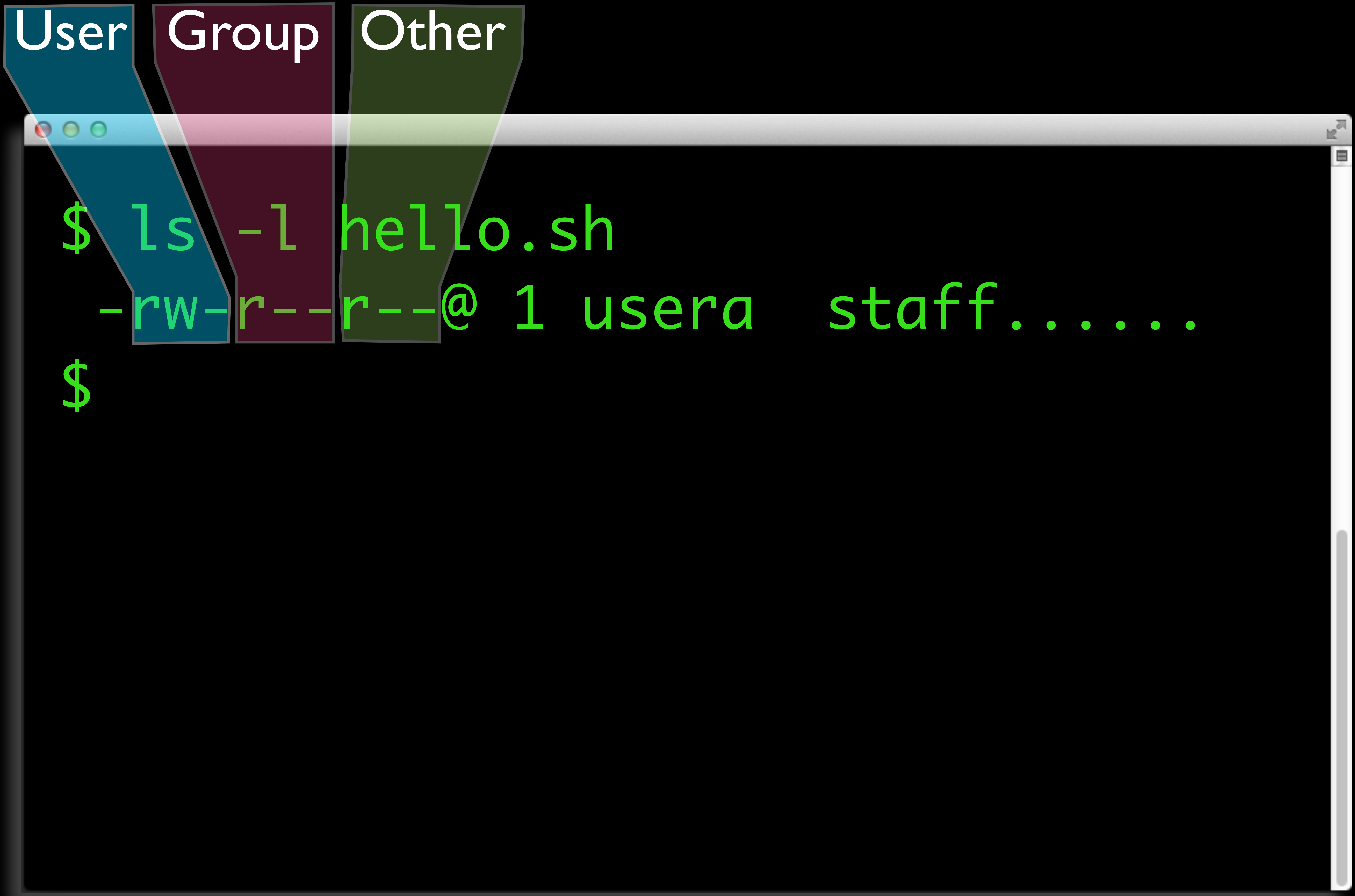
- Change Ownership:

```
chown owner:group filename
```

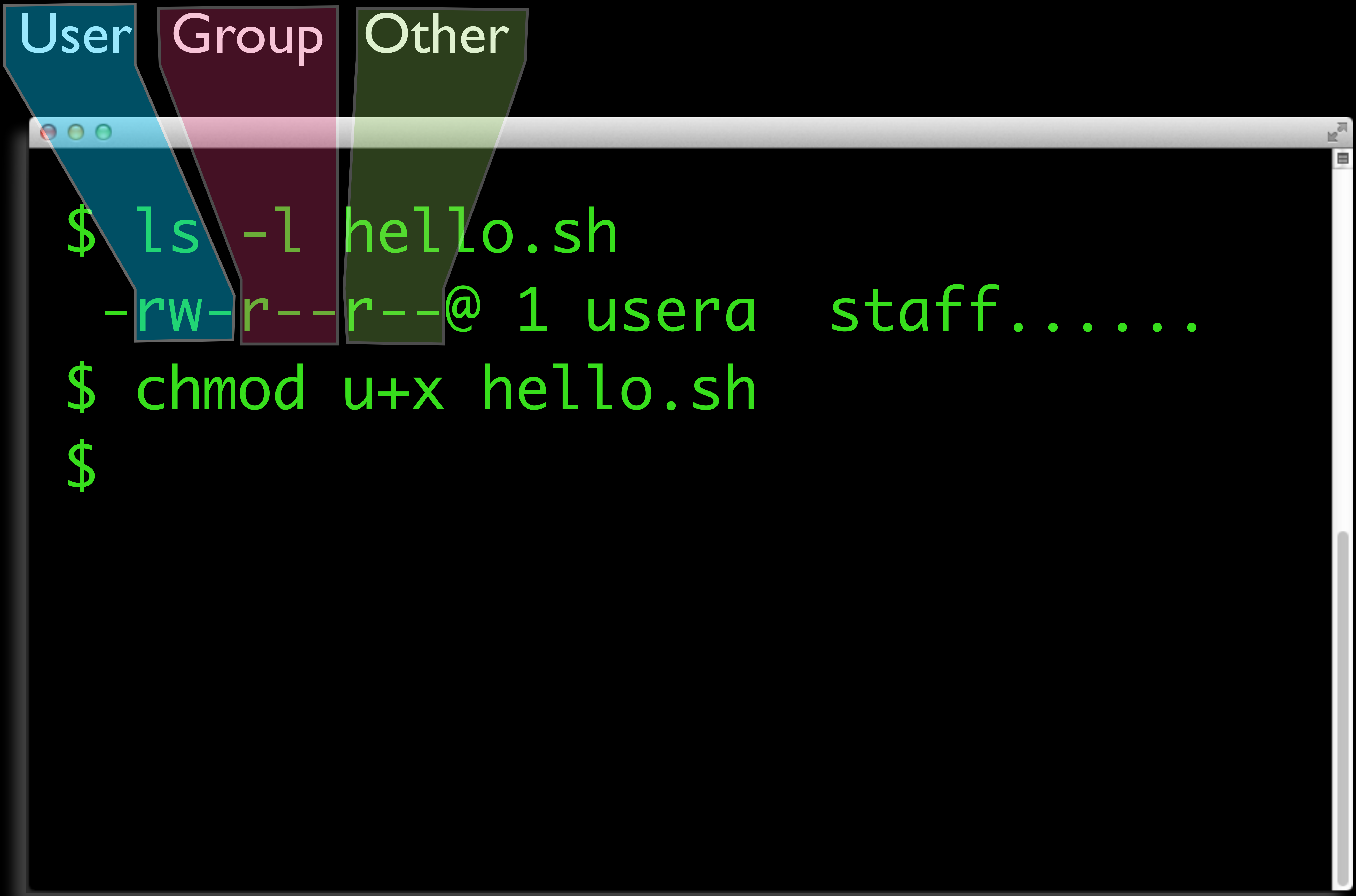
## Execute Bit!

### Permissions in a nutshell

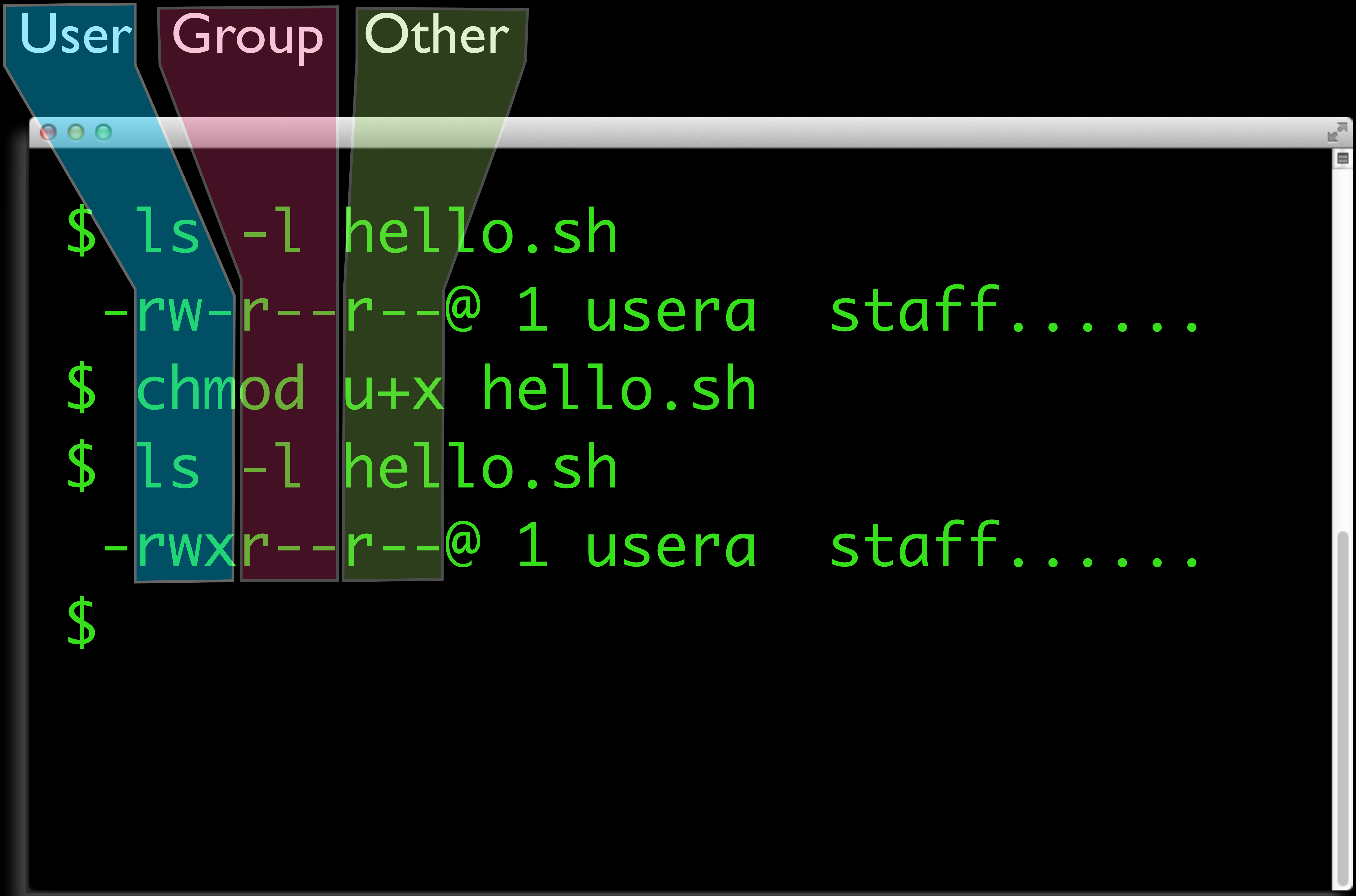
- 3 Fields: (u)ser, (g)roup, (o)ther
- 3 Bits/Field: (r)ead, (w)rite, e(x)ecute
- Execute by default not set



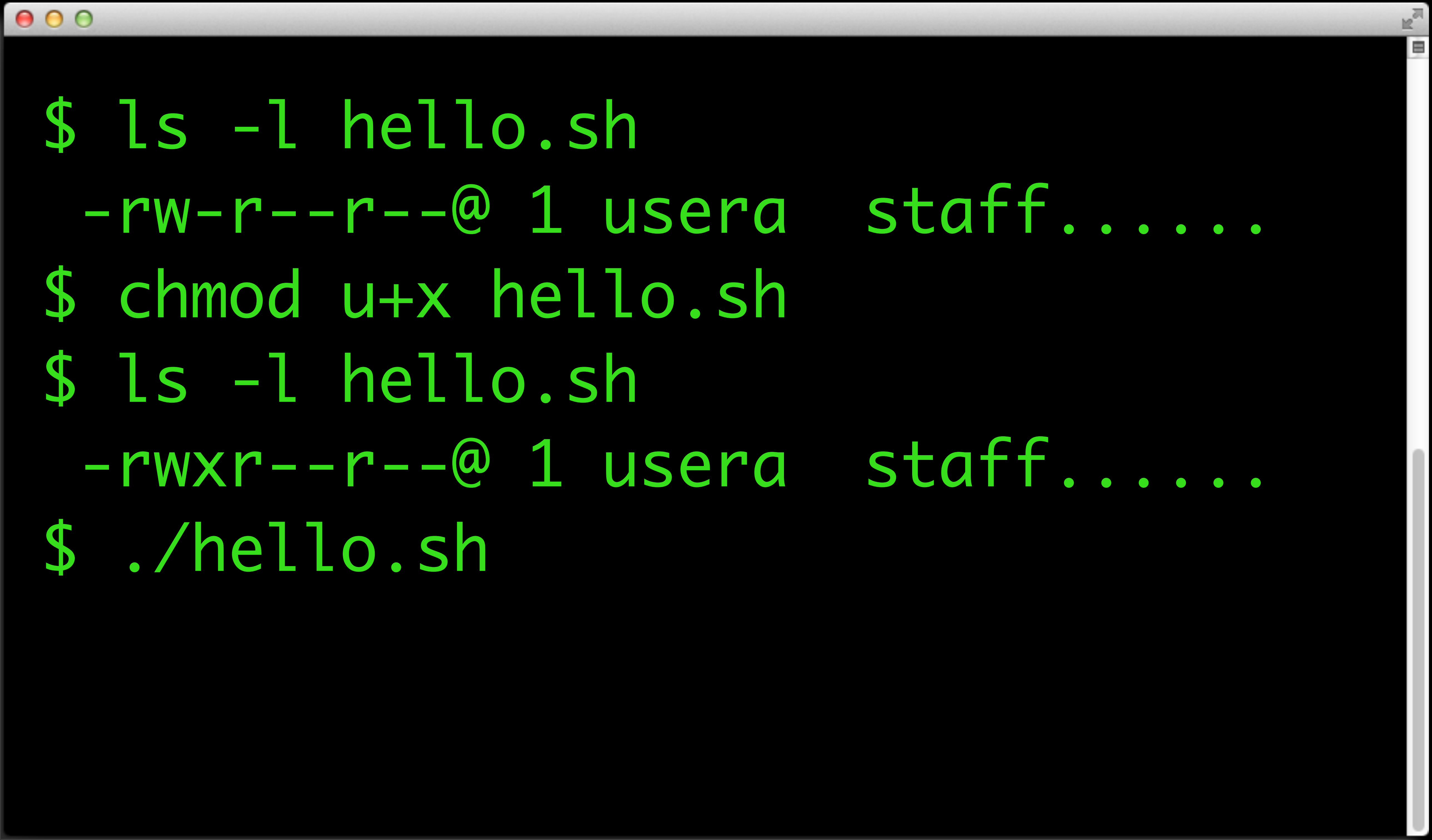
```
$ ls -l hello.sh
 -rw-r--r--@ 1 usera  staff......
$
```



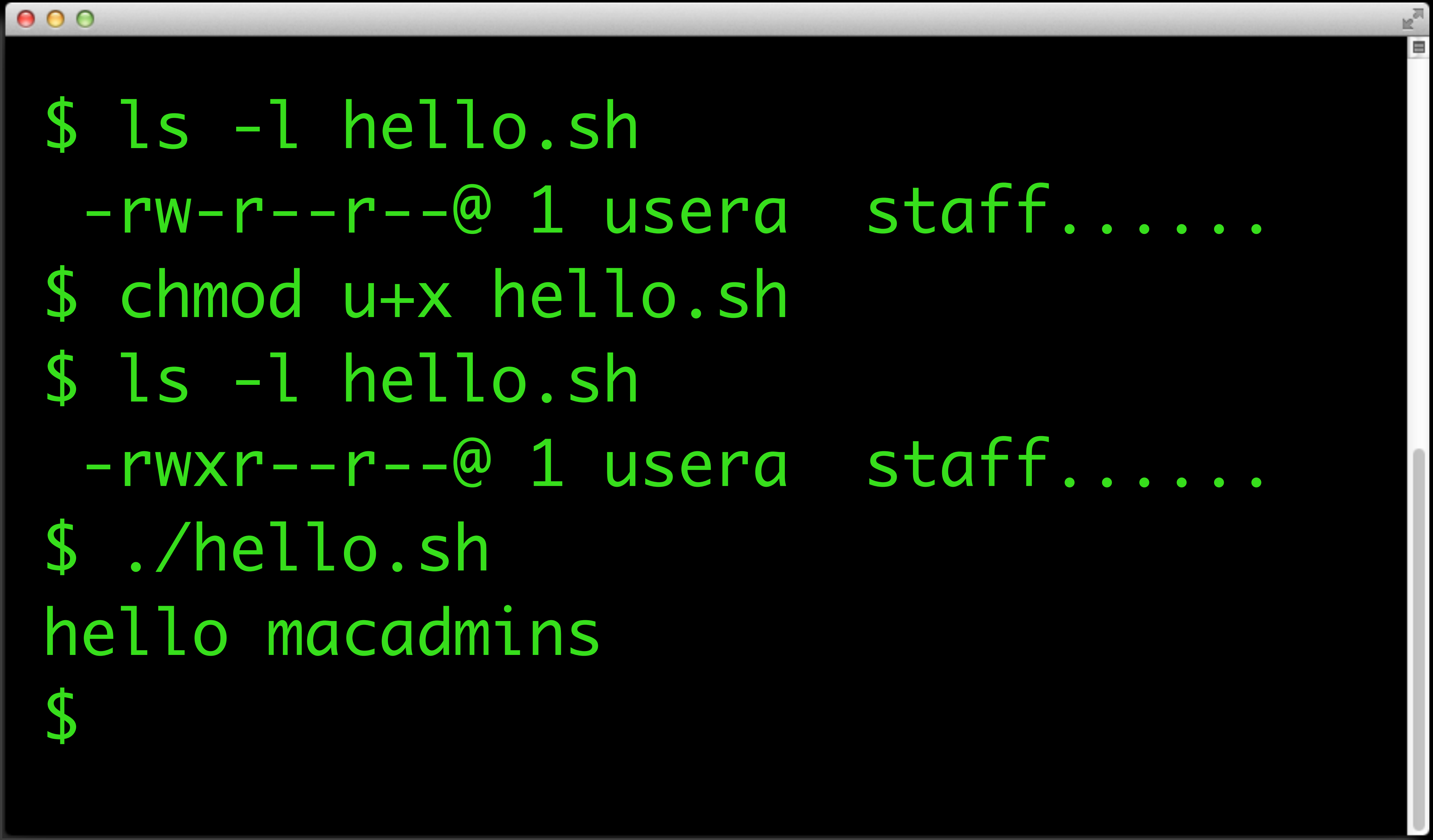
```
$ ls -l hello.sh
 -rw-r--r--@ 1 usera  staff......
$ chmod u+x hello.sh
$
```



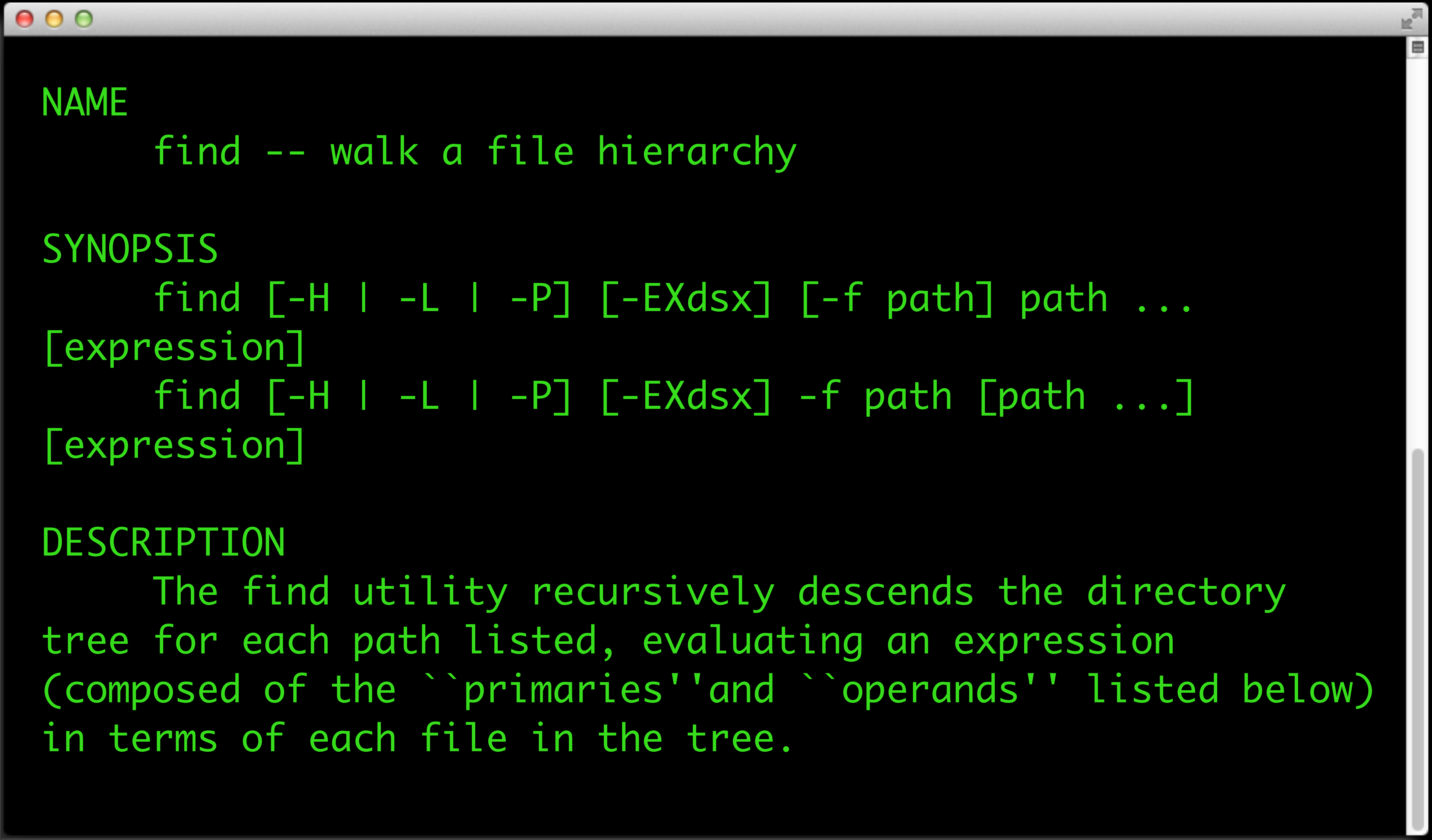
```
$ ls -l hello.sh
 -rw-r--r--@ 1 usera  staff......
$ chmod u+x hello.sh
$ ls -l hello.sh
 -rwxr--r--@ 1 usera  staff......
$
```



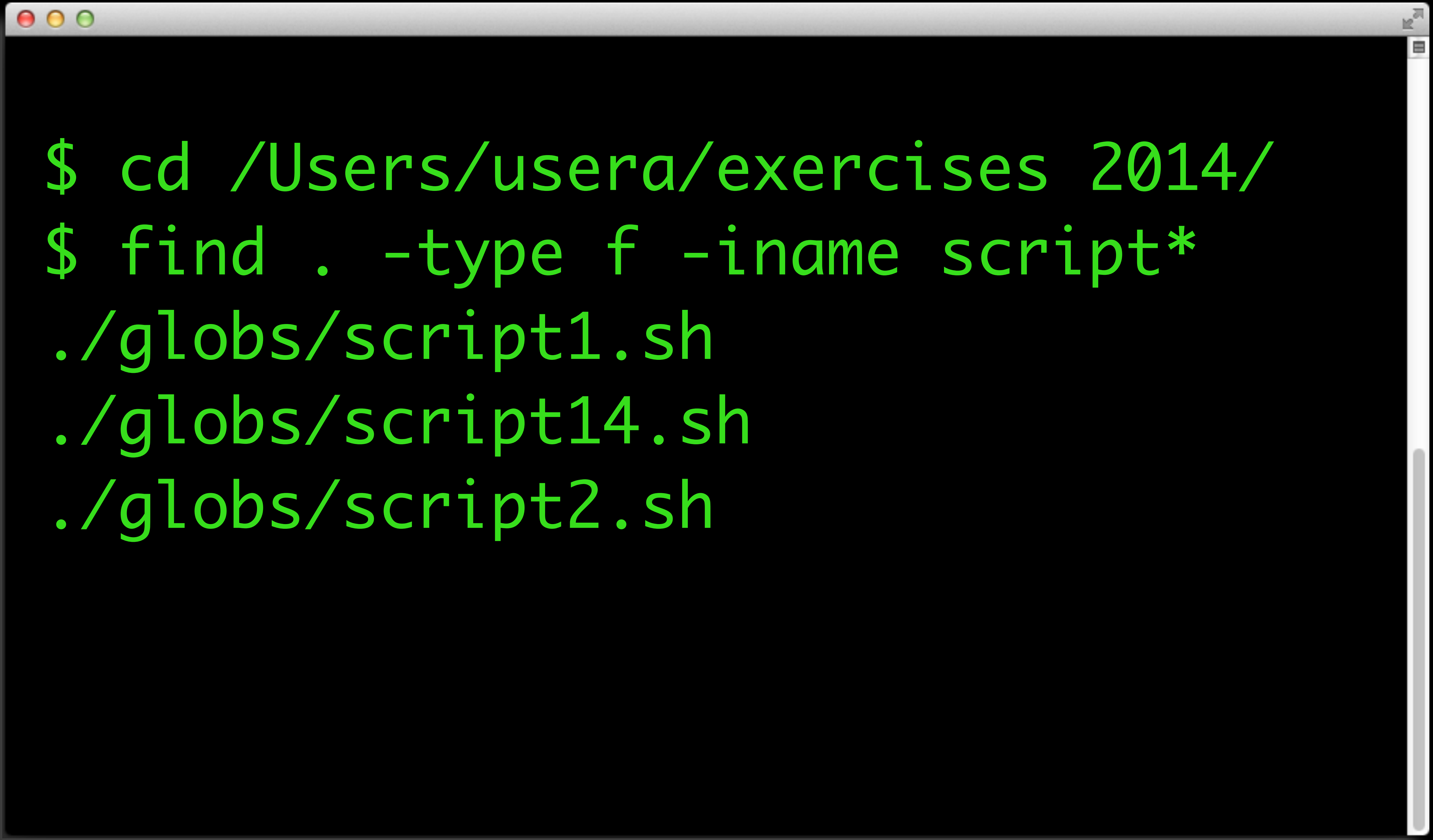
```
$ ls -l hello.sh
 -rw-r--r--@ 1 usera  staff......
$ chmod u+x hello.sh
$ ls -l hello.sh
 -rwxr--r--@ 1 usera  staff......
$ ./hello.sh
```



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



```
NAME
     find -- walk a file hierarchy

SYNOPSIS
     find [-H | -L | -P] [-EXdsx] [-f path] path ...
[expression]
     find [-H | -L | -P] [-EXdsx] -f path [path ...]
[expression]

DESCRIPTION
     The find utility recursively descends the directory
tree for each path listed, evaluating an expression
(composed of the ``primaries''and ``operands'' listed below)
in terms of each file in the tree.
```



```
$ cd /Users/usera/exercises 2014/
$ find . -type f -iname script*
./globs/script1.sh
./globs/script14.sh
./globs/script2.sh
```

## Starting Your Code

- #!/bin/bash of course!
- Write it in english
- Verbalize the problem
- Start with one small part

## Try It!

- Write hello.sh
- Save to Desktop
- Open Terminal
  - update permissions
  - run script

# hello.sh

```
#!/bin/bash

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

# echo hello MacAdmins to console
echo "hello MacAdmins"
```

break

# Part 2

- Special Characters
- Quoting
- Variables
- Command Substitution

# Terminal Trick

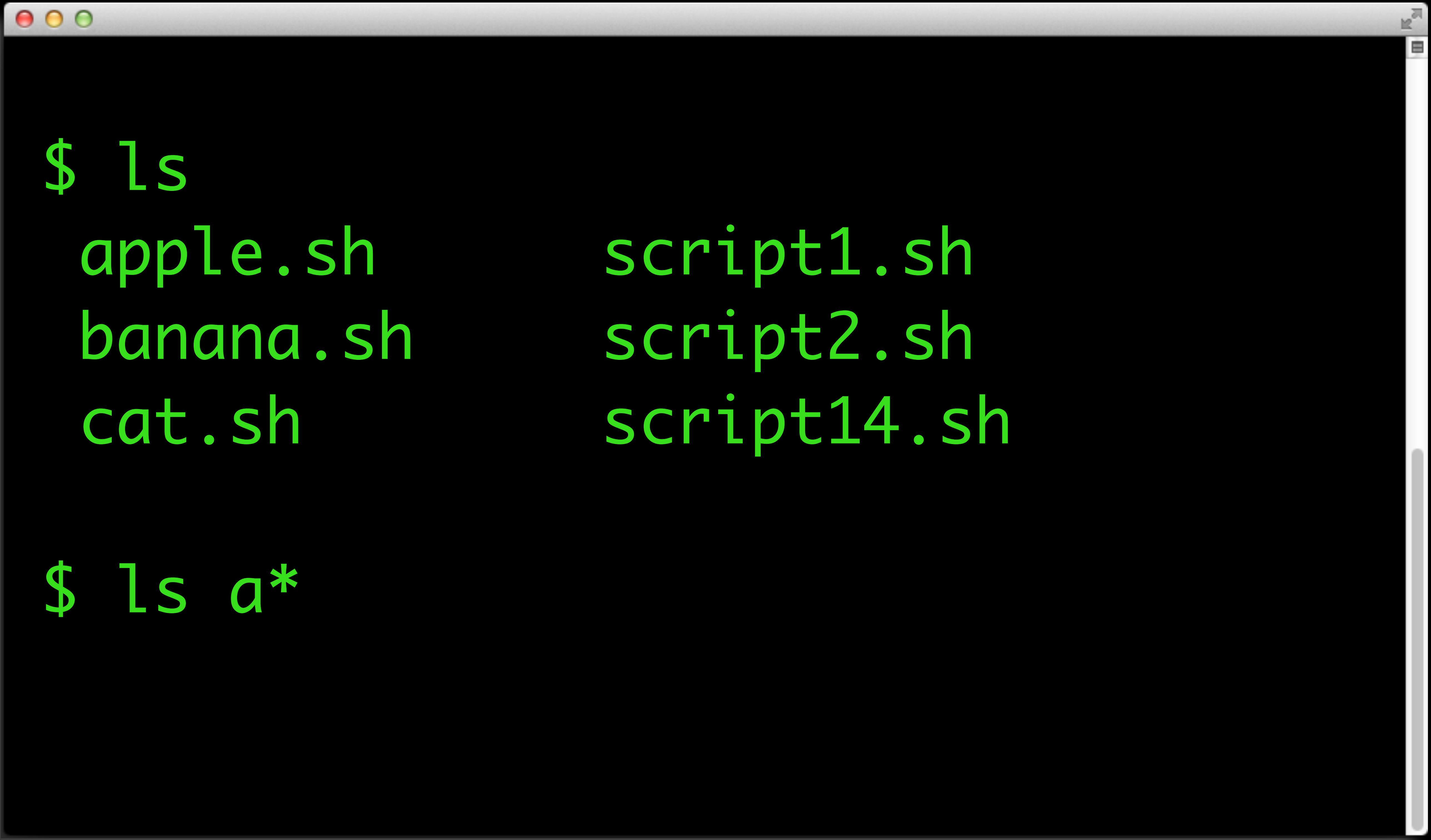
- Open Finder Window
  ```
  open /path/ = Open /path Fldr
  open . = Open Current Dir
  ```
- Open Application
  ```
  open /Applications/Safari.app
  ```
- Open File in Text Editor
  ```
  open -e Command\ Lists.txt
  ```

## Special Chars

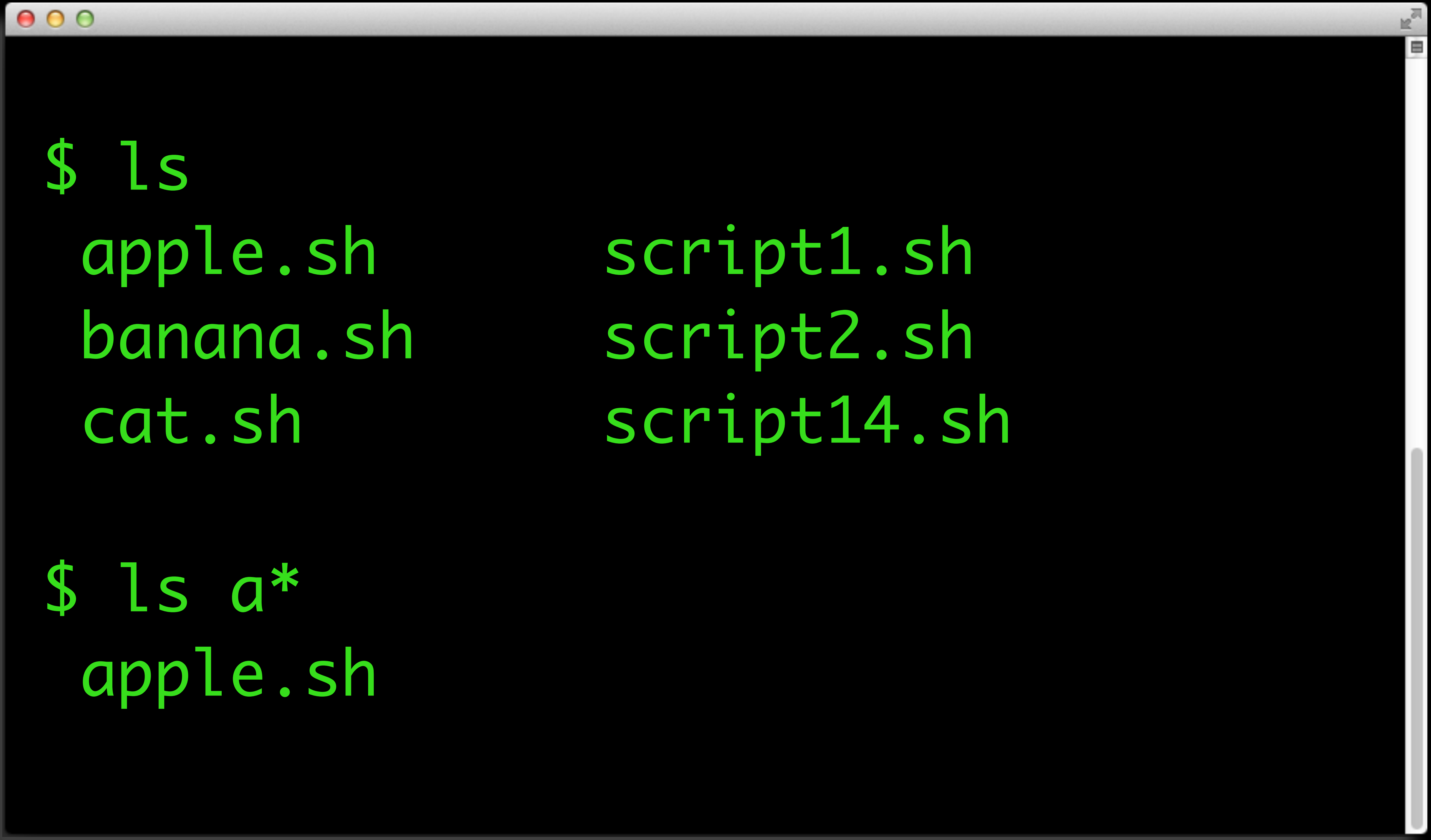
- What are they?
- Why Not?
- !&#|'"`~<>*$?\^()[]{}
  - Space, TAB

# Globs

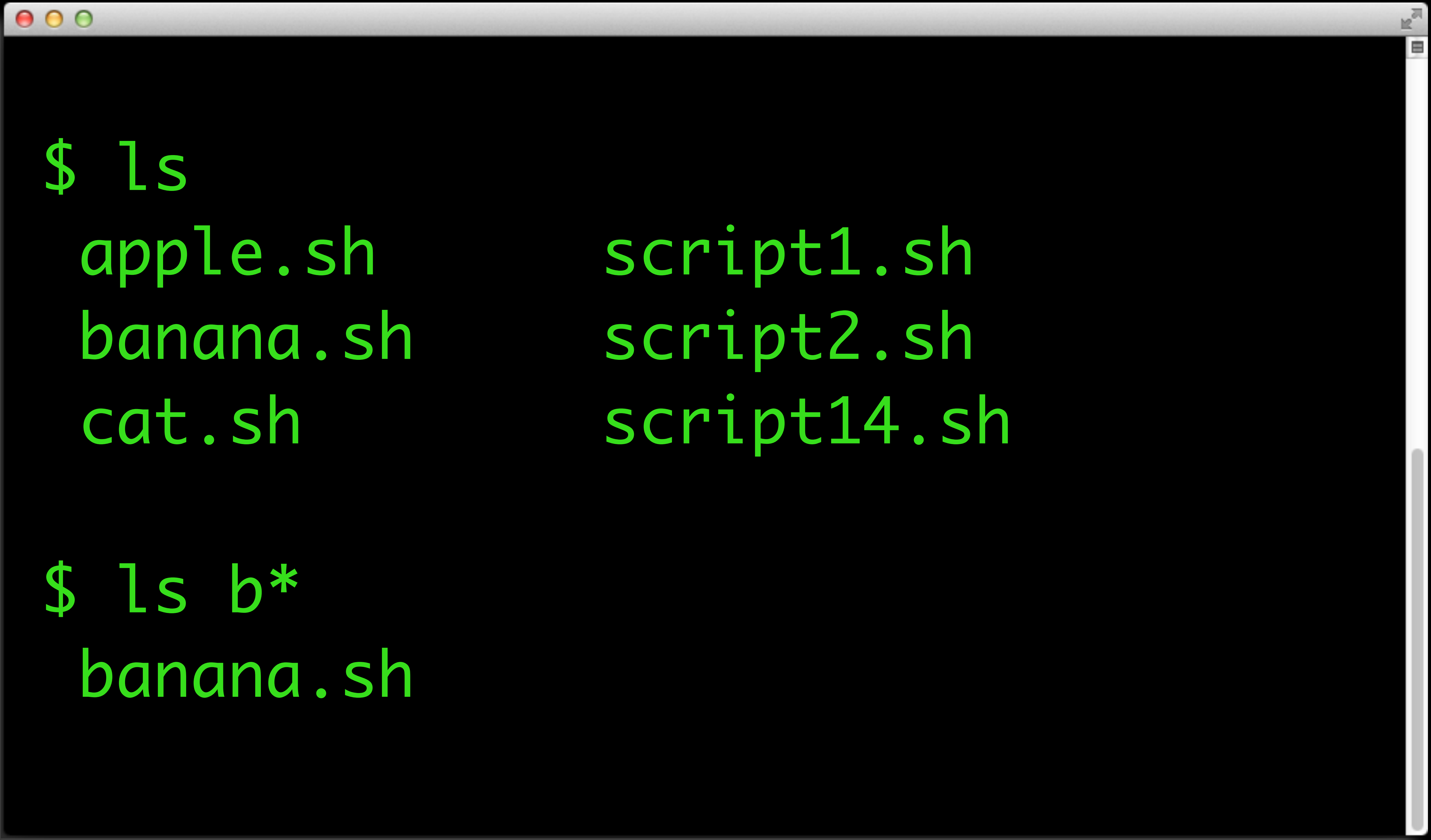
- Filename expansion by Bash
- Not Regular Expressions (RE)
- All Char: *
- One Char: ?
- Escape Char: \
- Group of Char: [ ]
- Negate Char: ^



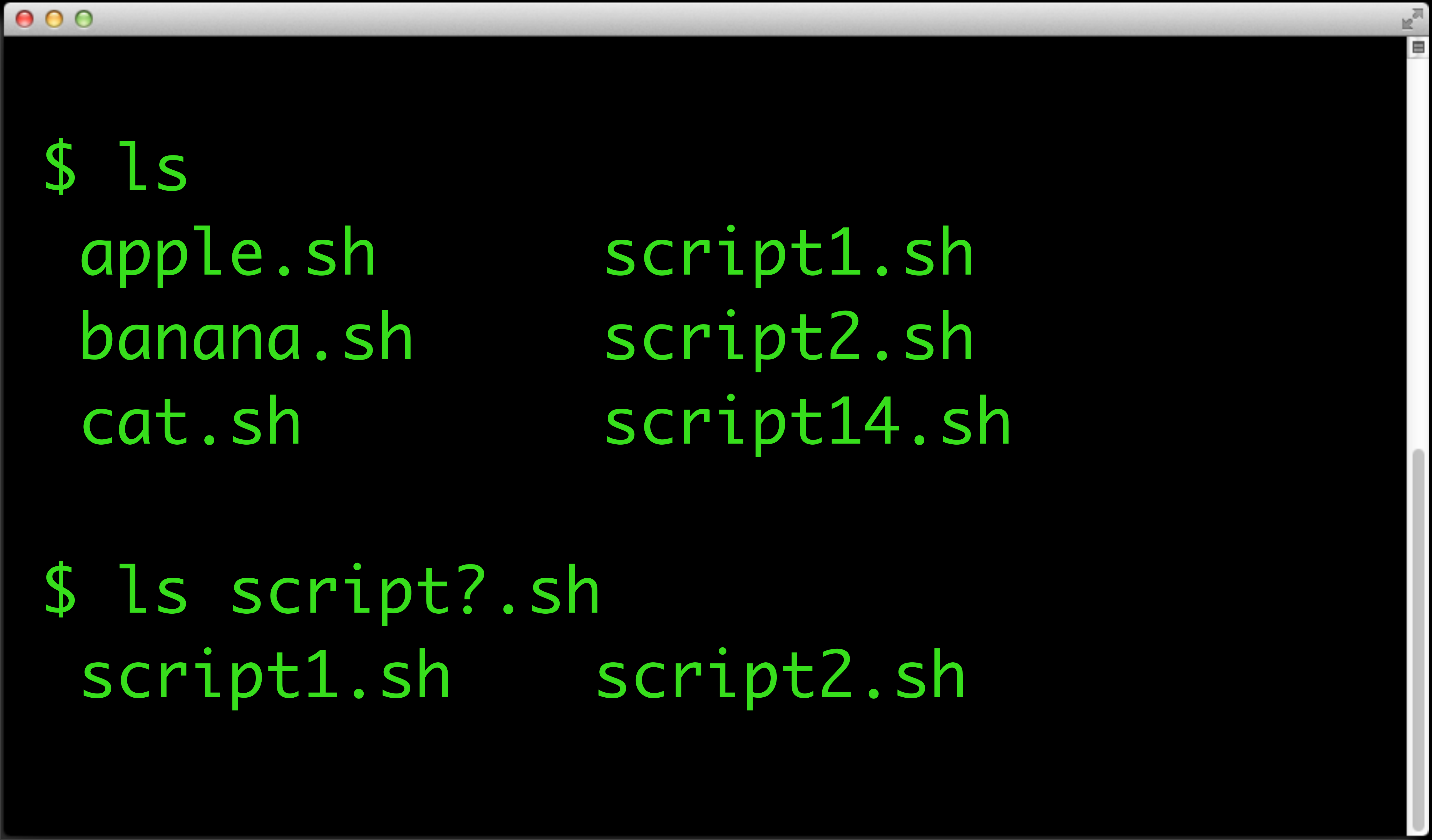
```
$ ls
 apple.sh         script1.sh
 banana.sh        script2.sh
 cat.sh           script14.sh

$ ls a*
```



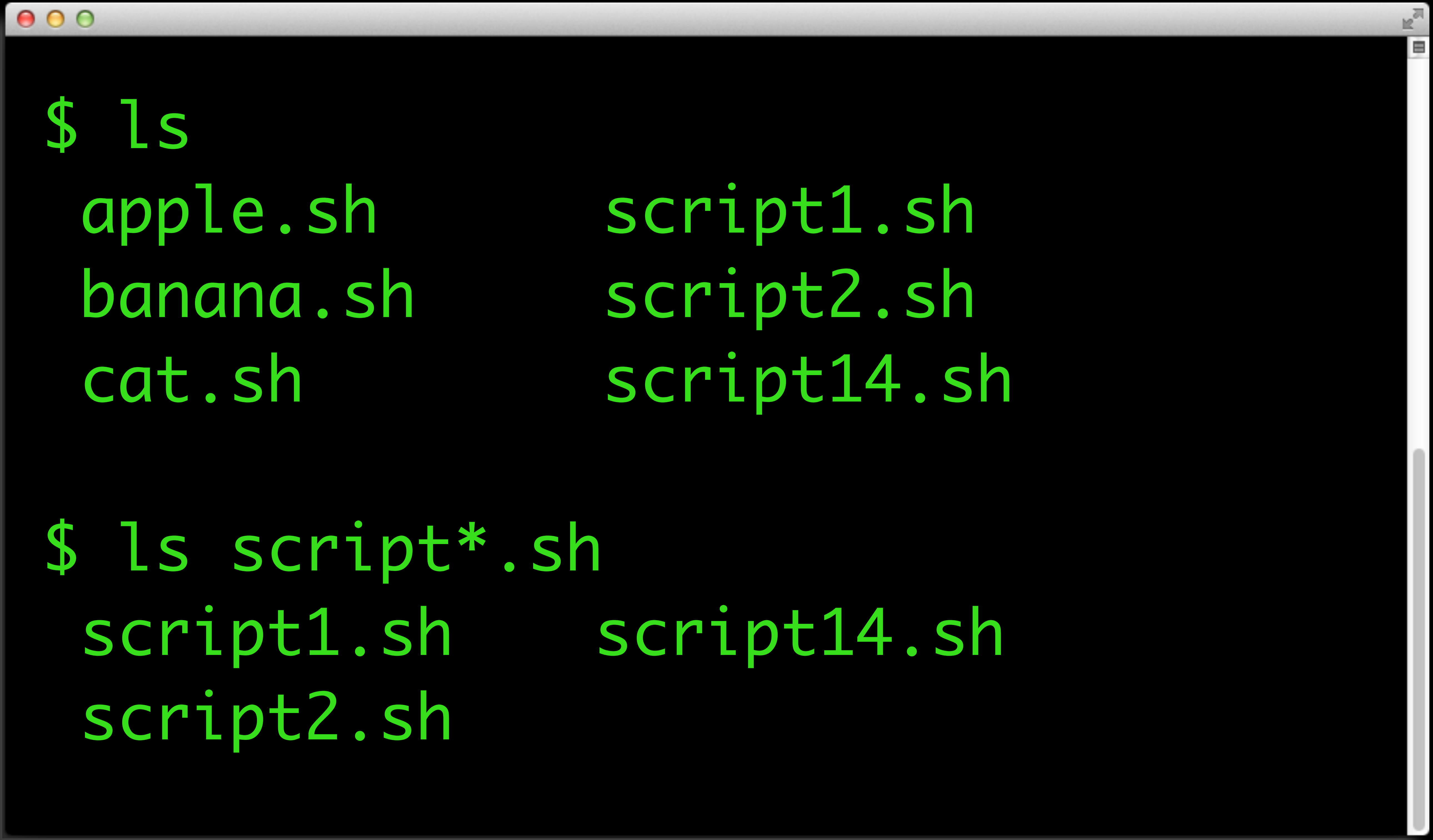
```
$ ls
 apple.sh         script1.sh
 banana.sh        script2.sh
 cat.sh           script14.sh

$ ls a*
 apple.sh
```



```
$ ls
 apple.sh        script1.sh
 banana.sh       script2.sh
 cat.sh          script14.sh

$ ls b*
 banana.sh
```



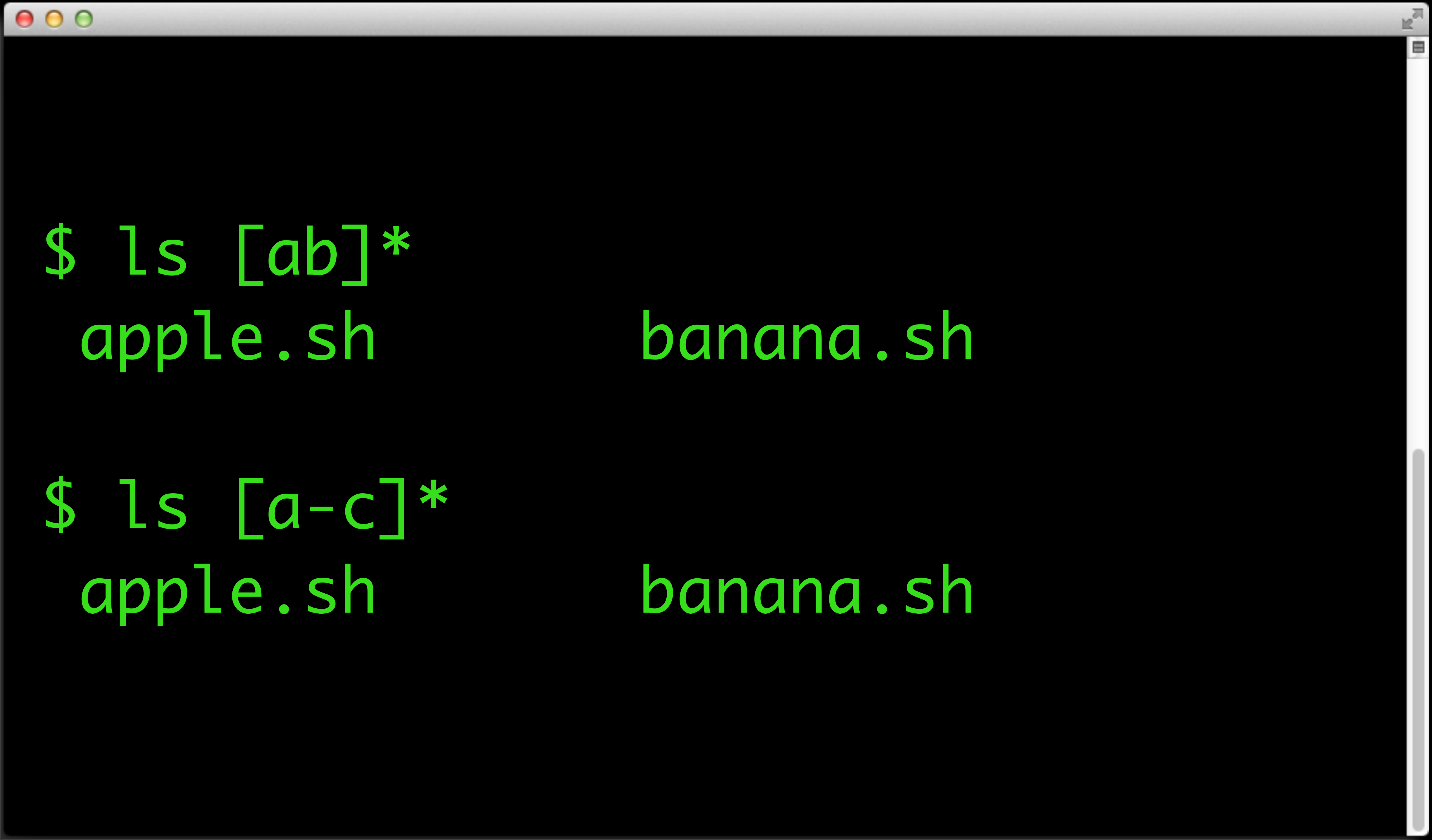
```
$ ls
 apple.sh        script1.sh
 banana.sh       script2.sh
 cat.sh          script14.sh

$ ls script?.sh
 script1.sh    script2.sh
```



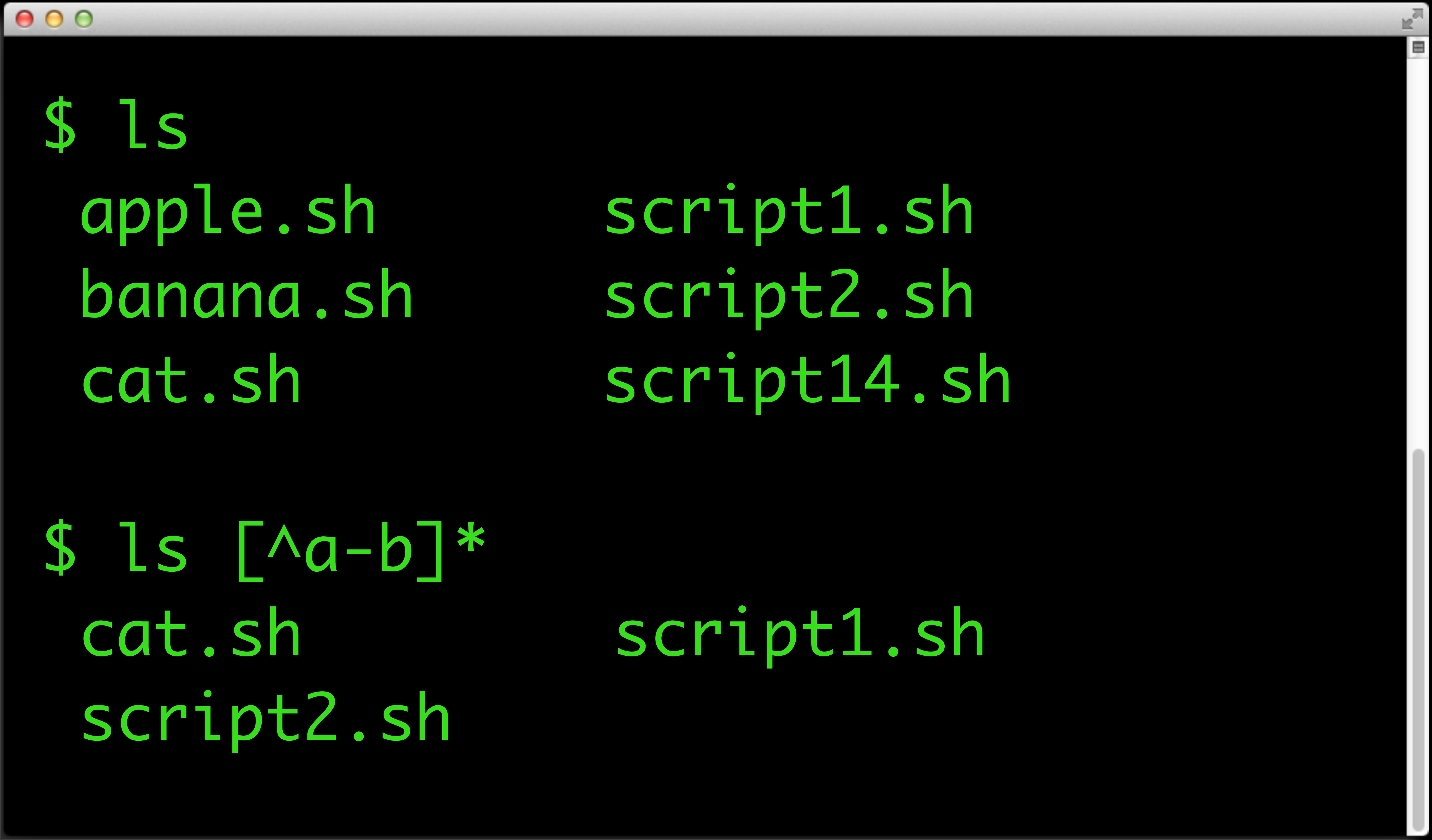
```
$ ls
 apple.sh        script1.sh
 banana.sh       script2.sh
 cat.sh          script14.sh

$ ls script*.sh
 script1.sh    script14.sh
 script2.sh
```

## Pattern Matching

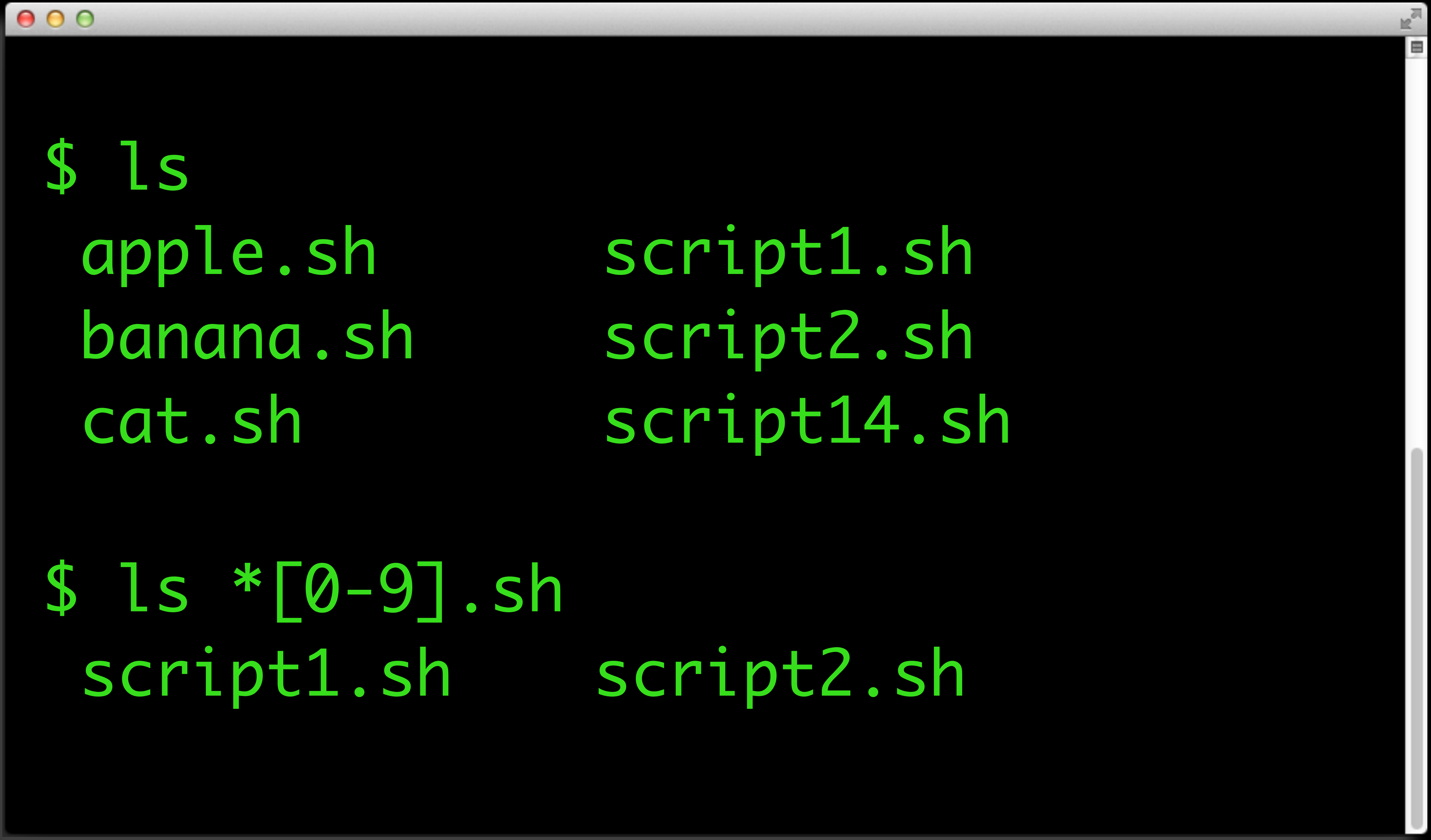
- Match single occurrence of char in [ ]
  - Find range [0-9]
    matches 0 through 9
  - Find range [a-z]
    matches a through z
- Find specific char [ab]
  - Finds a or b



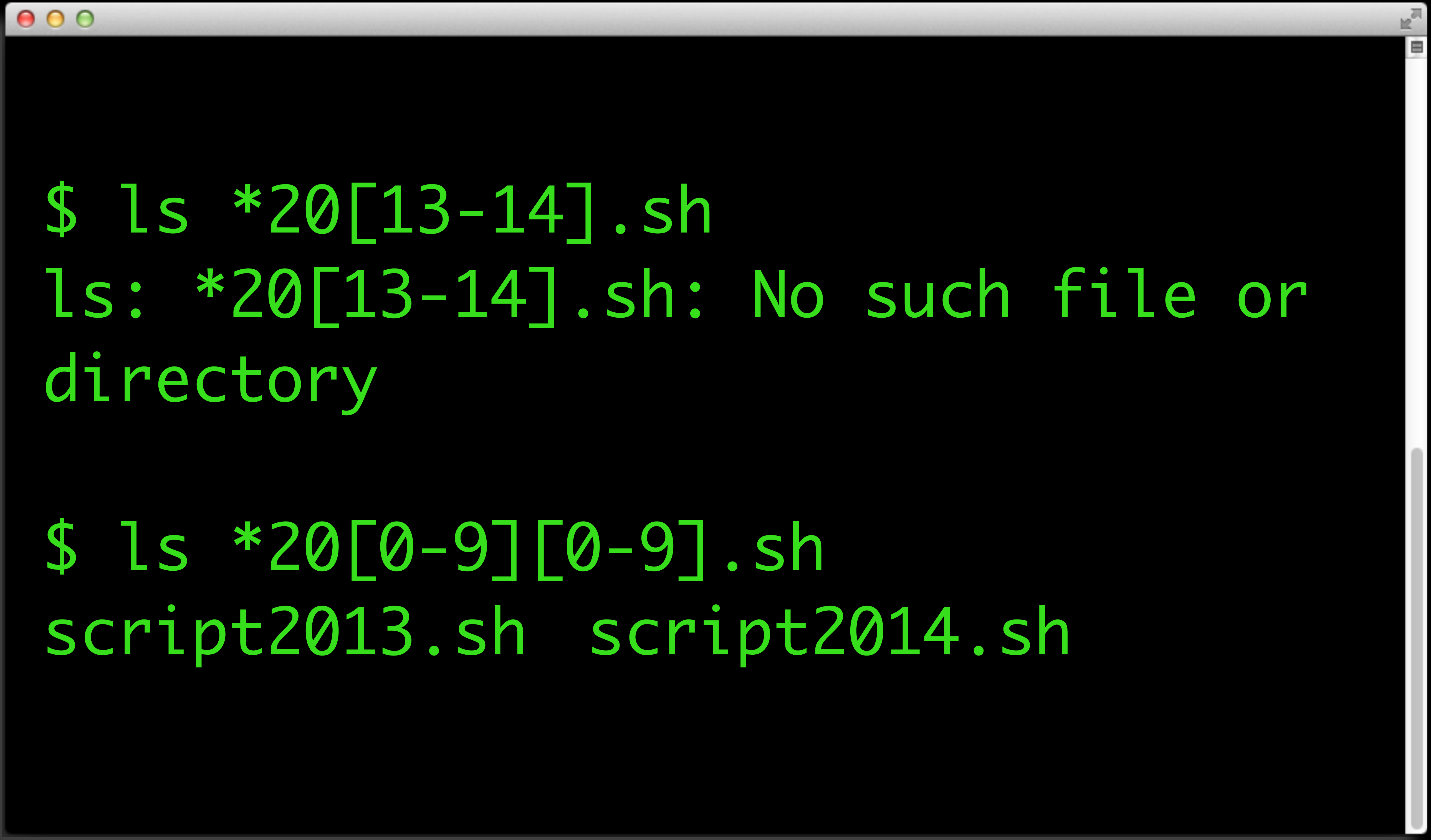
```
$ ls [ab]*
 apple.sh            banana.sh

$ ls [a-c]*
 apple.sh            banana.sh
```



```
$ ls
 apple.sh          script1.sh
 banana.sh         script2.sh
 cat.sh            script14.sh

$ ls [^a-b]*
 cat.sh             script1.sh
 script2.sh
```



```
$ ls
 apple.sh        script1.sh
 banana.sh       script2.sh
 cat.sh          script14.sh

$ ls *[0-9].sh
 script1.sh    script2.sh
```



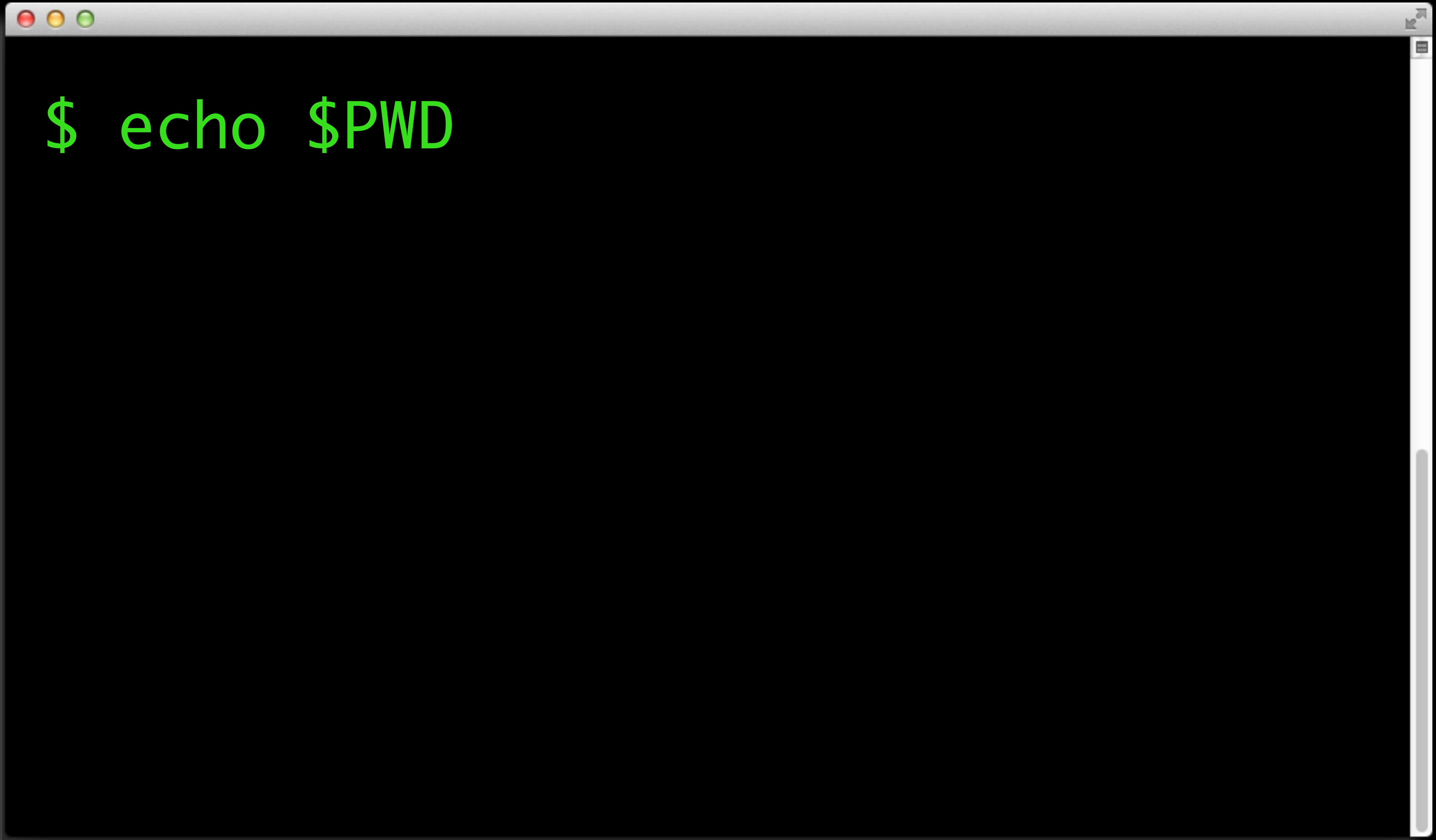
```
$ ls *20[13-14].sh
ls: *20[13-14].sh: No such file or
directory

$ ls *20[0-9][0-9].sh
script2013.sh  script2014.sh
```

## Try It!

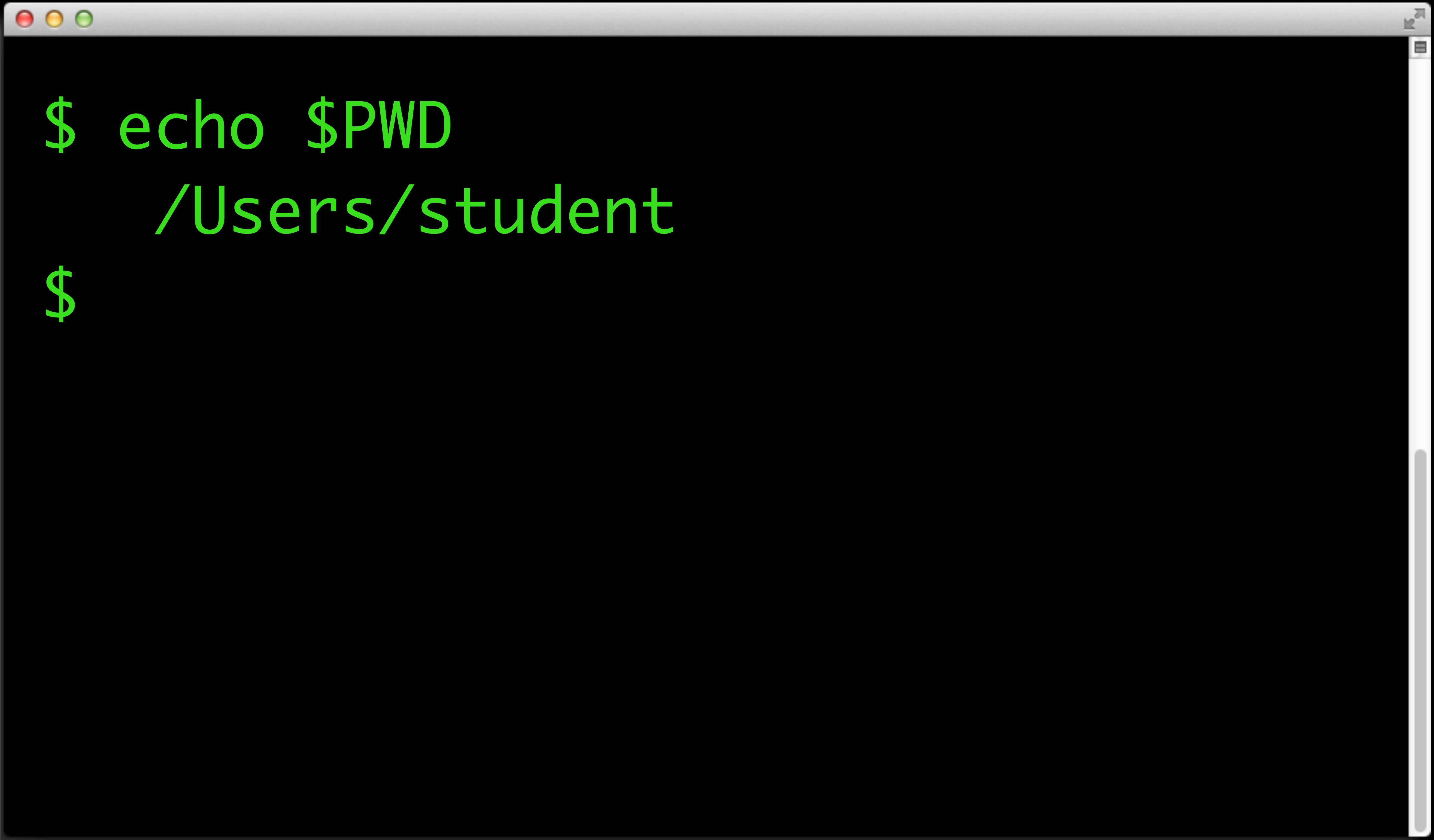
- List files in /bin
- How many start with 'b'? 'r'?
- How man end in 'sh'?

# Quoting

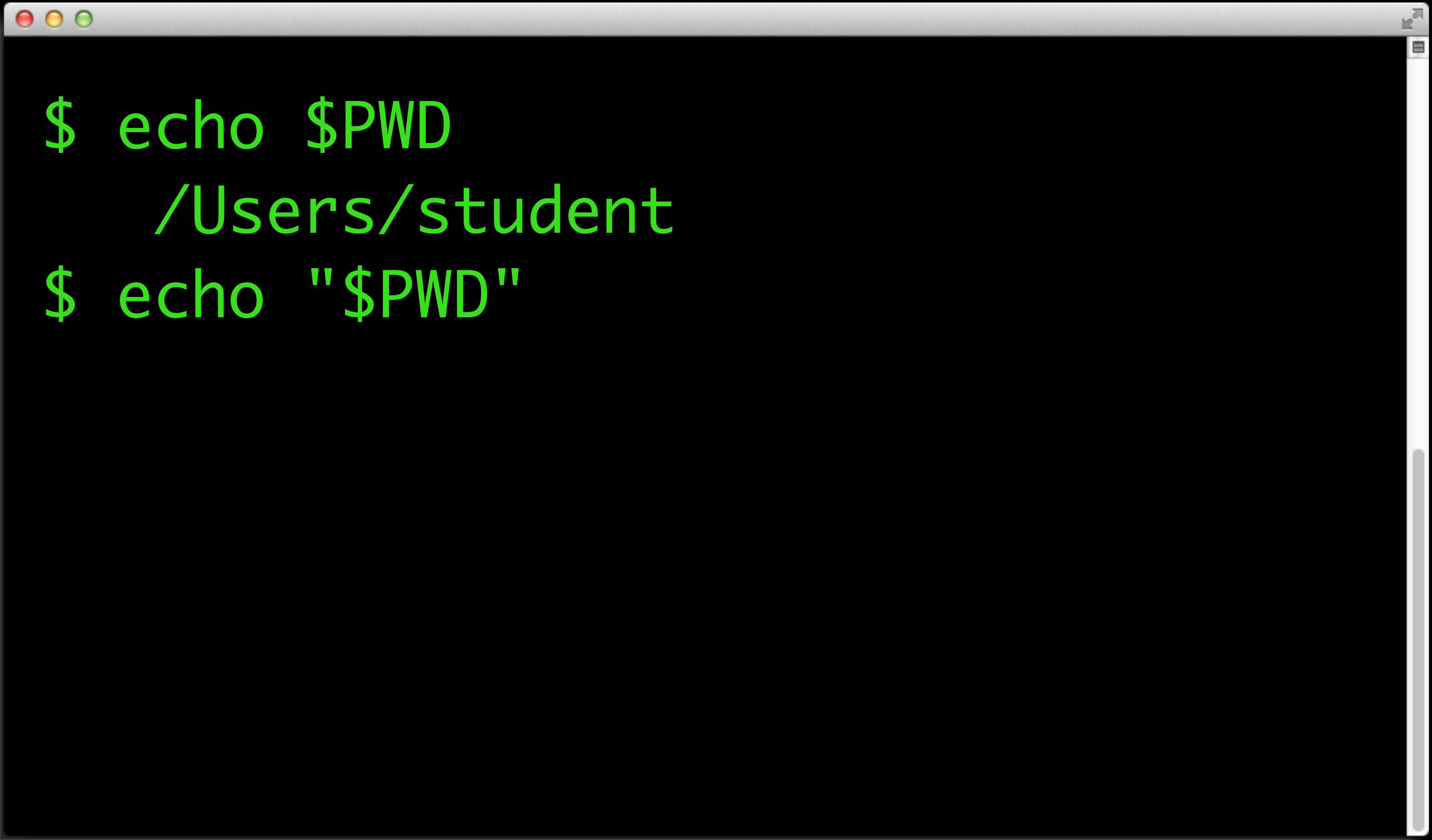
| | |
|---|---|
| Escape Next Char | \ |
| Escape All except $, `, \ | "abc" |
| Single Quotes | 'abc' |



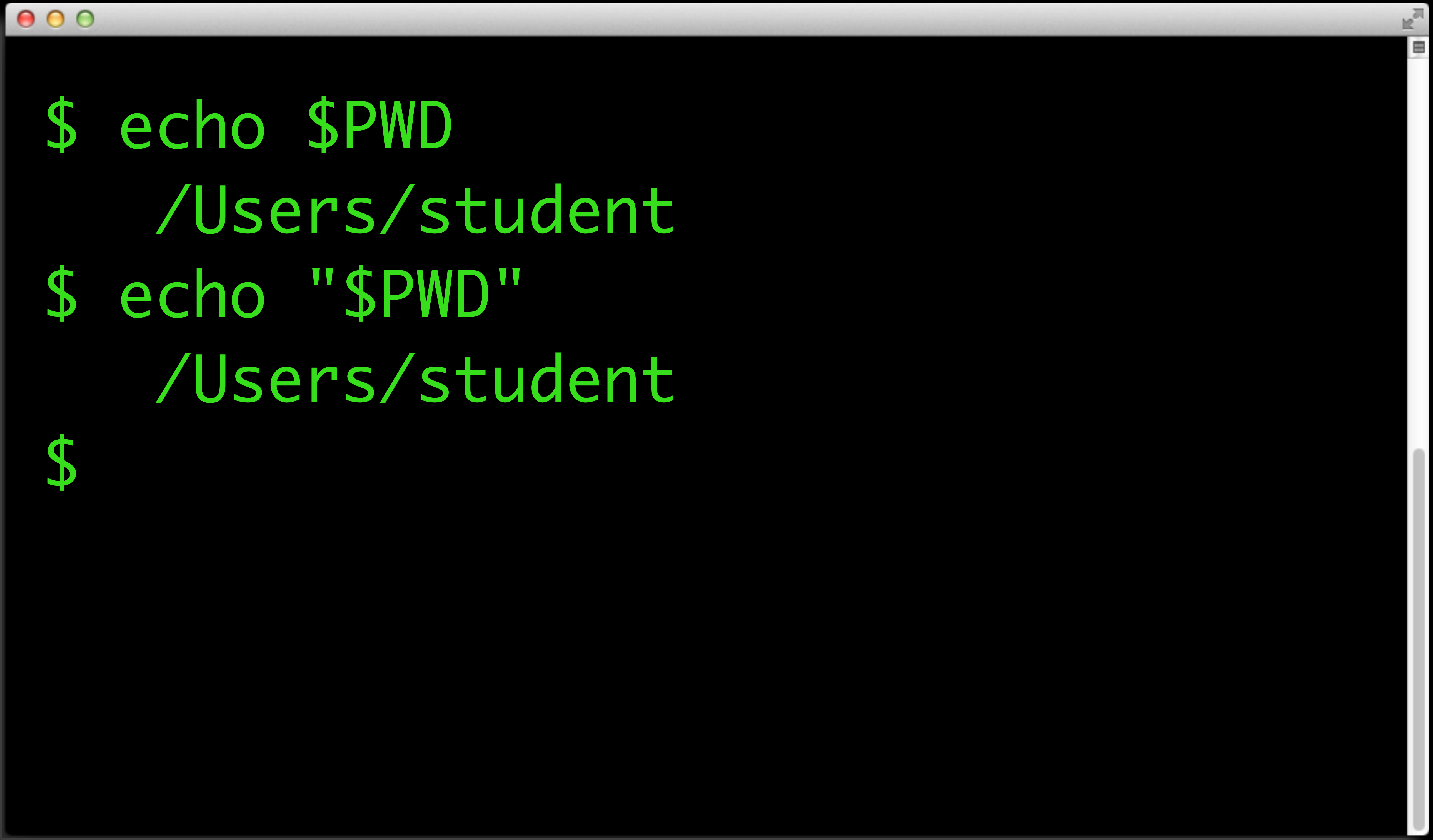
```
$ echo $PWD
```



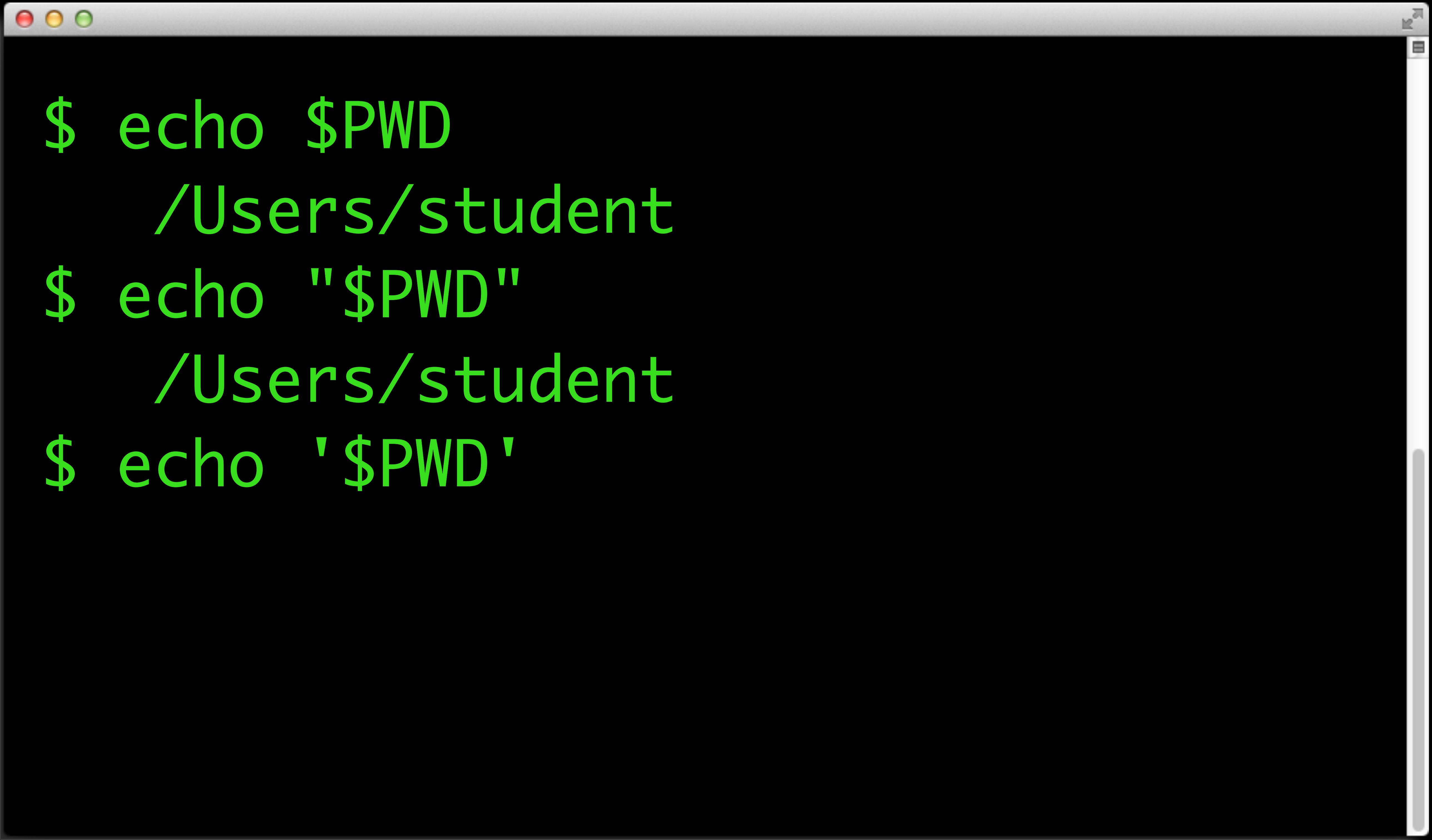
```
$ echo $PWD
   /Users/student
$
```



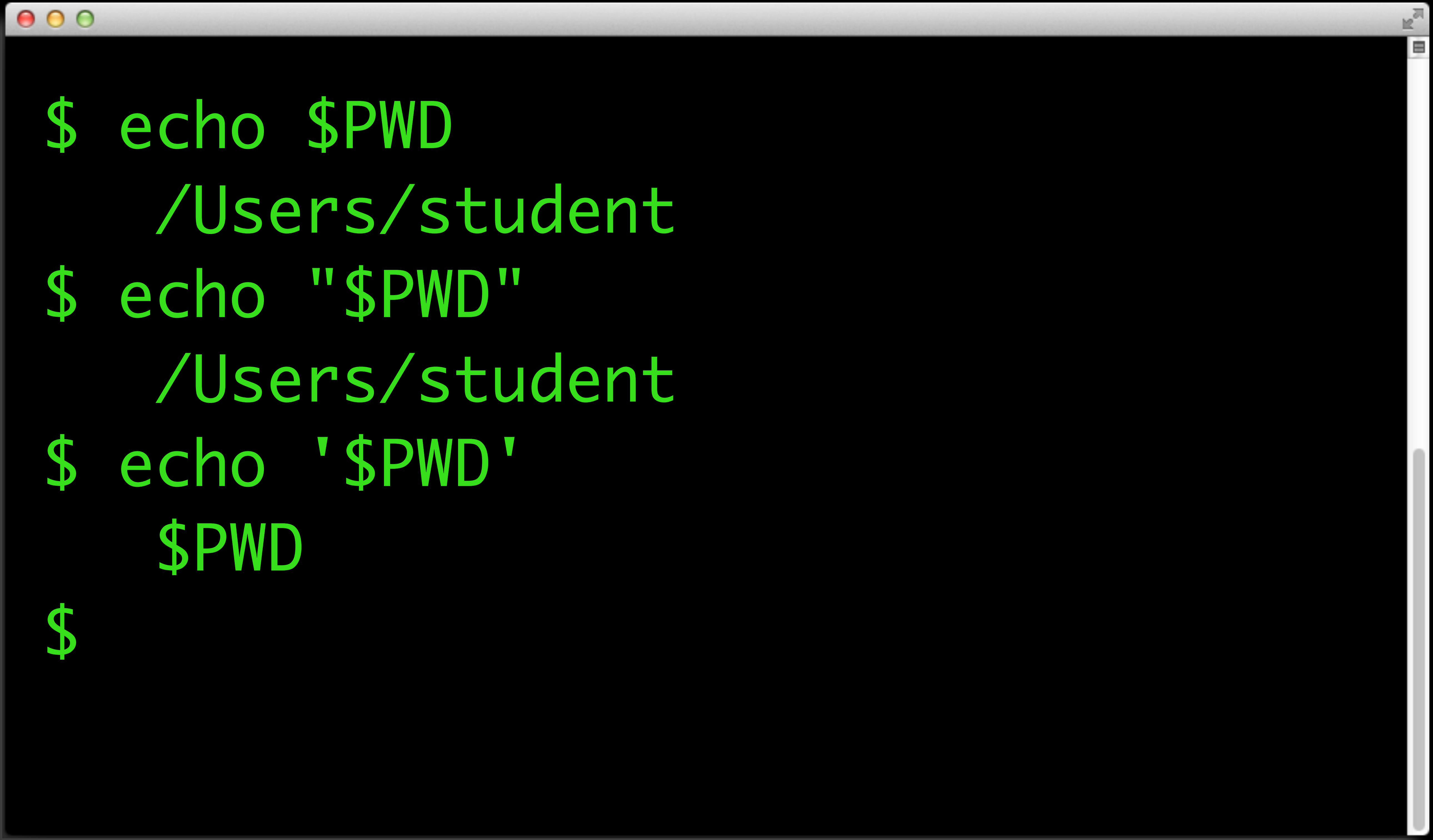
```
$ echo $PWD
   /Users/student
$ echo "$PWD"
```



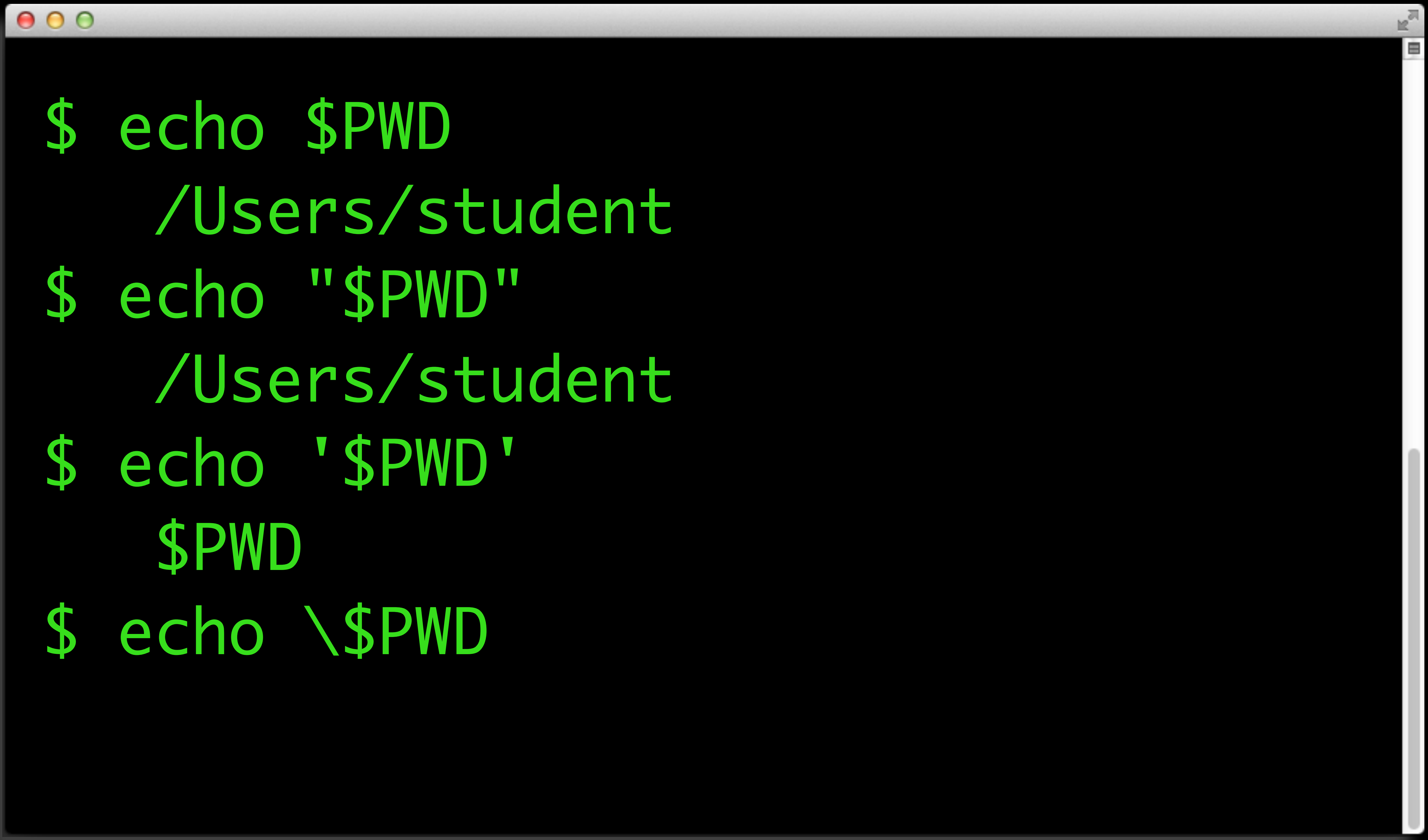
```
$ echo $PWD
   /Users/student
$ echo "$PWD"
   /Users/student
$
```



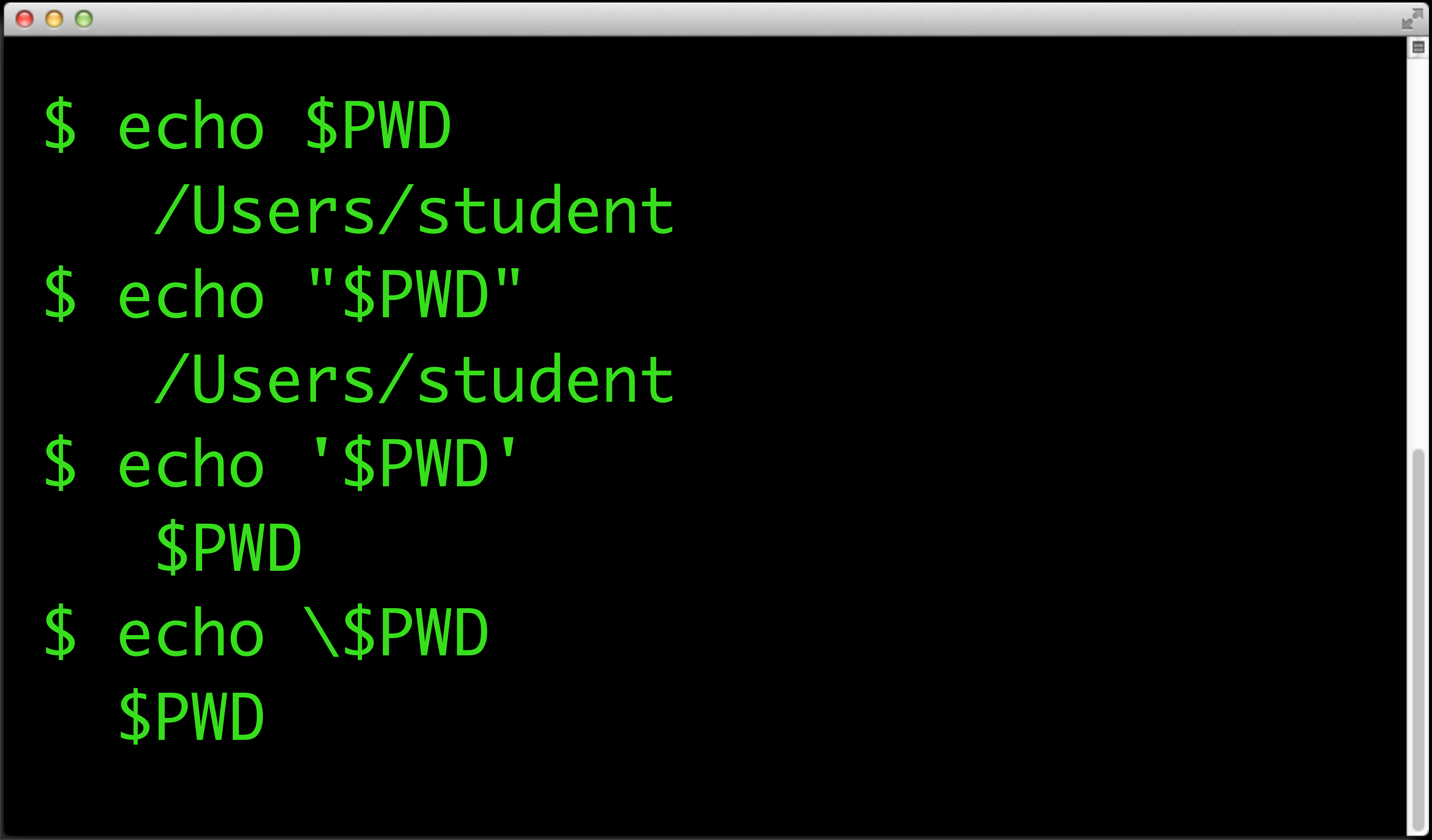
```
$ echo $PWD
   /Users/student
$ echo "$PWD"
   /Users/student
$ echo '$PWD'
```



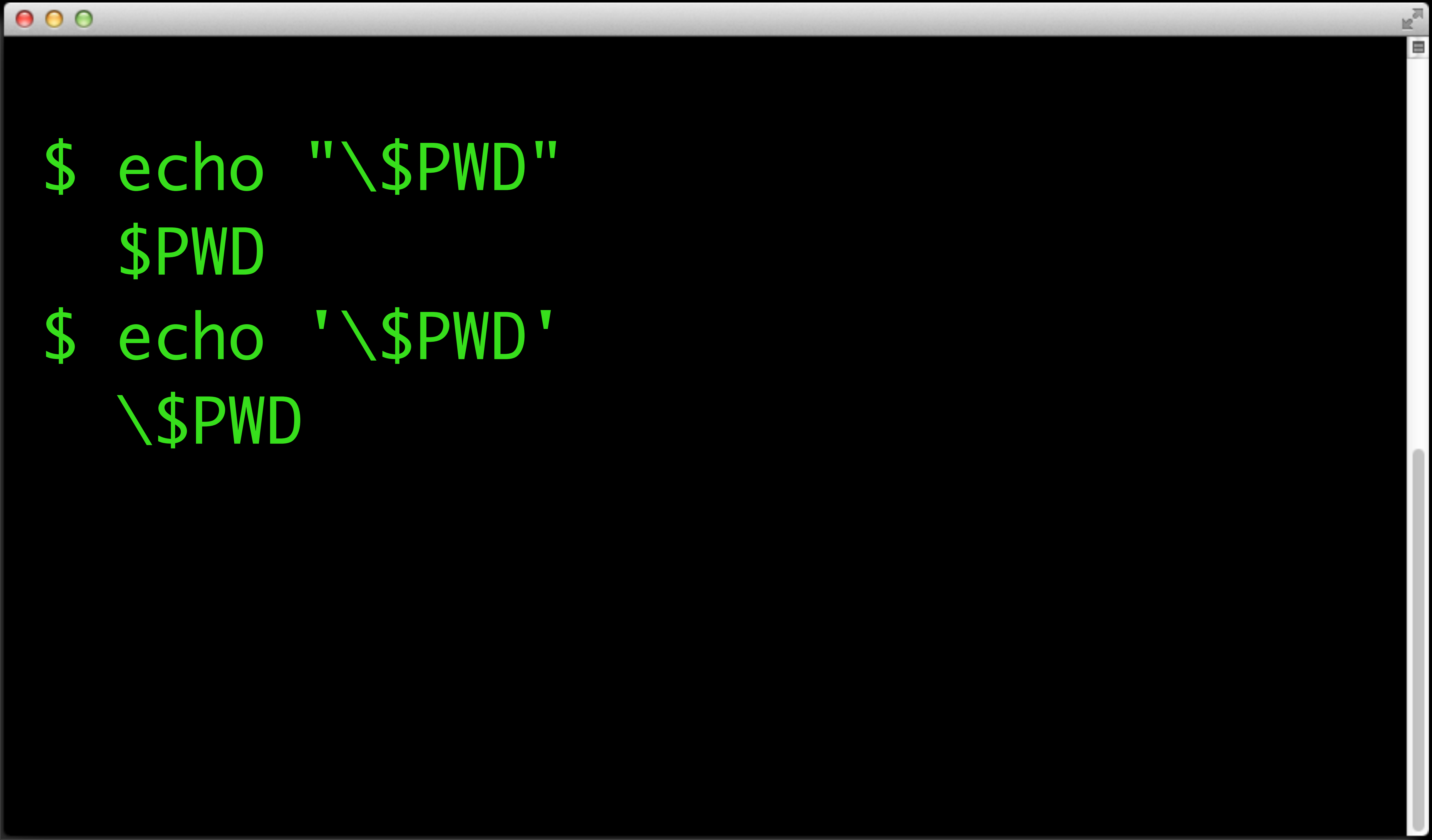
```
$ echo $PWD
   /Users/student
$ echo "$PWD"
   /Users/student
$ echo '$PWD'
   $PWD
$
```



```
$ 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
```



```
$ echo "\$PWD"
  $PWD
$ echo '\$PWD'
  \$PWD
```

# Try It!

- echo $PWD, $PATH, or $USER
- Try single/double quotes
- Try special characters
  ' " ~ * ? \ [ ]
- Name Grouping

# Shell Variables

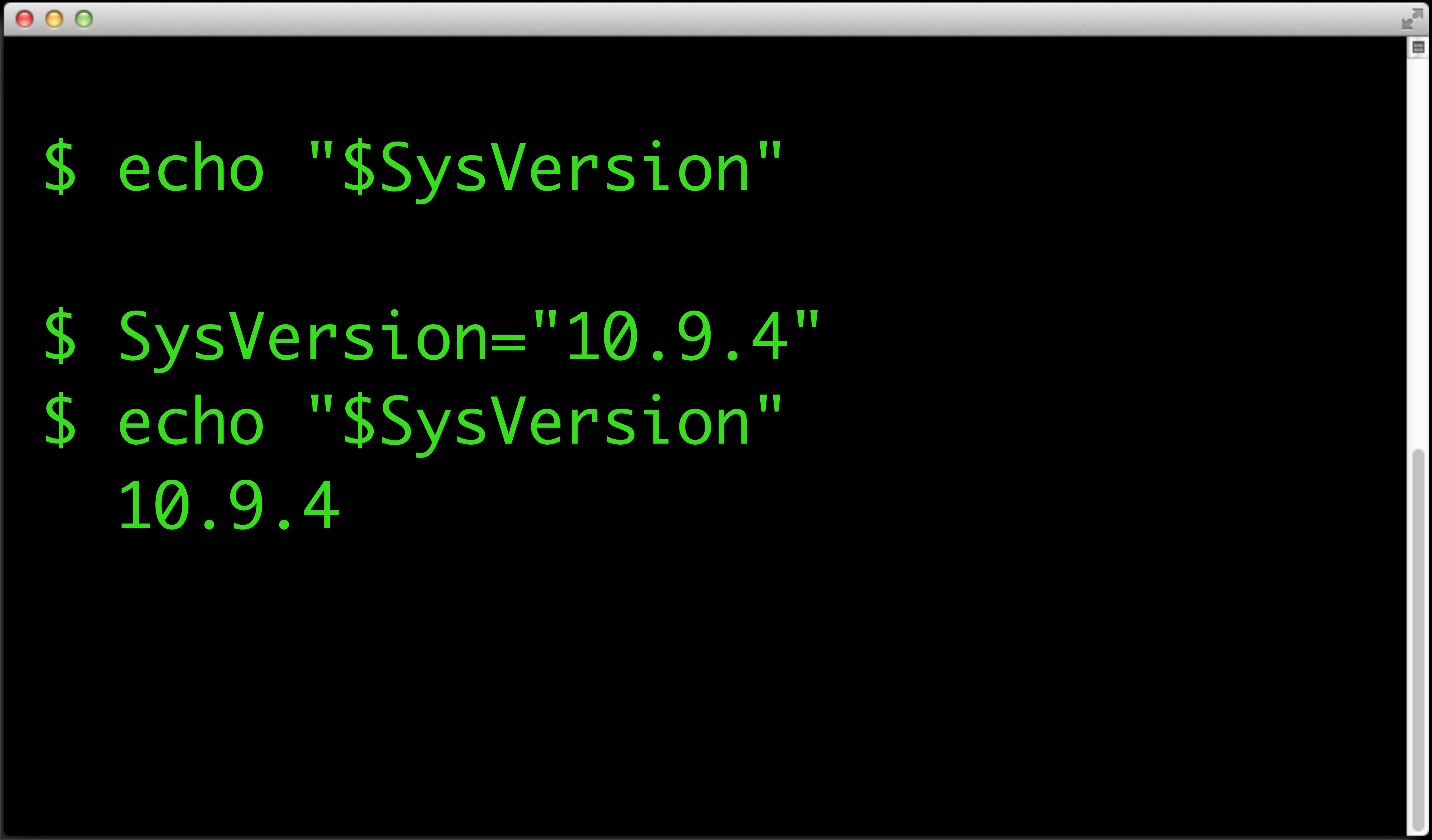
- echo $VARIABLE-NAME to show value
- run "env" to show current variables
    - Present Working Directory: $PWD
    - Current User: $USER
    - Current Shell: $SHELL
    - Search Path for commands: $PATH

## Variables

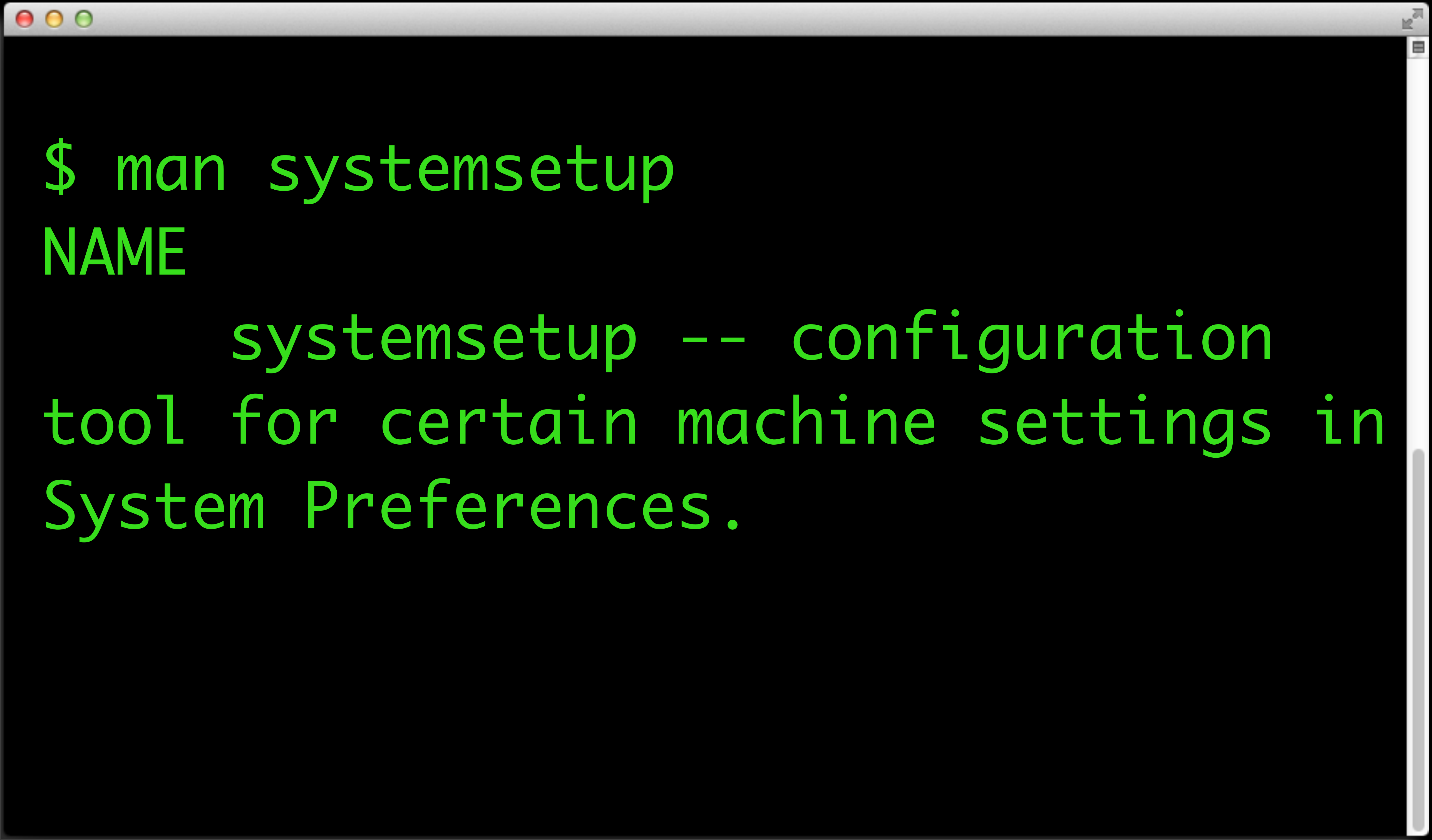
- At start of scripts
- Set with '='
  - VAR=10
- Precede Variable With '$' After Value Has Been Set
  - echo $VAR
  - Prints "10"

## unset

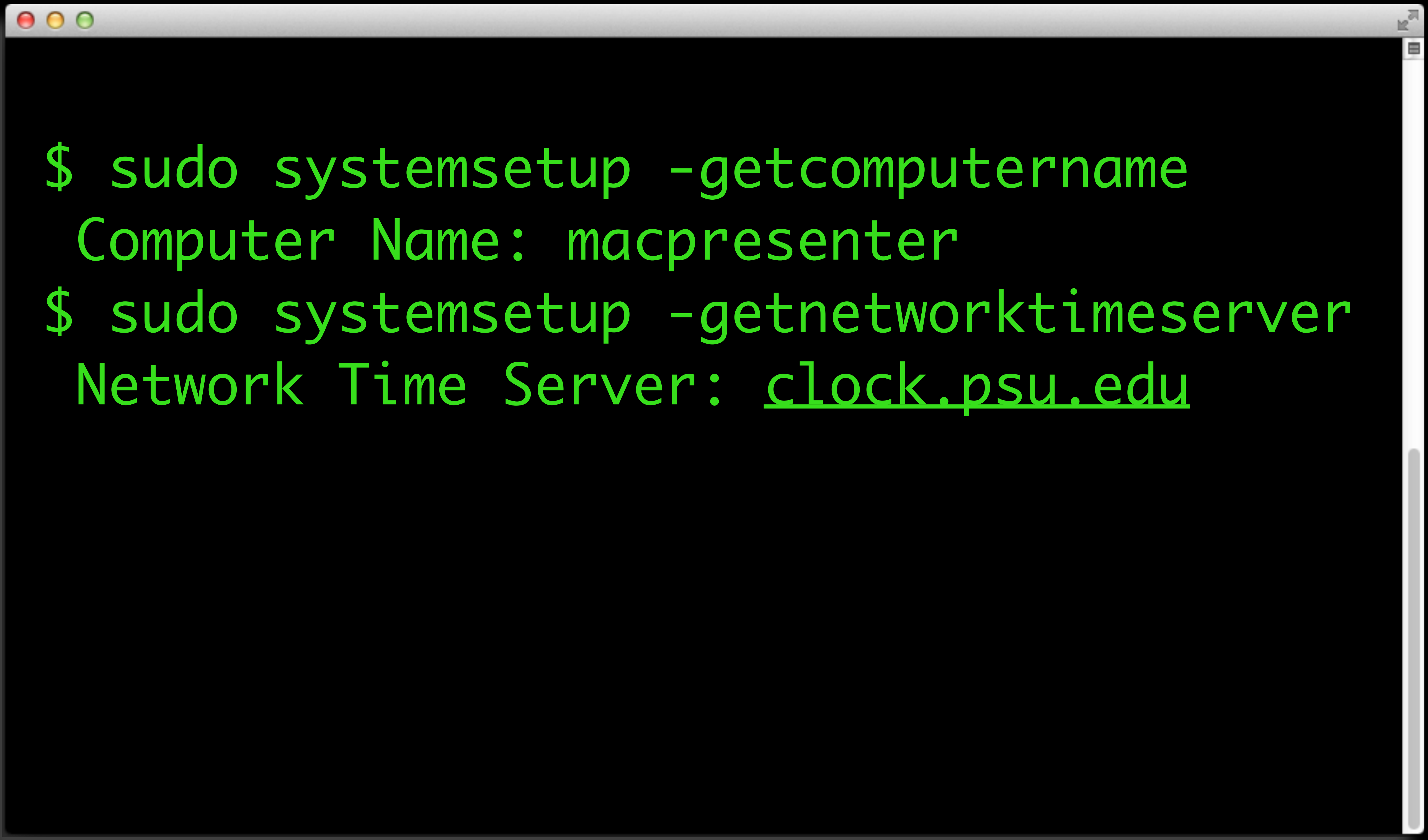
- Unset variables by name
- `unset argument1 argumentN`



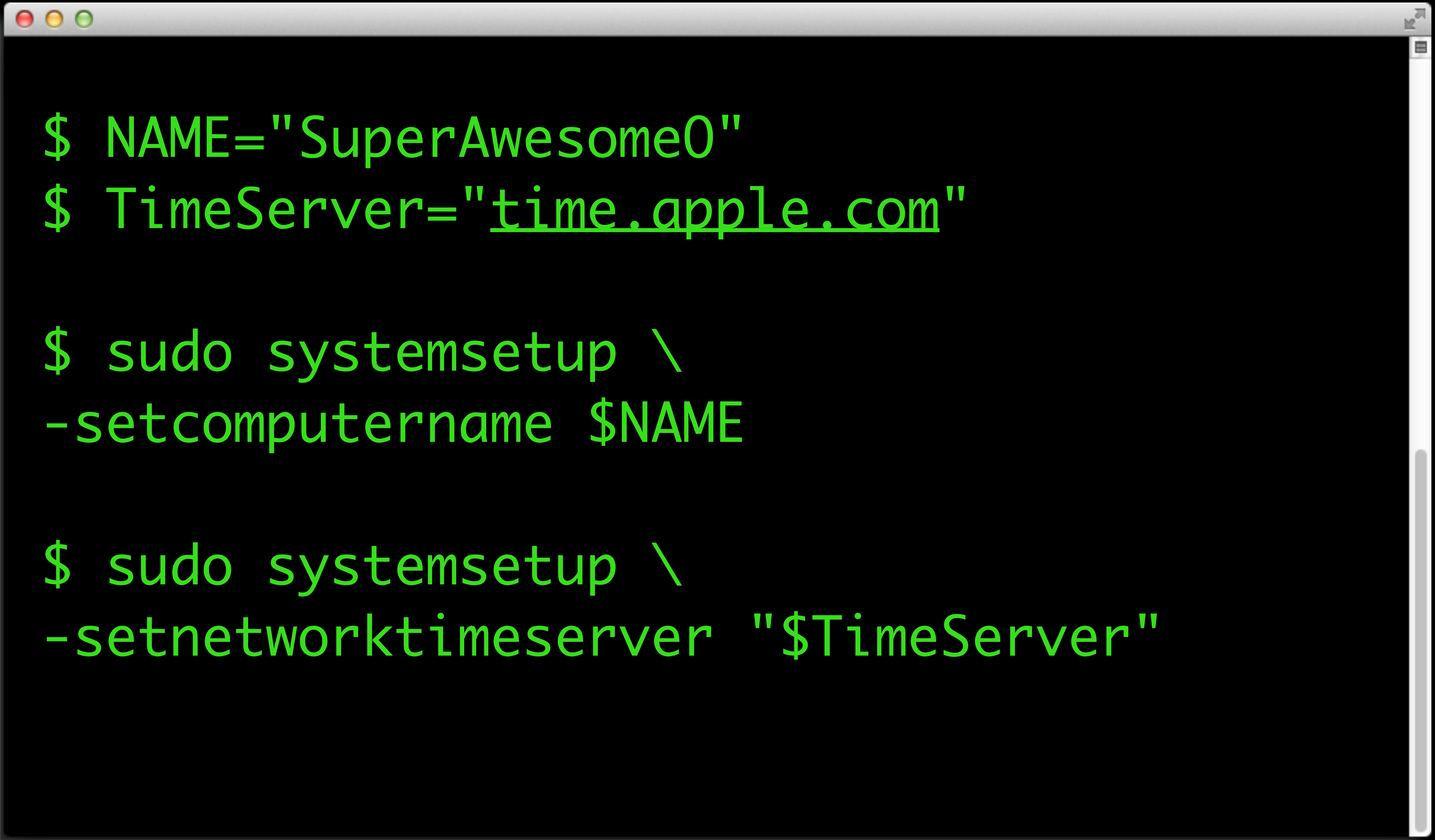
```
$ echo "$SysVersion"

$ SysVersion="10.9.4"
$ echo "$SysVersion"
  10.9.4
```



```
$ man systemsetup
NAME
     systemsetup -- configuration
tool for certain machine settings in
System Preferences.
```



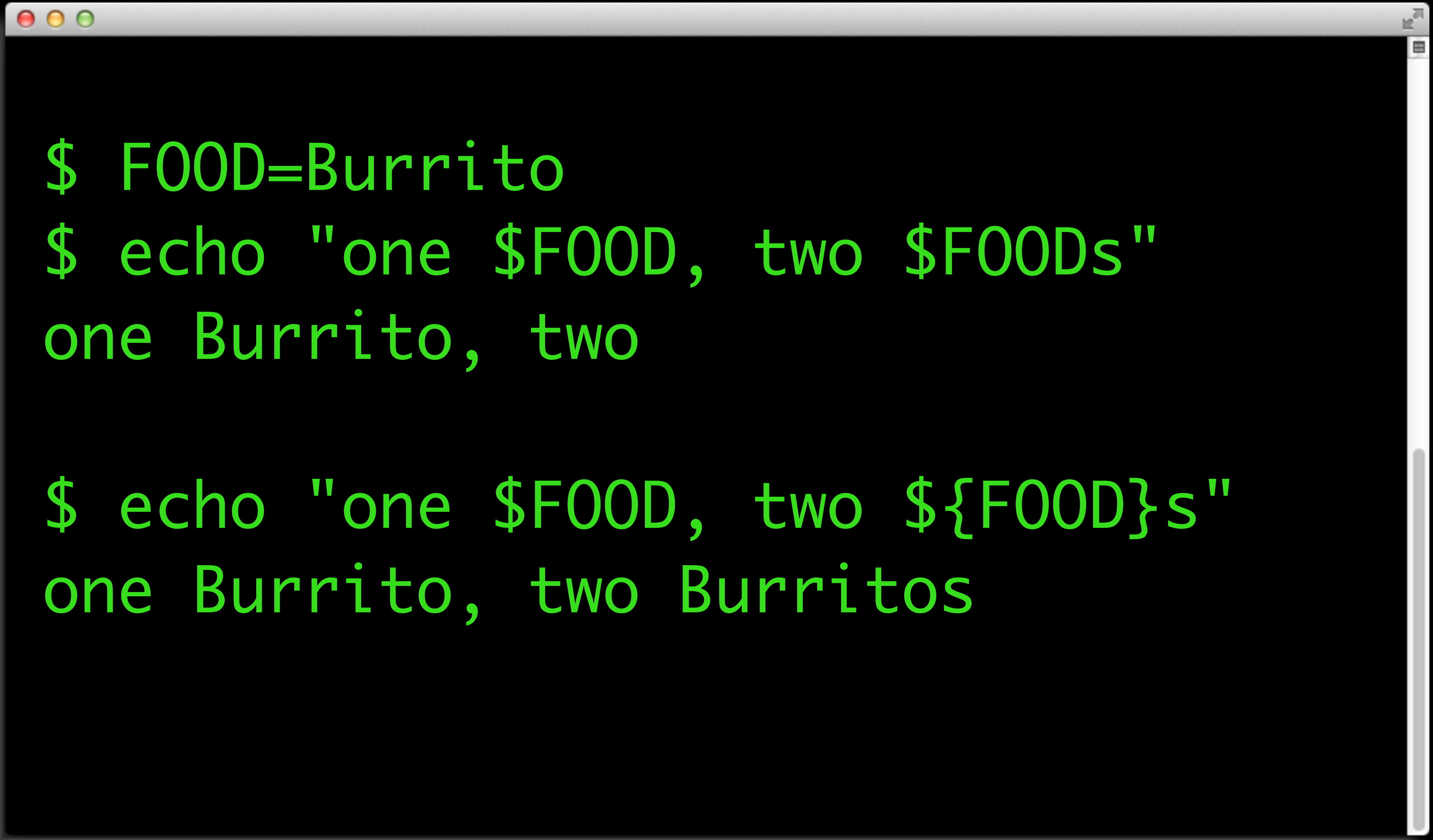
```
$ sudo systemsetup -getcomputername
 Computer Name: macpresenter
$ sudo systemsetup -getnetworktimeserver
 Network Time Server: clock.psu.edu
```



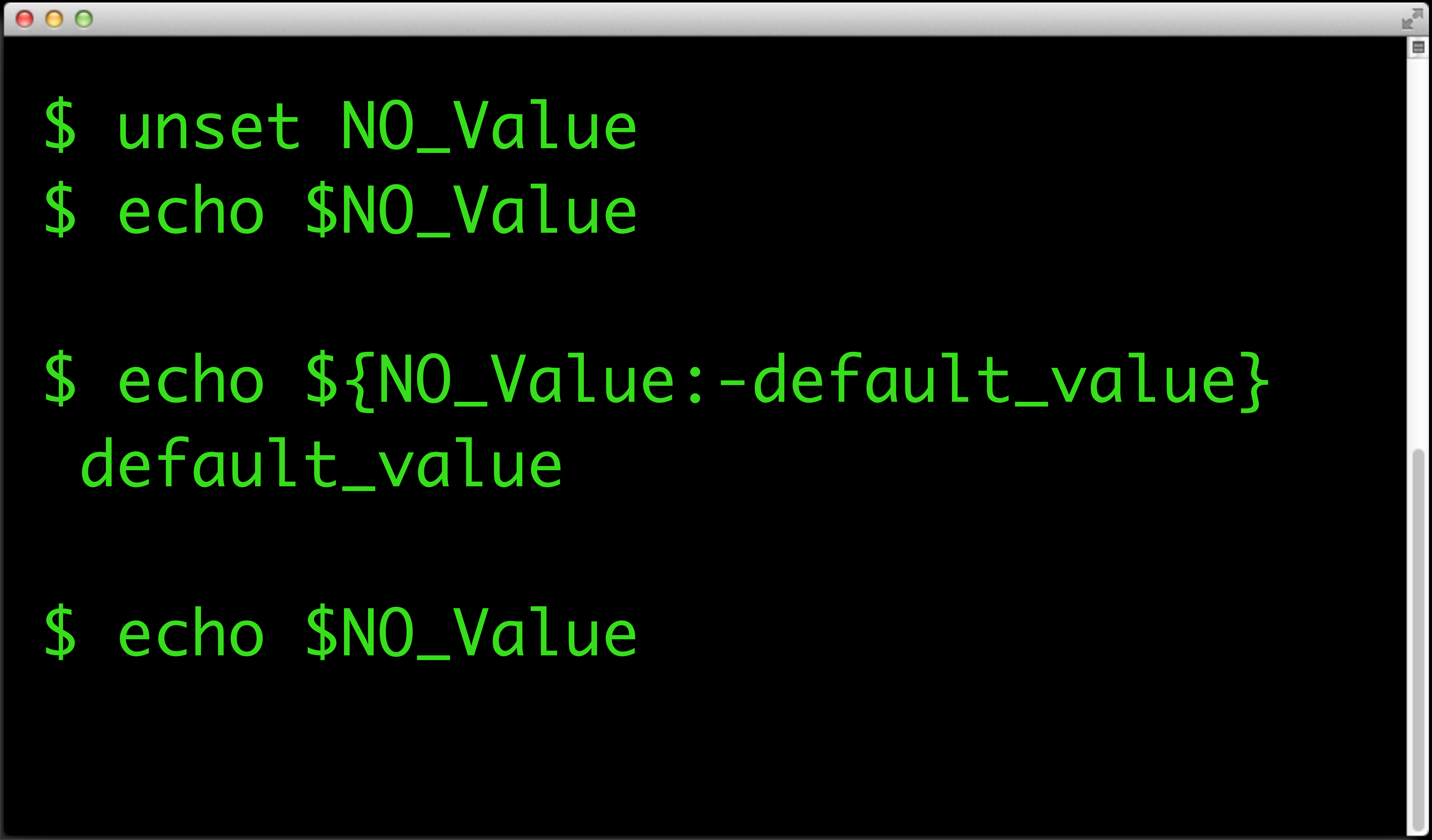
```
$ NAME="SuperAwesome0"
$ TimeServer="time.apple.com"

$ sudo systemsetup \
-setcomputername $NAME

$ sudo systemsetup \
-setnetworktimeserver "$TimeServer"
```

# Variable Expansion

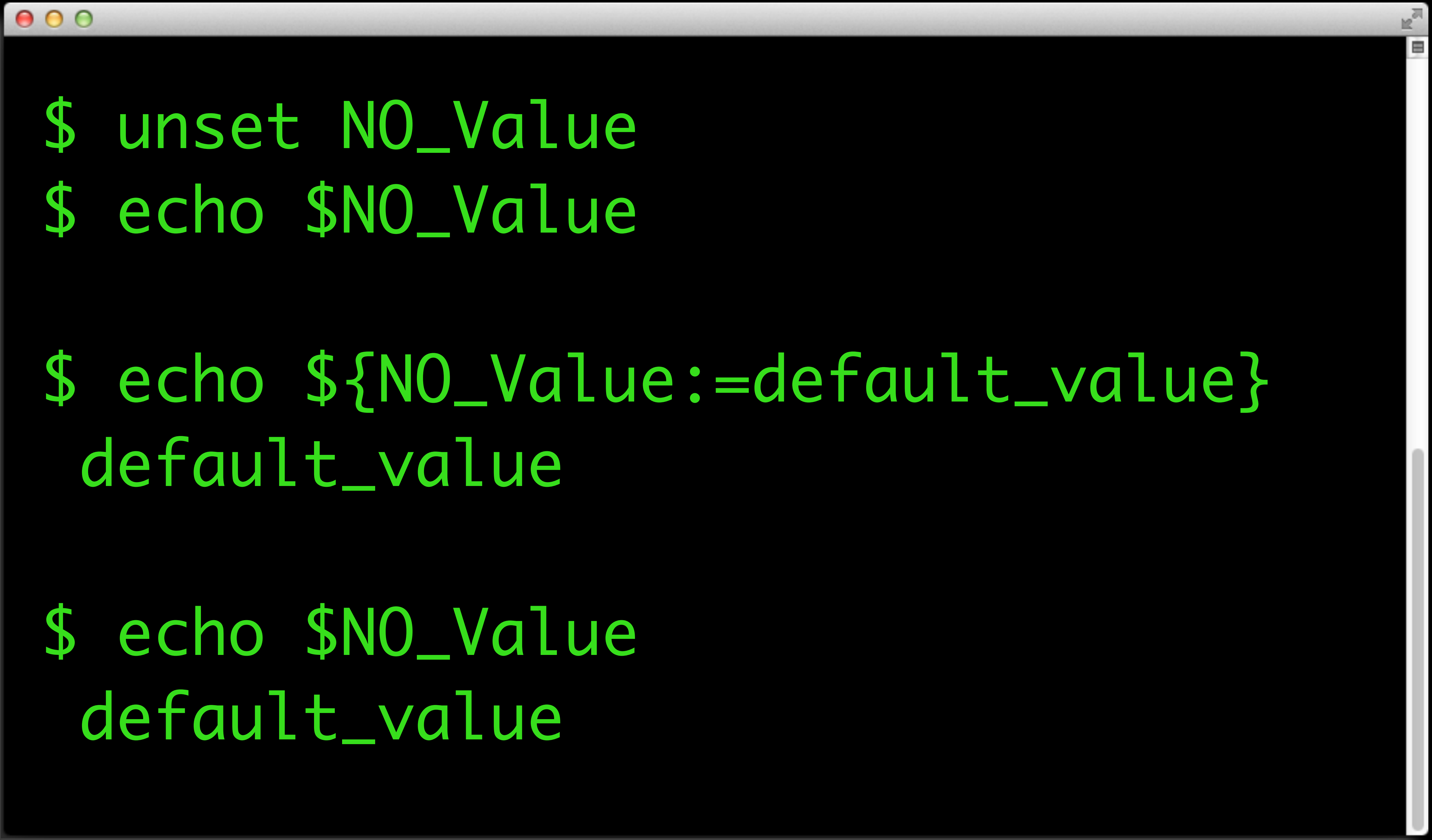
- Separate Variable from Text
  `${variable}`
- Print Default Value
  `${variable:-value}`
- Set Default Value
  `${variable:=value}`
- Error on unset variable
  `${variable:?message}`



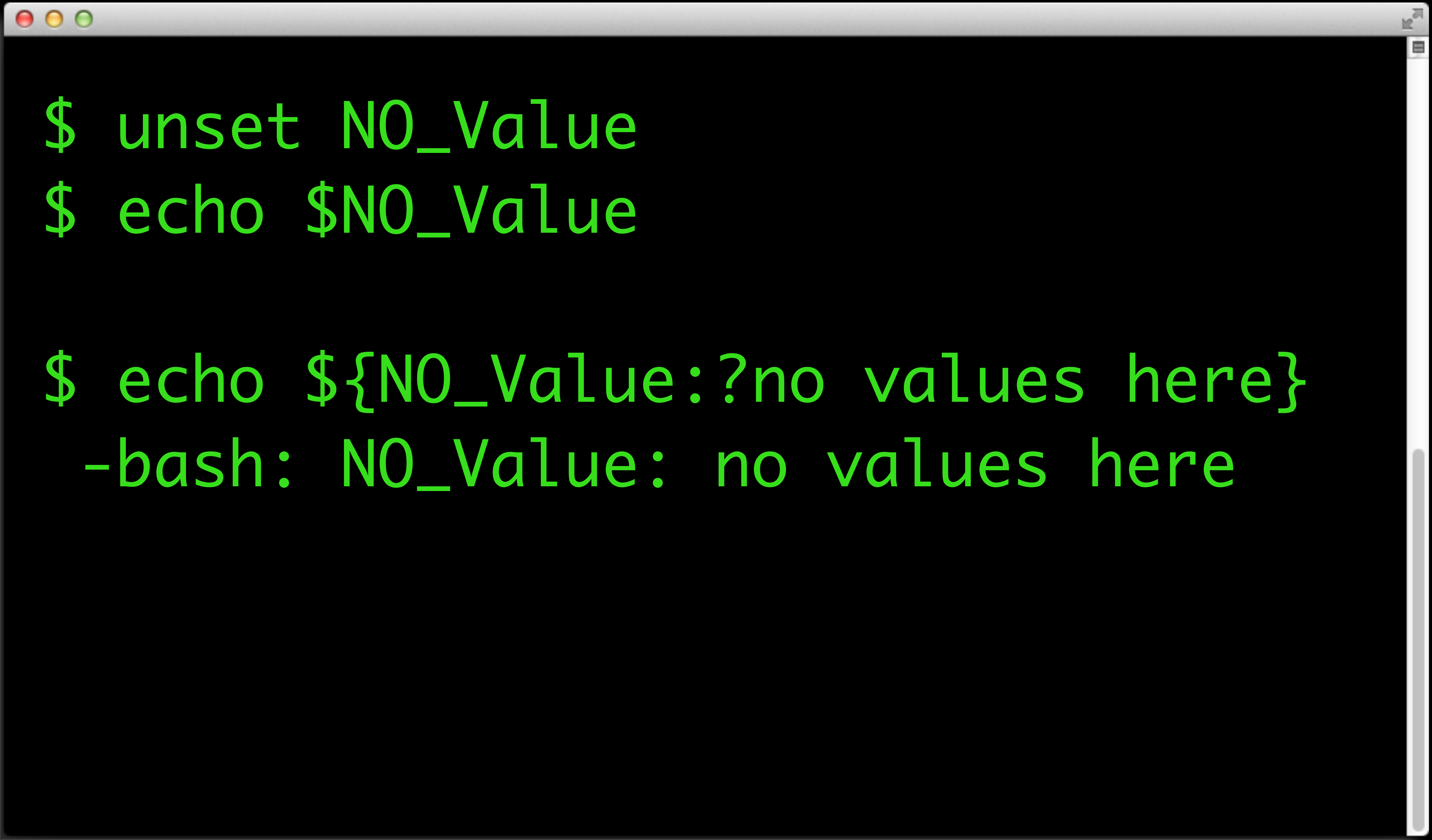
```
$ FOOD=Burrito
$ echo "one $FOOD, two $FOODs"
one Burrito, two

$ echo "one $FOOD, two ${FOOD}s"
one Burrito, two Burritos
```



```
$ unset NO_Value
$ echo $NO_Value

$ echo ${NO_Value:-default_value}
 default_value

$ echo $NO_Value
```



```
$ unset NO_Value
$ echo $NO_Value

$ echo ${NO_Value:=default_value}
 default_value

$ echo $NO_Value
 default_value
```



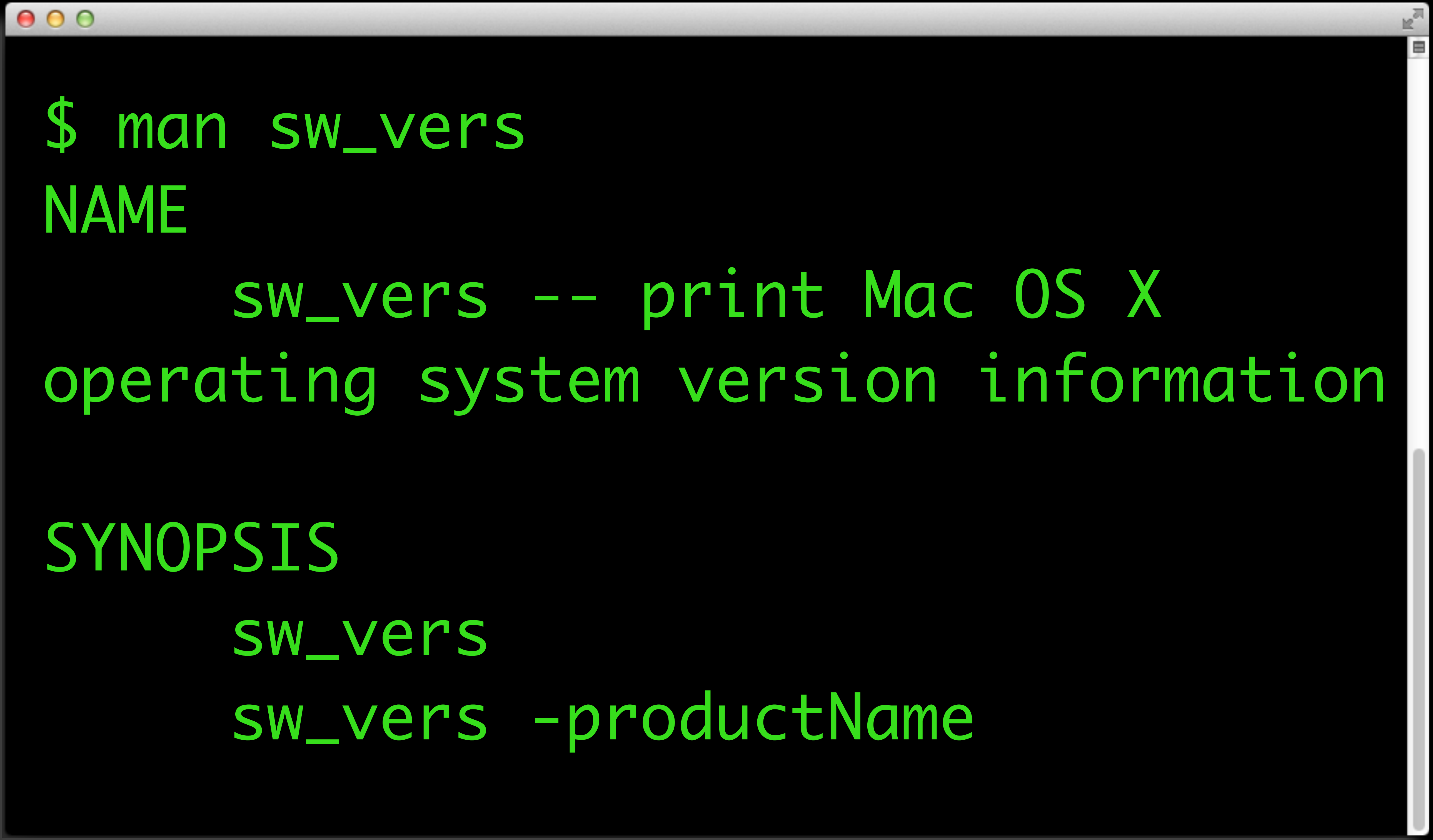
```
$ unset NO_Value
$ echo $NO_Value

$ echo ${NO_Value:?no values here}
 -bash: NO_Value: no values here
```

# Try It!

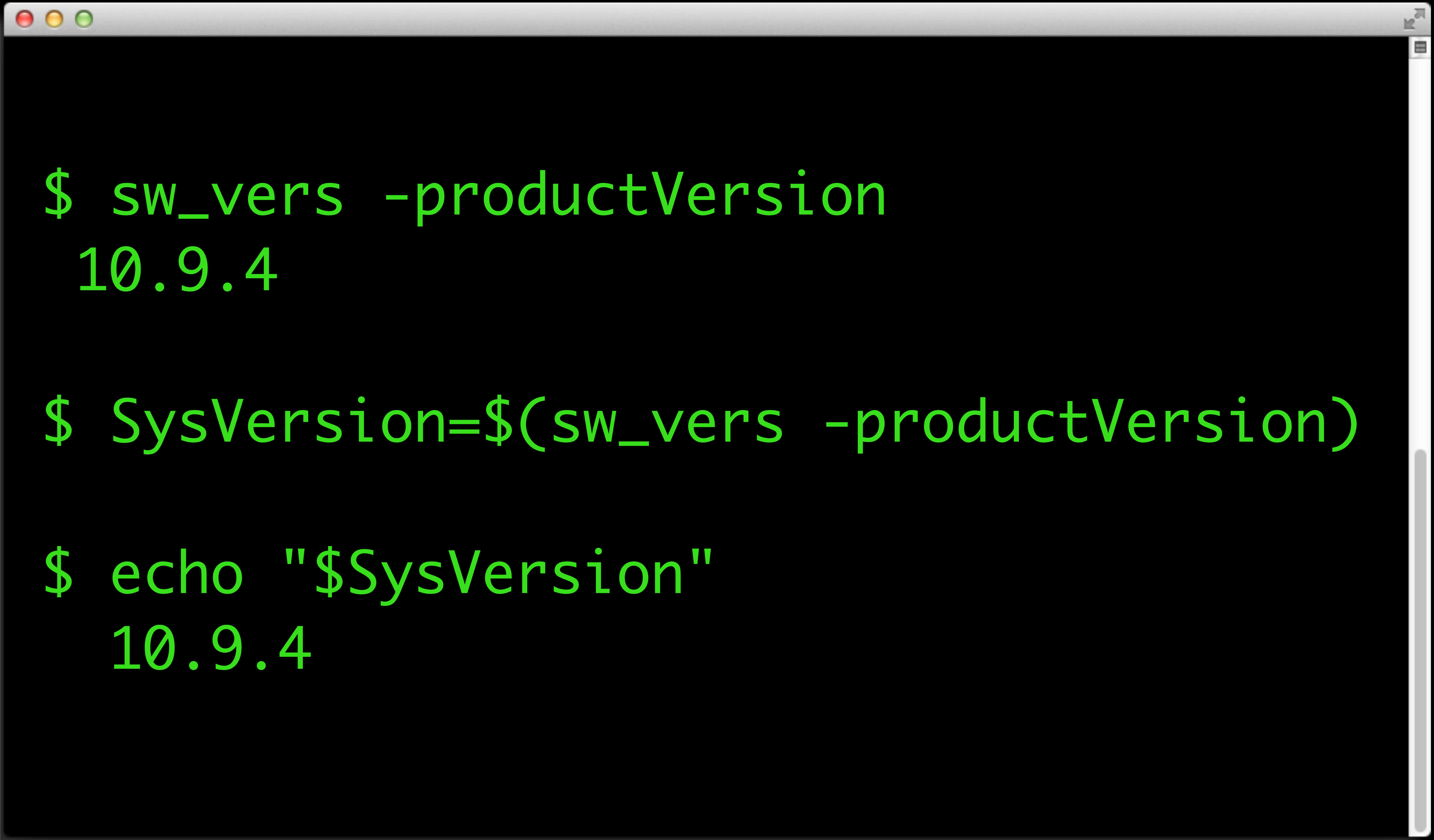
- Create New Variable
- Echo Variable
- Try with single/double Quotes

## Command Substitution

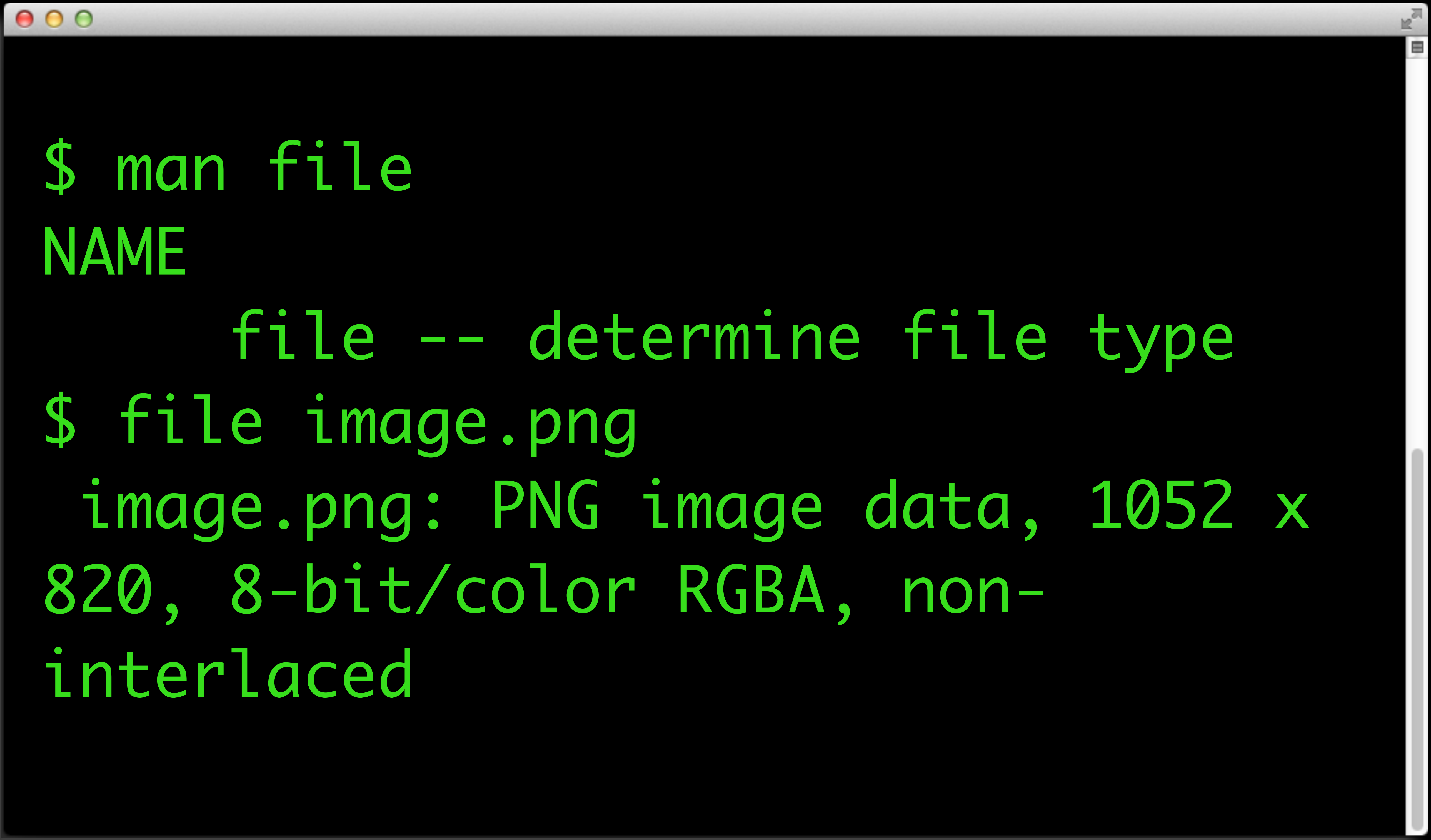
- Inserts output of one command into another
- `echo "$( commands )"`



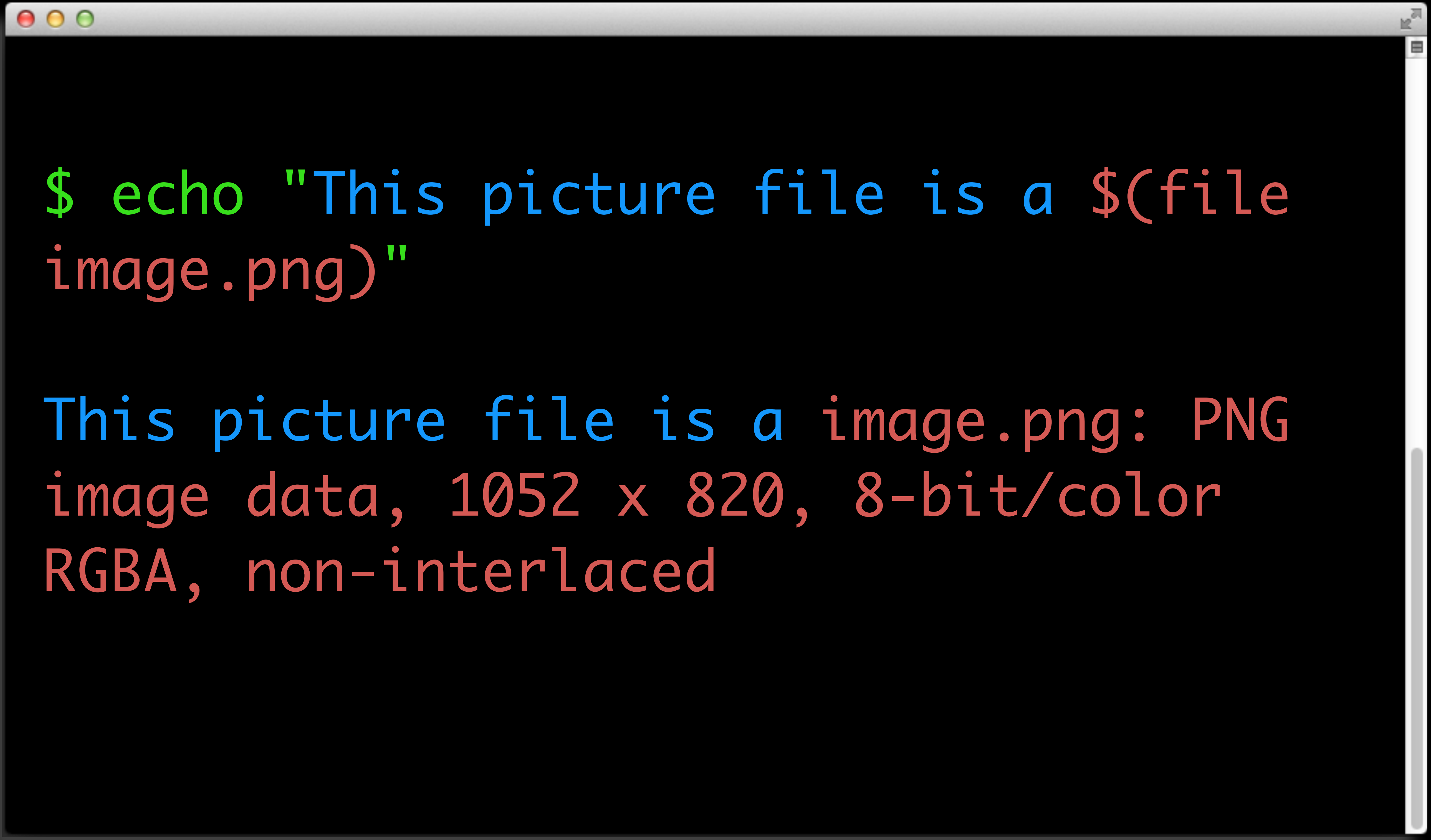
```
$ man sw_vers
NAME
     sw_vers -- print Mac OS X
operating system version information

SYNOPSIS
     sw_vers
     sw_vers -productName
```



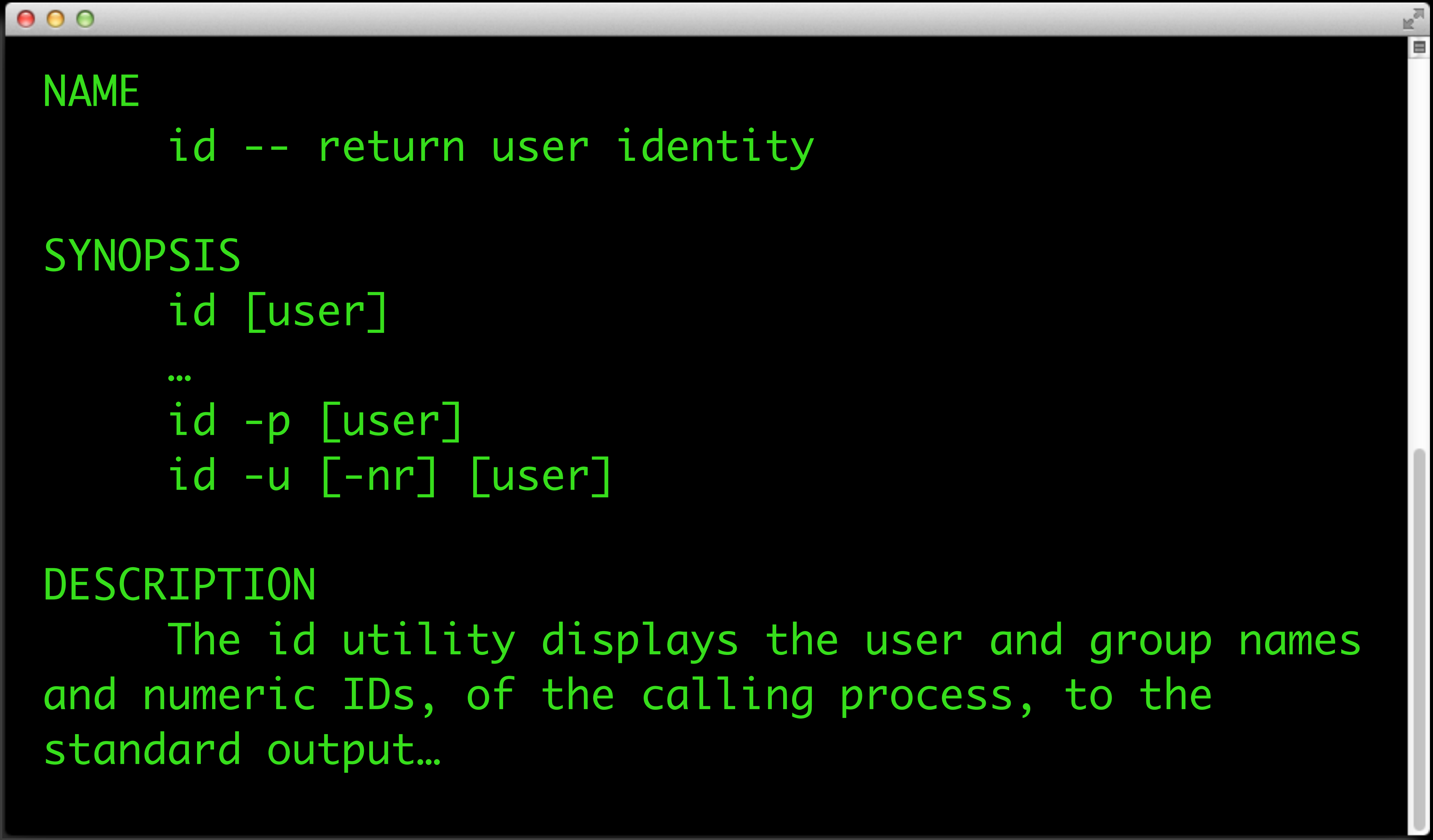
```
$ sw_vers -productVersion
 10.9.4

$ SysVersion=$(sw_vers -productVersion)

$ echo "$SysVersion"
  10.9.4
```



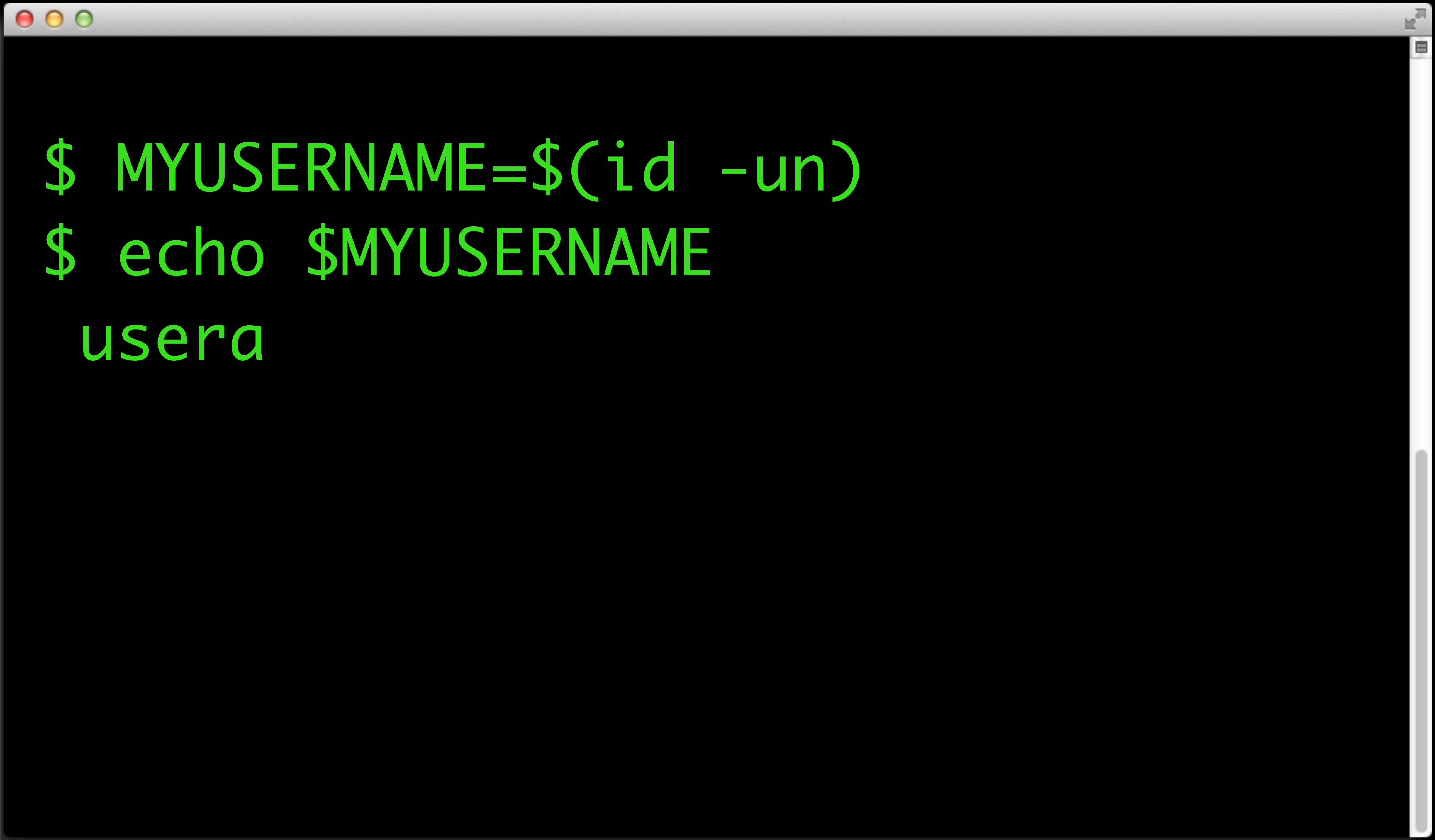
```
$ man file
NAME
     file -- determine file type
$ file image.png
 image.png: PNG image data, 1052 x
820, 8-bit/color RGBA, non-
interlaced
```



```
$ echo "This picture file is a $(file
image.png)"

This picture file is a image.png: PNG
image data, 1052 x 820, 8-bit/color
RGBA, non-interlaced
```

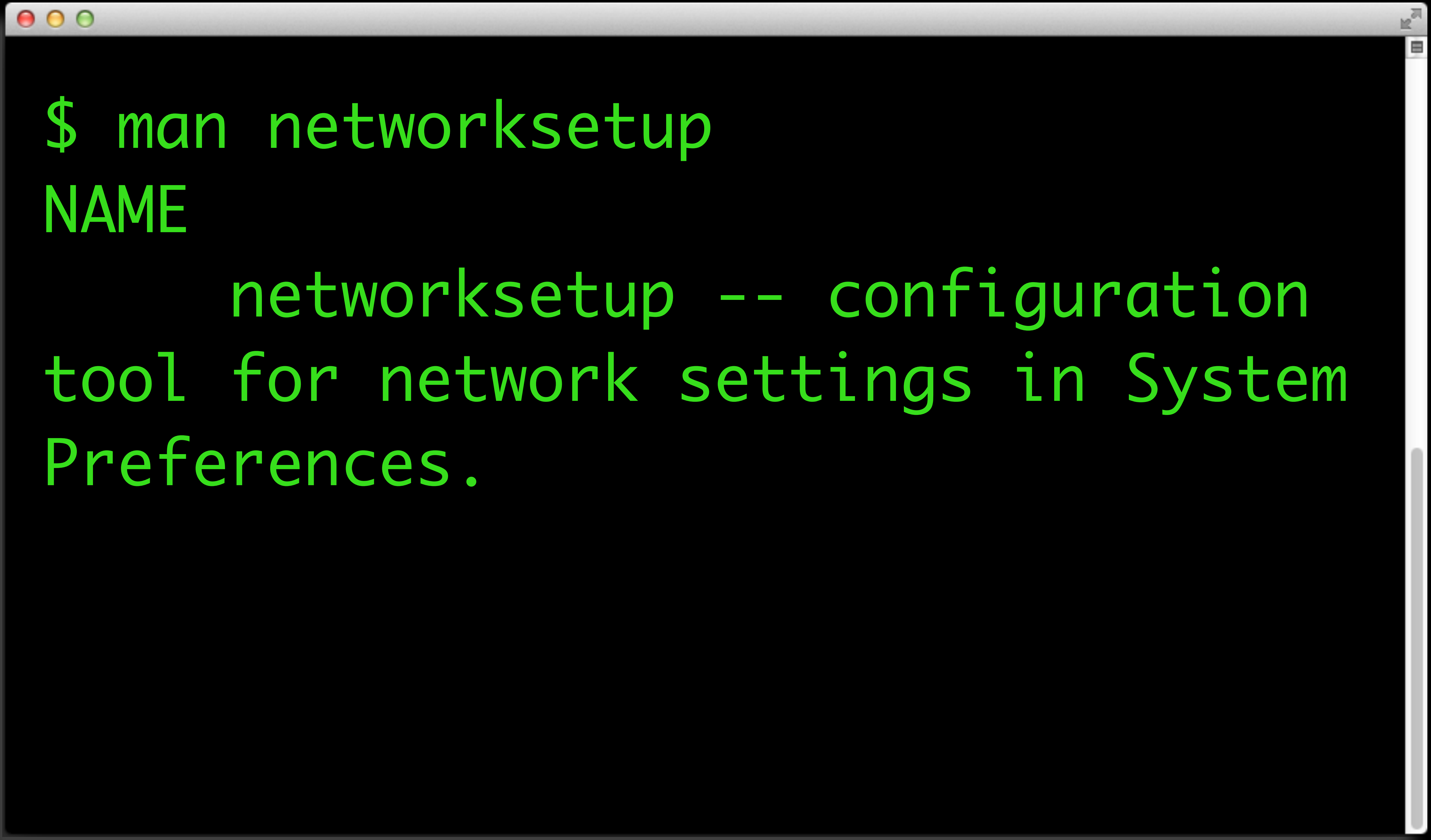


```
NAME
     id -- return user identity

SYNOPSIS
     id [user]
     …
     id -p [user]
     id -u [-nr] [user]

DESCRIPTION
     The id utility displays the user and group names
and numeric IDs, of the calling process, to the
standard output…
```

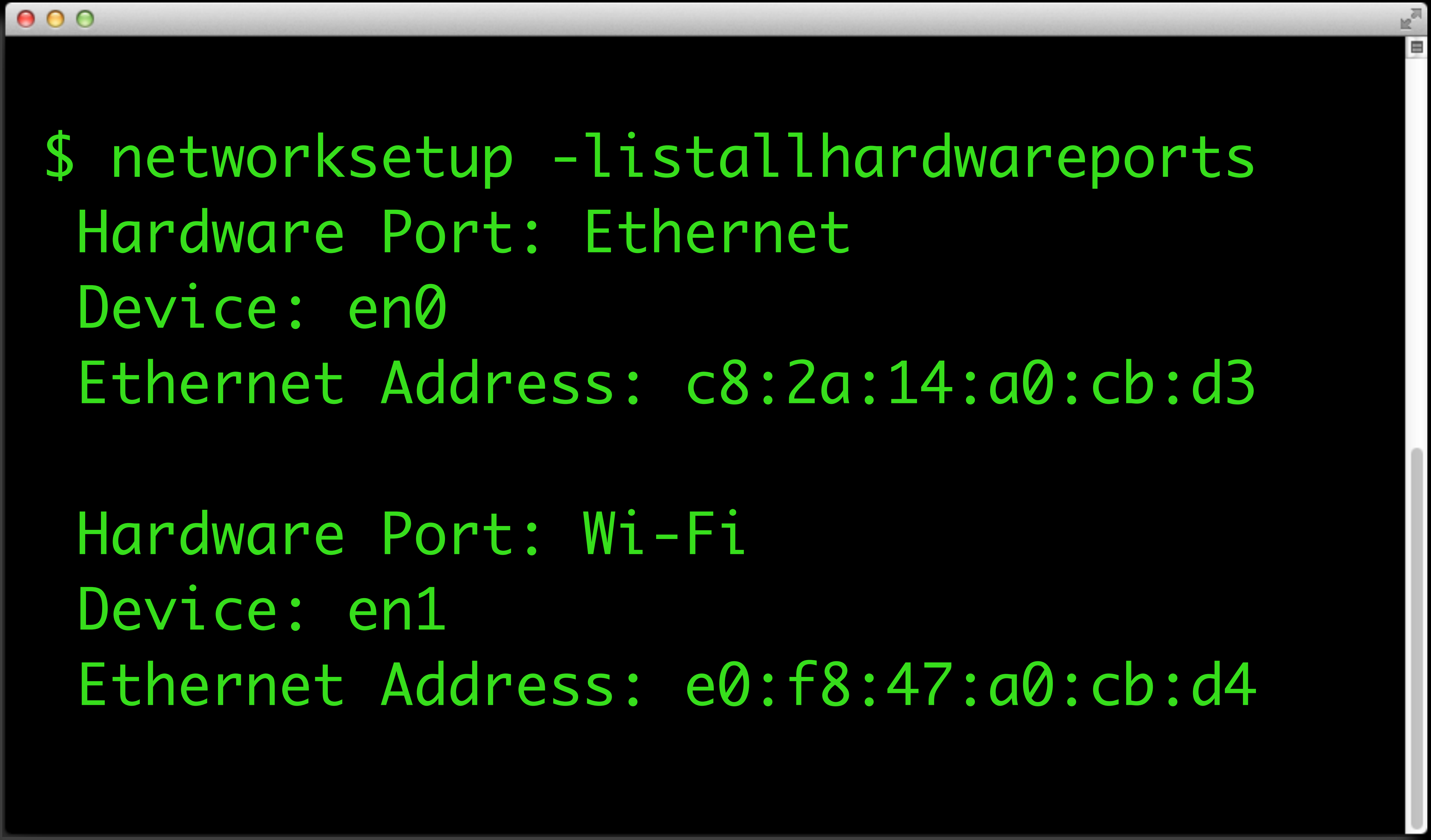


```
$ MYUSERNAME=$(id -un)
$ echo $MYUSERNAME
 usera
```

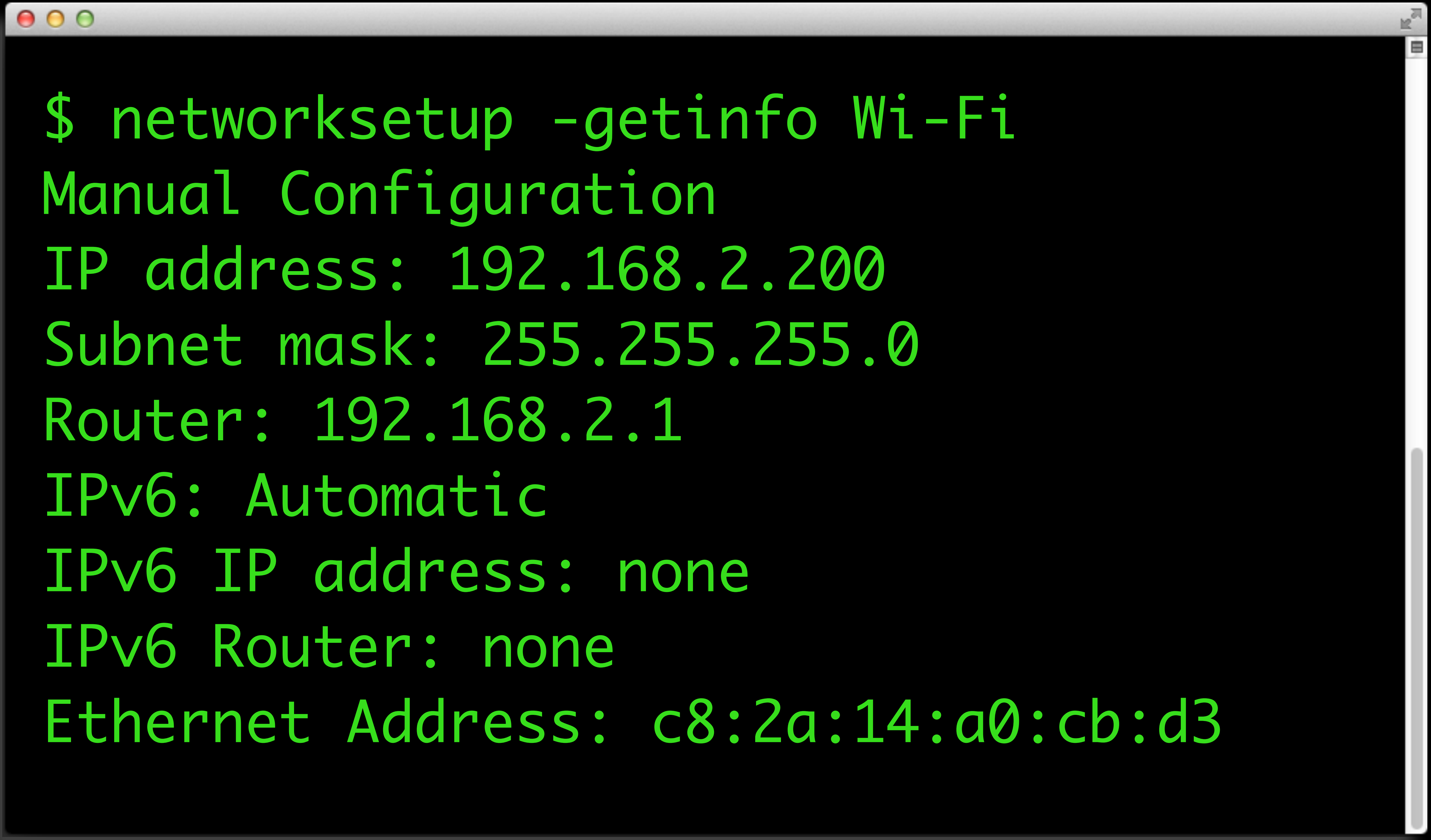
## Helpful OS X Command



```
$ man networksetup
NAME
     networksetup -- configuration
tool for network settings in System
Preferences.
```

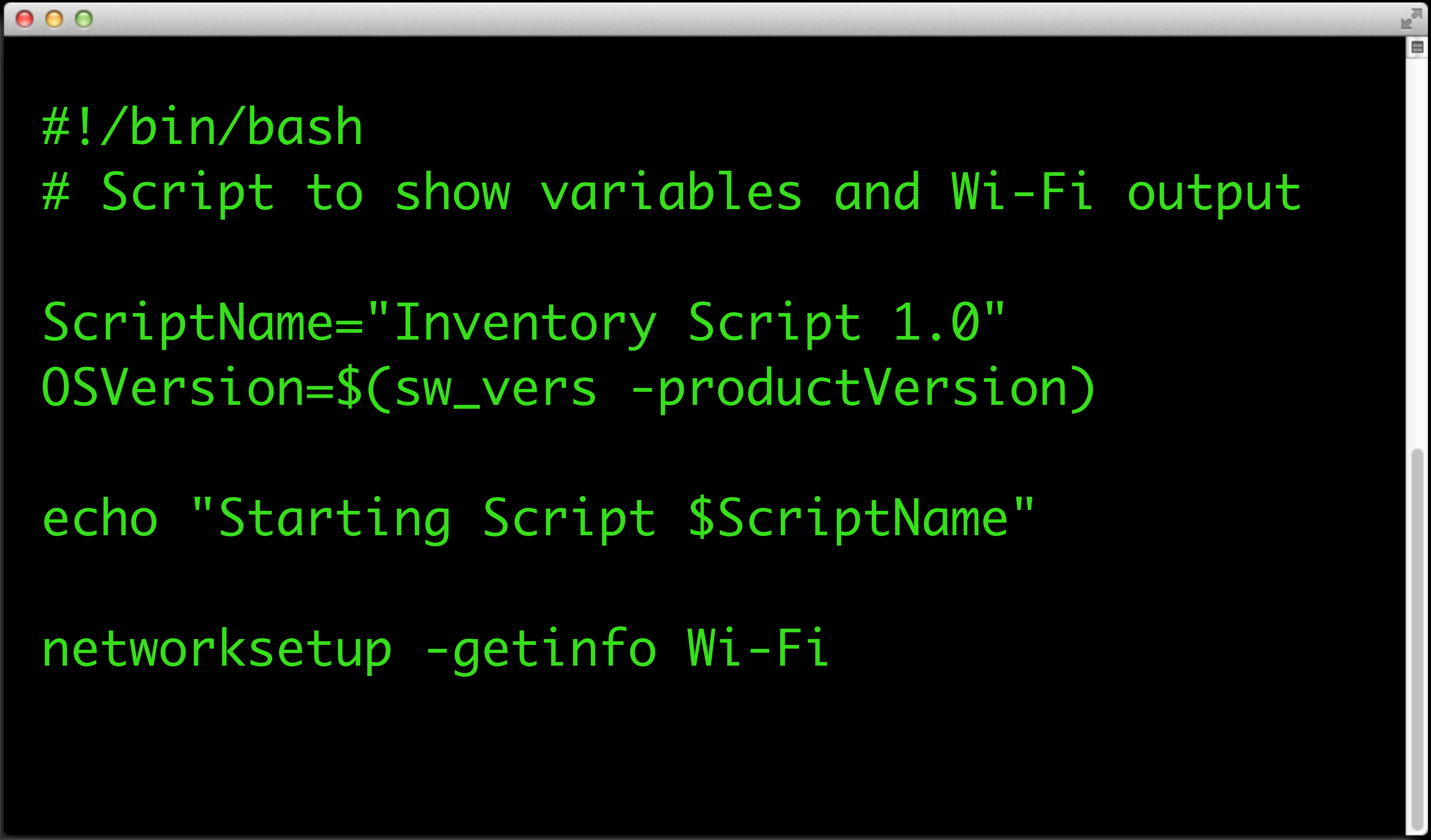


```
$ networksetup -listallhardwareports
 Hardware Port: Ethernet
 Device: en0
 Ethernet Address: c8:2a:14:a0:cb:d3

 Hardware Port: Wi-Fi
 Device: en1
 Ethernet Address: e0:f8:47:a0:cb:d4
```


```
$ networksetup -getinfo Wi-Fi
Manual Configuration
IP address: 192.168.2.200
Subnet mask: 255.255.255.0
Router: 192.168.2.1
IPv6: Automatic
IPv6 IP address: none
IPv6 Router: none
Ethernet Address: c8:2a:14:a0:cb:d3
```

## Try It!

- Create New inventory.sh
  - Use a hard coded and programmatically generated variable
  - Use networksetup to find WiFi IP address
  - Use echo to output what the script is doing
  - Test Script (Don't Forget x Bit!)



```
#!/bin/bash
# Script to show variables and Wi-Fi output

ScriptName="Inventory Script 1.0"
OSVersion=$(sw_vers -productVersion)

echo "Starting Script $ScriptName"

networksetup -getinfo Wi-Fi
```

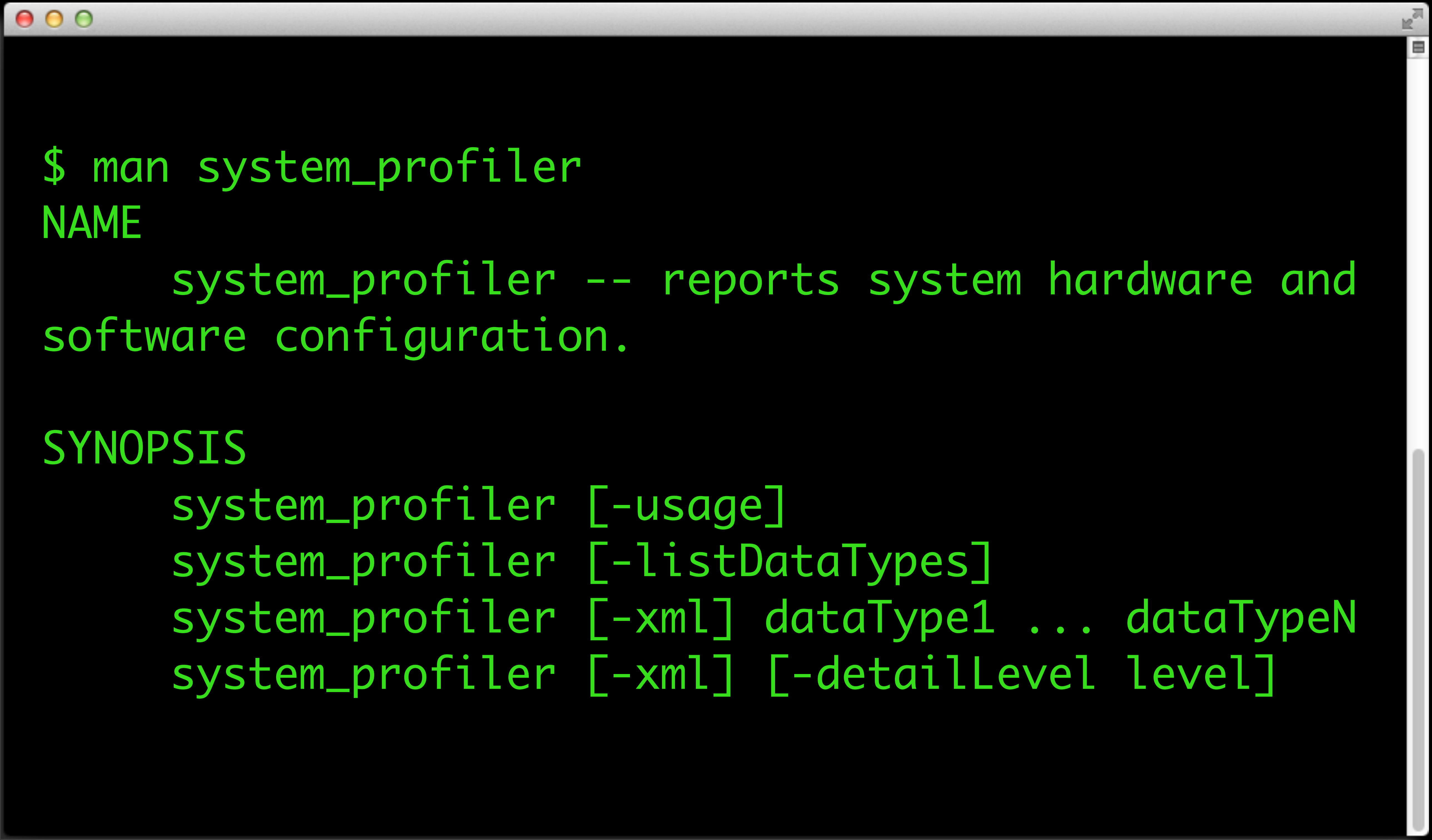
break

## Part 3

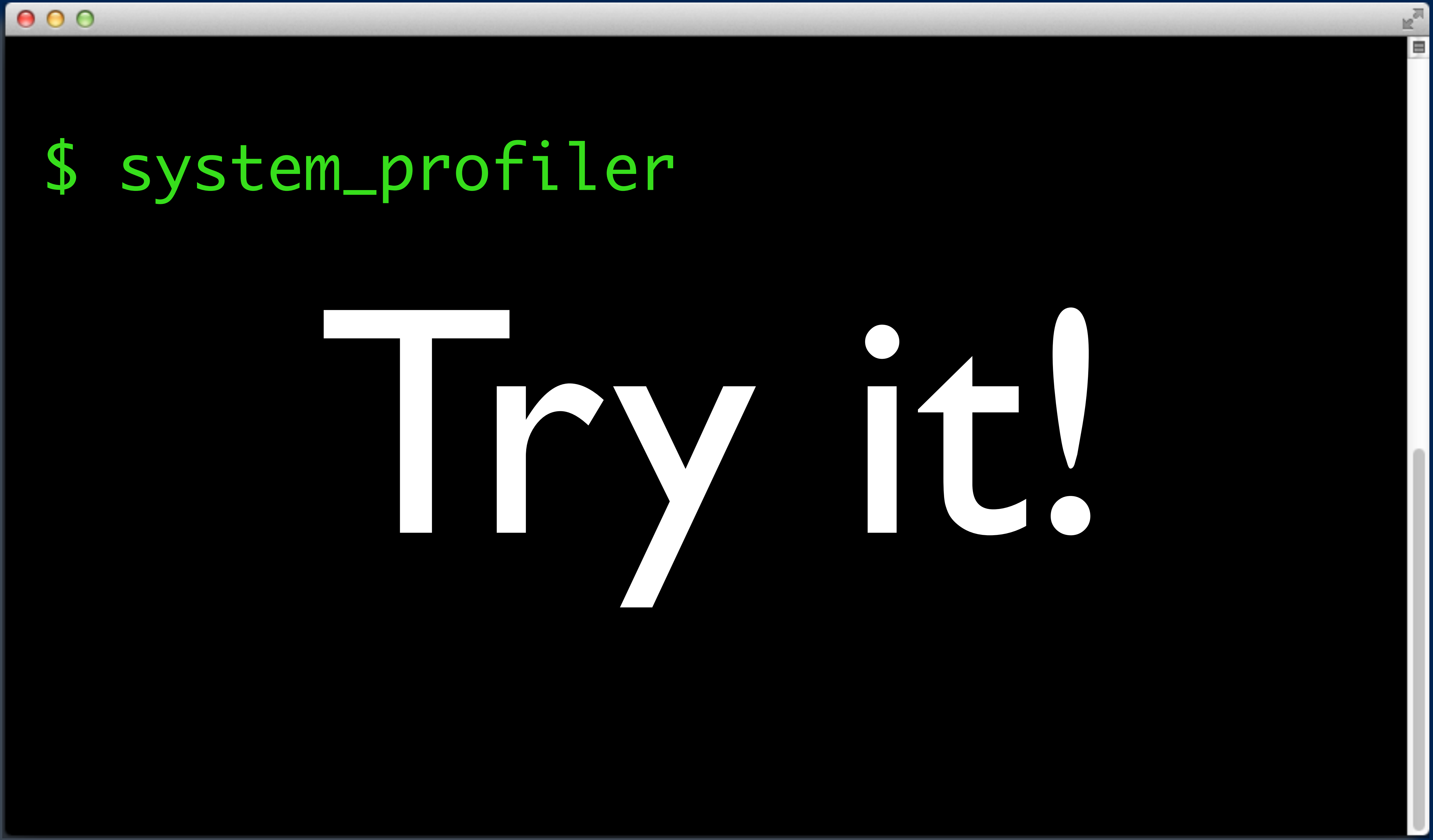
- Text Manipulation
- Piping
- Redirection
- Exit Values

## Controlling Text

- Command unwieldy output!
- Set Variables with output



```
$ man system_profiler
NAME
     system_profiler -- reports system hardware and
software configuration.

SYNOPSIS
     system_profiler [-usage]
     system_profiler [-listDataTypes]
     system_profiler [-xml] dataType1 ... dataTypeN
     system_profiler [-xml] [-detailLevel level]
```



```
$ system_profiler
```

Try it!

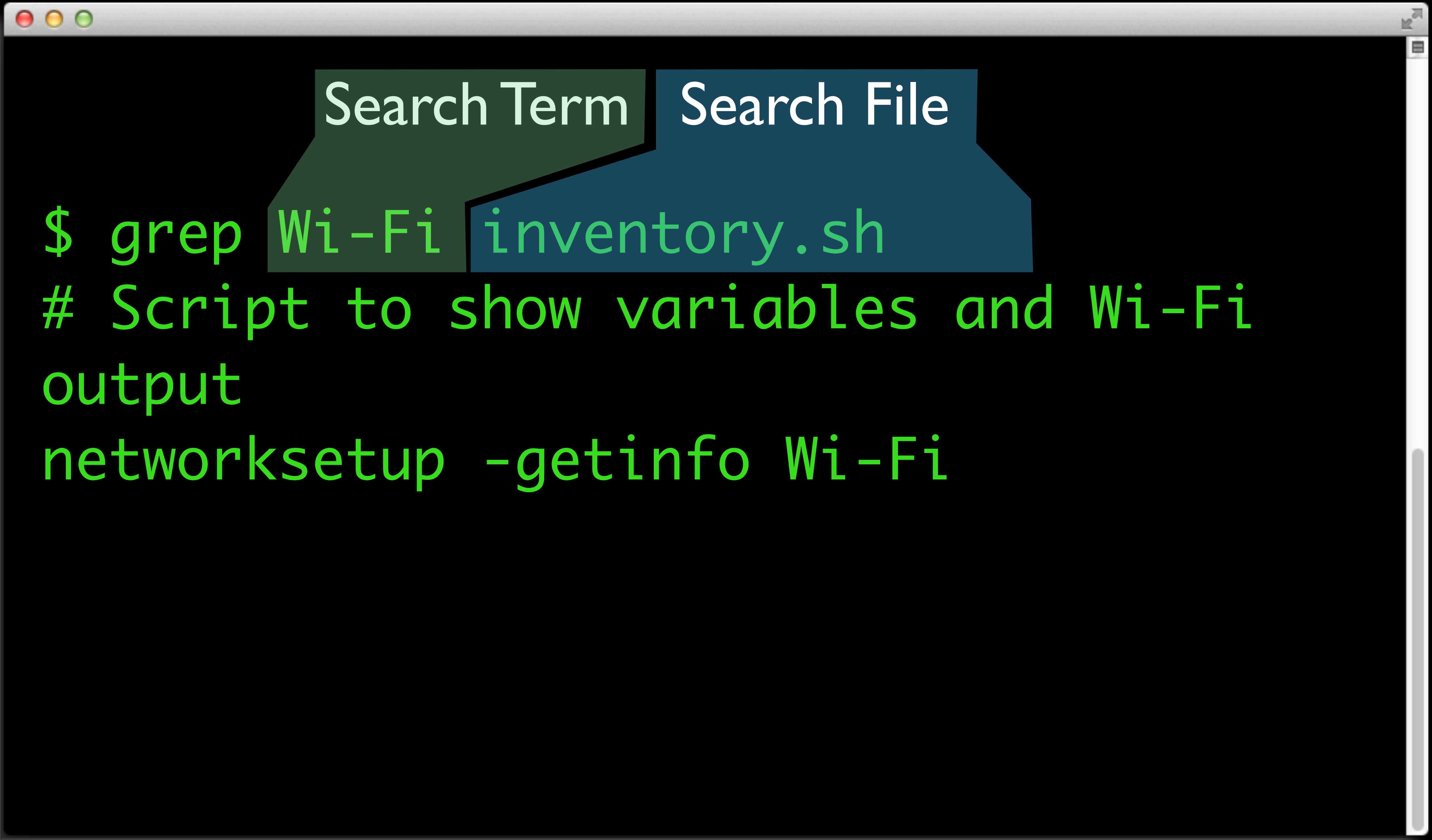
## Now, Control the Output!

## Grep

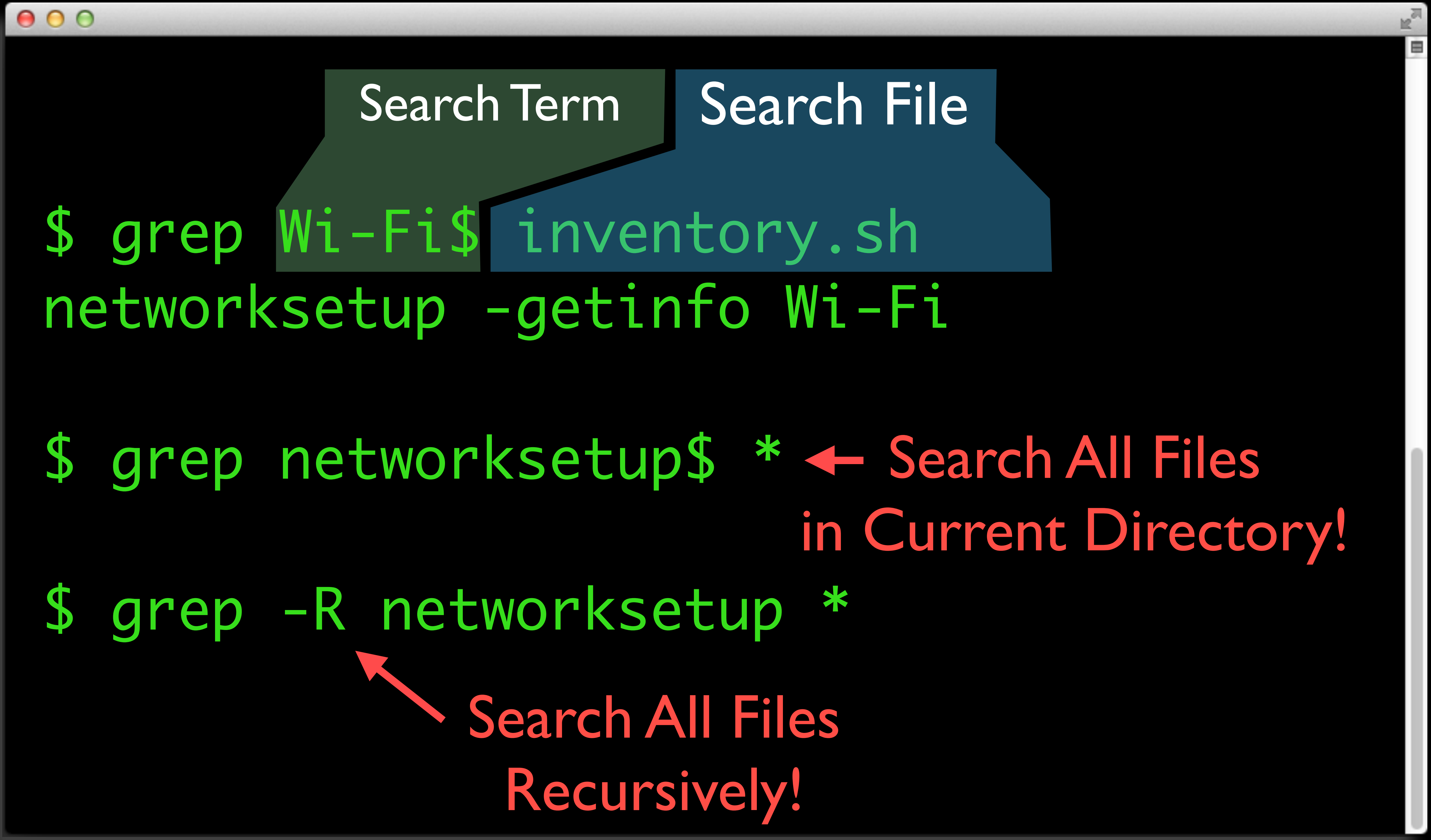
- Search & Match Patterns with RE
- Prints Match to stdout
- Ignore Case: -i
- Print 5 Lines After Match: -A5
- Print 5 Lines Before Match: -B5

## Basic Regular Expressions (RE/RegExp)

- Interpreted by certain programs
- Beginning of Line: ^char
- End of Line: char$
- Any Char Except New Line: .
- Group of Char: [abc]



```
$ grep Wi-Fi inventory.sh
# Script to show variables and Wi-Fi
output
networksetup -getinfo Wi-Fi
```


Search Term
Search File
Search All Files in Current Directory!
Search All Files Recursively!

```
$ grep Wi-Fi$ inventory.sh
networksetup -getinfo Wi-Fi

$ grep networksetup$ *

$ grep -R networksetup *
```

## Try It!

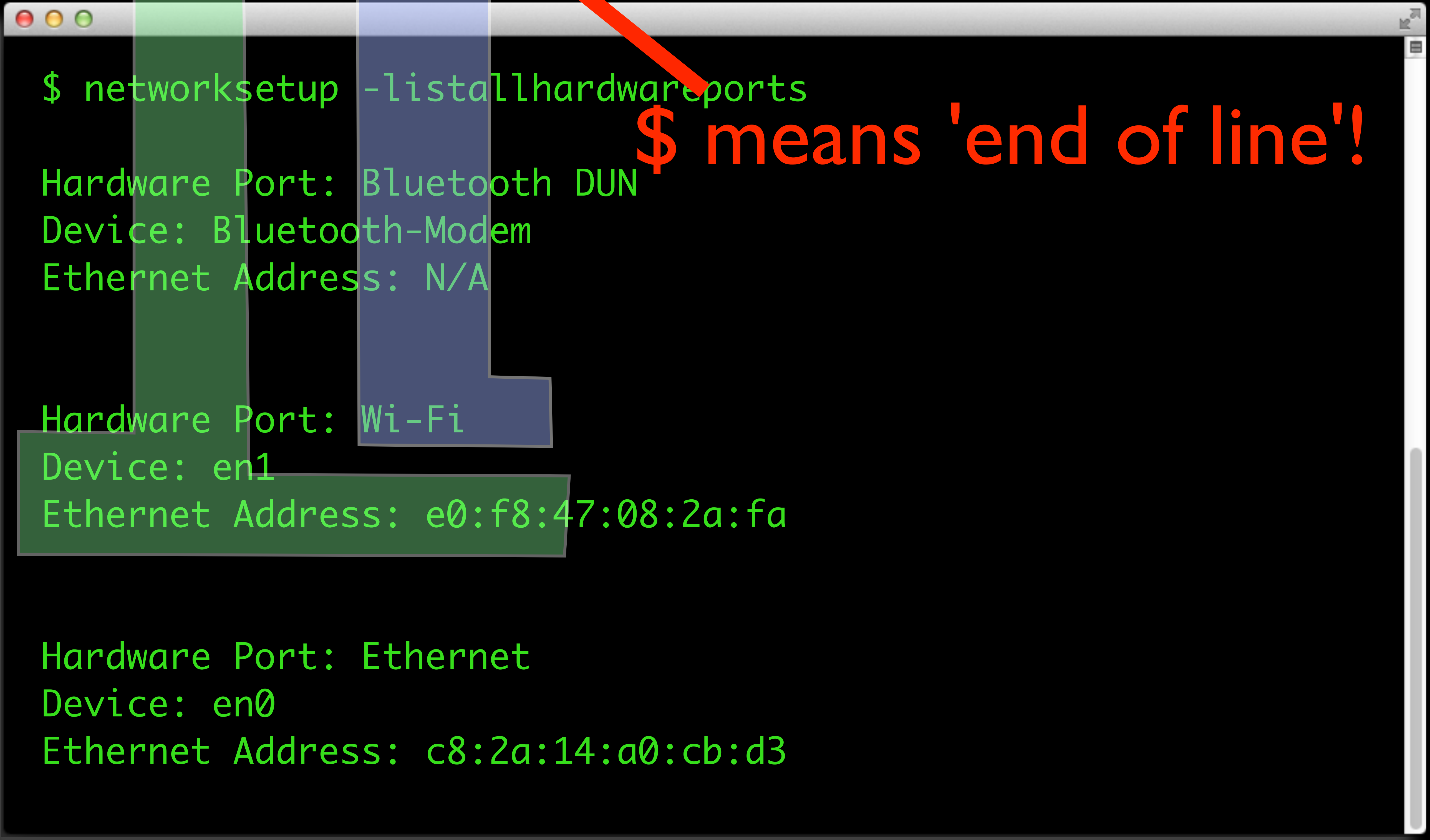
- Change Directory into PSUMAC2014 Scripts
- grep all scripts for a keyword
- Try "^networksetup" or "Wi-Fi$"

## Piping

## Pipe

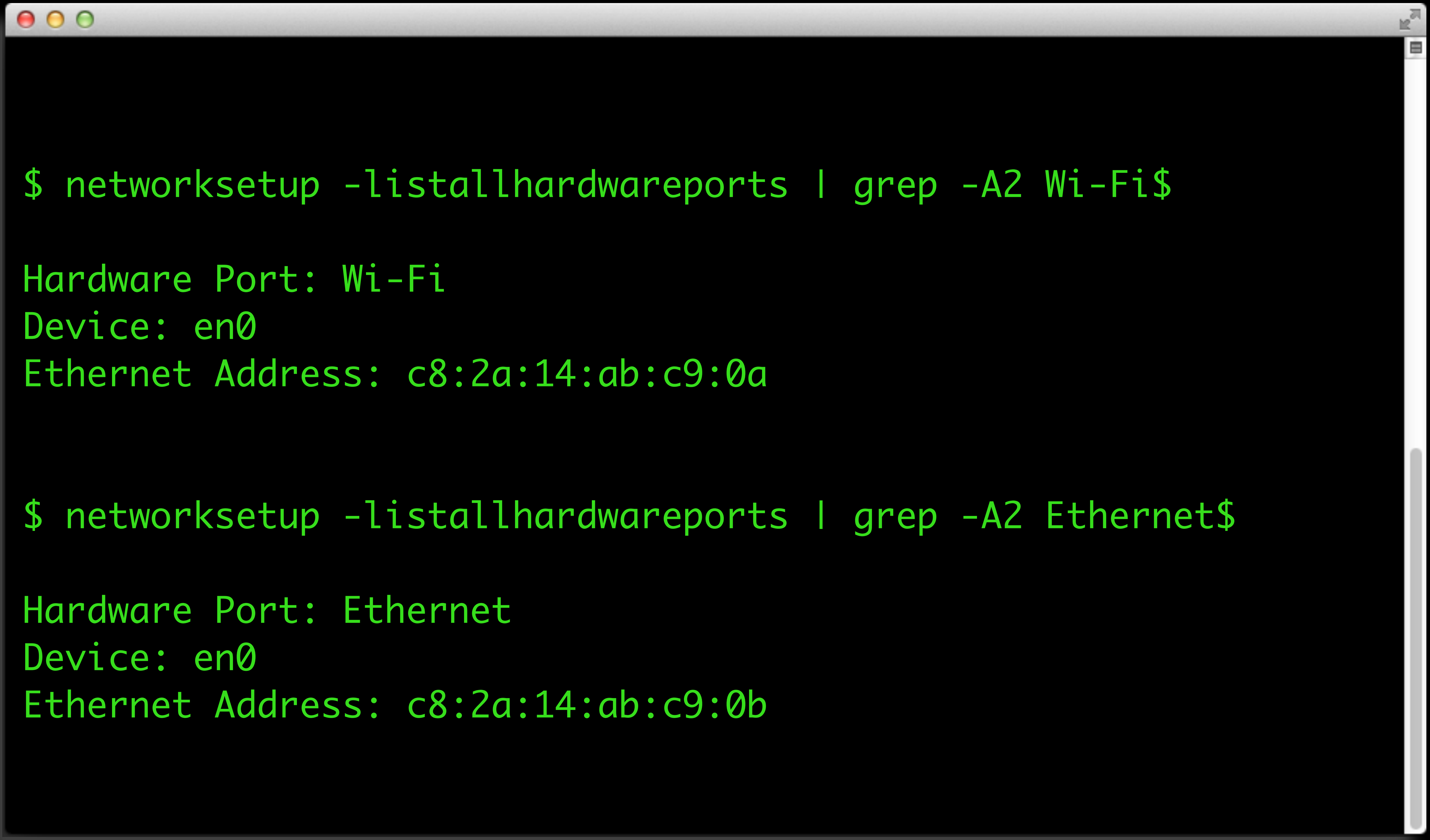
- A pipe is: |
- Pass output of left side to input of right side
- String multiple commands together

grep -A2 Wi-Fi$

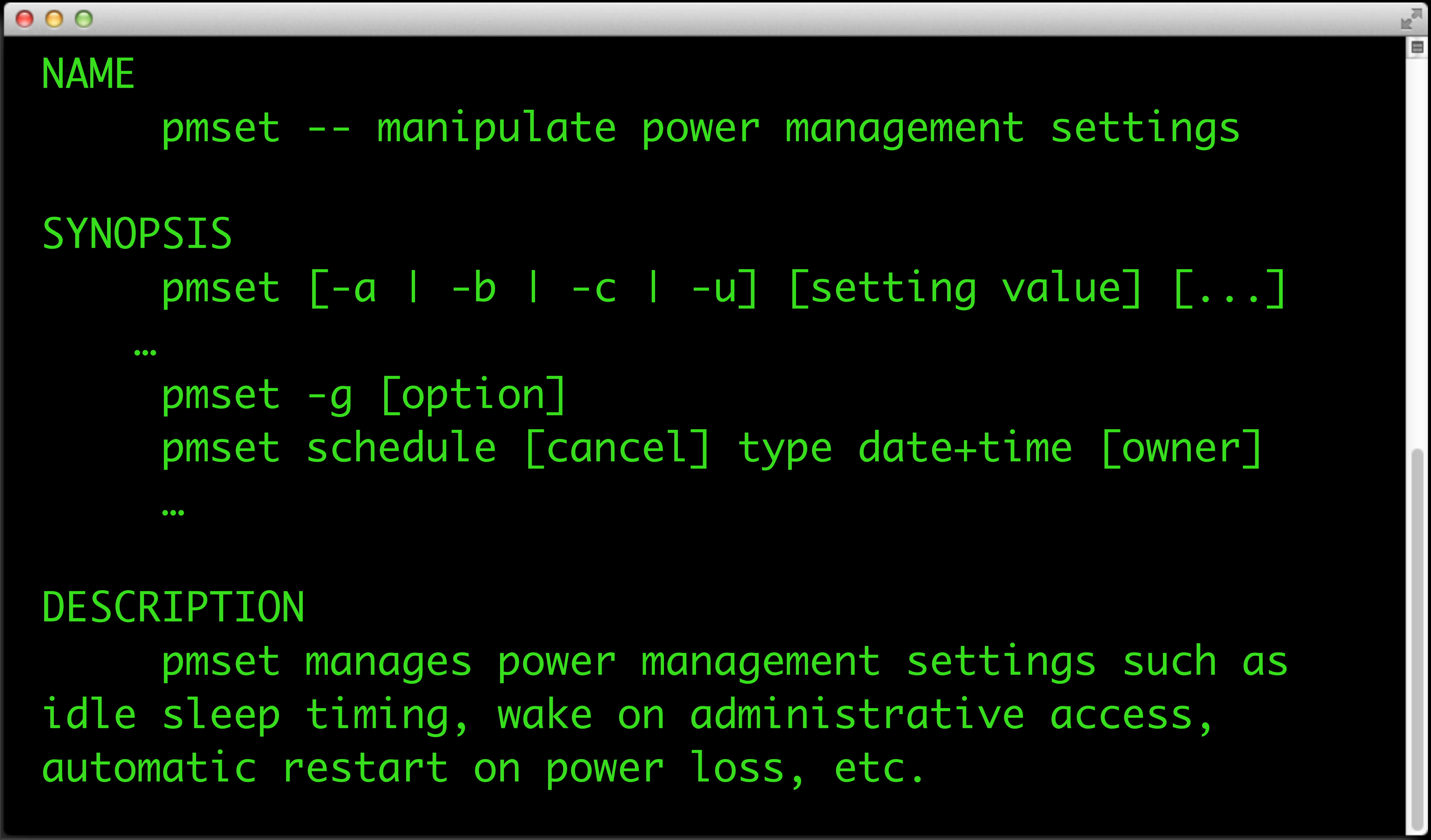


```
$ networksetup -listallhardwareports

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


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


Hardware Port: Ethernet
Device: en0
Ethernet Address: c8:2a:14:a0:cb:d3
```

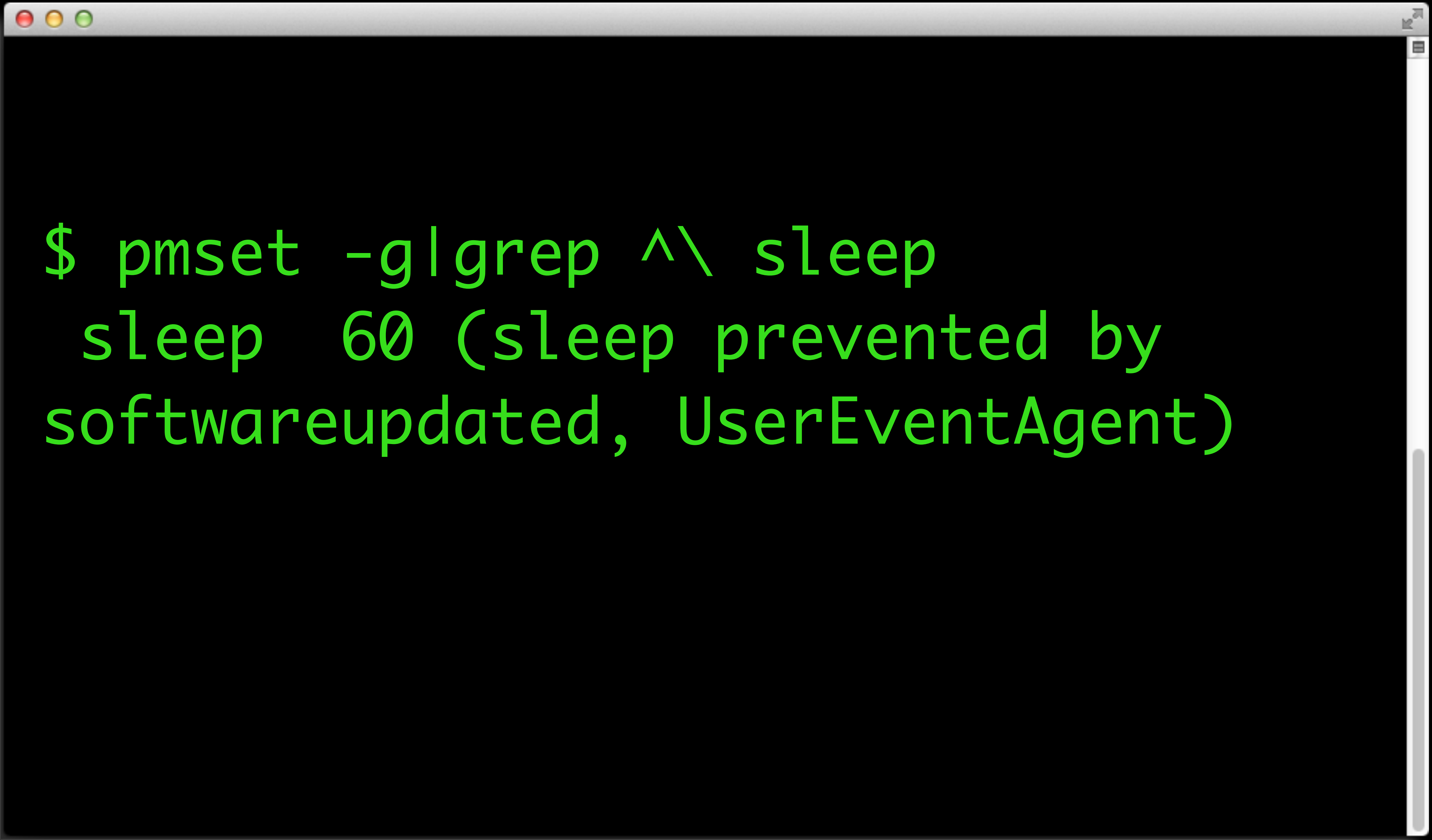
$ means 'end of line'!



```
$ networksetup -listallhardwareports | grep -A2 Wi-Fi$

Hardware Port: Wi-Fi
Device: en0
Ethernet Address: c8:2a:14:ab:c9:0a


$ networksetup -listallhardwareports | grep -A2 Ethernet$

Hardware Port: Ethernet
Device: en0
Ethernet Address: c8:2a:14:ab:c9:0b
```



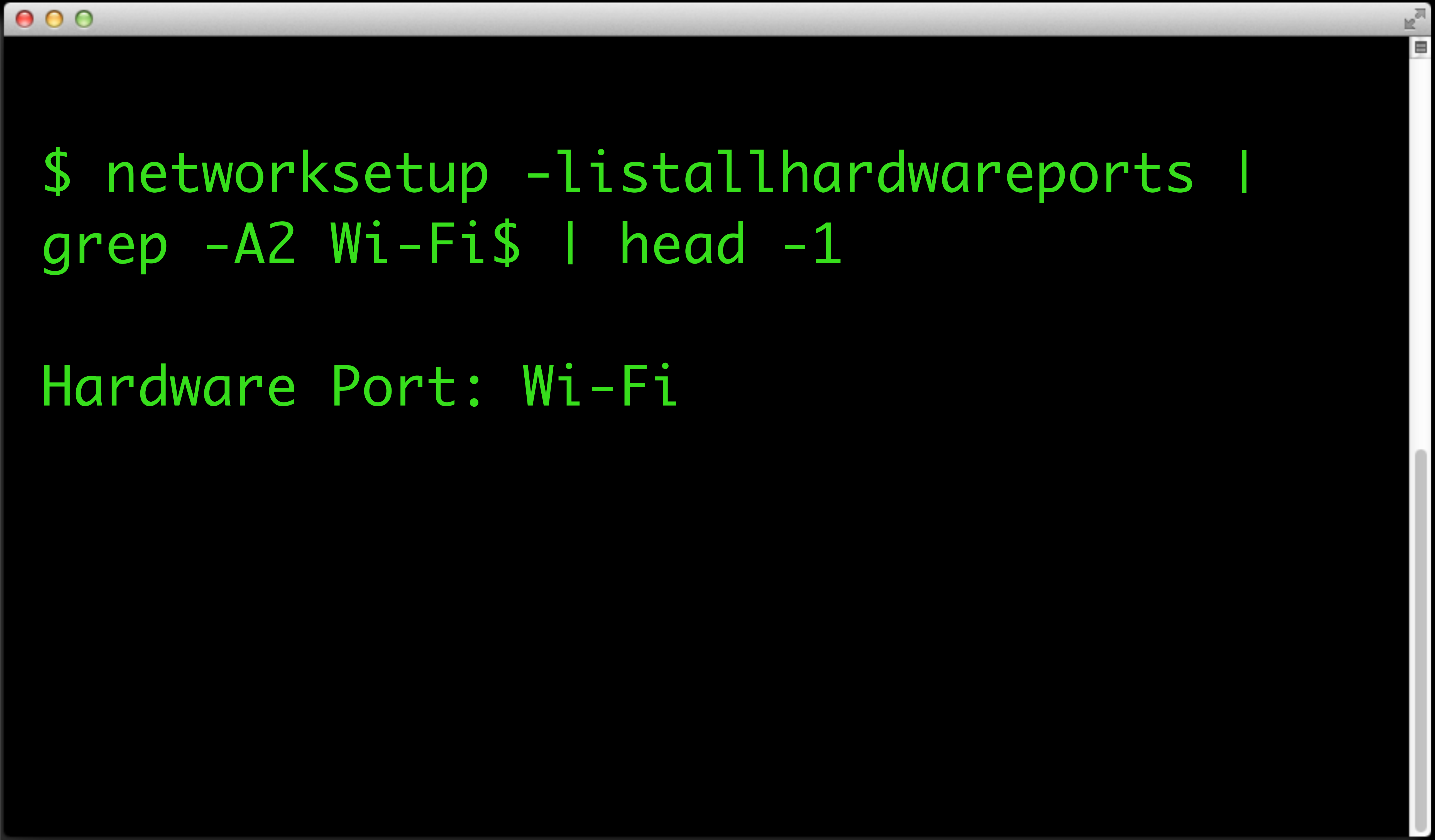
```
NAME
     pmset -- manipulate power management settings

SYNOPSIS
     pmset [-a | -b | -c | -u] [setting value] [...]
    …
     pmset -g [option]
     pmset schedule [cancel] type date+time [owner]
     …

DESCRIPTION
     pmset manages power management settings such as
idle sleep timing, wake on administrative access,
automatic restart on power loss, etc.
```



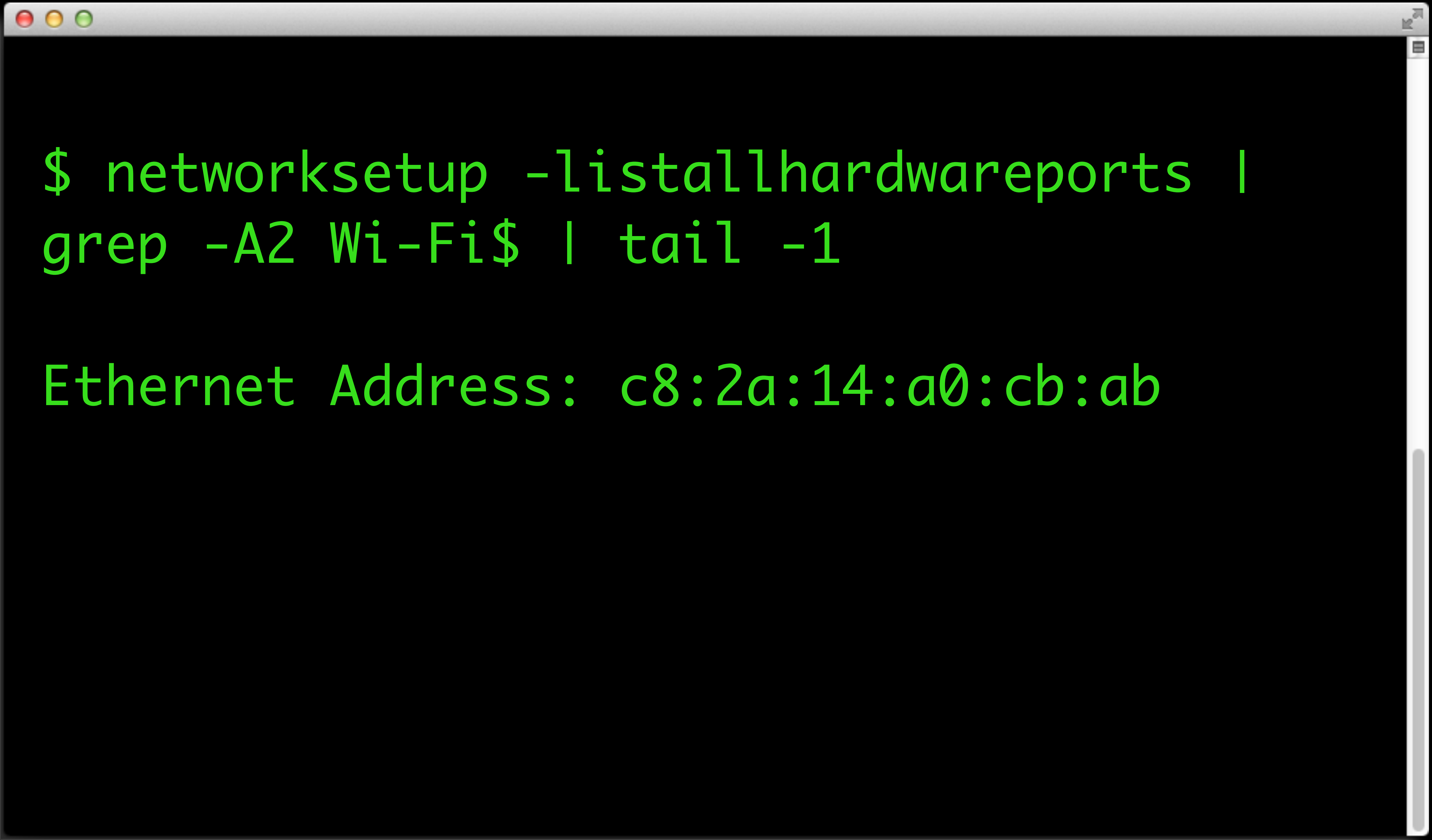
```
$ pmset -g|grep ^\ sleep
 sleep  60 (sleep prevented by
softwareupdated, UserEventAgent)
```

# head & tail

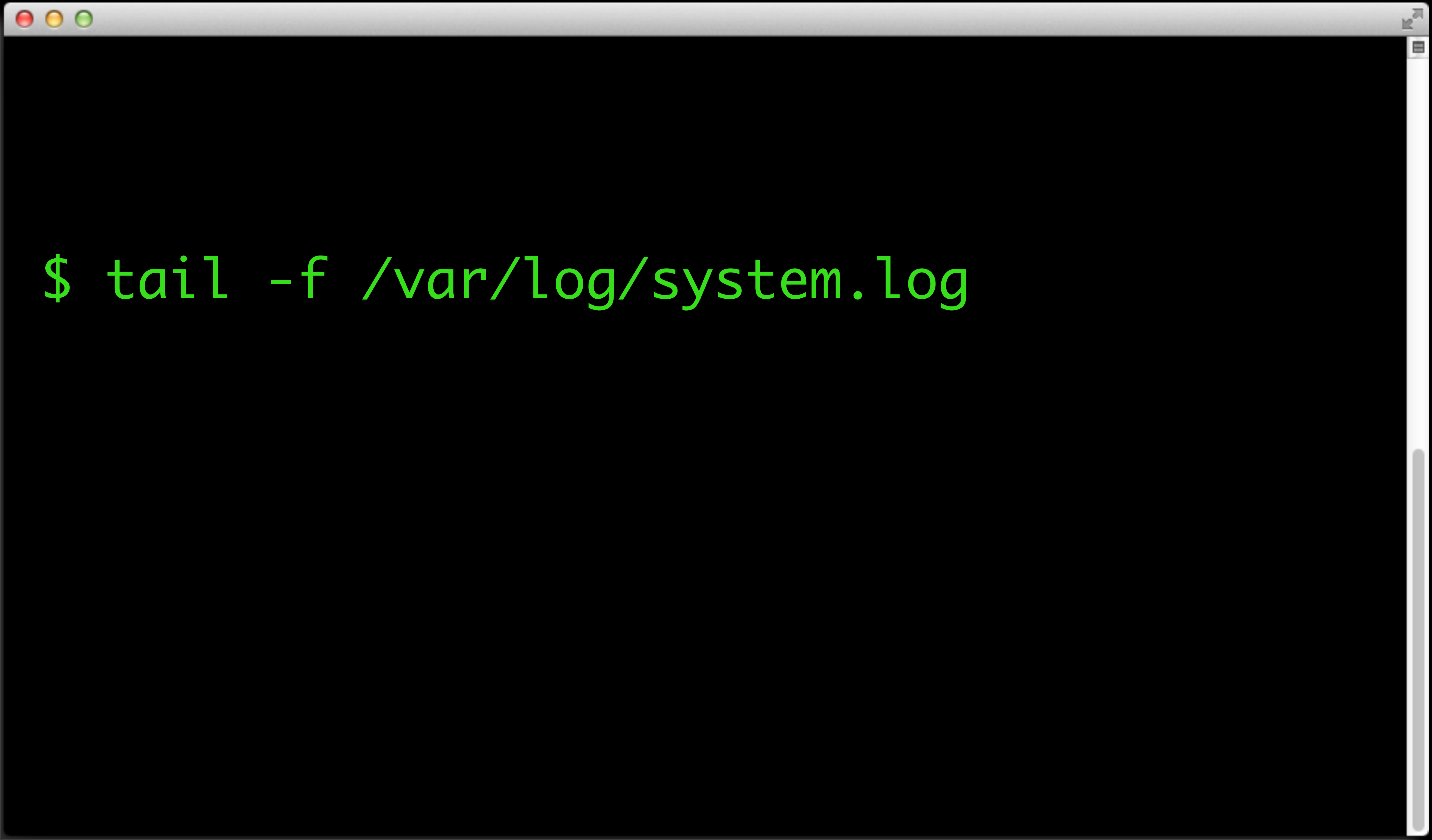
- `head -N`
  shows N lines top down
- `tail -N`
  shows N lines bottom up



```
$ networksetup -listallhardwareports |
grep -A2 Wi-Fi$ | head -1

Hardware Port: Wi-Fi
```



```
$ networksetup -listallhardwareports |
grep -A2 Wi-Fi$ | tail -1

Ethernet Address: c8:2a:14:a0:cb:ab
```



```
$ tail -f /var/log/system.log
```

## Try It!

- Grep output of:

`networksetup`

- `-getcomputername`
- `-getinfo Wi-Fi`
- `-getmacaddress`

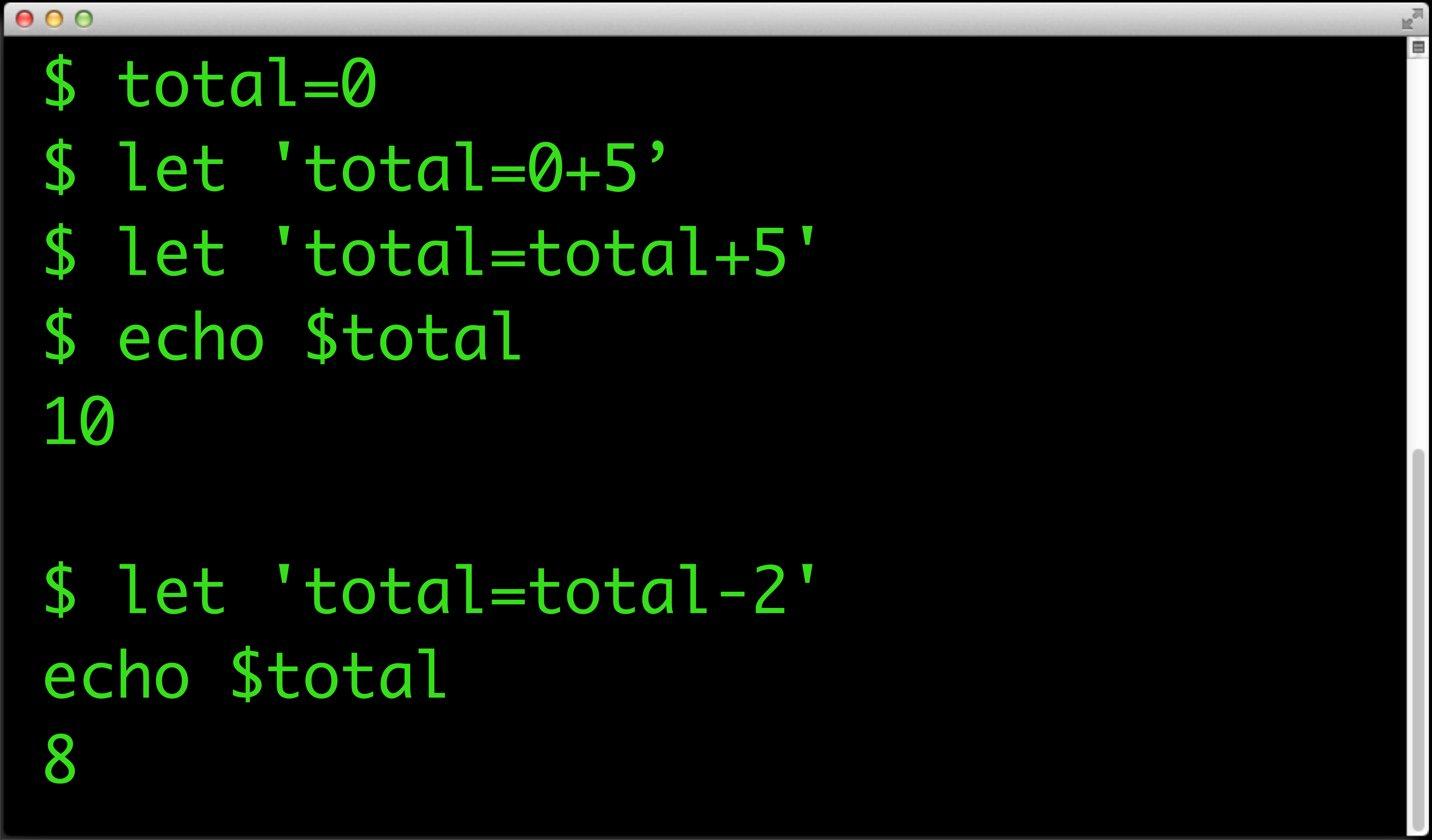
`system_profiler`

- `Serial`
- `.app`
- `USB`
- `Model Identifier:`

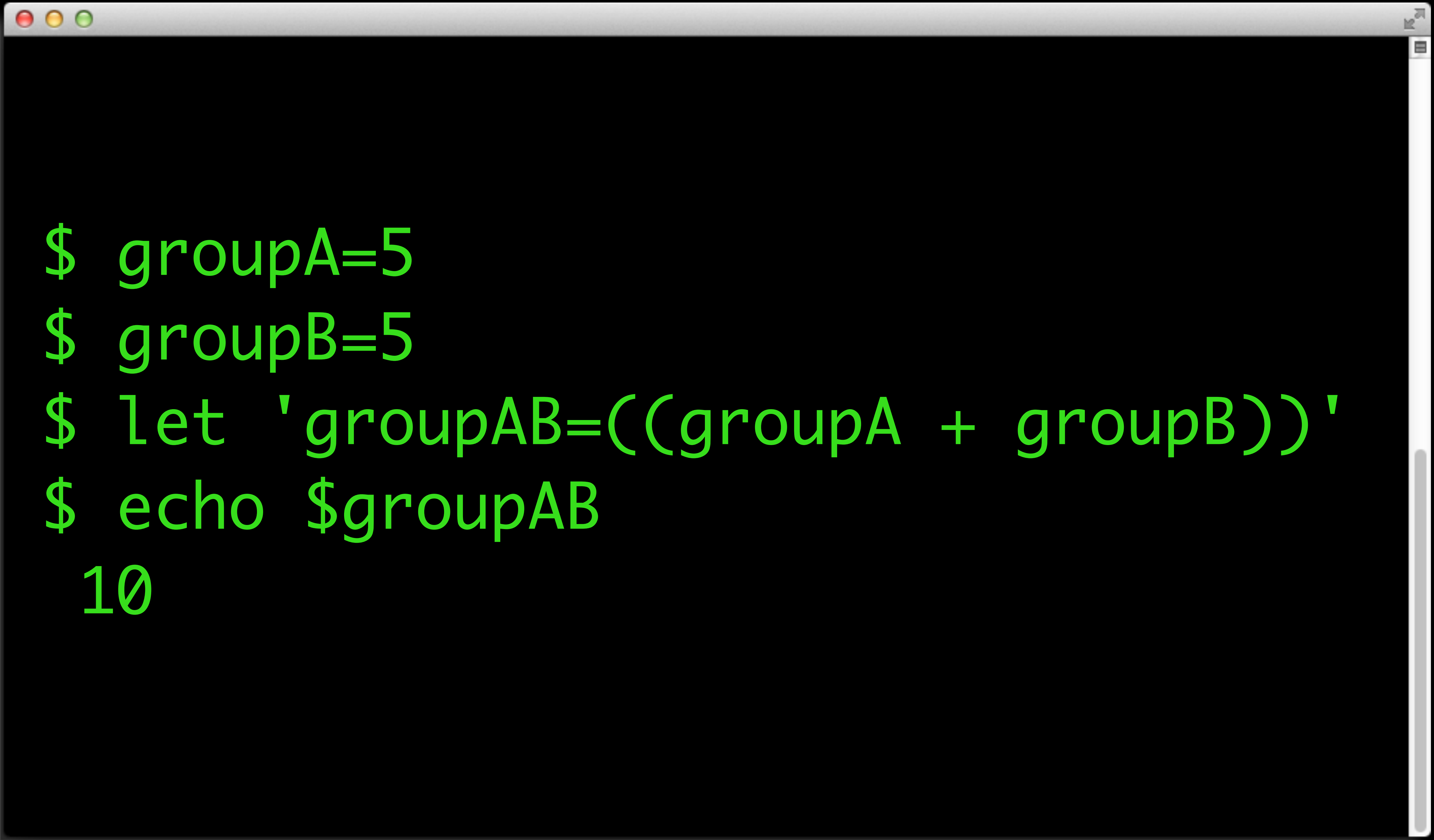
Hint: `system_profiler -listDataTypes`

# Math

- Bash supports integer only
- `let answer=num1+-/*num2`
- Variables don't need $



```
$ total=0
$ let 'total=0+5’
$ let 'total=total+5'
$ echo $total
10

$ let 'total=total-2'
echo $total
8
```


```
$ groupA=5
$ groupB=5
$ let 'groupAB=((groupA + groupB))'
$ echo $groupAB
 10
```

## Try It!

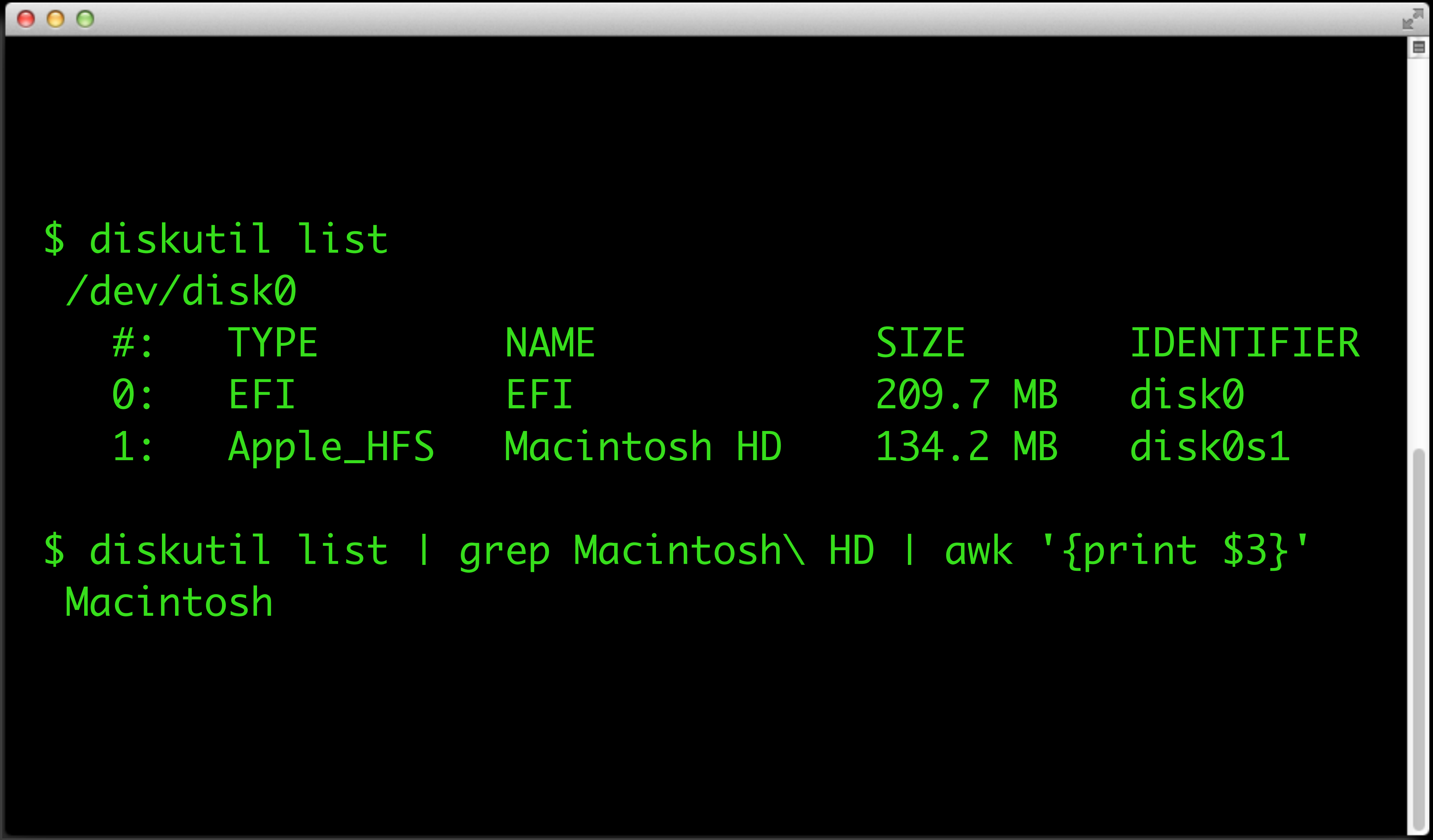
- Do some math!
- `total=0`
- `let 'total=total+1'`
- `echo $total`

# awk

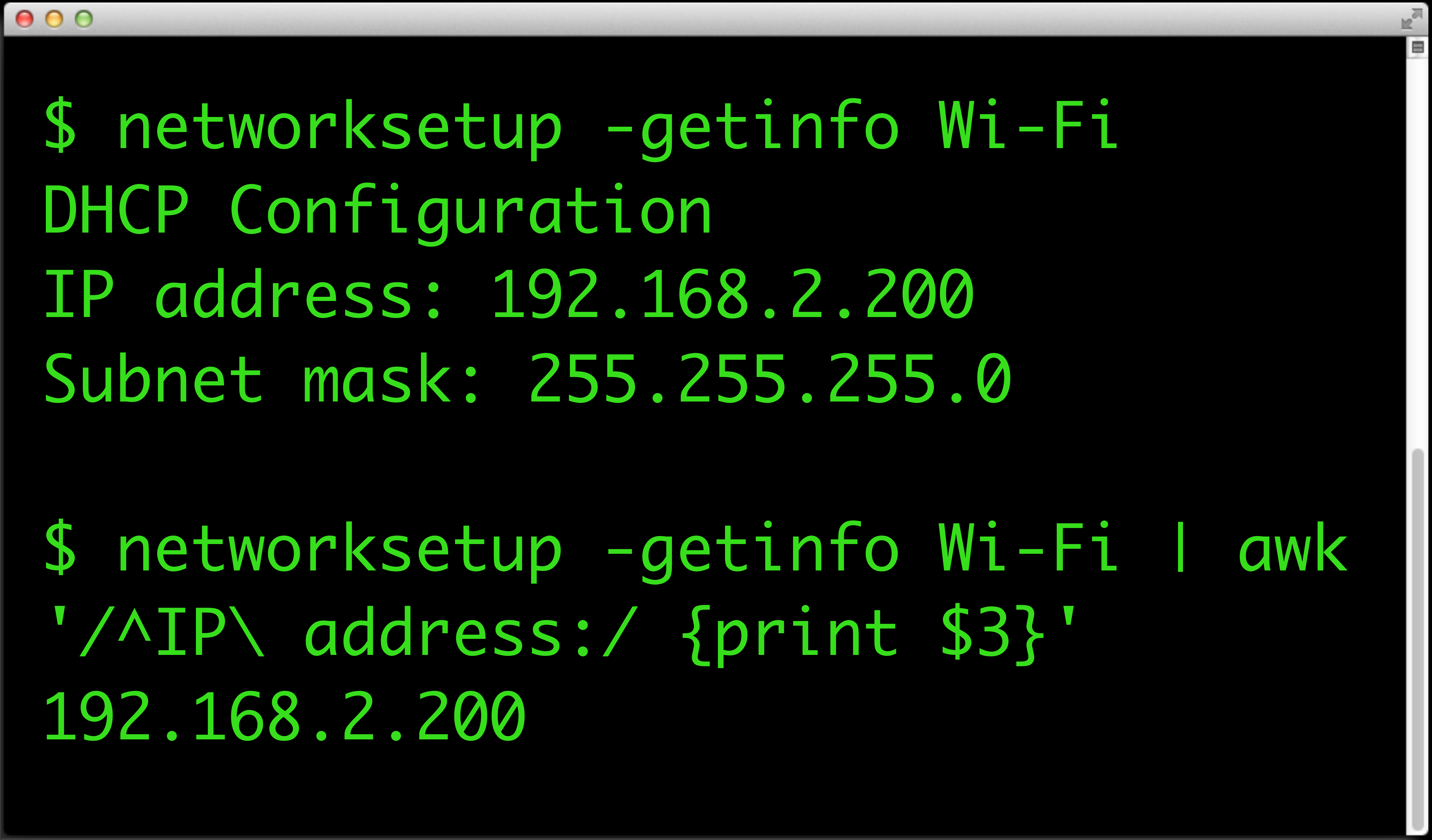
- Print parts of string
- Change delimiter
- You can even search!
- `awk '/search/{print $field#}'`

## diskutil

- Manipulates structure of local disks
- Provides information on:
  - partitioning
  - schemes
  - layouts
  - formats
- CoreStorage and AppleRAID



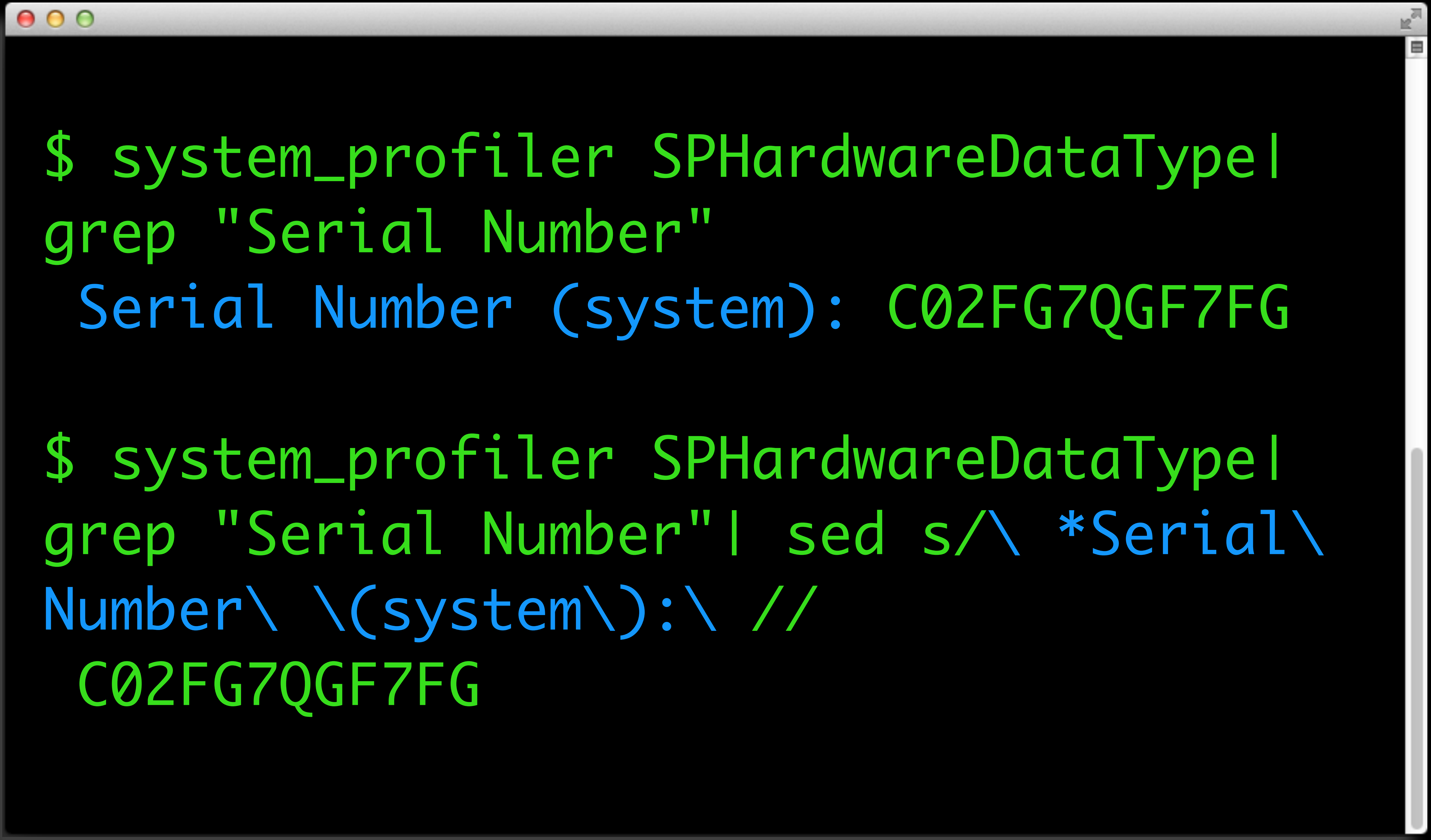
```
$ diskutil list
 /dev/disk0
   #:    TYPE         NAME              SIZE       IDENTIFIER
   0:    EFI          EFI               209.7 MB   disk0
   1:    Apple_HFS    Macintosh HD      134.2 MB   disk0s1

$ diskutil list | grep Macintosh\ HD | awk '{print $3}'
 Macintosh
```



```
$ networksetup -getinfo Wi-Fi
DHCP Configuration
IP address: 192.168.2.200
Subnet mask: 255.255.255.0

$ networksetup -getinfo Wi-Fi | awk
'/^IP\ address:/ {print $3}'
192.168.2.200
```

## sed

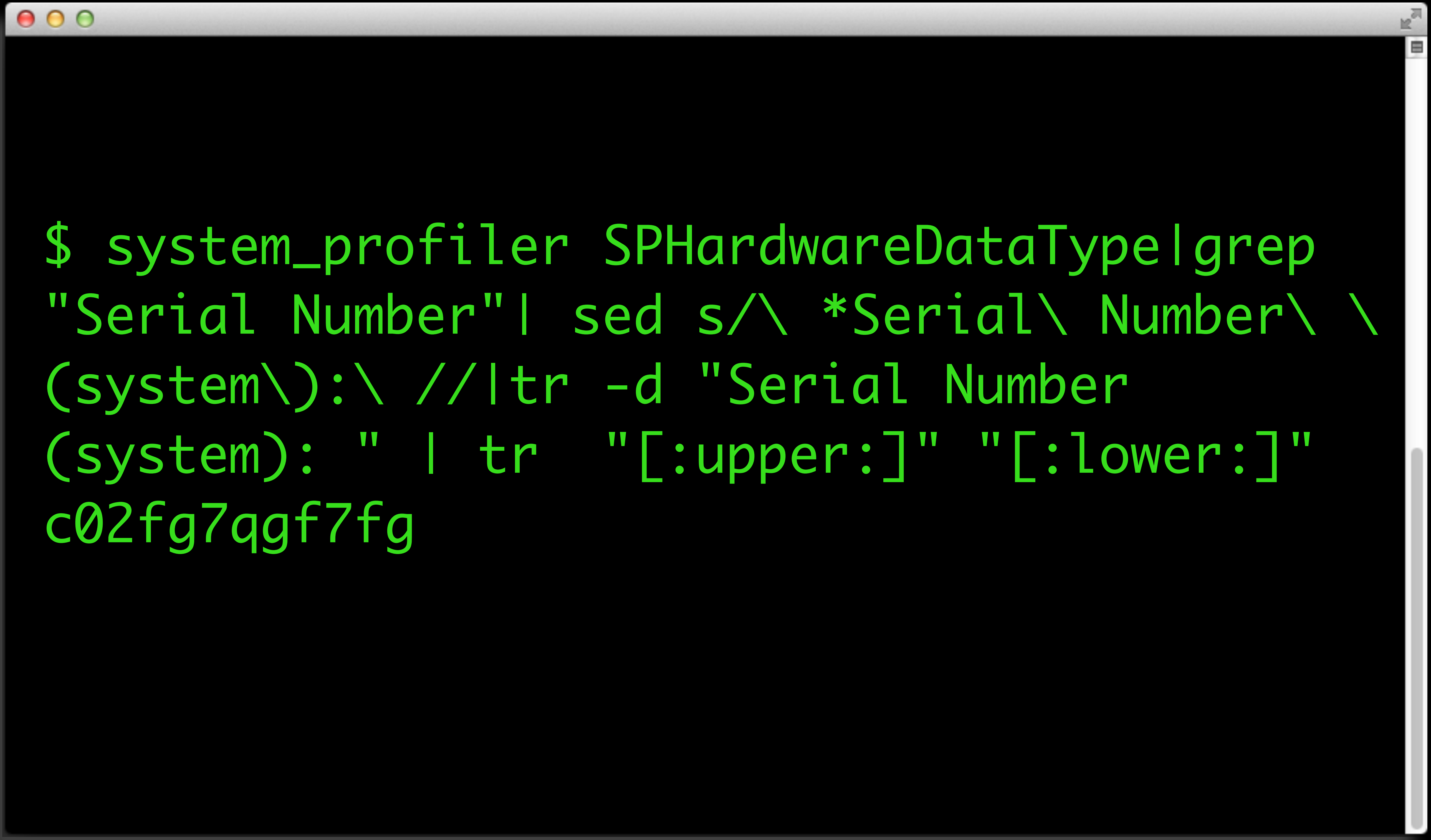
- Stream Editor
- Filters and Transforms Text
- Most Commonly Used to Substitute
  - sed s/regex/replacement/



```
$ system_profiler SPHardwareDataType|
grep "Serial Number"
 Serial Number (system): C02FG7QGF7FG

$ system_profiler SPHardwareDataType|
grep "Serial Number"| sed s/\ *Serial\
Number\ \(system\):\ //
 C02FG7QGF7FG
```

# tr

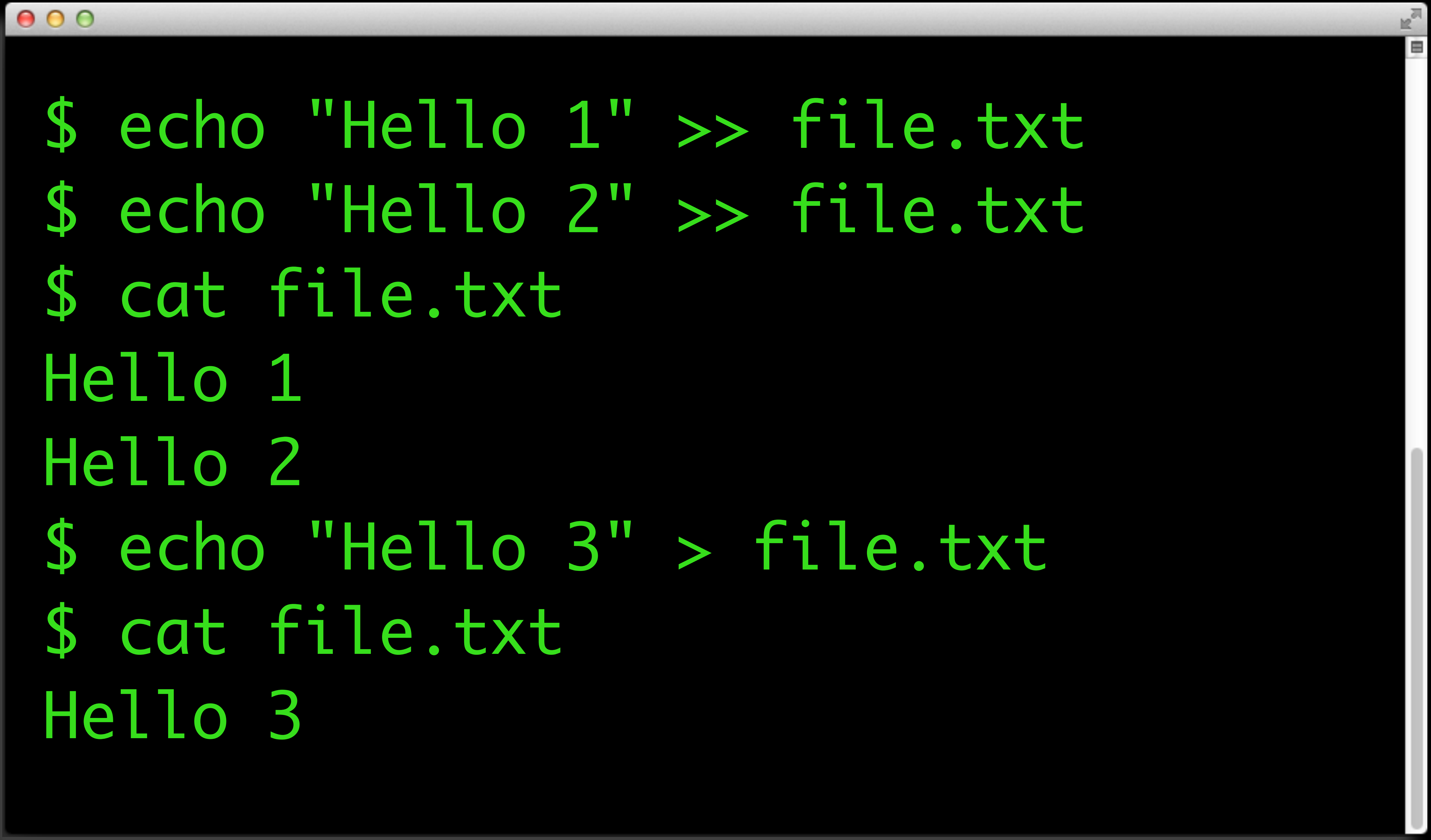
- Translate Characters
- Substitute: tr string1 string2
- Delete Charachters: tr -d "abc"



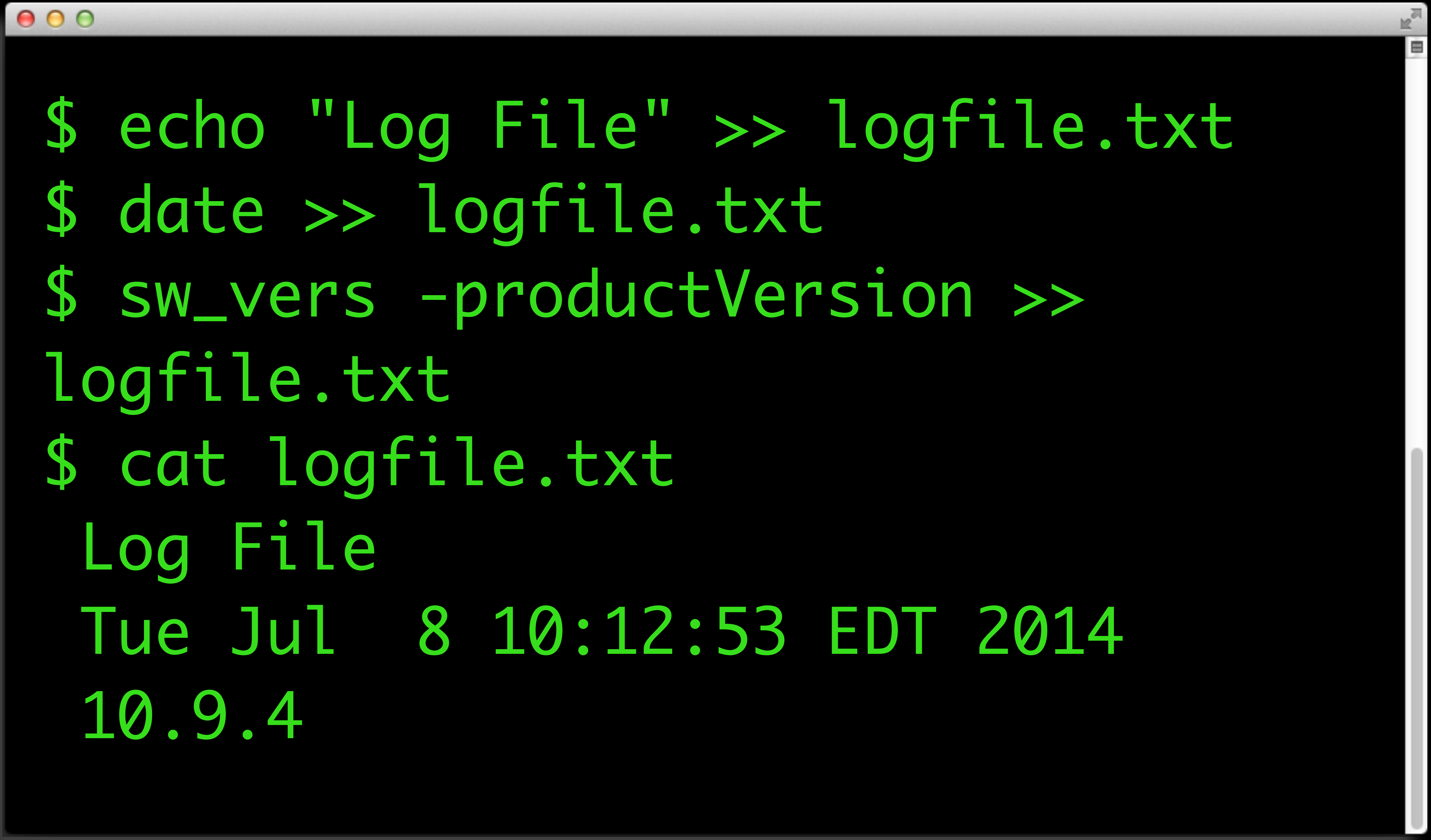
```
$ system_profiler SPHardwareDataType|grep
"Serial Number"| sed s/\ *Serial\ Number\ \
(system\):\ //|tr -d "Serial Number
(system): " | tr  "[:upper:]" "[:lower:]"
c02fg7qgf7fg
```

## Redirection

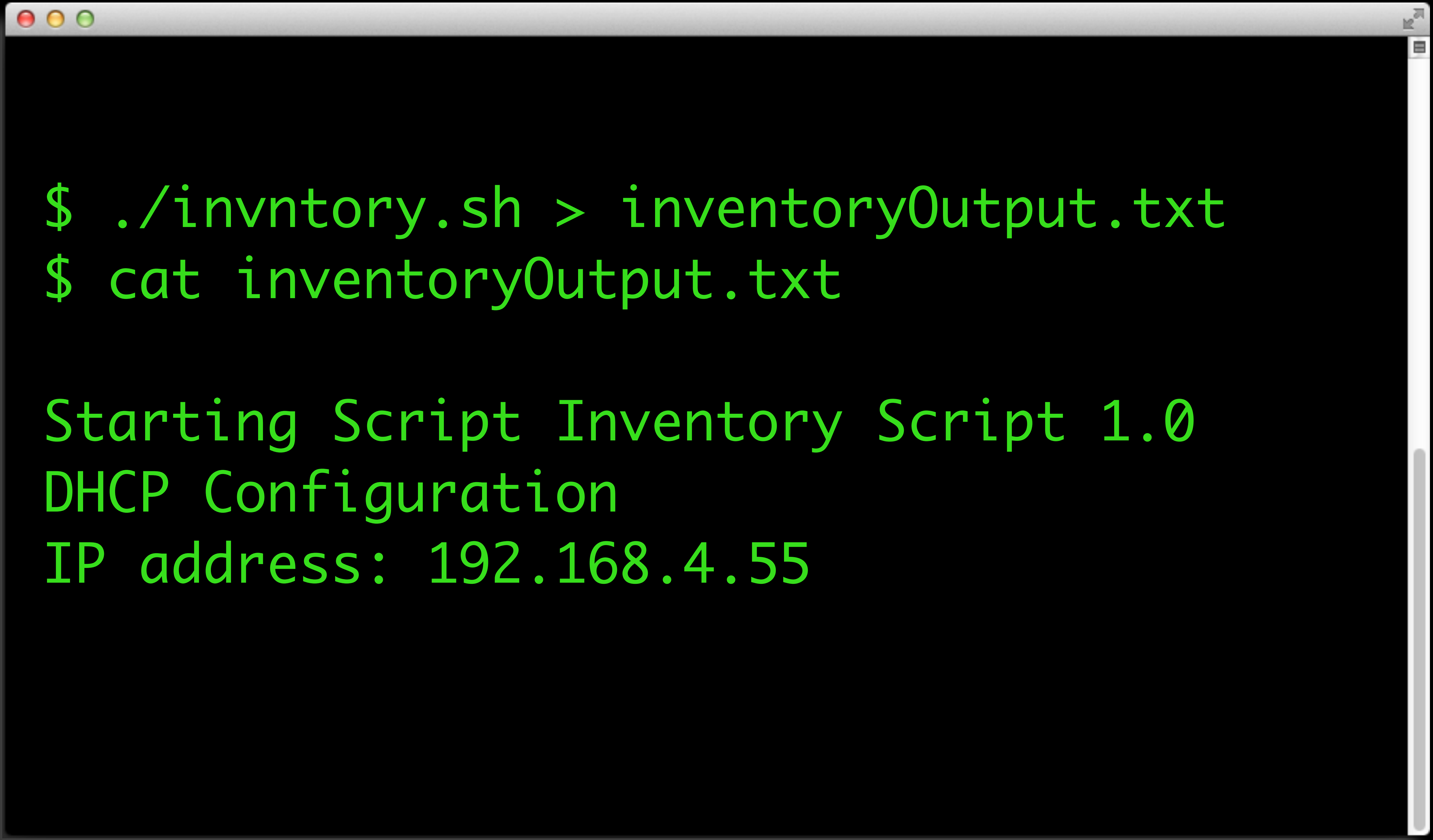
- Overwrite File: >
- Append to File: >>
- Command Input: <



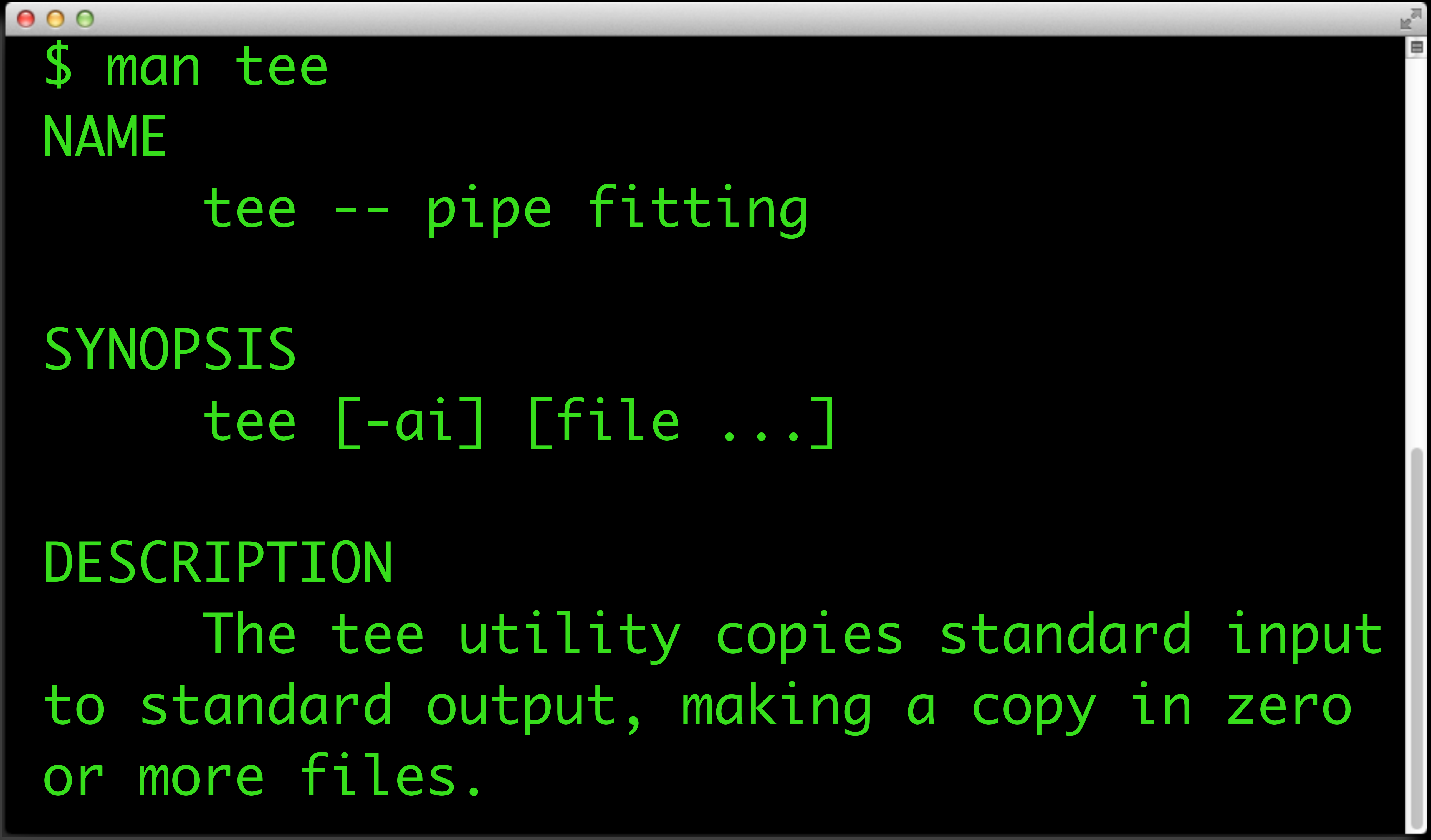
```
$ echo "Hello 1" >> file.txt
$ echo "Hello 2" >> file.txt
$ cat file.txt
Hello 1
Hello 2
$ echo "Hello 3" > file.txt
$ cat file.txt
Hello 3
```



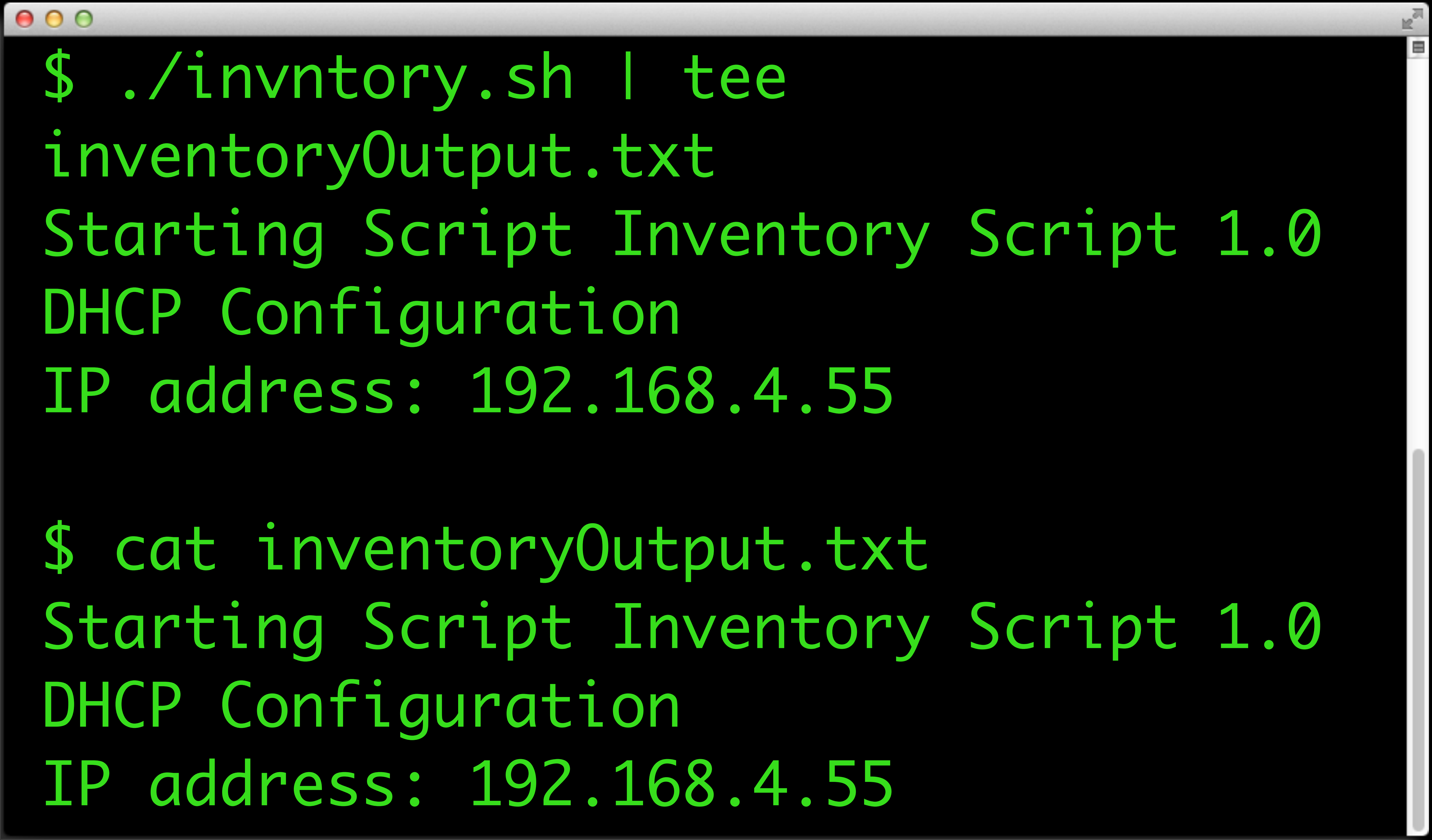
```
$ echo "Log File" >> logfile.txt
$ date >> logfile.txt
$ sw_vers -productVersion >>
logfile.txt
$ cat logfile.txt
 Log File
 Tue Jul  8 10:12:53 EDT 2014
 10.9.4
```



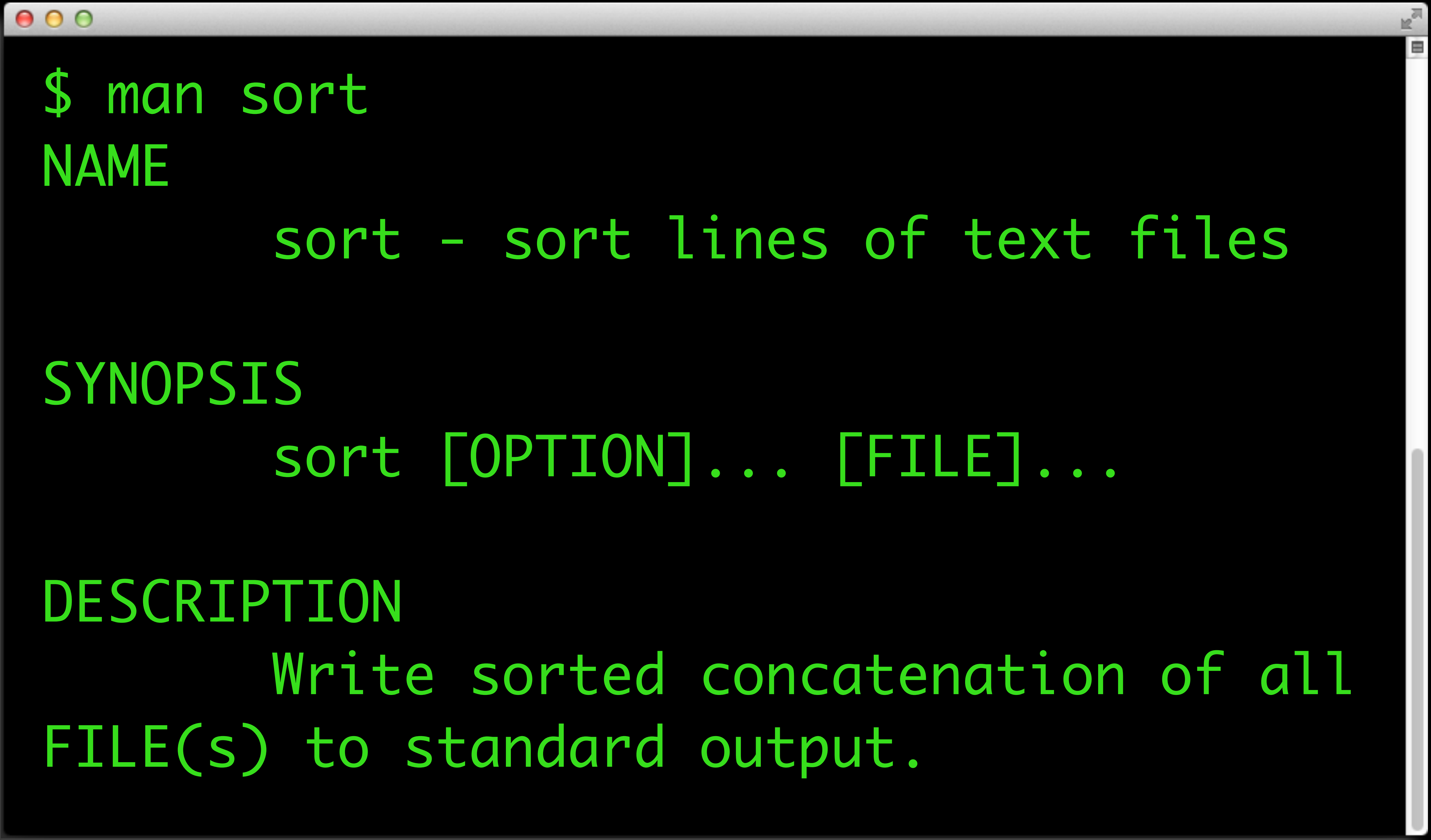
```
$ ./invntory.sh > inventoryOutput.txt
$ cat inventoryOutput.txt

Starting Script Inventory Script 1.0
DHCP Configuration
IP address: 192.168.4.55
```



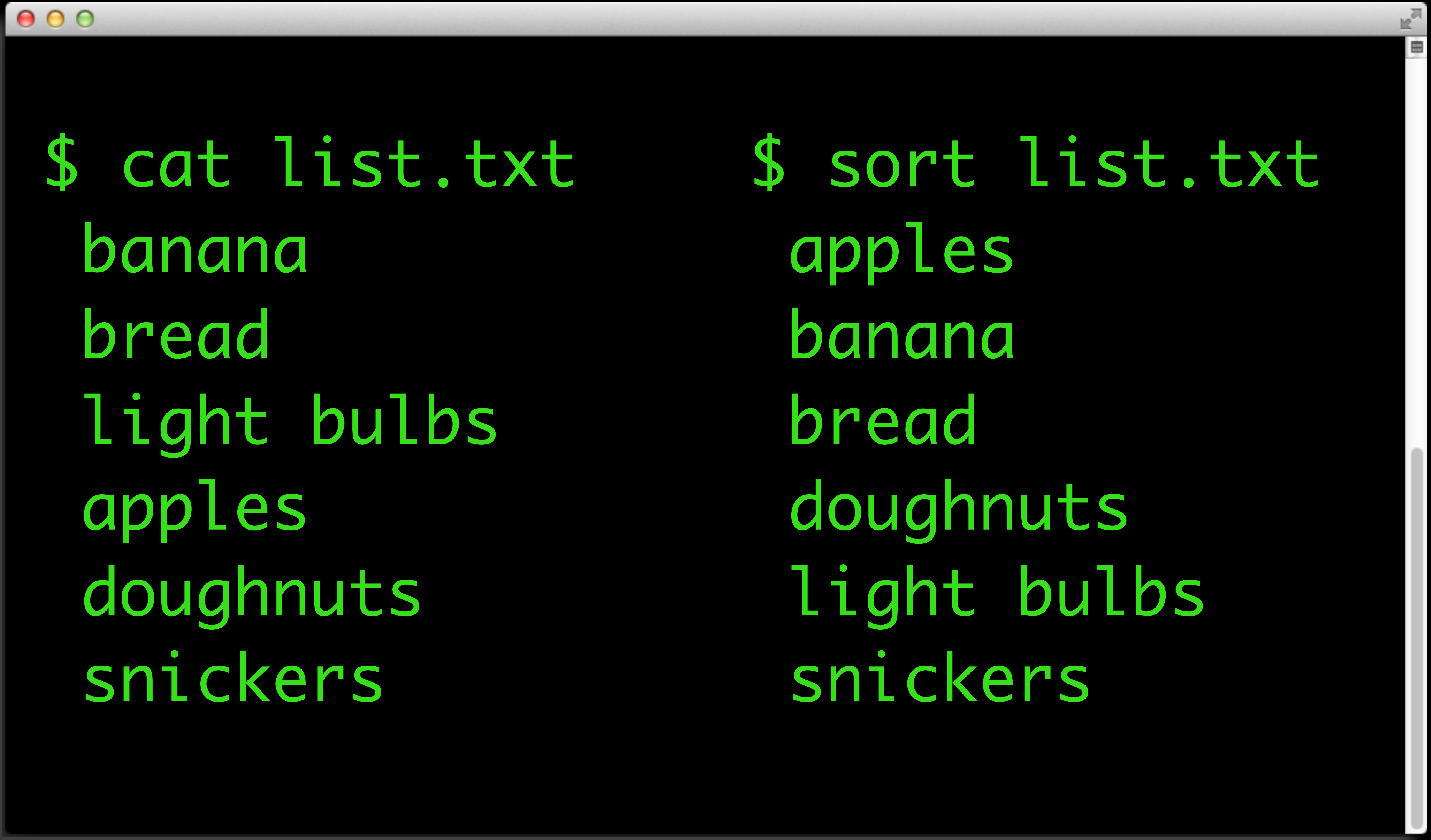
```
$ man tee
NAME
     tee -- pipe fitting

SYNOPSIS
     tee [-ai] [file ...]

DESCRIPTION
     The tee utility copies standard input
to standard output, making a copy in zero
or more files.
```



```
$ ./invntory.sh | tee
inventoryOutput.txt
Starting Script Inventory Script 1.0
DHCP Configuration
IP address: 192.168.4.55

$ cat inventoryOutput.txt
Starting Script Inventory Script 1.0
DHCP Configuration
IP address: 192.168.4.55
```



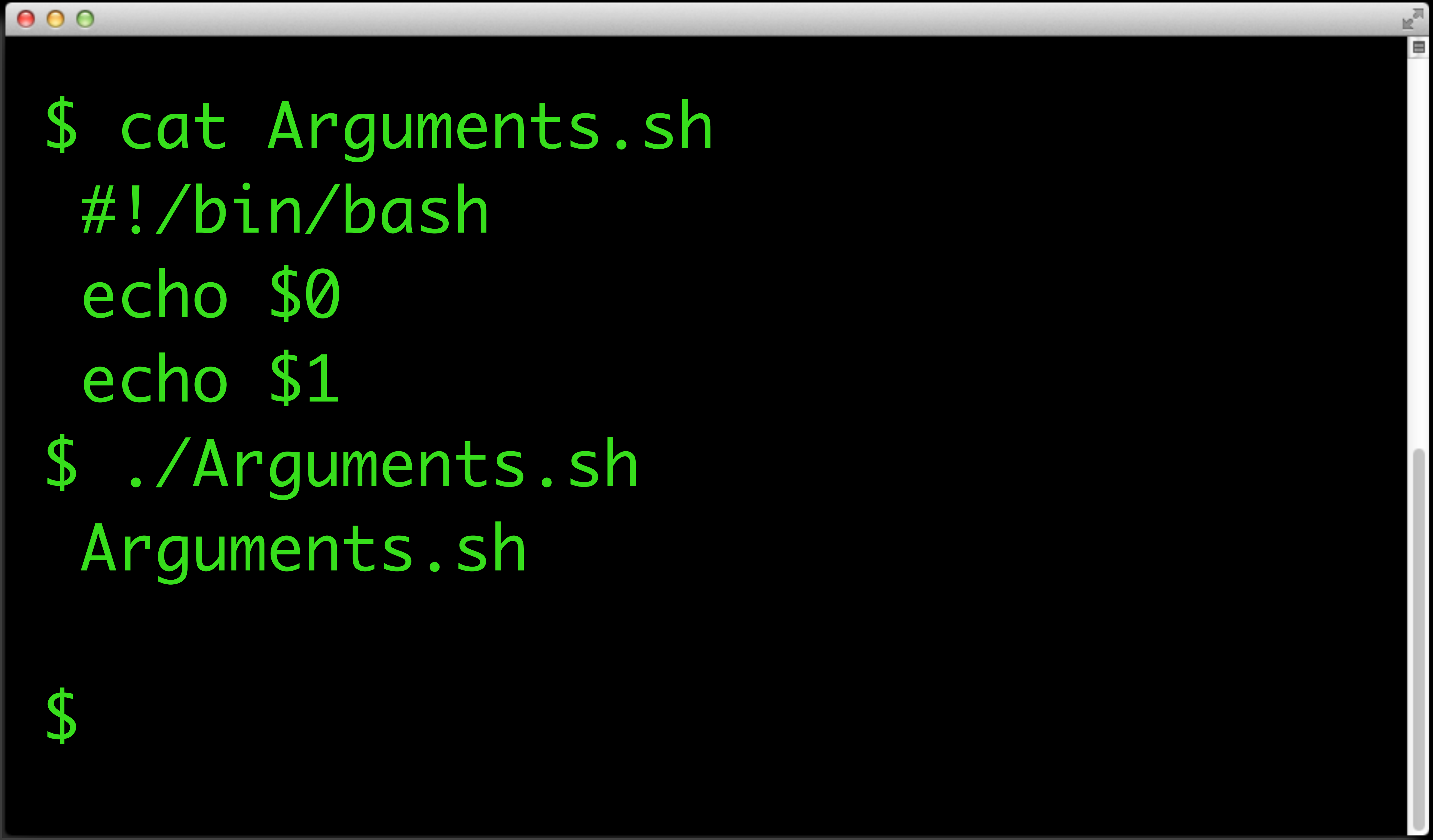
```
$ man sort
NAME
       sort - sort lines of text files

SYNOPSIS
       sort [OPTION]... [FILE]...

DESCRIPTION
       Write sorted concatenation of all
FILE(s) to standard output.
```



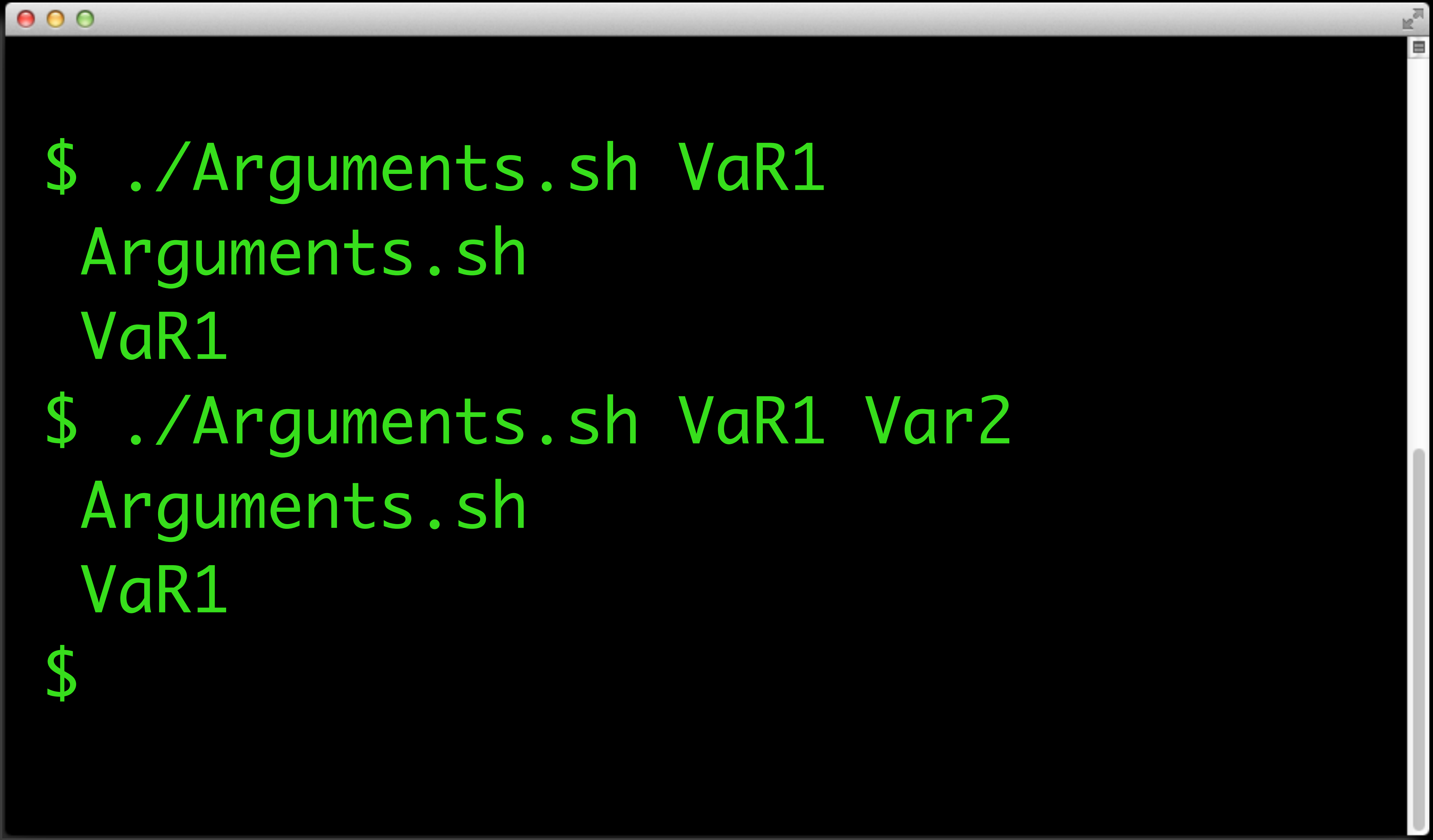
```
$ cat list.txt        $ sort list.txt
 banana                apples
 bread                 banana
 light bulbs           bread
 apples                doughnuts
 doughnuts             light bulbs
 snickers              snickers
```

## Positional Parameters

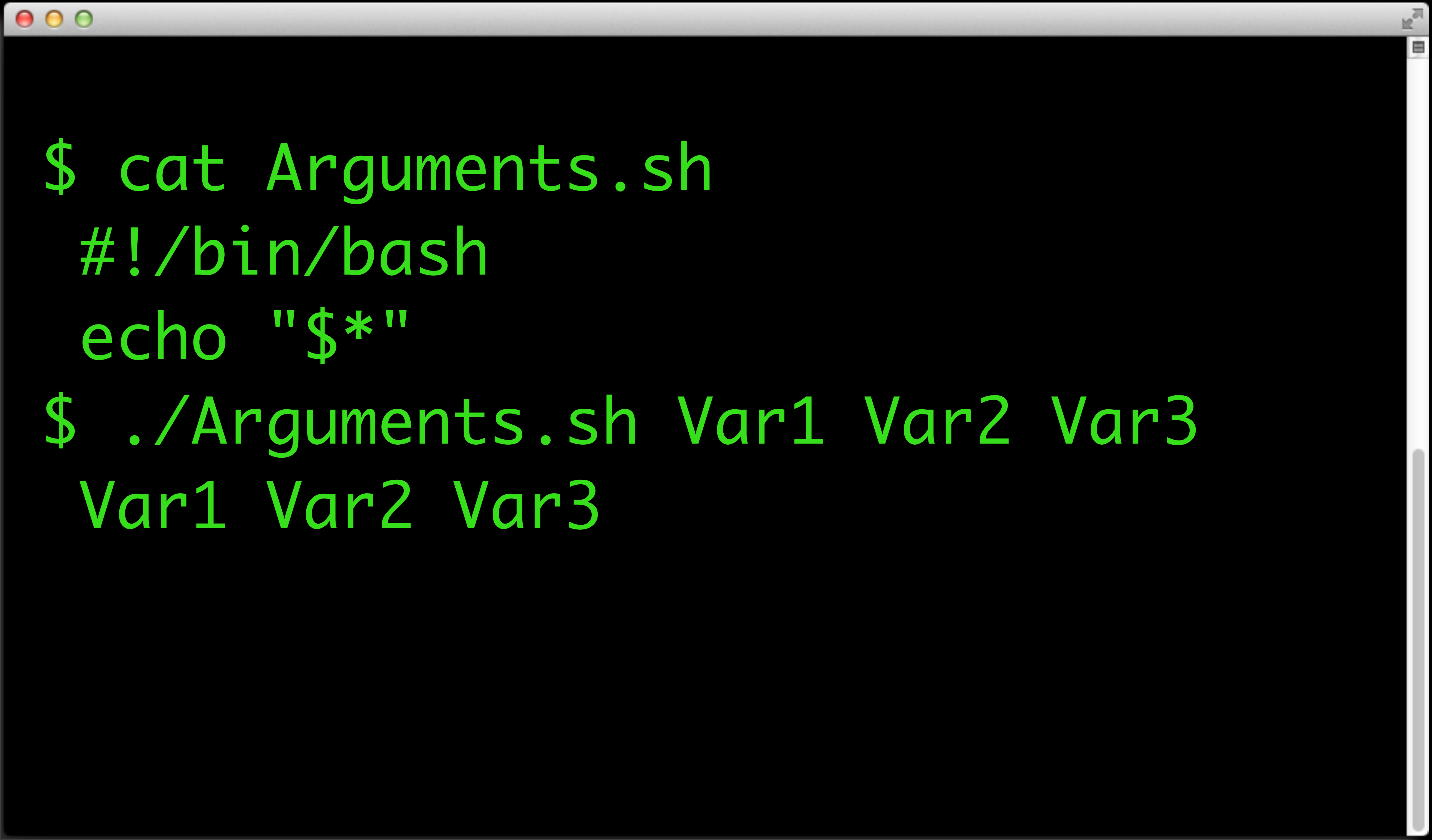
- Pass arguments to scripts
- $0 = Script Name
- $1 = First Argument (0-9)
- ${10} = 10th Argument (10+)
- $* = All Arguments (Single String)
- $@ = All Arguments (white space splits string)



```
$ cat Arguments.sh
 #!/bin/bash
 echo $0
 echo $1
$ ./Arguments.sh
 Arguments.sh


$
```



```
$ ./Arguments.sh VaR1
 Arguments.sh
 VaR1
$ ./Arguments.sh VaR1 Var2
 Arguments.sh
 VaR1
$
```



```
$ cat Arguments.sh
 #!/bin/bash
 echo "$*"
$ ./Arguments.sh Var1 Var2 Var3
 Var1 Var2 Var3
```

## Try It!

- Write system_profiler output to file
- Take output as first argument
- sed, awk, grep, sort, tee tr
- Output text to new file named: $day-$serial#.txt

break

## Part 4

- Loops
- Tests
- Case
- Functions

## More Tricks

- Ctrl-e = Move cursor to EOL
- Ctrl-a = Move cursor to BOL
- Ctrl-l = Clear Screen
- Ctrl-w = Delete 1 Previous Word

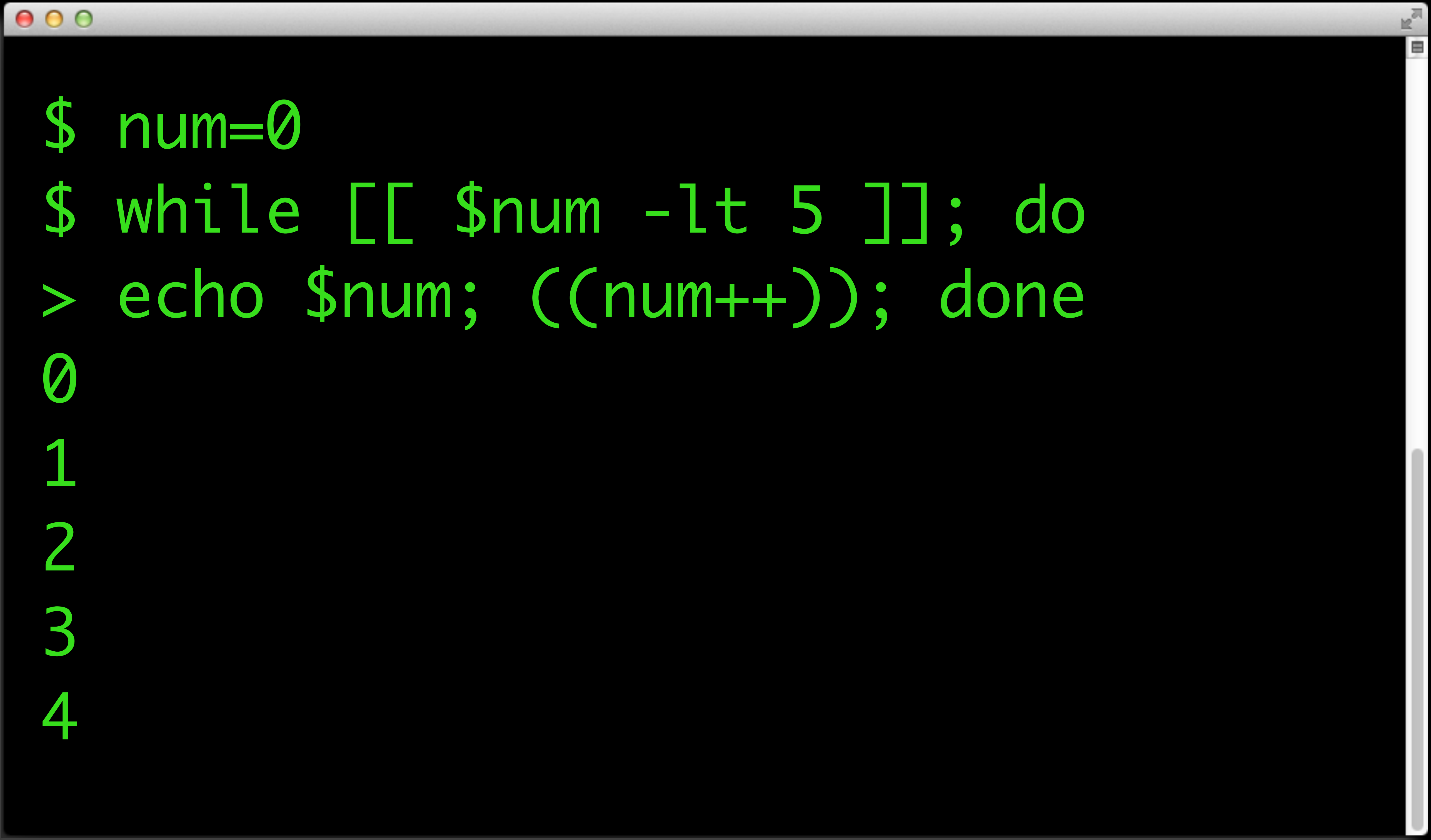
## While Loops

- Execute commands repeatedly
- While Control-Command is true, Consequent-Commands run

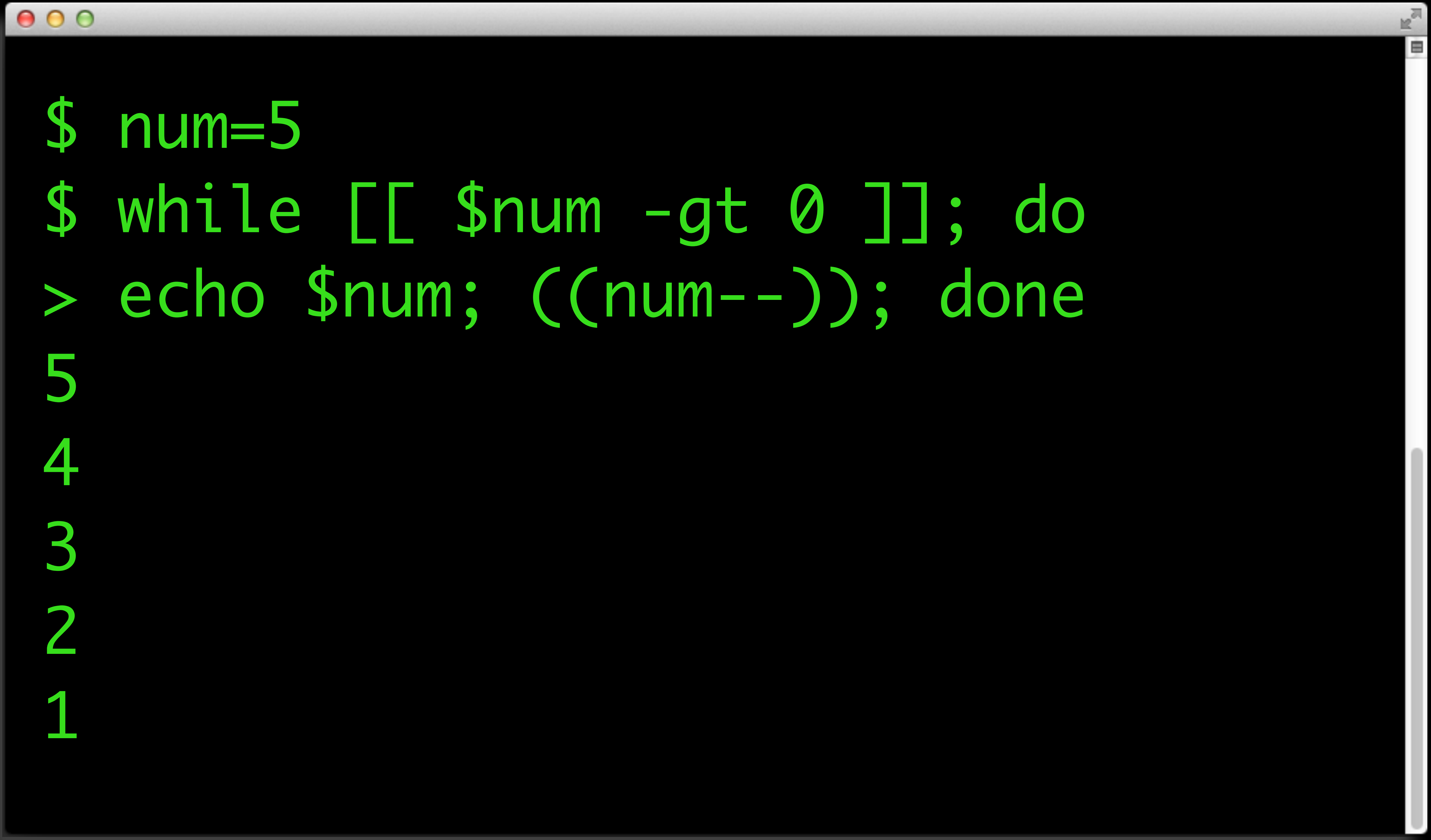
```
while CONTROL-COMMAND; do
CONSEQUENT-COMMANDS; done
```

## helpful let

- ((variable++)) Increment by 1
- ((variable--)) Decrement by 1



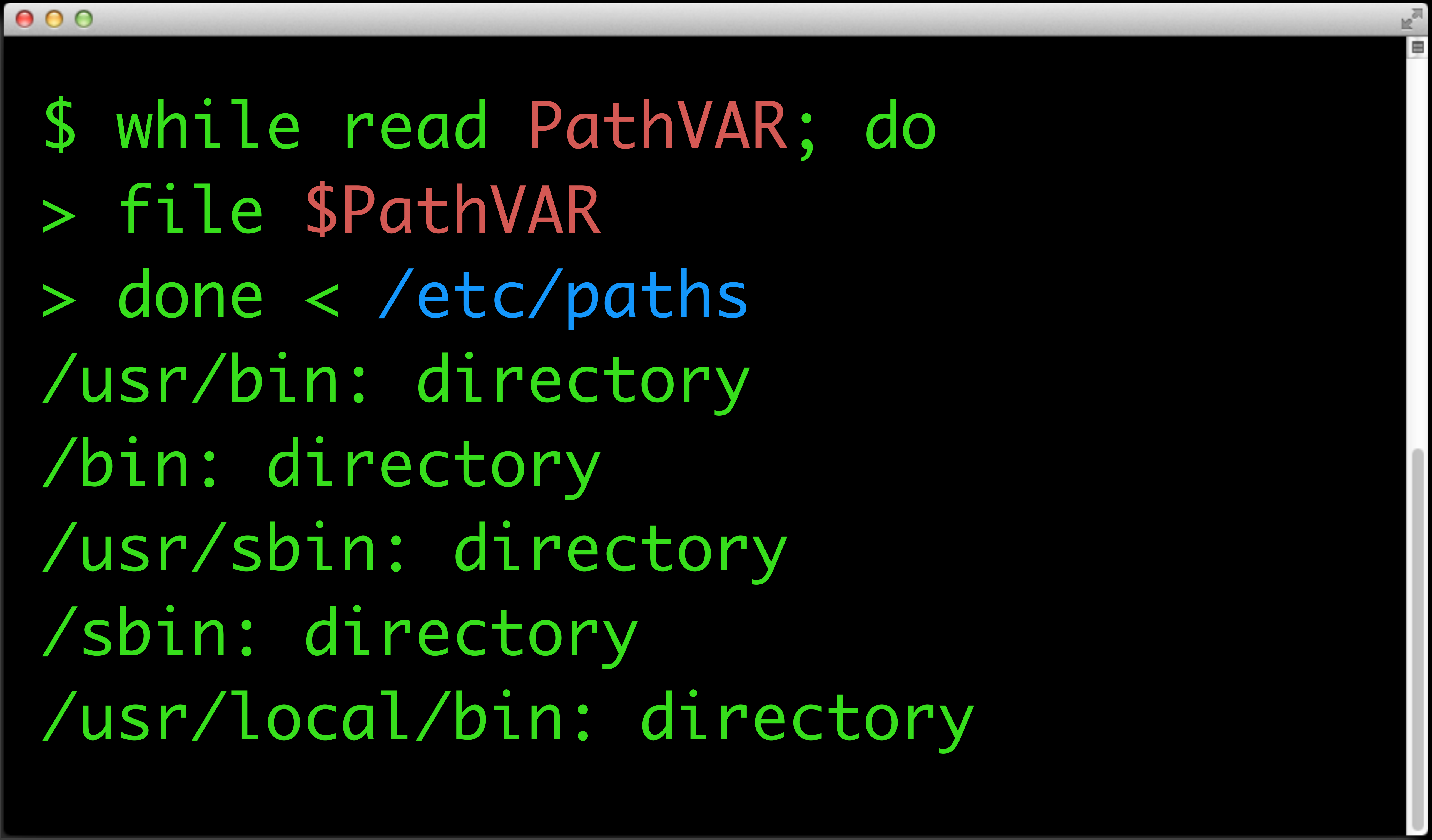
```
$ num=0
$ while [[ $num -lt 5 ]]; do
> echo $num; ((num++)); done
0
1
2
3
4
```



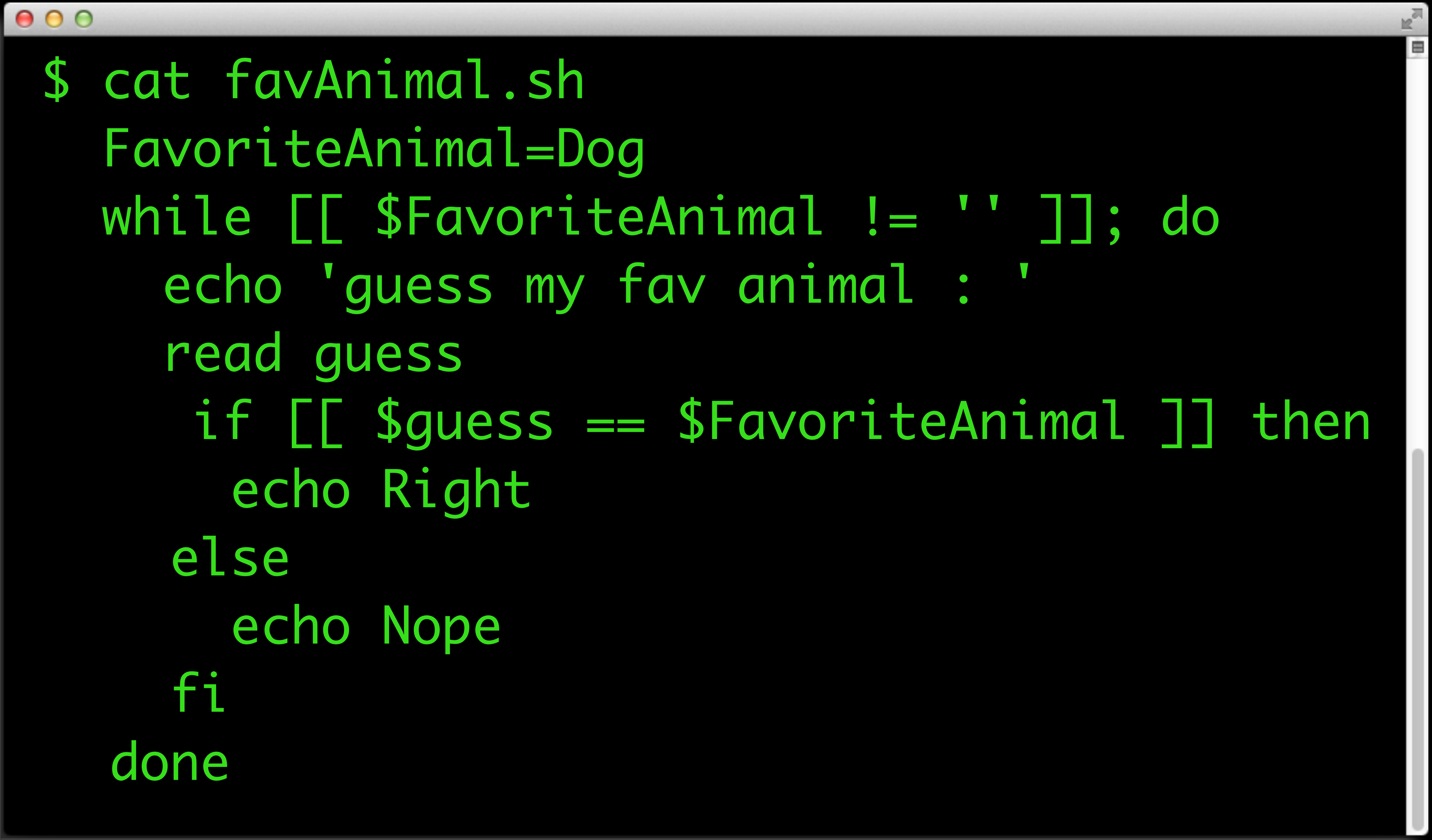
```
$ num=5
$ while [[ $num -gt 0 ]]; do
> echo $num; ((num--)); done
5
4
3
2
1
```

## read

- Take Input from Terminal
- Set into Variable



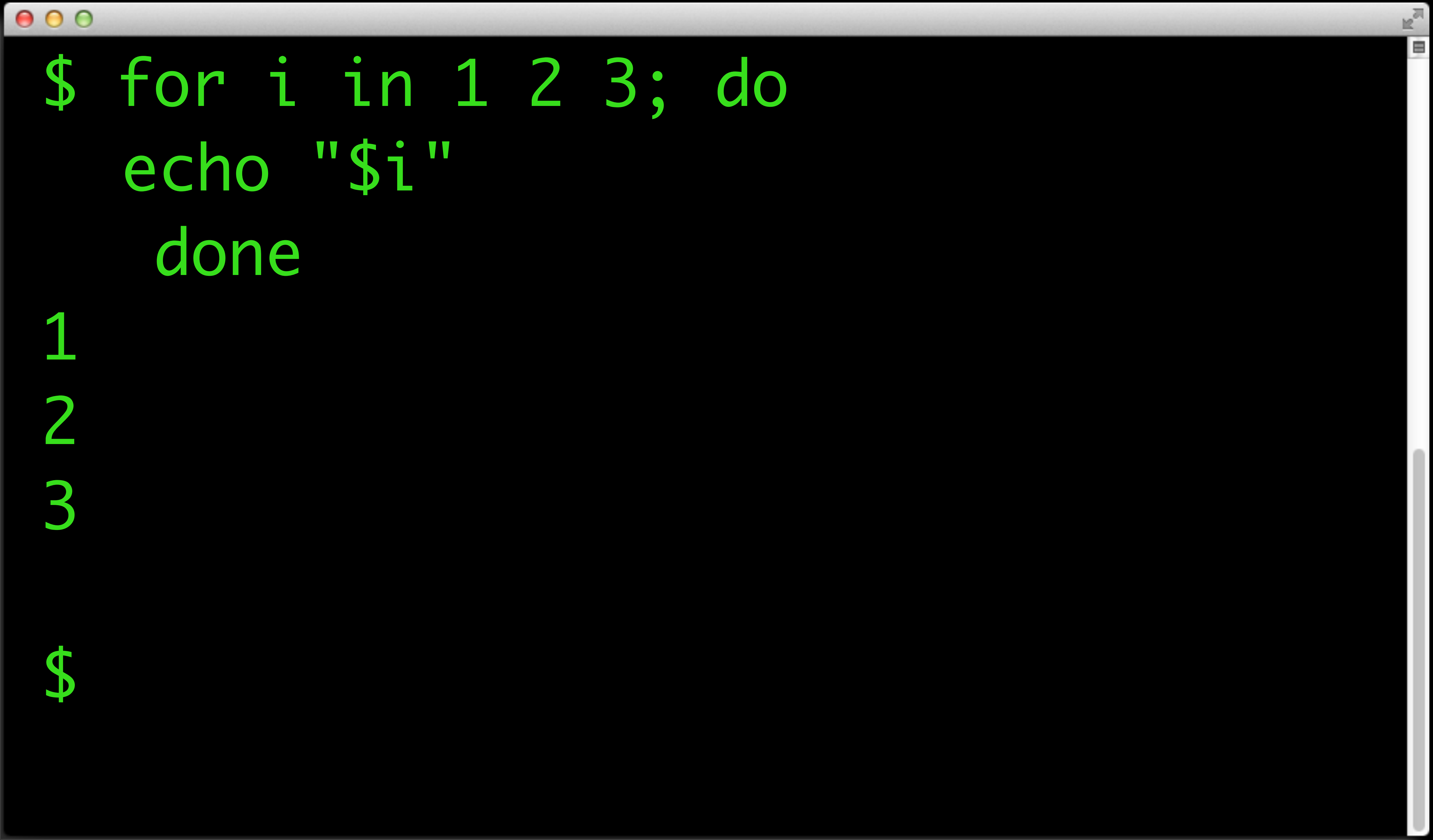
```
$ while read PathVAR; do
> file $PathVAR
> done < /etc/paths
/usr/bin: directory
/bin: directory
/usr/sbin: directory
/sbin: directory
/usr/local/bin: directory
```



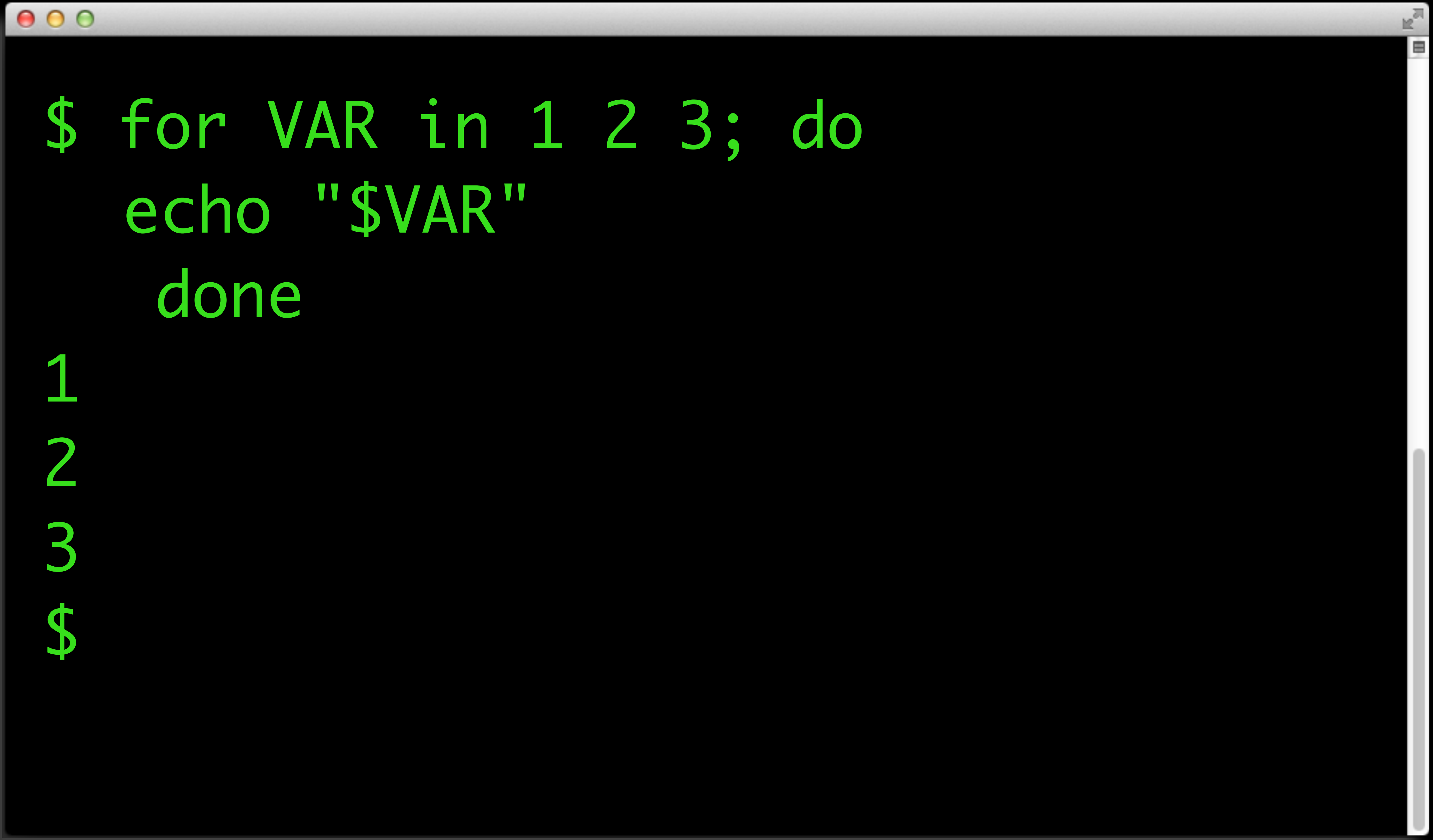
```
$ cat favAnimal.sh
  FavoriteAnimal=Dog
  while [[ $FavoriteAnimal != '' ]]; do
    echo 'guess my fav animal : '
    read guess
     if [[ $guess == $FavoriteAnimal ]] then
      echo Right
     else
      echo Nope
     fi
   done
```

## For Loops

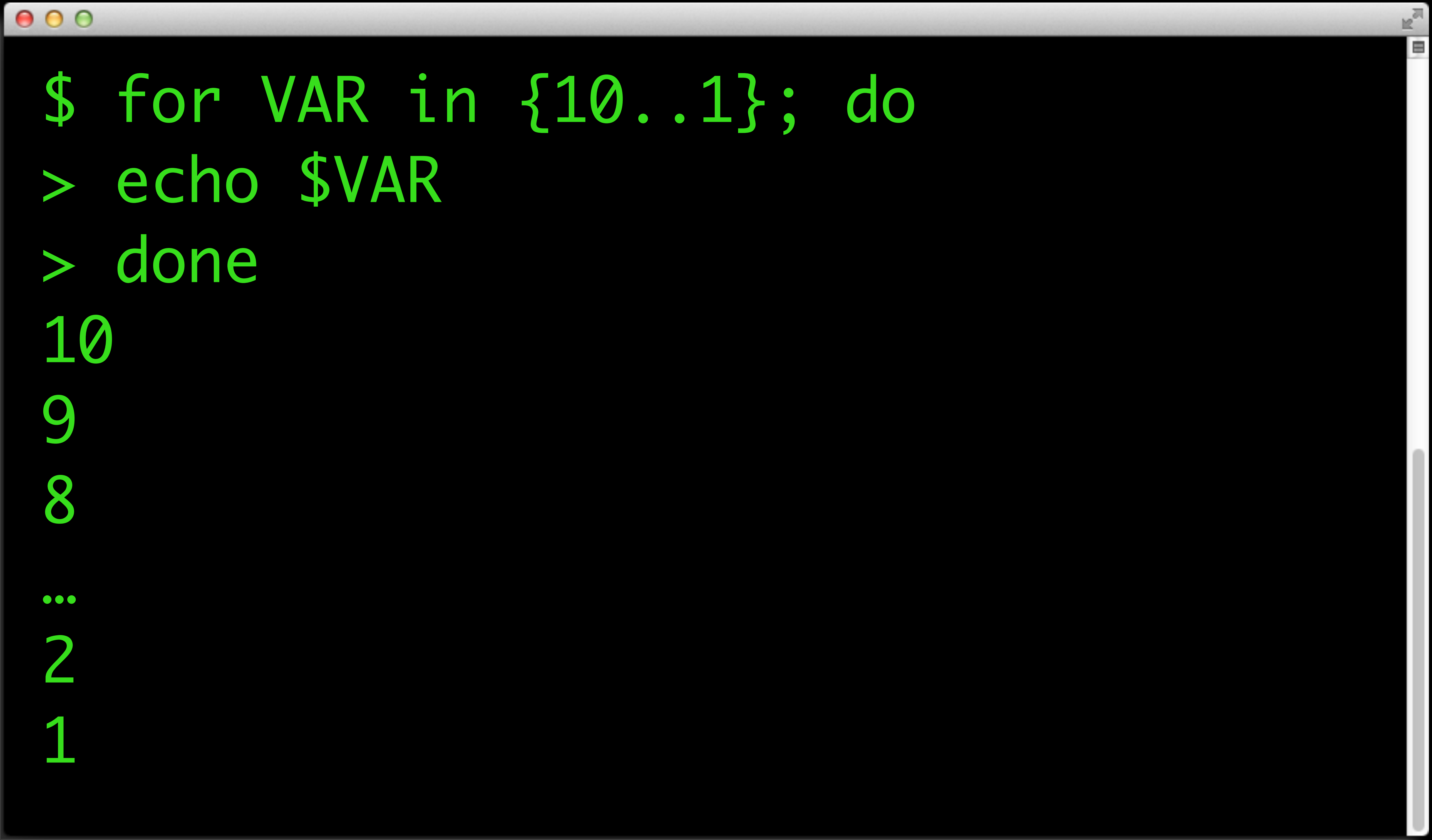
- Repeat Commands
- Pass Arguments for each loop from:
  - A command substation
  - A list/range of characters



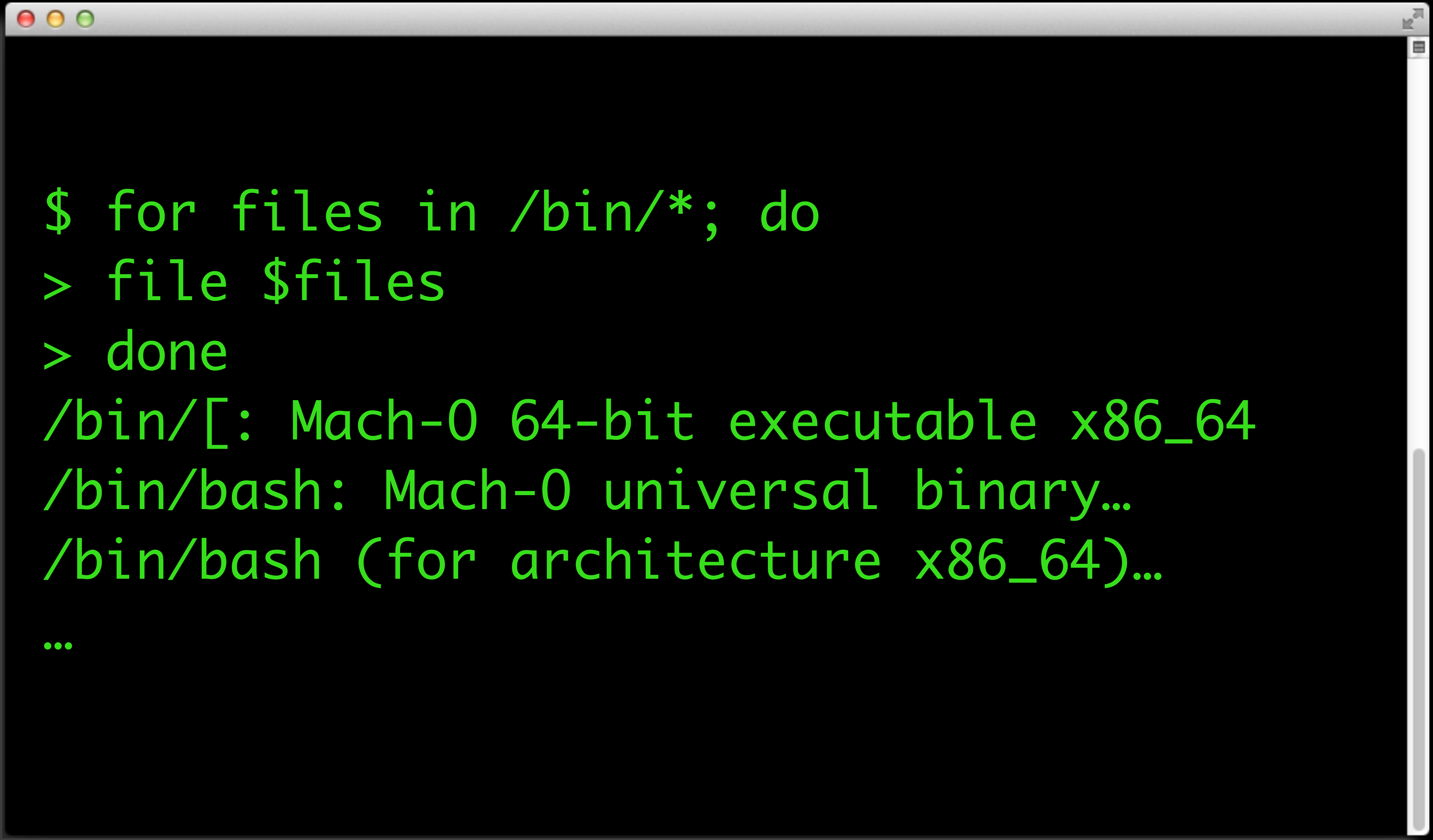
```
$ for i in 1 2 3; do
   echo "$i"
    done
1
2
3

$
```



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



```
$ for VAR in {10..1}; do
> echo $VAR
> done
10
9
8
…
2
1
```



```
$ for files in /bin/*; do
> file $files
> done
/bin/[: Mach-O 64-bit executable x86_64
/bin/bash: Mach-O universal binary…
/bin/bash (for architecture x86_64)…
…
```

## defaults

- Read/Write OS X Preferences
- CFPreferences
- .plist

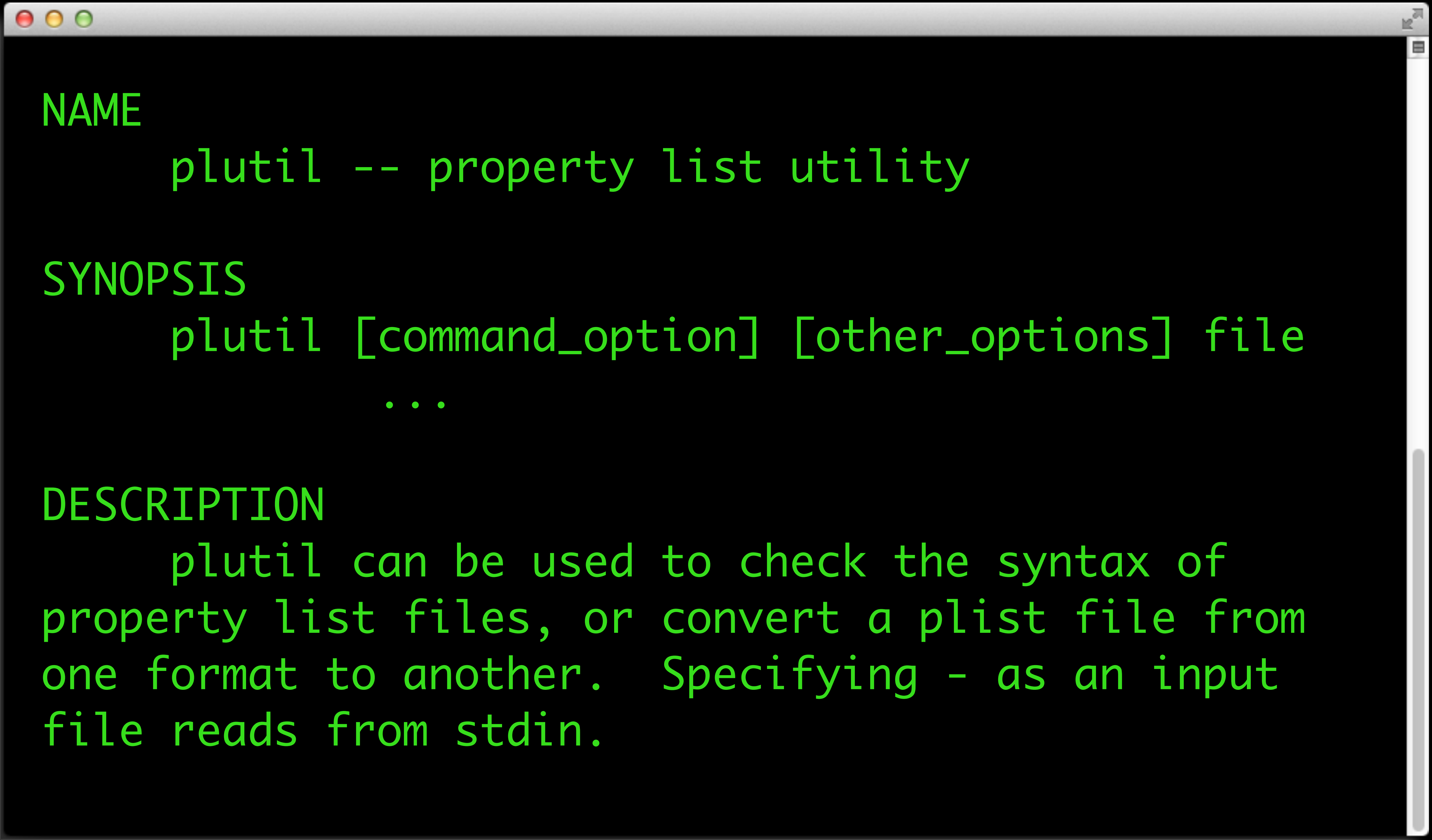
```
defaults read preference key
defaults write preference key -type value
```

## Plists

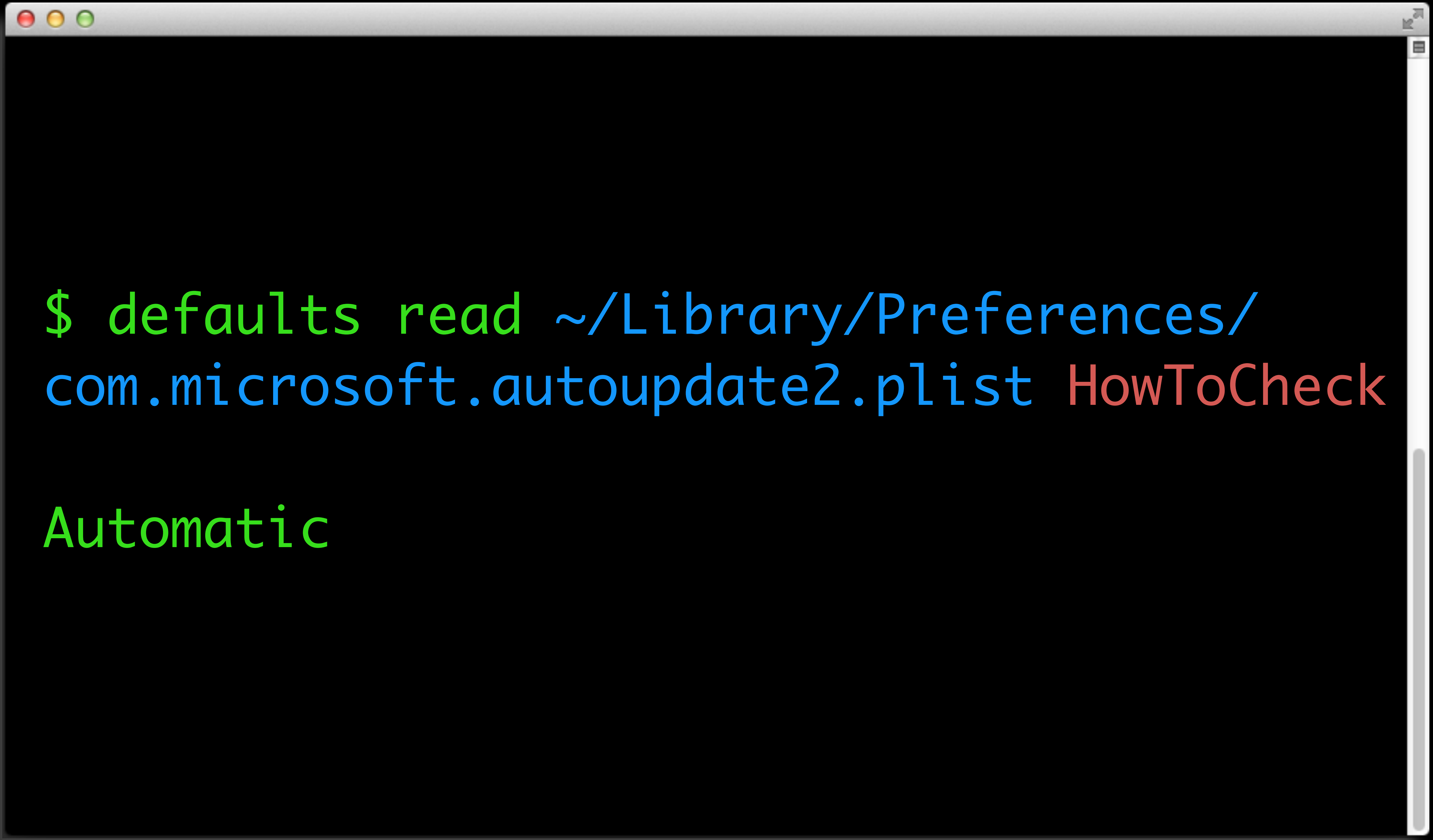
- App/User/System Preferences
  - Key/Value pairs
    - Strings
    - Integers
    - Boolean
    - Arrays
- Binary or XLM format

# plutil

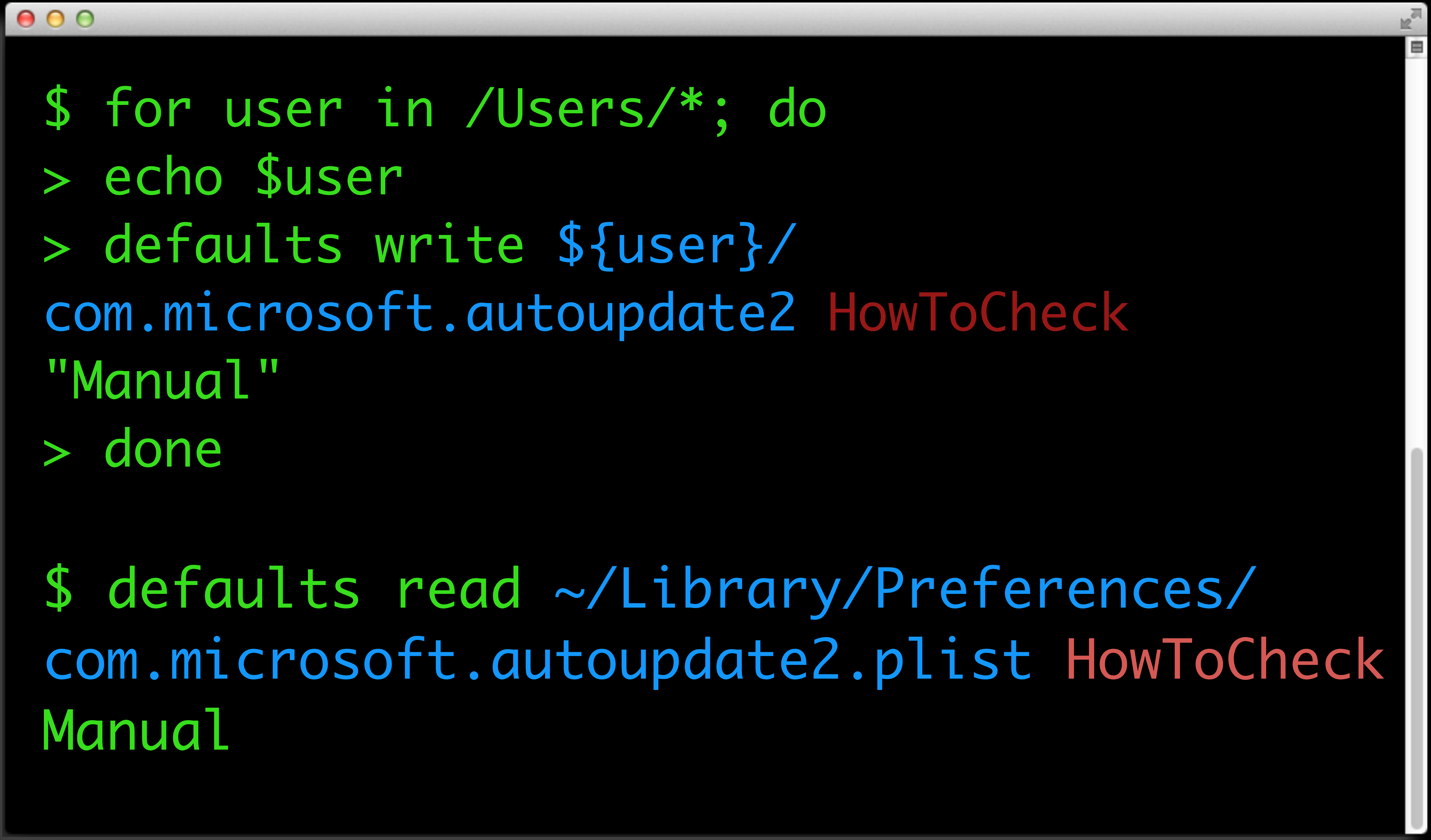
- `plutil -convert xml1 filename`
- `plutil -convert binary filename`



```
NAME
     plutil -- property list utility

SYNOPSIS
     plutil [command_option] [other_options] file
             ...

DESCRIPTION
     plutil can be used to check the syntax of
property list files, or convert a plist file from
one format to another.  Specifying - as an input
file reads from stdin.
```



```
$ defaults read ~/Library/Preferences/
com.microsoft.autoupdate2.plist HowToCheck

Automatic
```

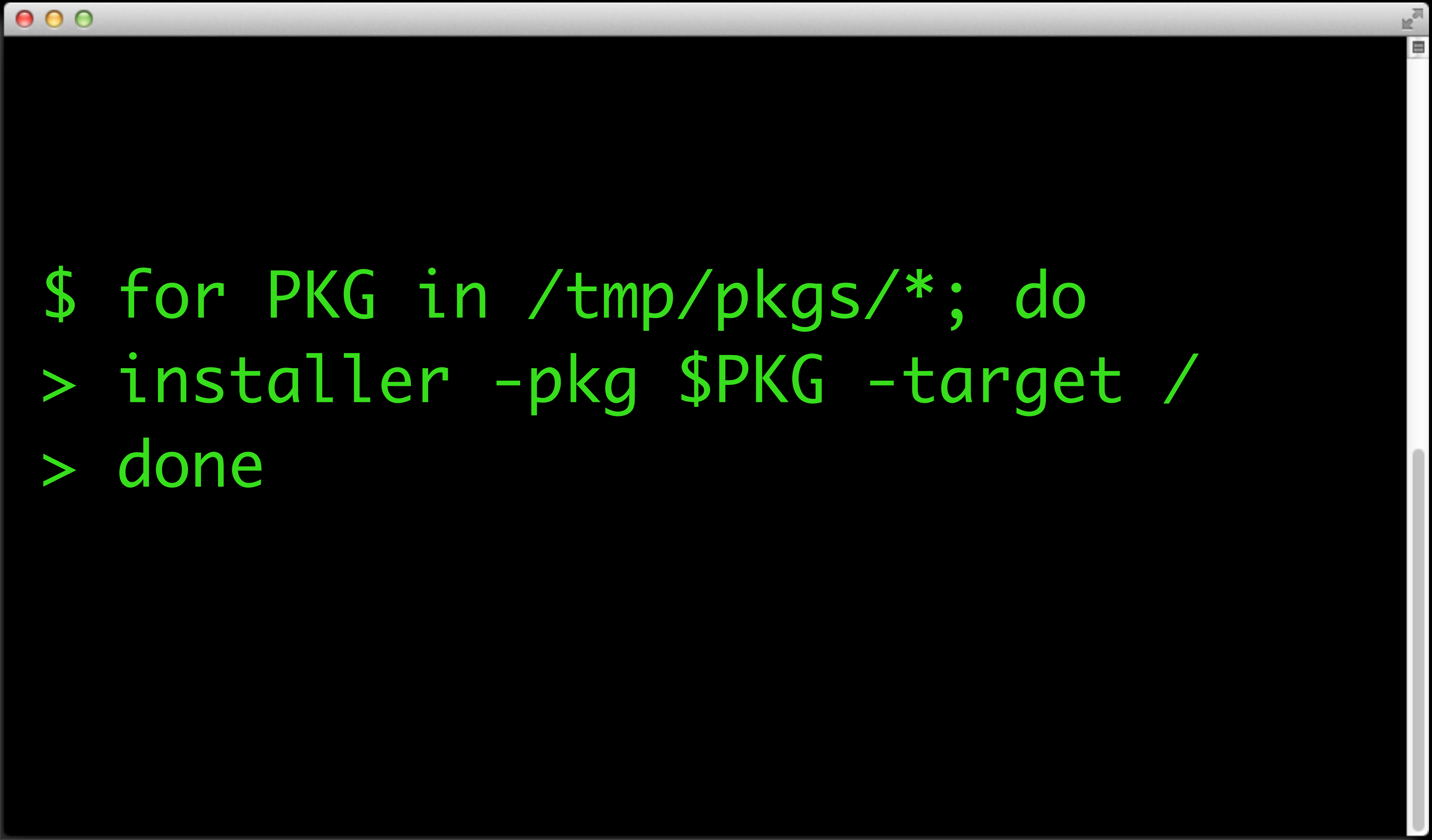


```
$ for user in /Users/*; do
> echo $user
> defaults write ${user}/
com.microsoft.autoupdate2 HowToCheck
"Manual"
> done

$ defaults read ~/Library/Preferences/
com.microsoft.autoupdate2.plist HowToCheck
Manual
```

## installer

- Package installer command
- `installer -pkg arg1 -target arg2`

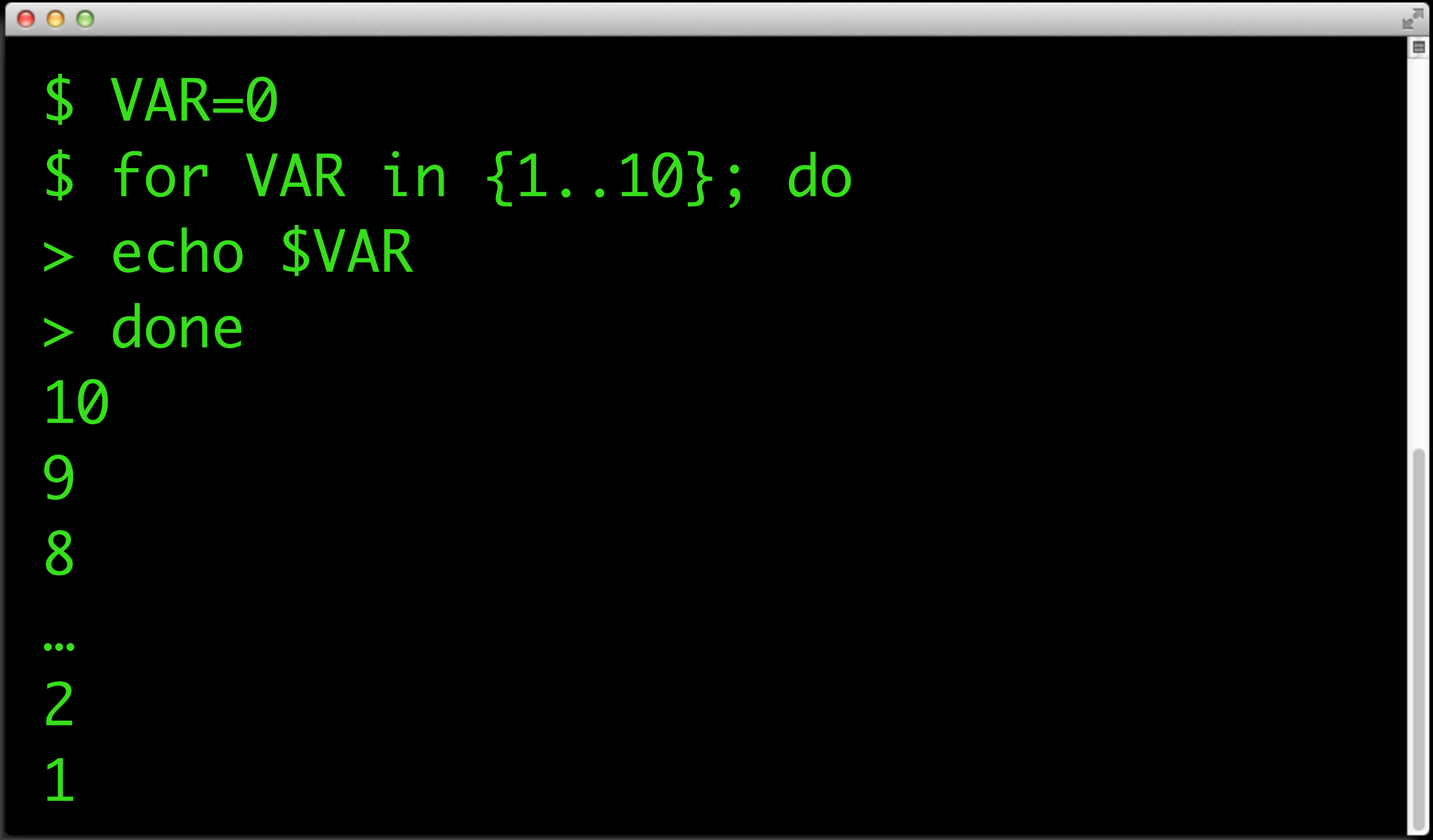
```
installer -pkg /tmp/k2Client.pkg
 -target /
```



```
$ for PKG in /tmp/pkgs/*; do
> installer -pkg $PKG -target /
> done
```

## Try It!

- for loop over directory
- for i in /path/*
- for loop 10 times



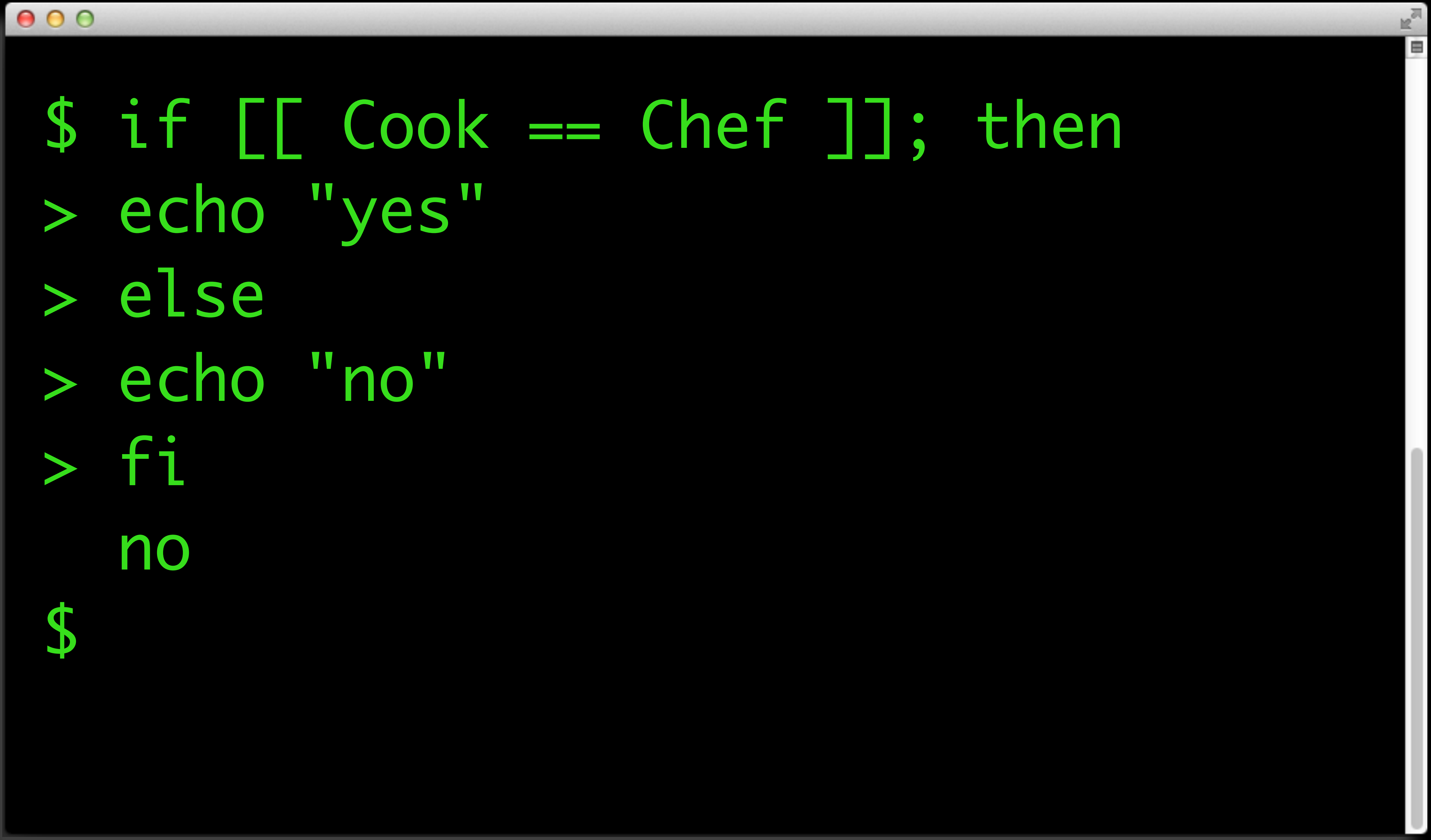
```
$ VAR=0
$ for VAR in {1..10}; do
> echo $VAR
> done
10
9
8
…
2
1
```

## Tests

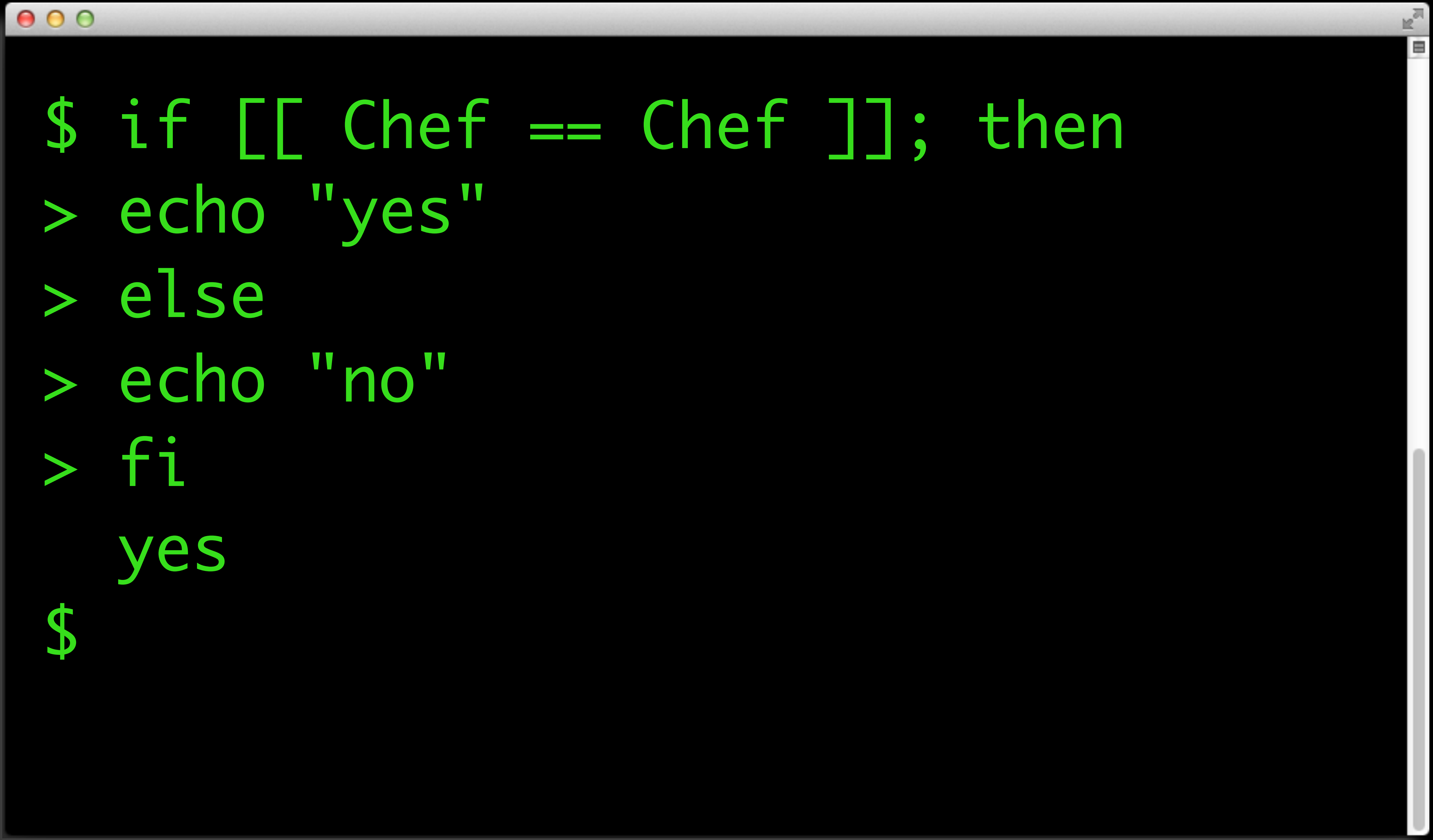
- True/False
- If condition is true
  - do something!
- else
  - do something else!

## String Comparison

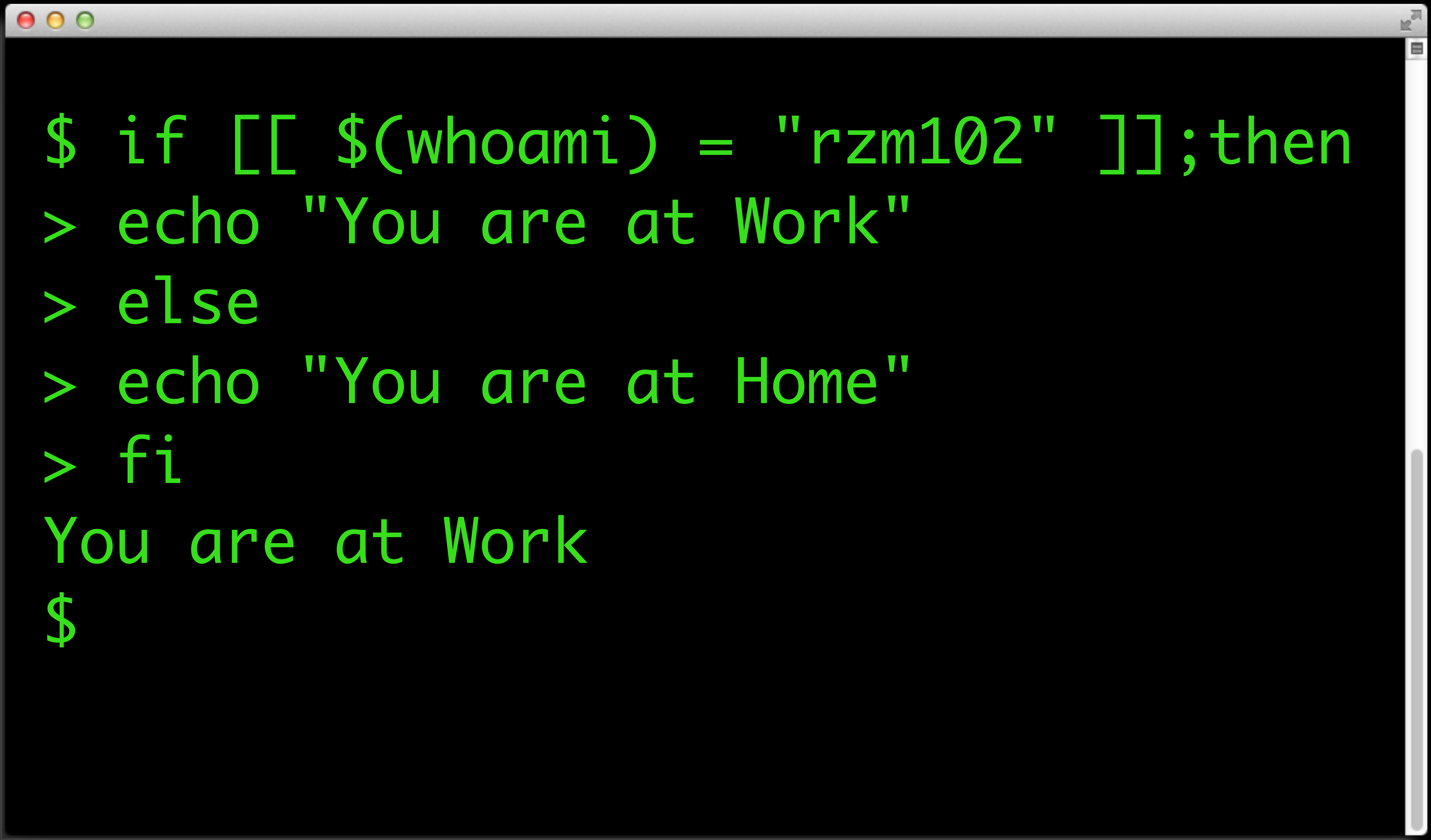
- Is Equal To:
  `[[ "$string1" == "$string2" ]]`
- Is NOT Equal:
  `[[ "$string1" != "$string2" ]]`
- String is Null:
  `[[ -n "$string1" ]]`



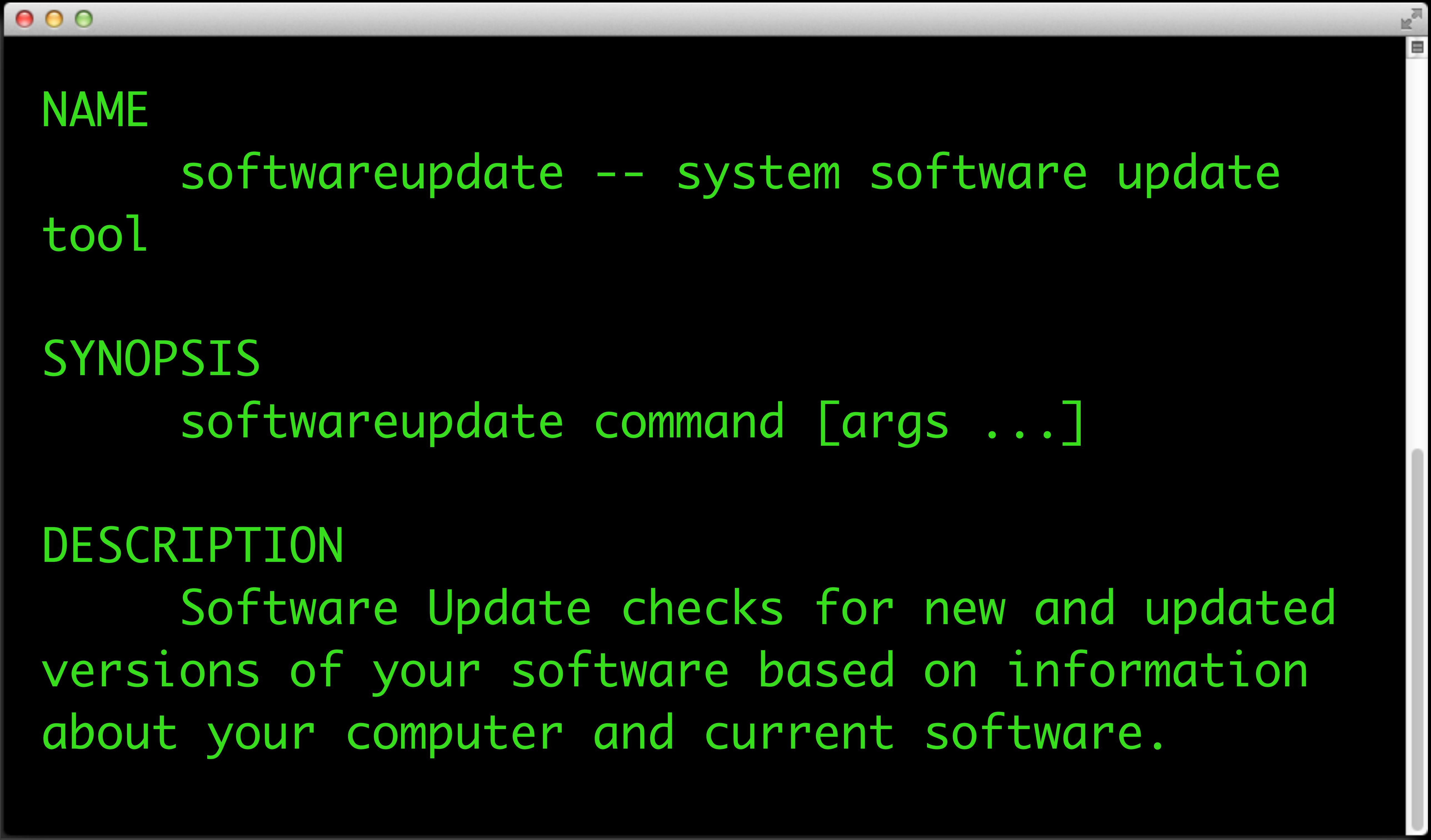
```
$ if [[ Cook == Chef ]]; then
> echo "yes"
> else
> echo "no"
> fi
  no
$
```



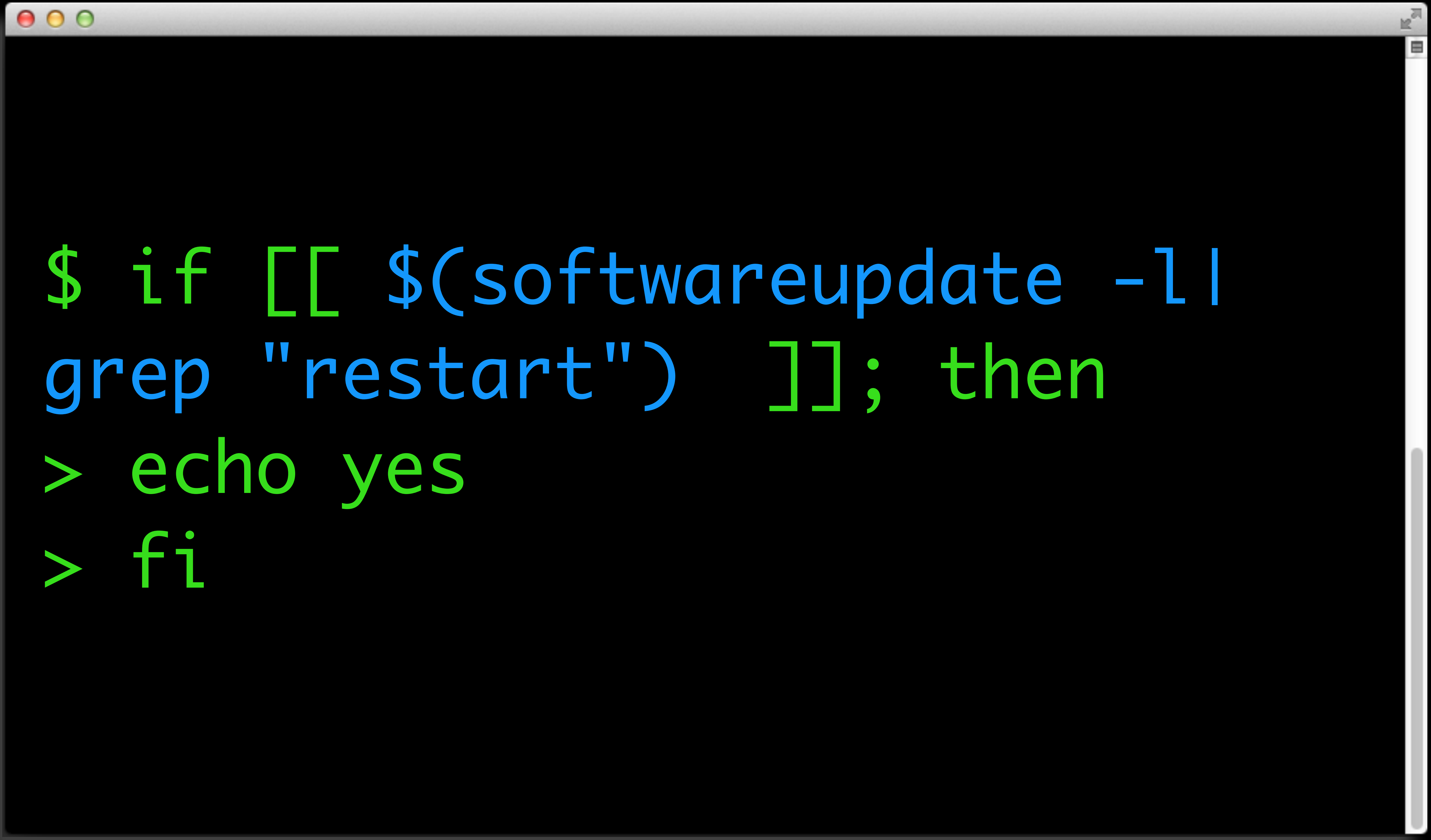
```
$ if [[ Chef == Chef ]]; then
> echo "yes"
> else
> echo "no"
> fi
  yes
$
```



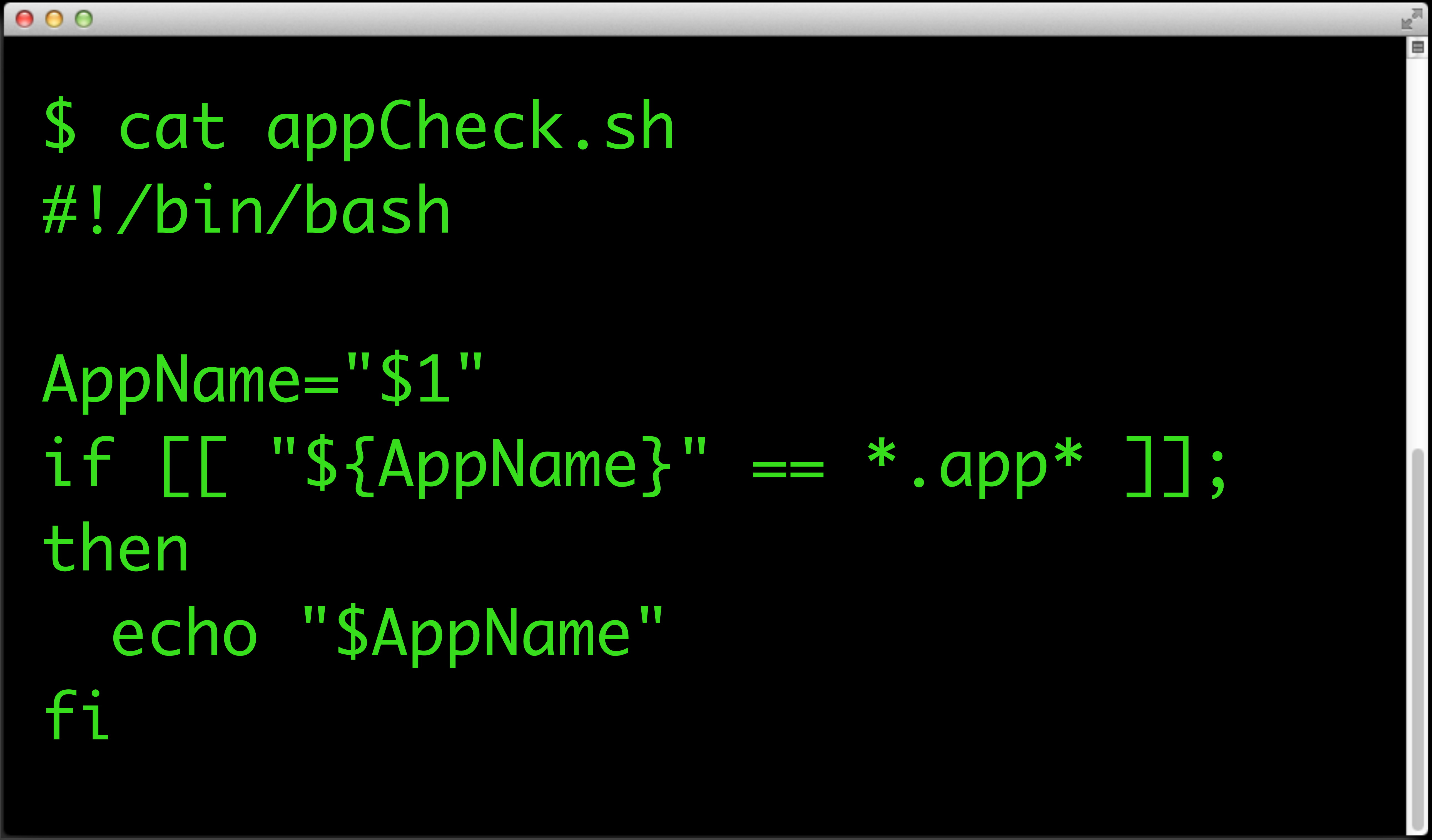
```
$ if [[ $(whoami) = "rzm102" ]];then
> echo "You are at Work"
> else
> echo "You are at Home"
> fi
You are at Work
$
```



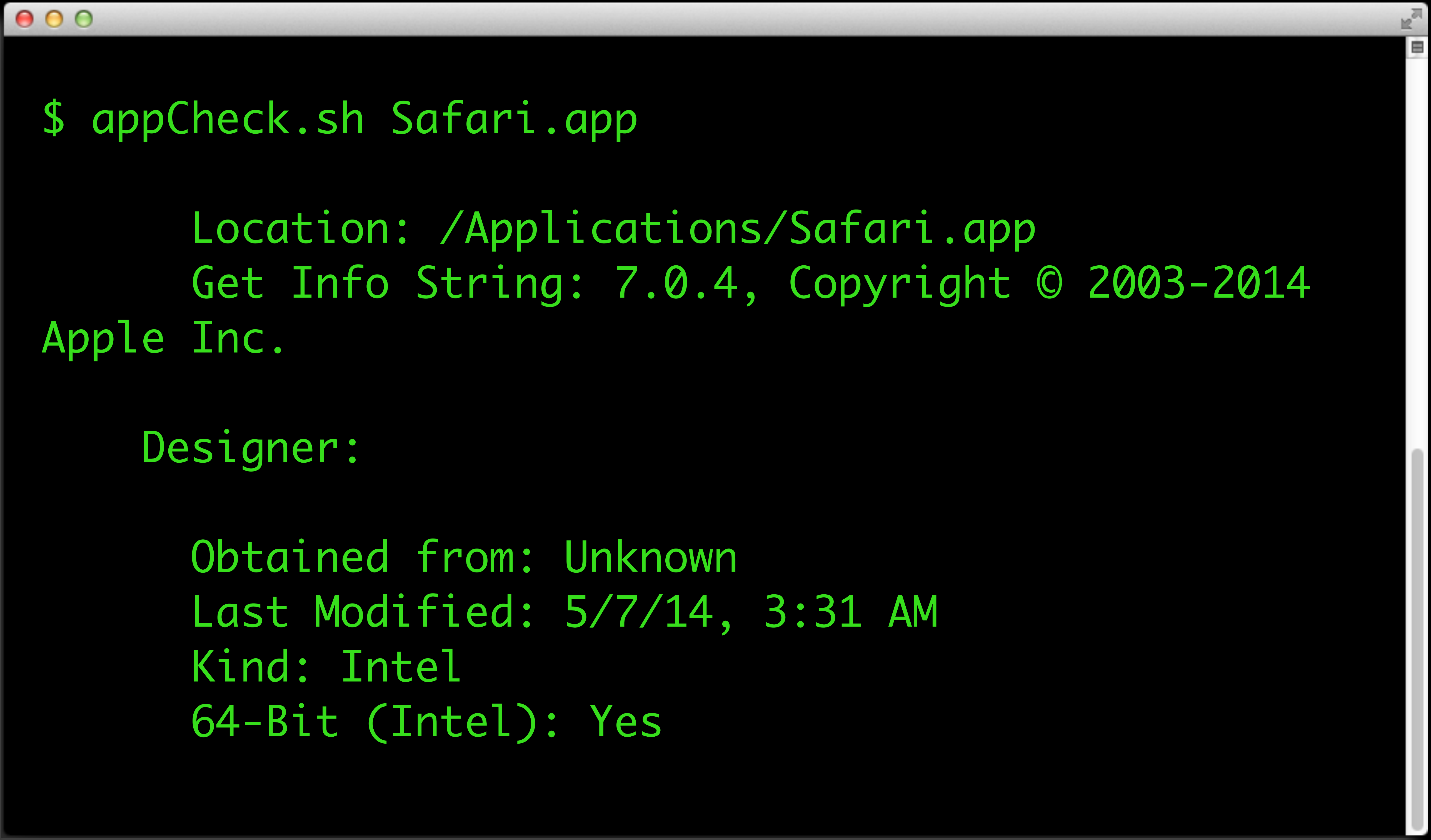
```
NAME
     softwareupdate -- system software update
tool

SYNOPSIS
     softwareupdate command [args ...]

DESCRIPTION
     Software Update checks for new and updated
versions of your software based on information
about your computer and current software.
```



```
$ if [[ $(softwareupdate -l|
grep "restart")  ]]; then
> echo yes
> fi
```



```
$ cat appCheck.sh
#!/bin/bash

AppName="$1"
if [[ "${AppName}" == *.app* ]];
then
  echo "$AppName"
fi
```

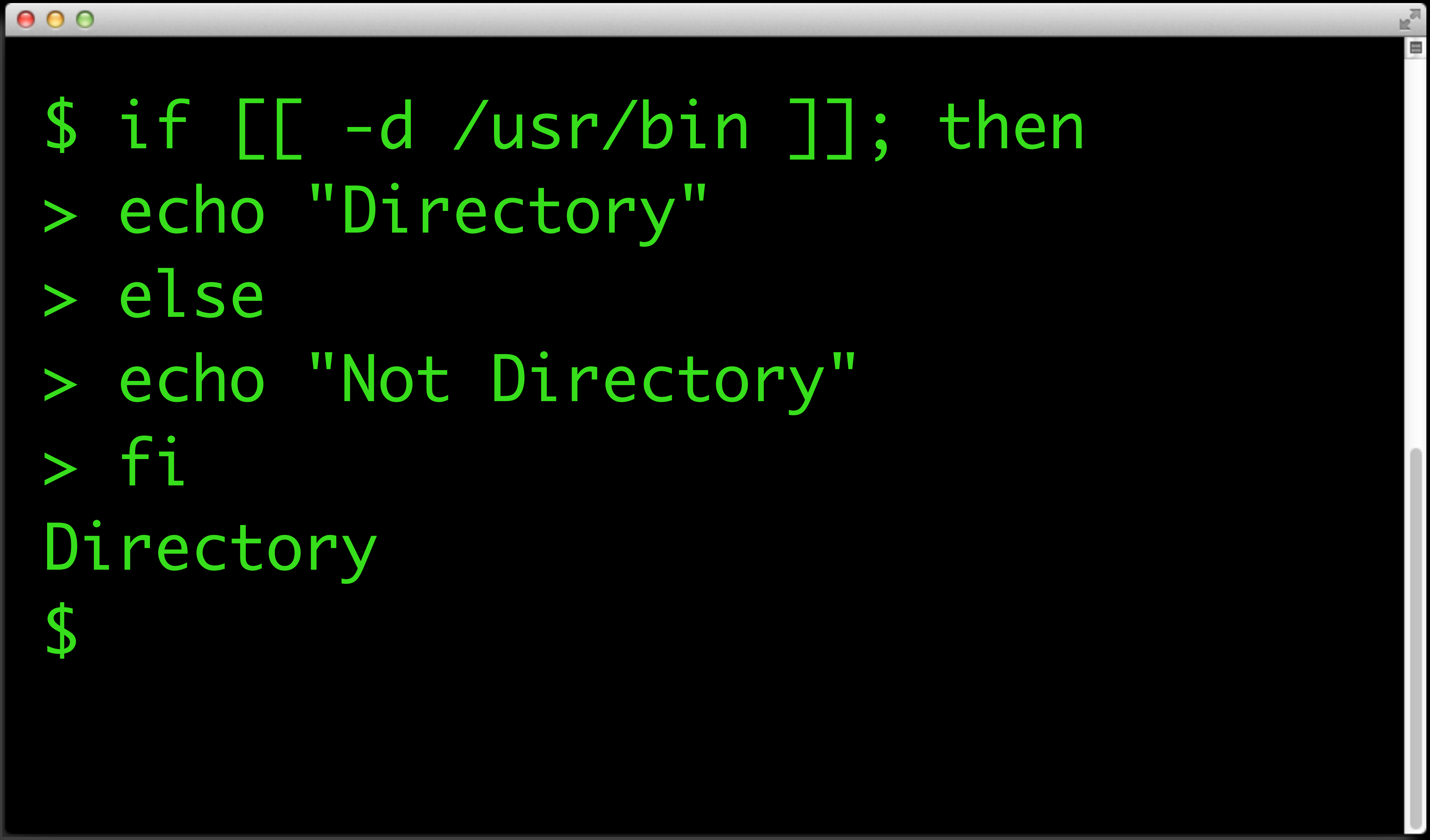


```
$ appCheck.sh Safari.app

      Location: /Applications/Safari.app
      Get Info String: 7.0.4, Copyright © 2003-2014
Apple Inc.

    Designer:

      Obtained from: Unknown
      Last Modified: 5/7/14, 3:31 AM
      Kind: Intel
      64-Bit (Intel): Yes
```

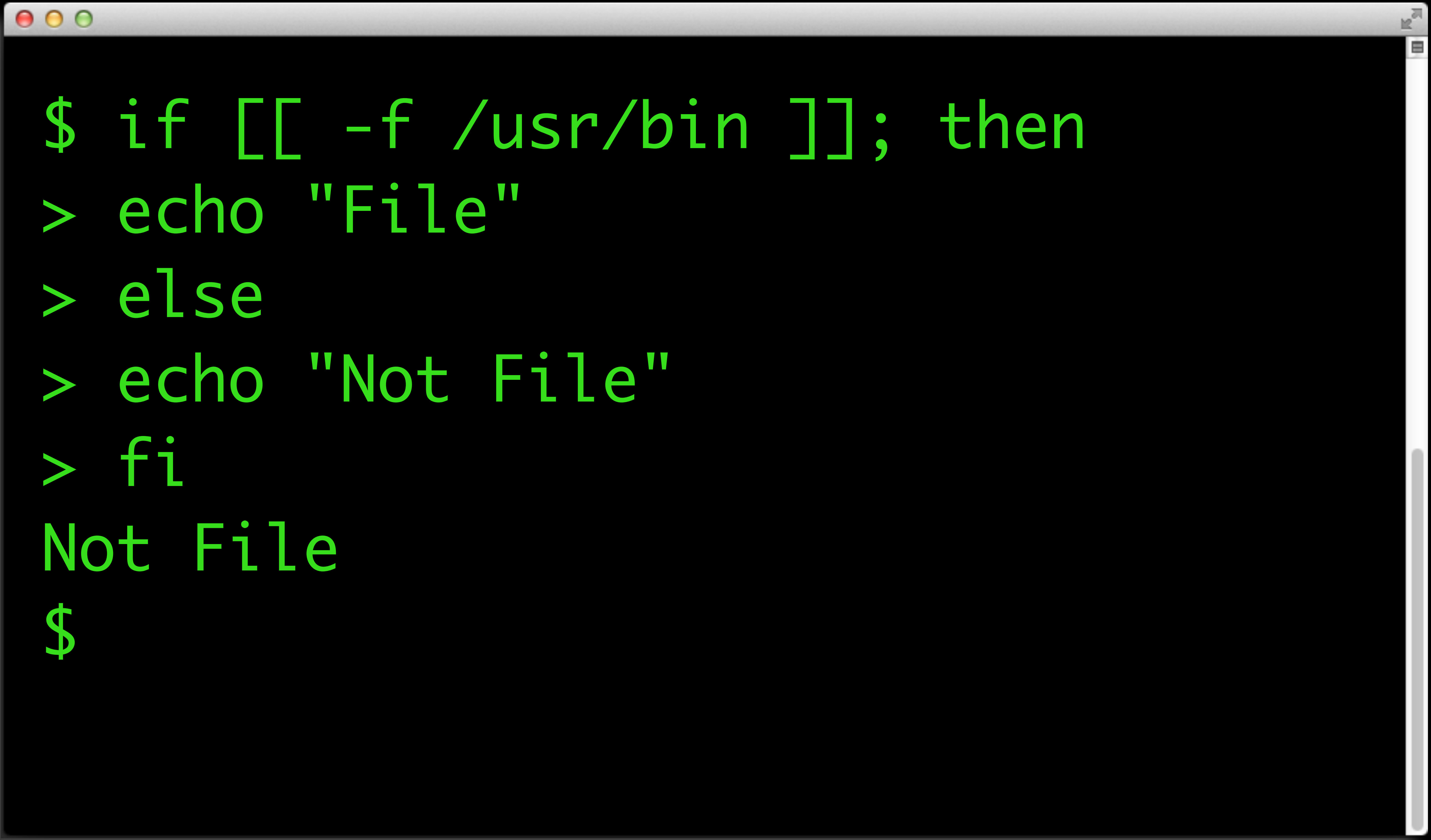
# File Tests

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

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



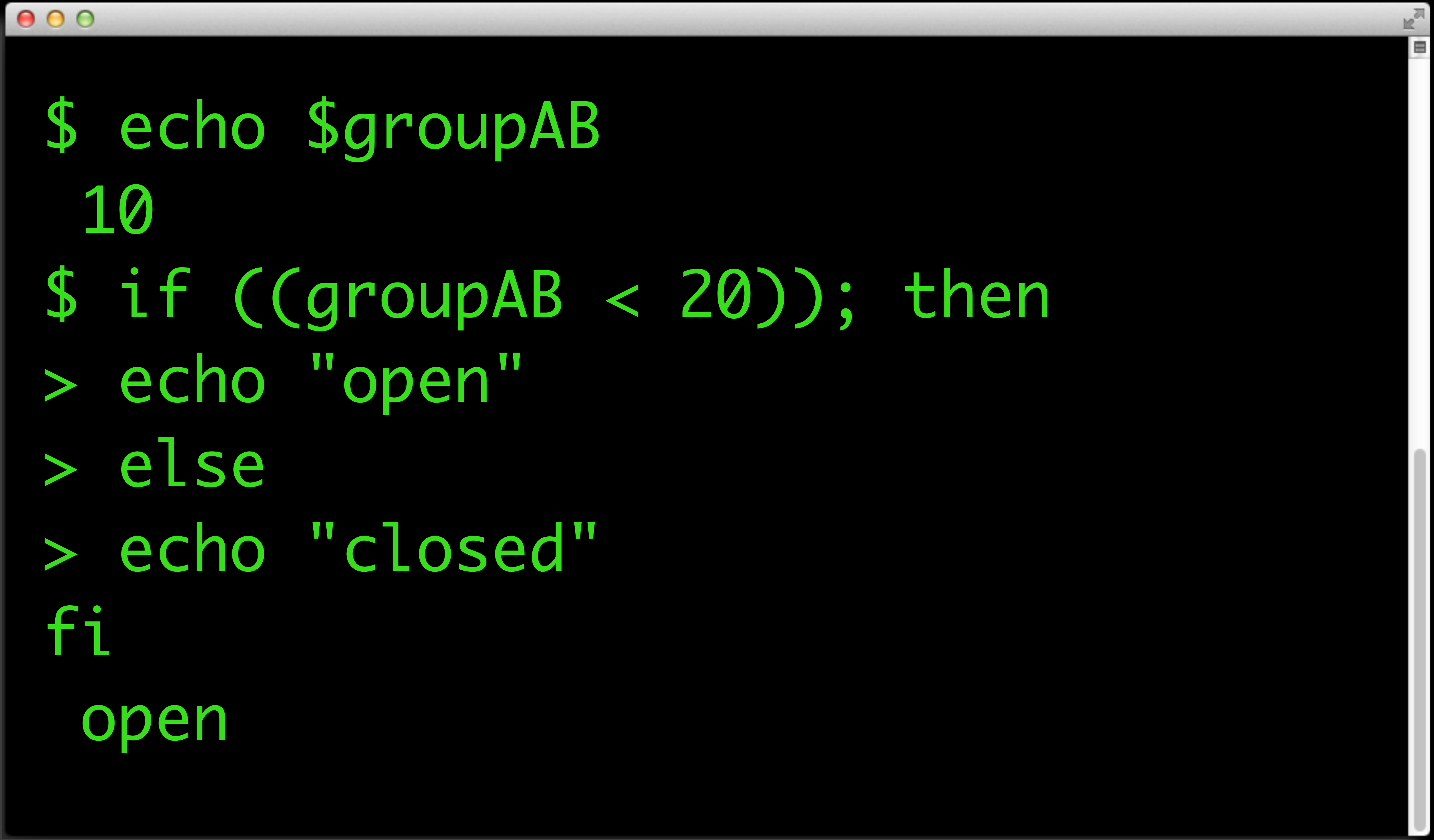
```
$ if [[ -d /usr/bin ]]; then
> echo "Directory"
> else
> echo "Not Directory"
> fi
Directory
$
```



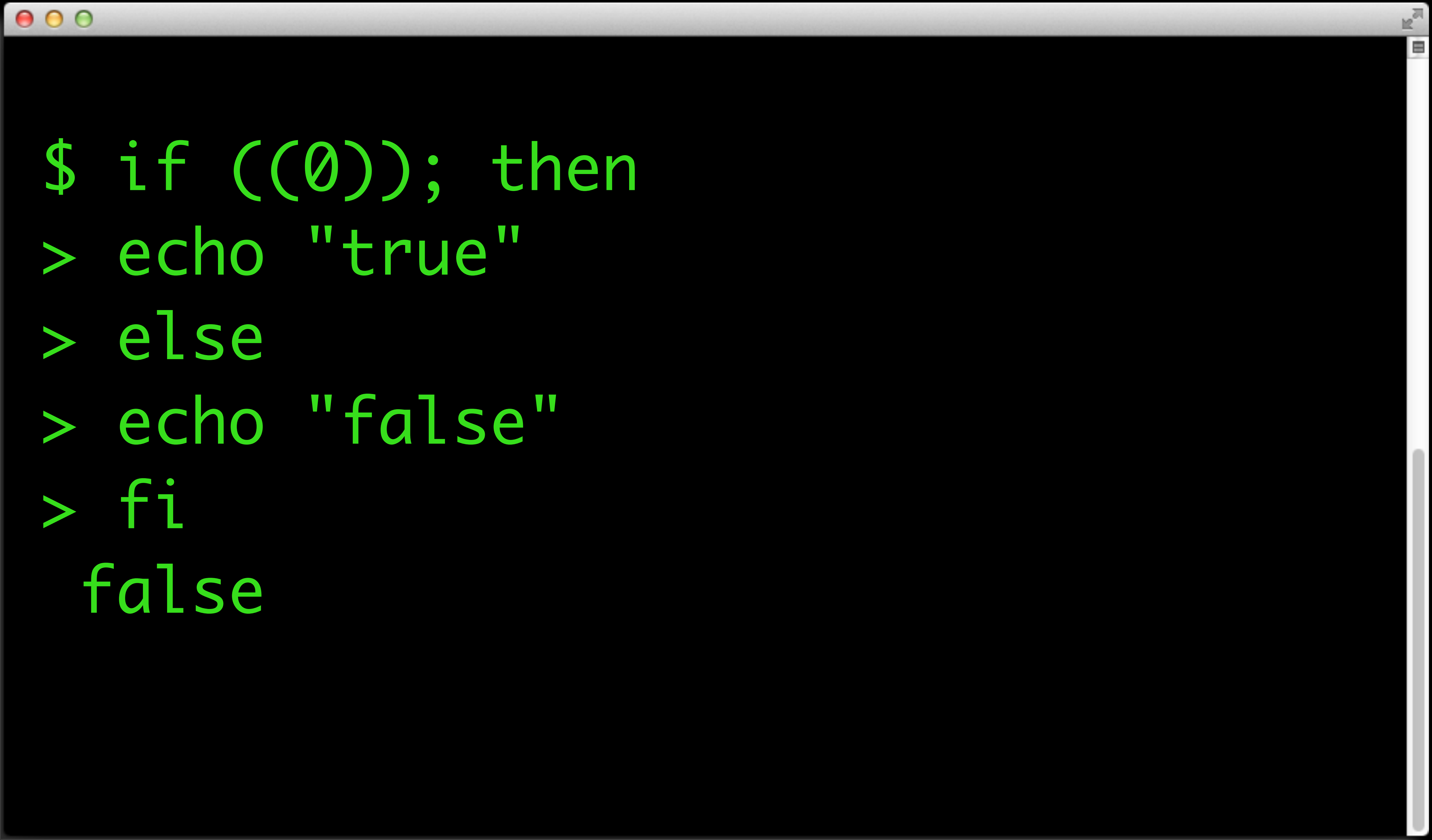
```
$ if [[ -f /usr/bin ]]; then
> echo "File"
> else
> echo "Not File"
> fi
Not File
$
```

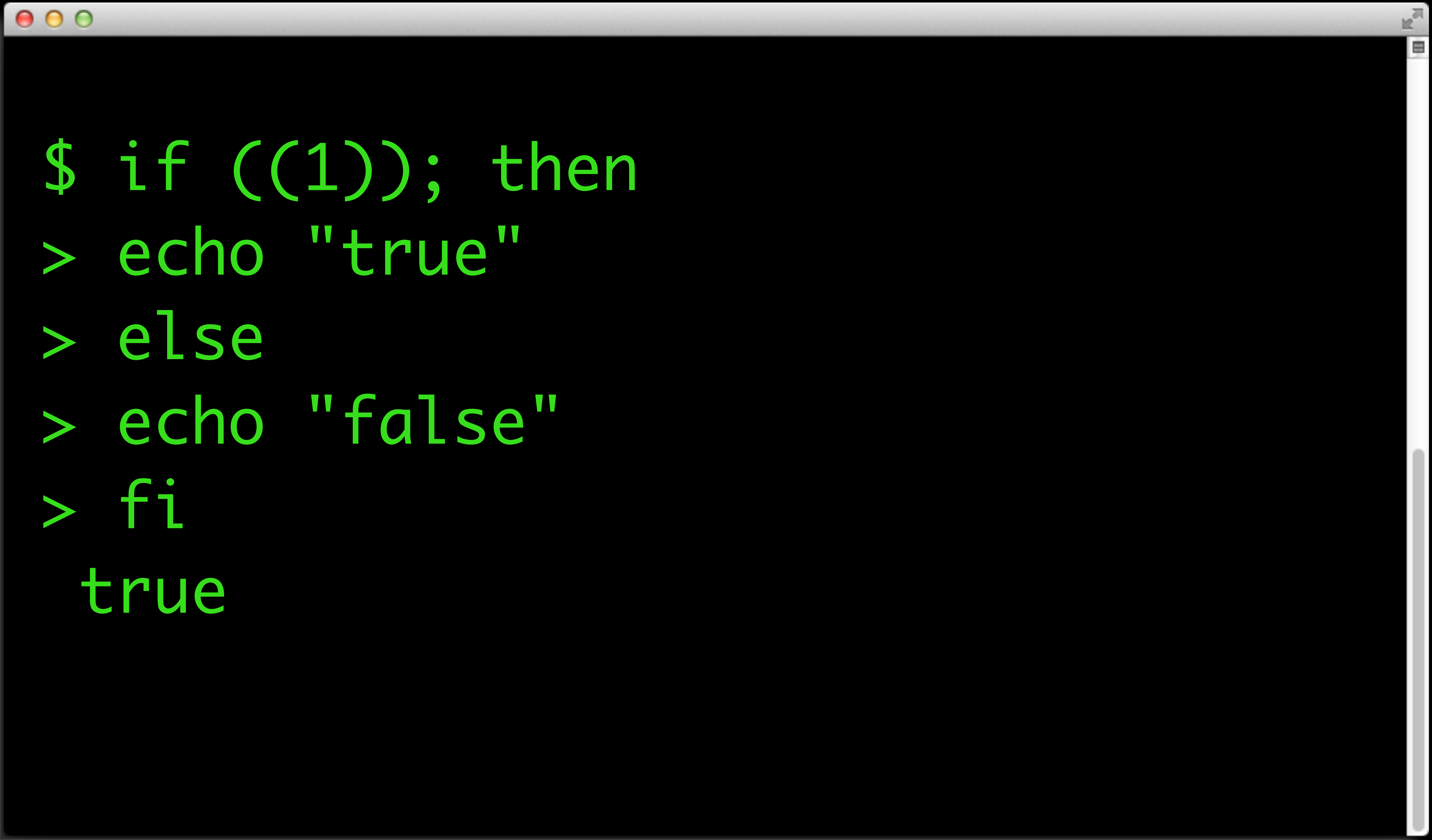
## Math Tests

- Check equations
- Supports variables
- Boolean: False = 0 / True >= 1



```
$ echo $groupAB
 10
$ if ((groupAB < 20)); then
> echo "open"
> else
> echo "closed"
fi
 open
```

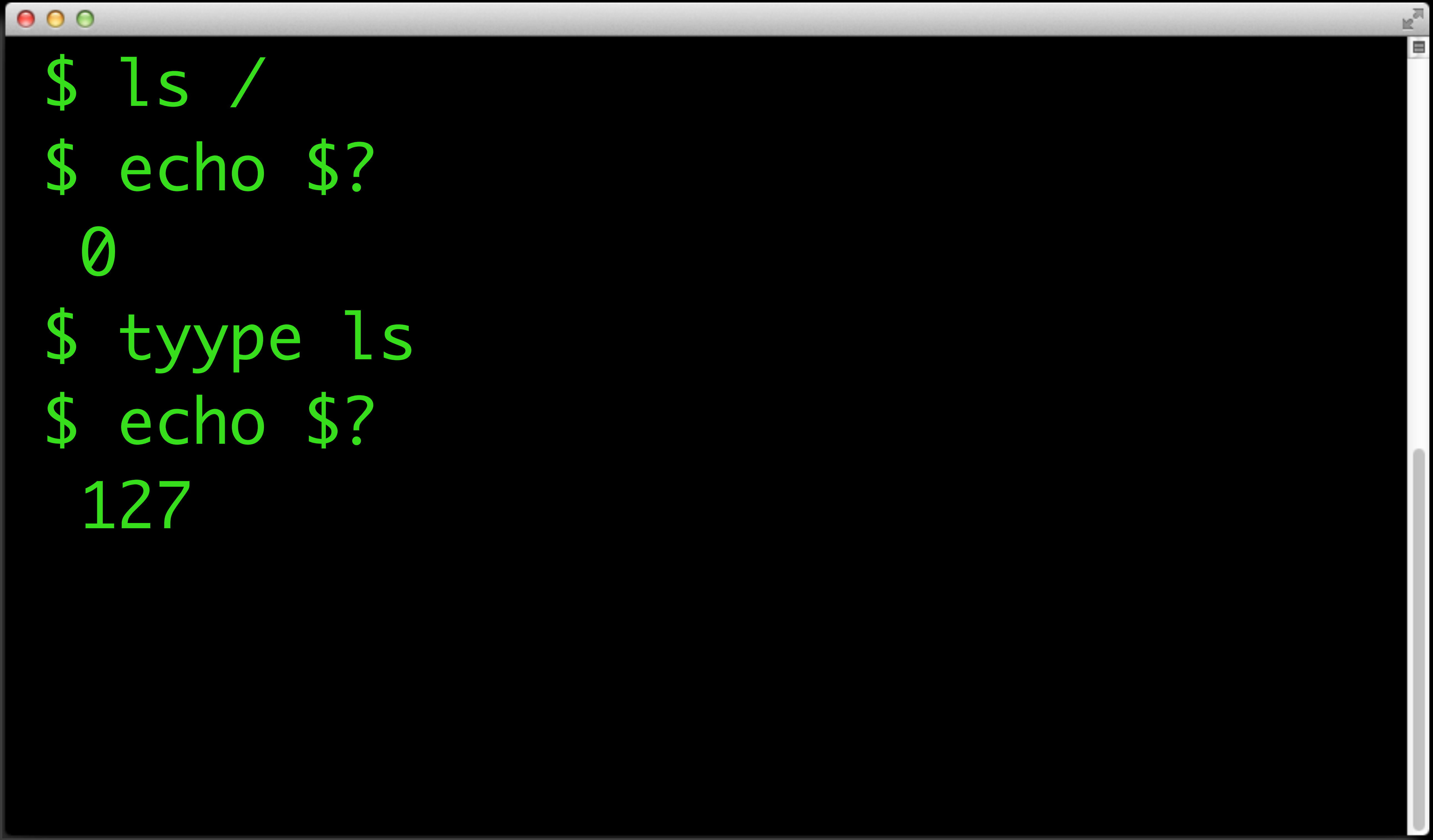

```
$ if ((0)); then
> echo "true"
> else
> echo "false"
> fi
 false
```



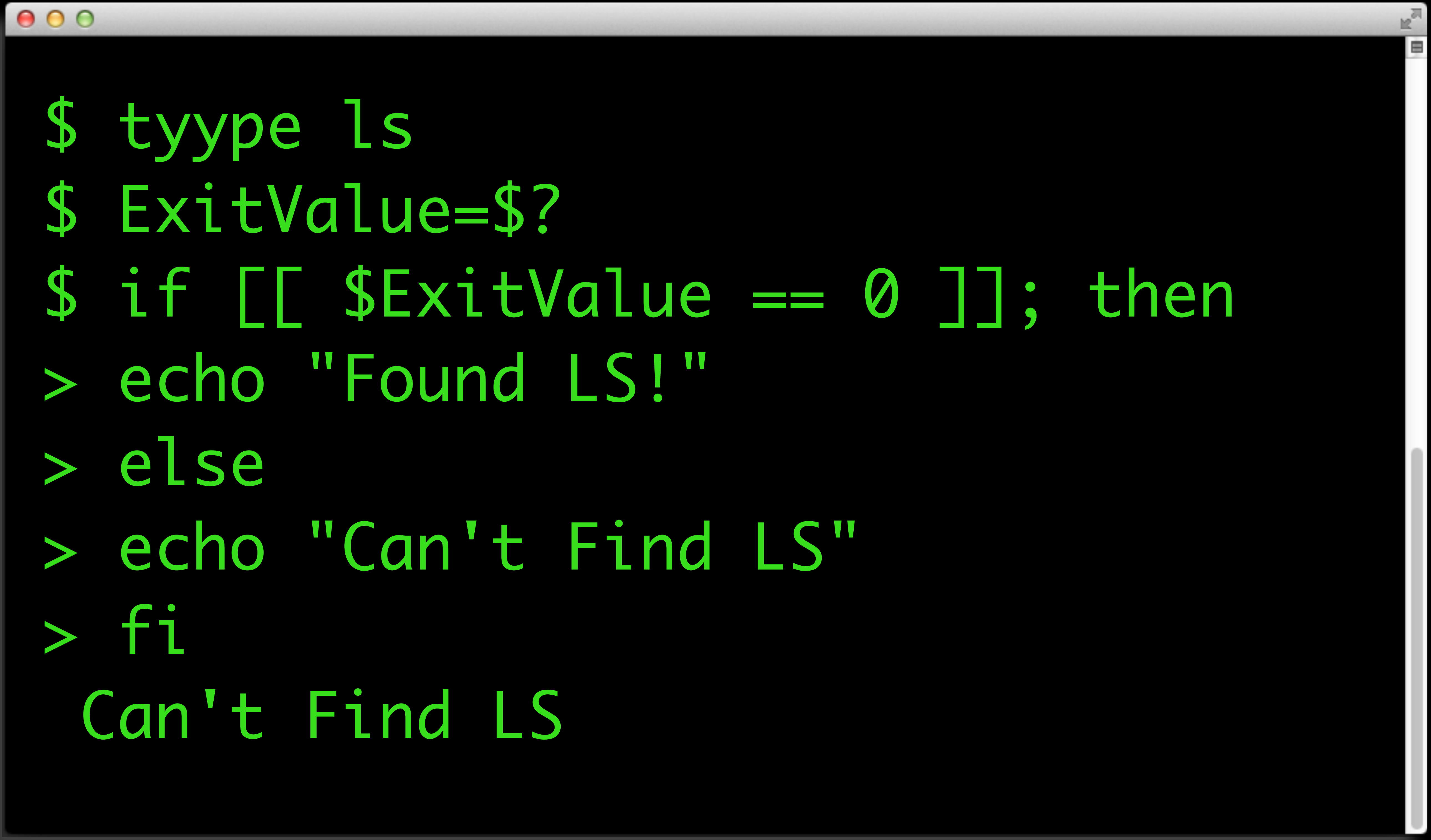
```
$ if ((1)); then
> echo "true"
> else
> echo "false"
> fi
 true
```

## Exit Status

- Value returned by executed command
- Numerical between 0-255
- Exit 0 means Success
- Number > 0 is Failure
- echo $? = Show Exit Value



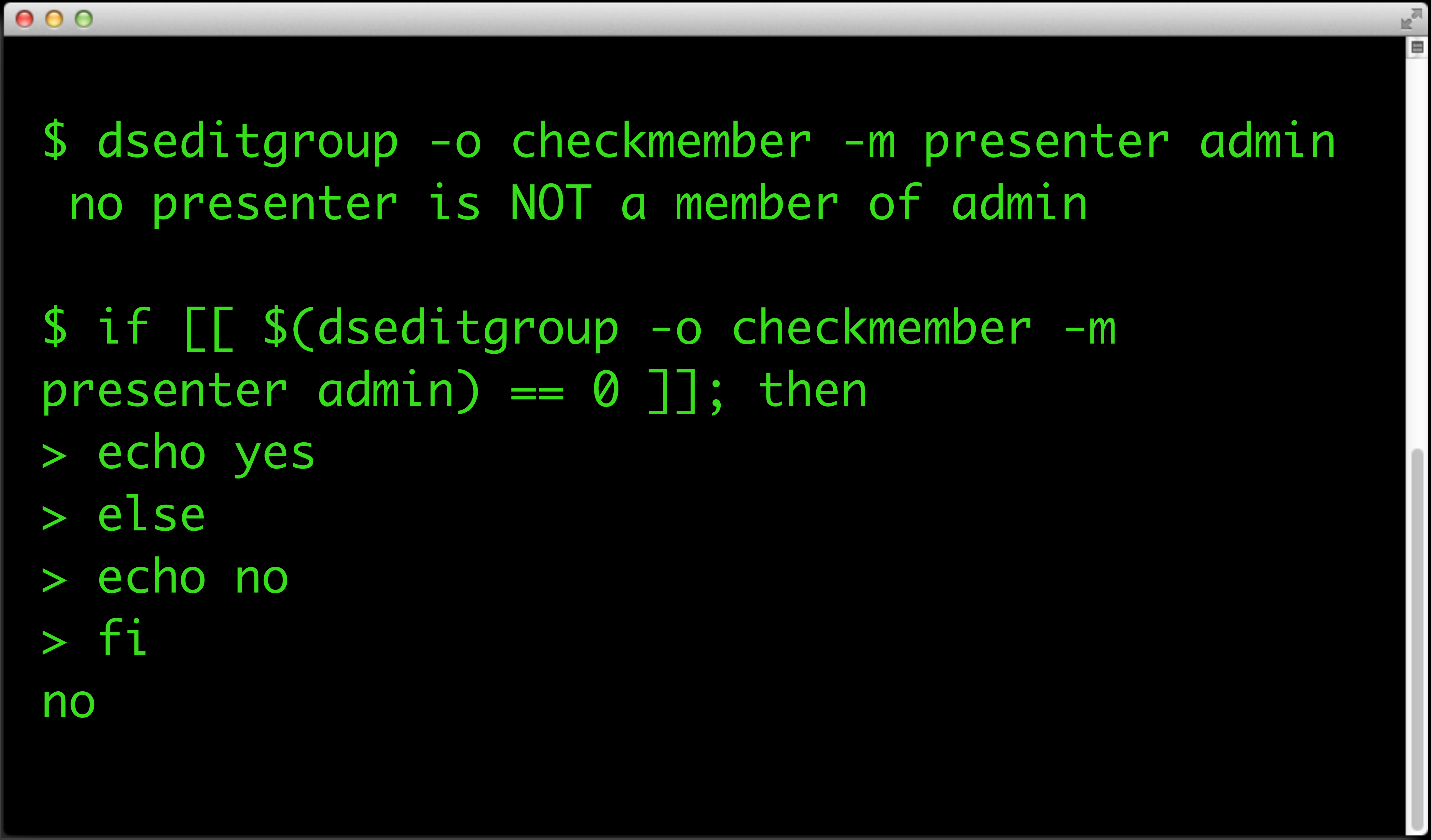
```
$ ls /
$ echo $?
 0
$ tyype ls
$ echo $?
 127
```



```
$ tyype ls
$ ExitValue=$?
$ if [[ $ExitValue == 0 ]]; then
> echo "Found LS!"
> else
> echo "Can't Find LS"
> fi
 Can't Find LS
```

## deseditgroup

- `dseditgroup -o checkmember -m user group`
- `dseditgroup -o checkmember -m presenter admin`



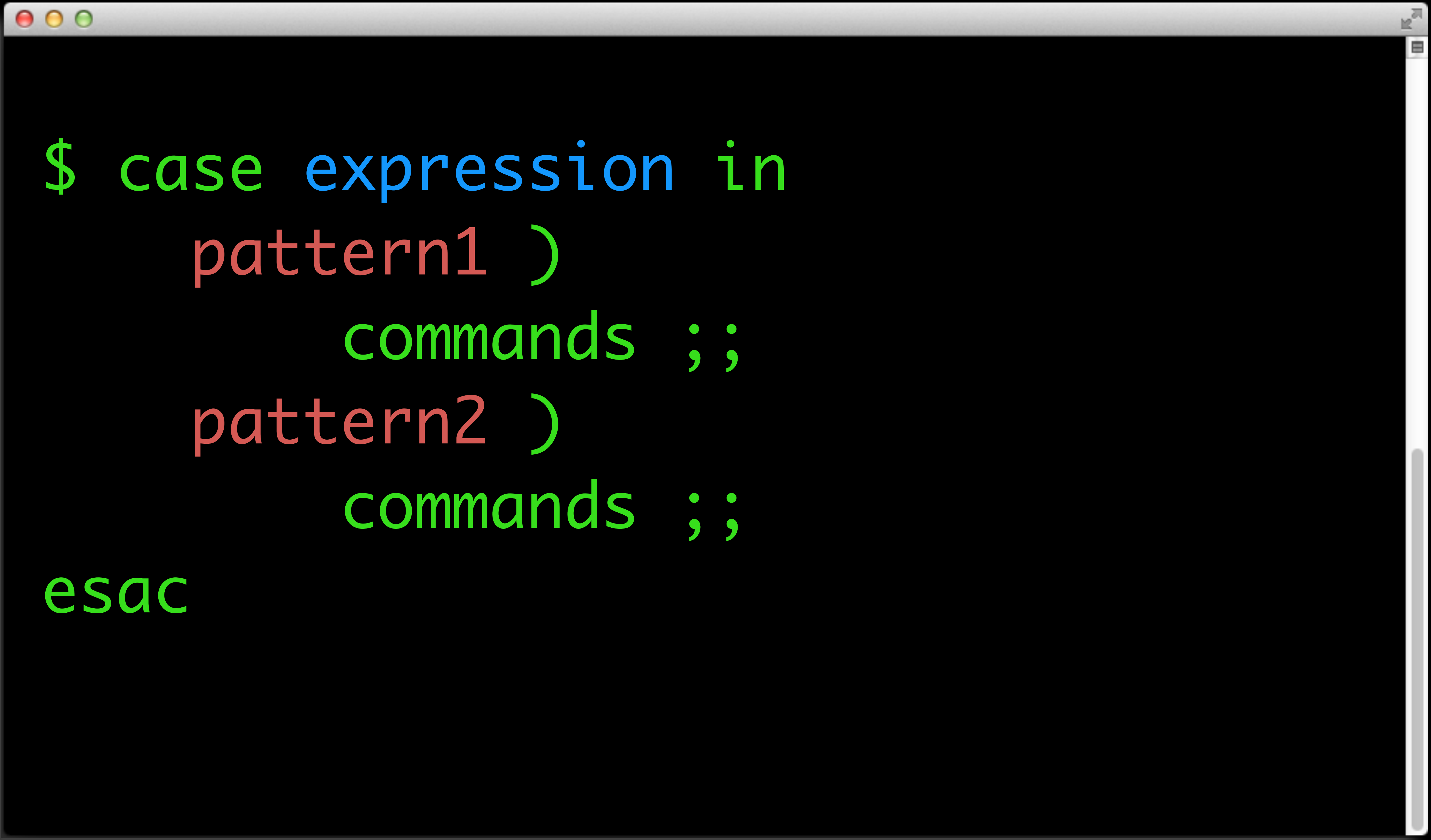
```
$ dseditgroup -o checkmember -m presenter admin
 no presenter is NOT a member of admin

$ if [[ $(dseditgroup -o checkmember -m
presenter admin) == 0 ]]; then
> echo yes
> else
> echo no
> fi
no
```

# Try It!

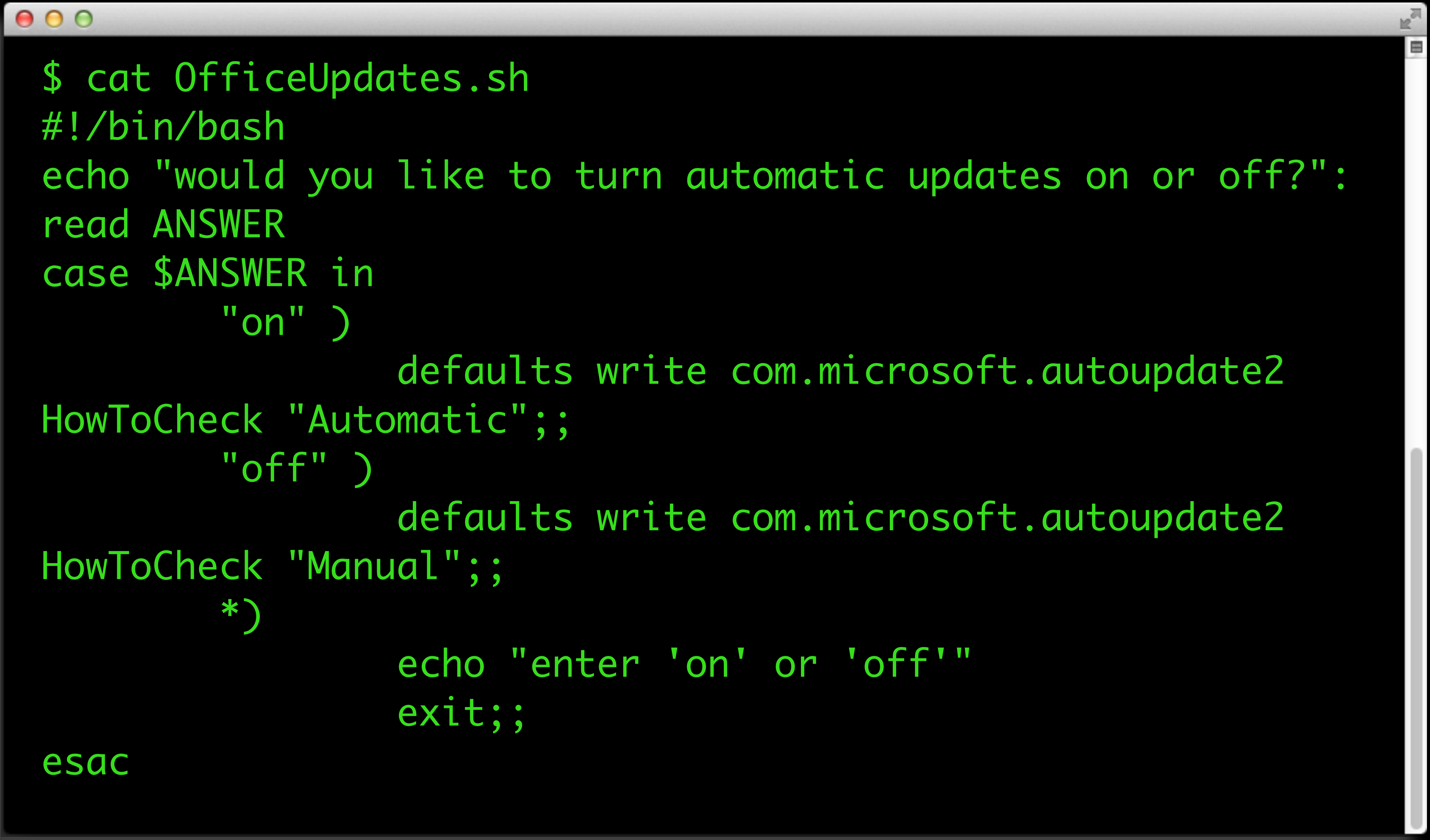
- Test a String and File
- Look at an exit status
- man dseditgroup

## case

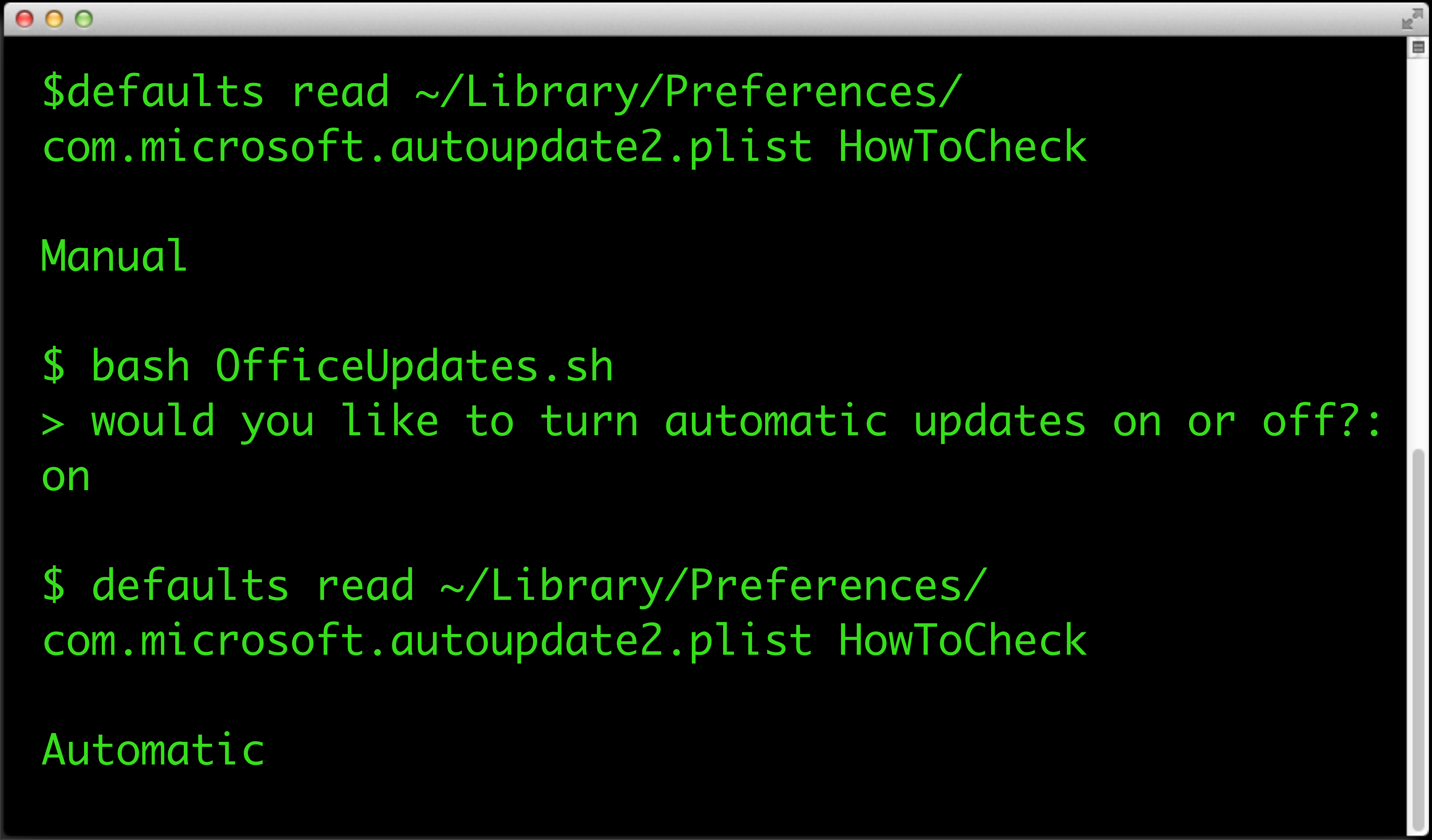
- Matching Patterns & Execute Commands
- Create Menu for script



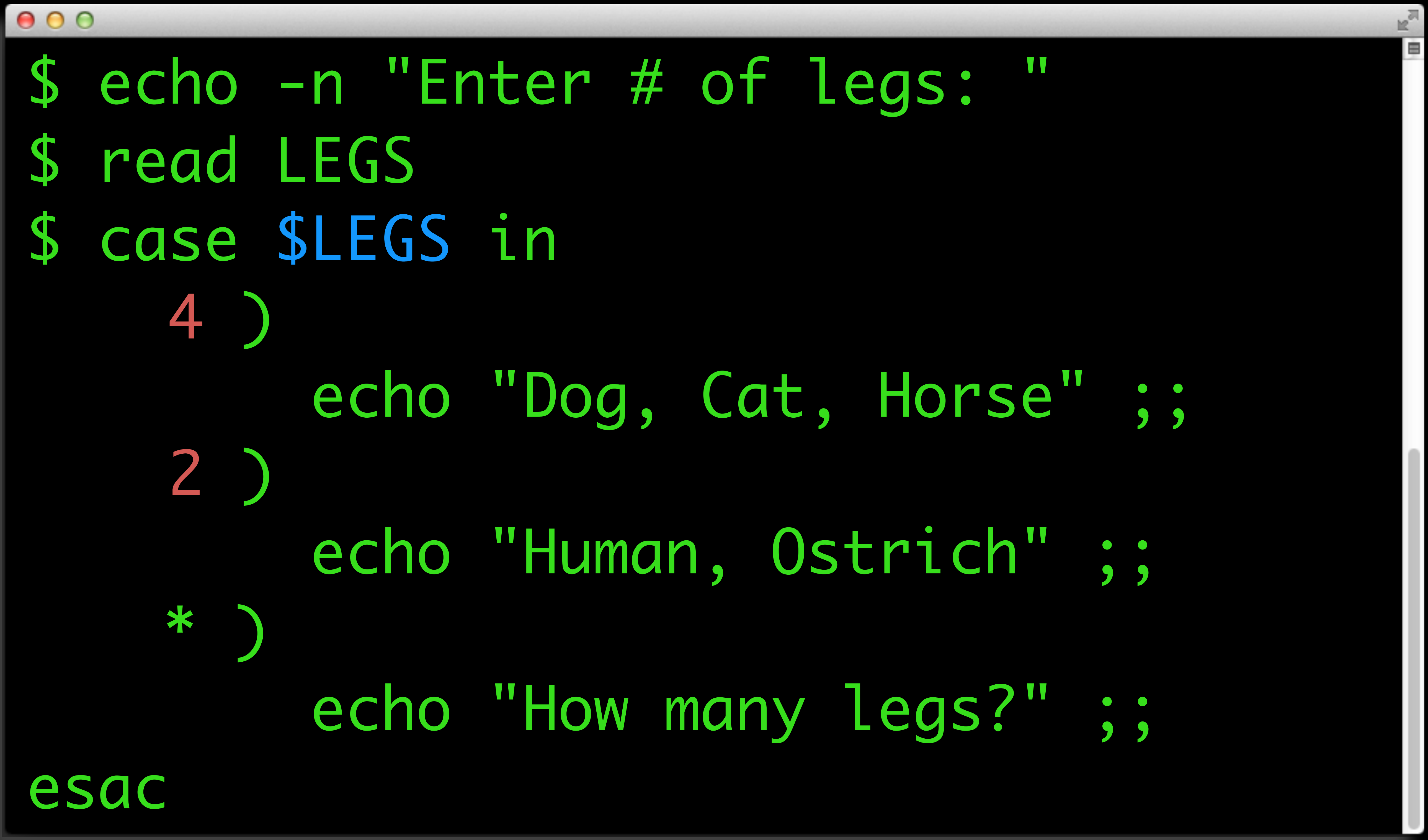
```
$ case expression in
    pattern1 )
        commands ;;
    pattern2 )
        commands ;;
esac
```



```
$ cat OfficeUpdates.sh
#!/bin/bash
echo "would you like to turn automatic updates on or off?":
read ANSWER
case $ANSWER in
        "on" )
                defaults write com.microsoft.autoupdate2
HowToCheck "Automatic";;
        "off" )
                defaults write com.microsoft.autoupdate2
HowToCheck "Manual";;
        *)
                echo "enter 'on' or 'off'"
                exit;;
esac
```



```
$defaults read ~/Library/Preferences/
com.microsoft.autoupdate2.plist HowToCheck

Manual

$ bash OfficeUpdates.sh
> would you like to turn automatic updates on or off?:
on

$ defaults read ~/Library/Preferences/
com.microsoft.autoupdate2.plist HowToCheck

Automatic
```



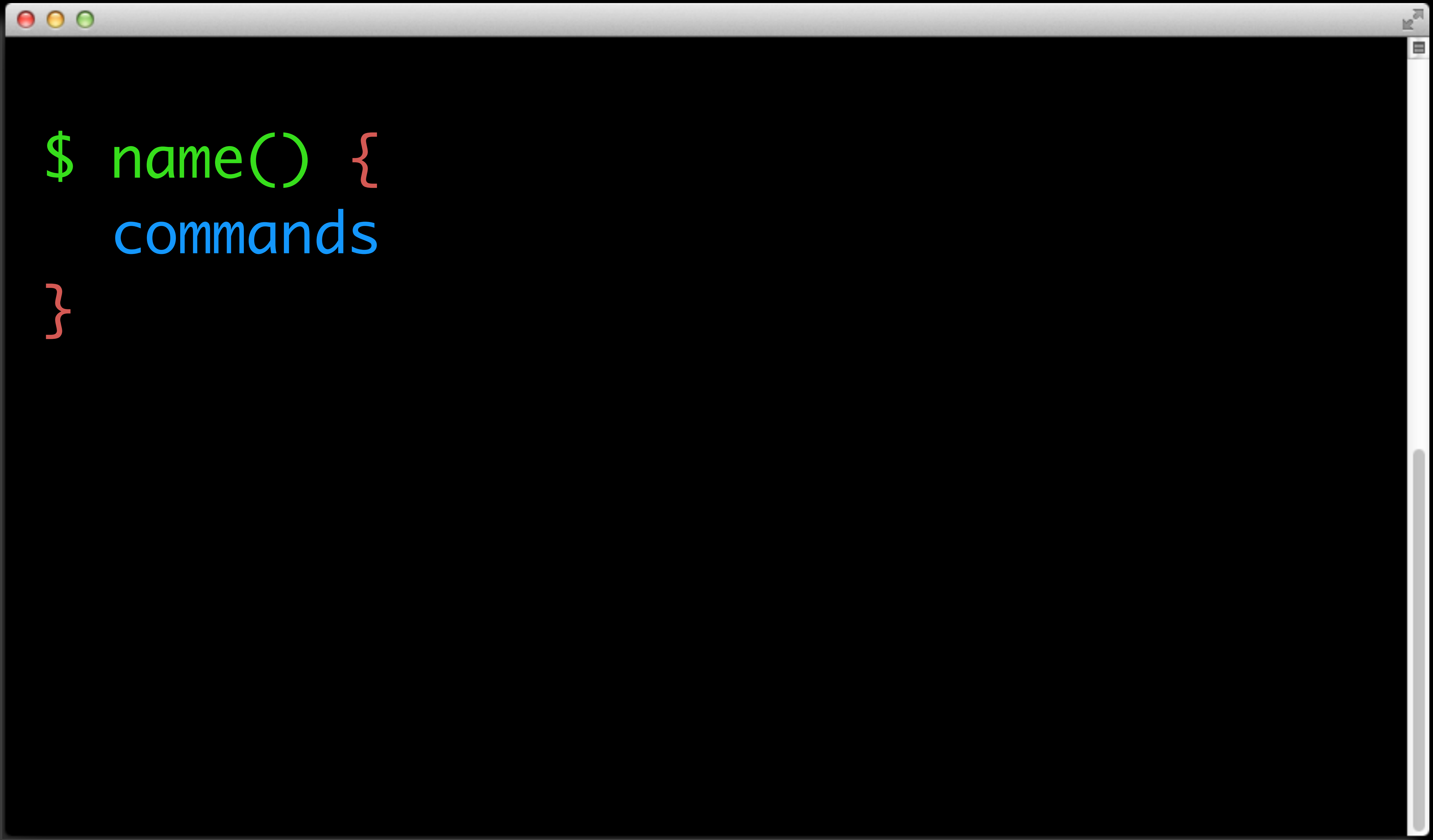
```
$ echo -n "Enter # of legs: "
$ read LEGS
$ case $LEGS in
    4 )
        echo "Dog, Cat, Horse" ;;
    2 )
        echo "Human, Ostrich" ;;
    * )
        echo "How many legs?" ;;
esac
```

## Try It!

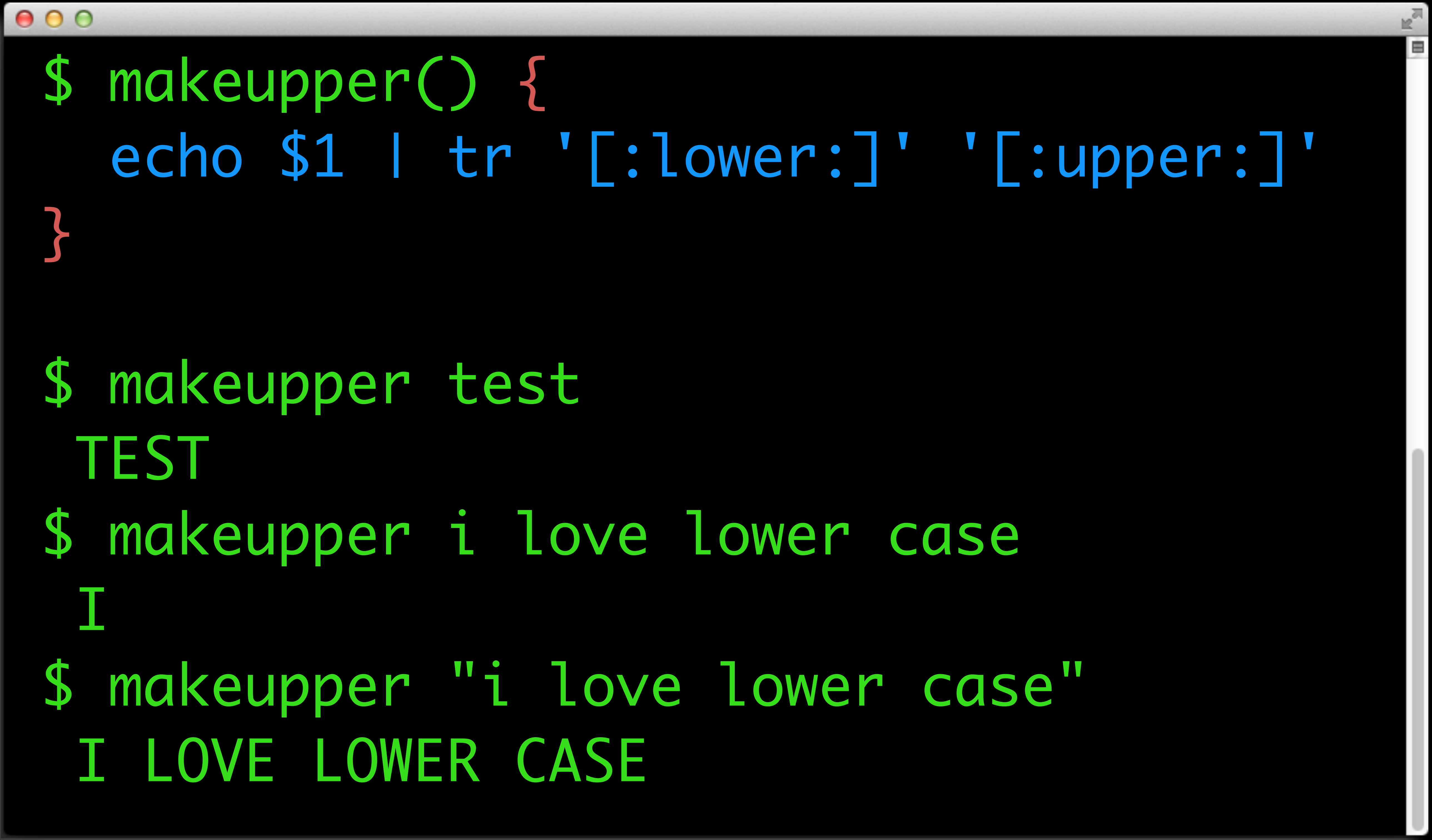
- Create a case statement
  - Read input
  - Output with echo

## Functions

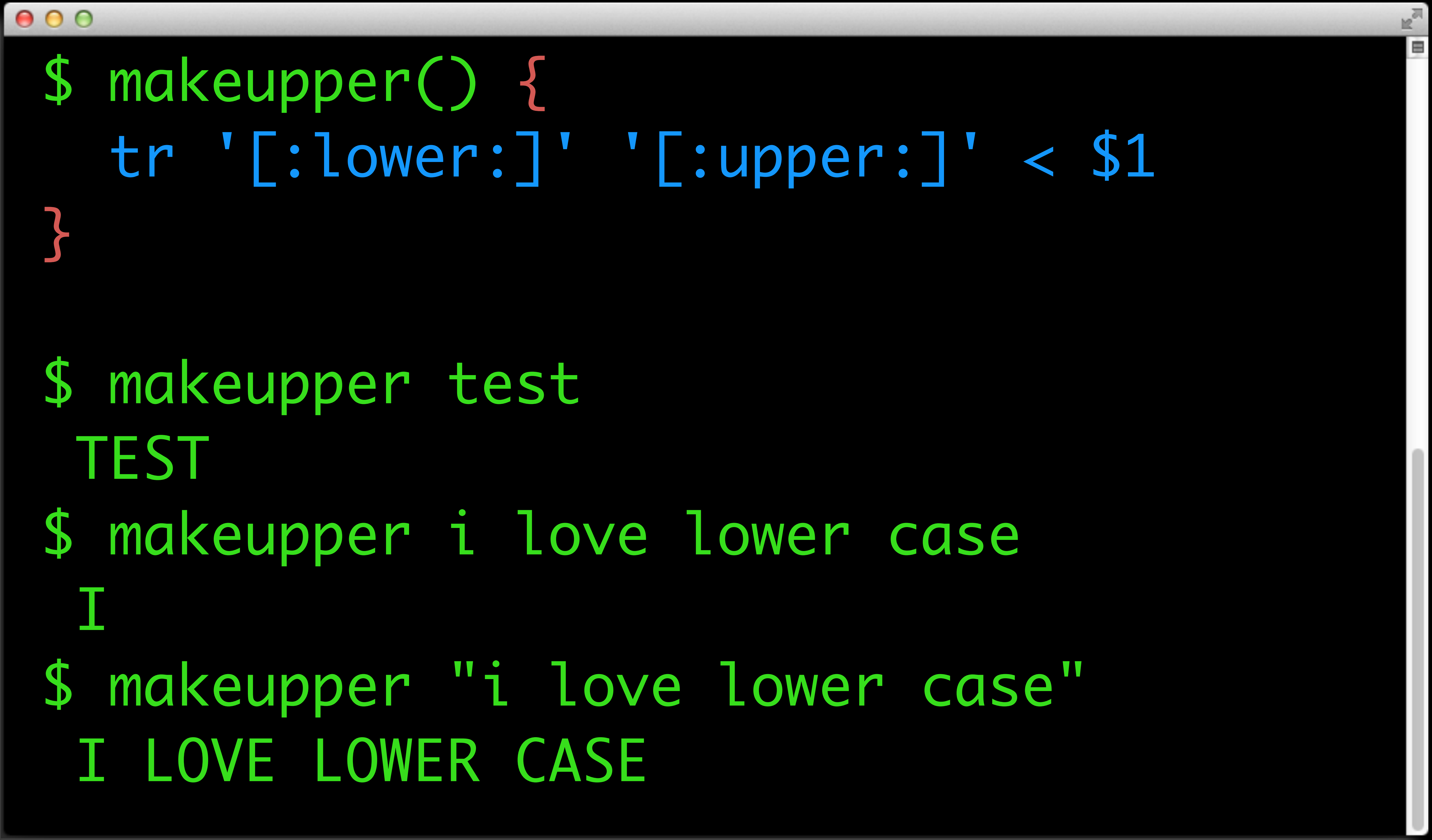
- Modularize Code
- Repetitive Tasks
- Update in One Place!



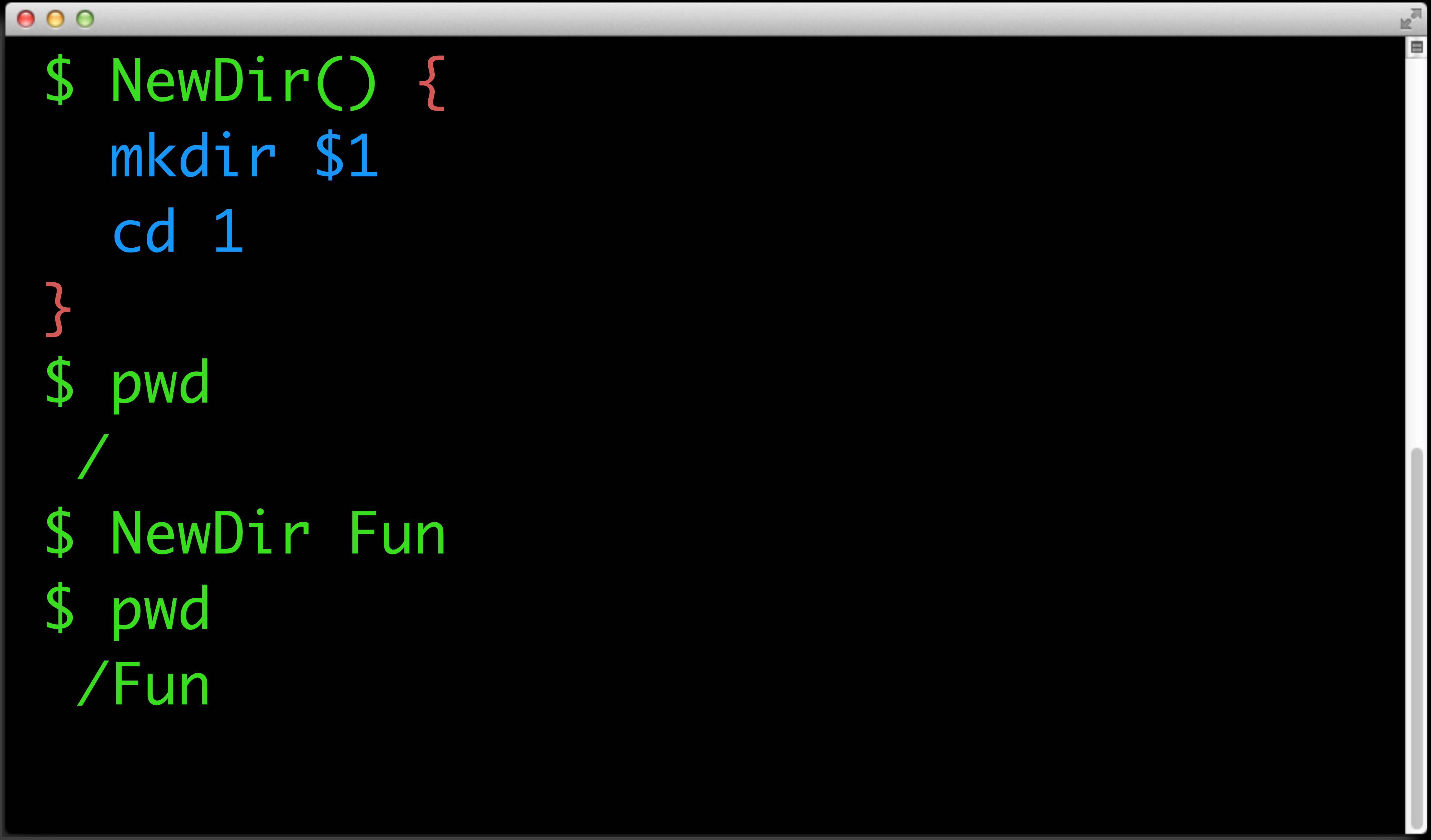
```
$ name() {
    commands
}
```



```
$ makeupper() {
    echo $1 | tr '[:lower:]' '[:upper:]'
}

$ makeupper test
 TEST
$ makeupper i love lower case
 I
$ makeupper "i love lower case"
 I LOVE LOWER CASE
```



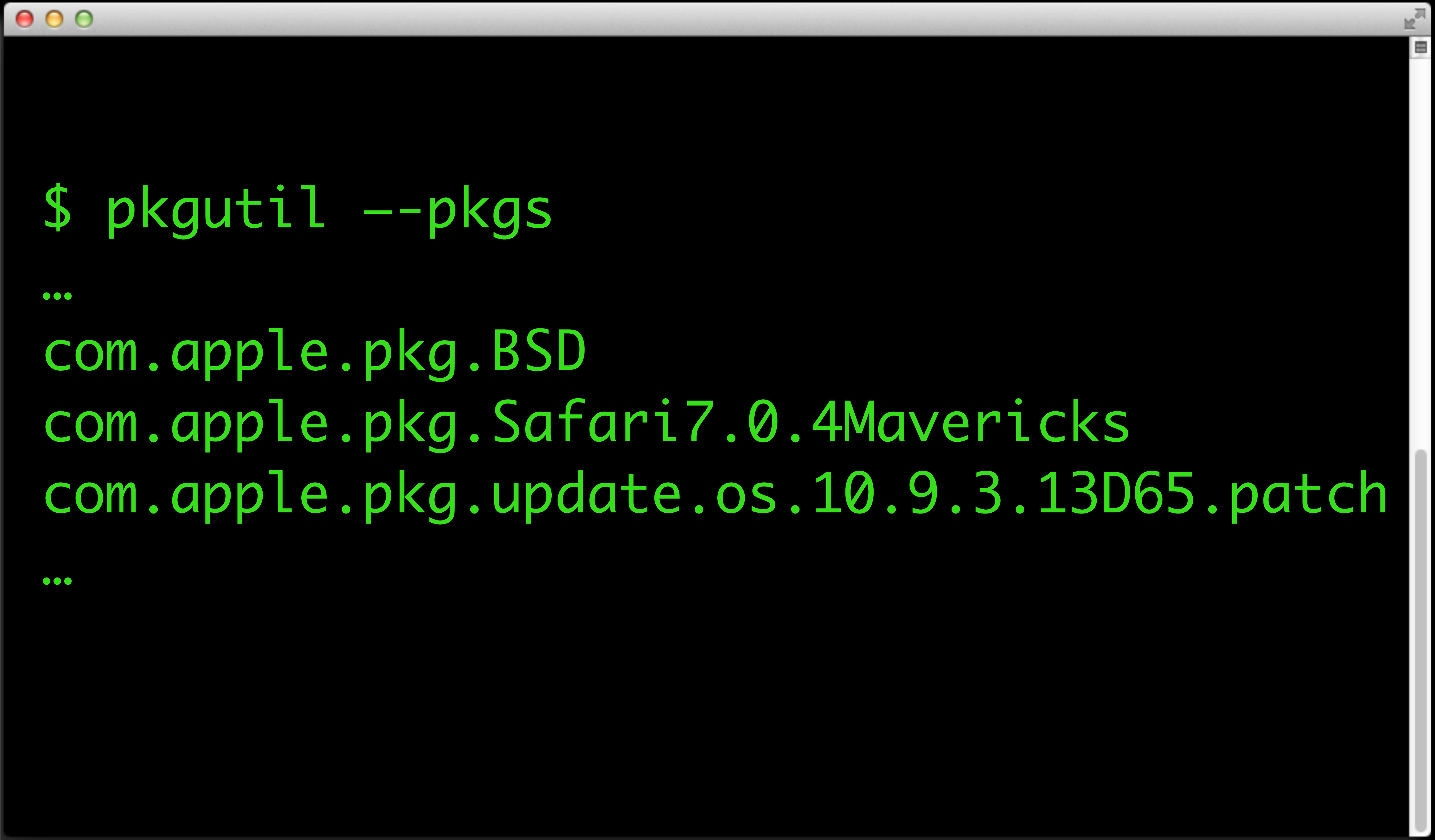
```
$ makeupper() {
   tr '[:lower:]' '[:upper:]' < $1
}


$ makeupper test
 TEST
$ makeupper i love lower case
 I
$ makeupper "i love lower case"
 I LOVE LOWER CASE
```



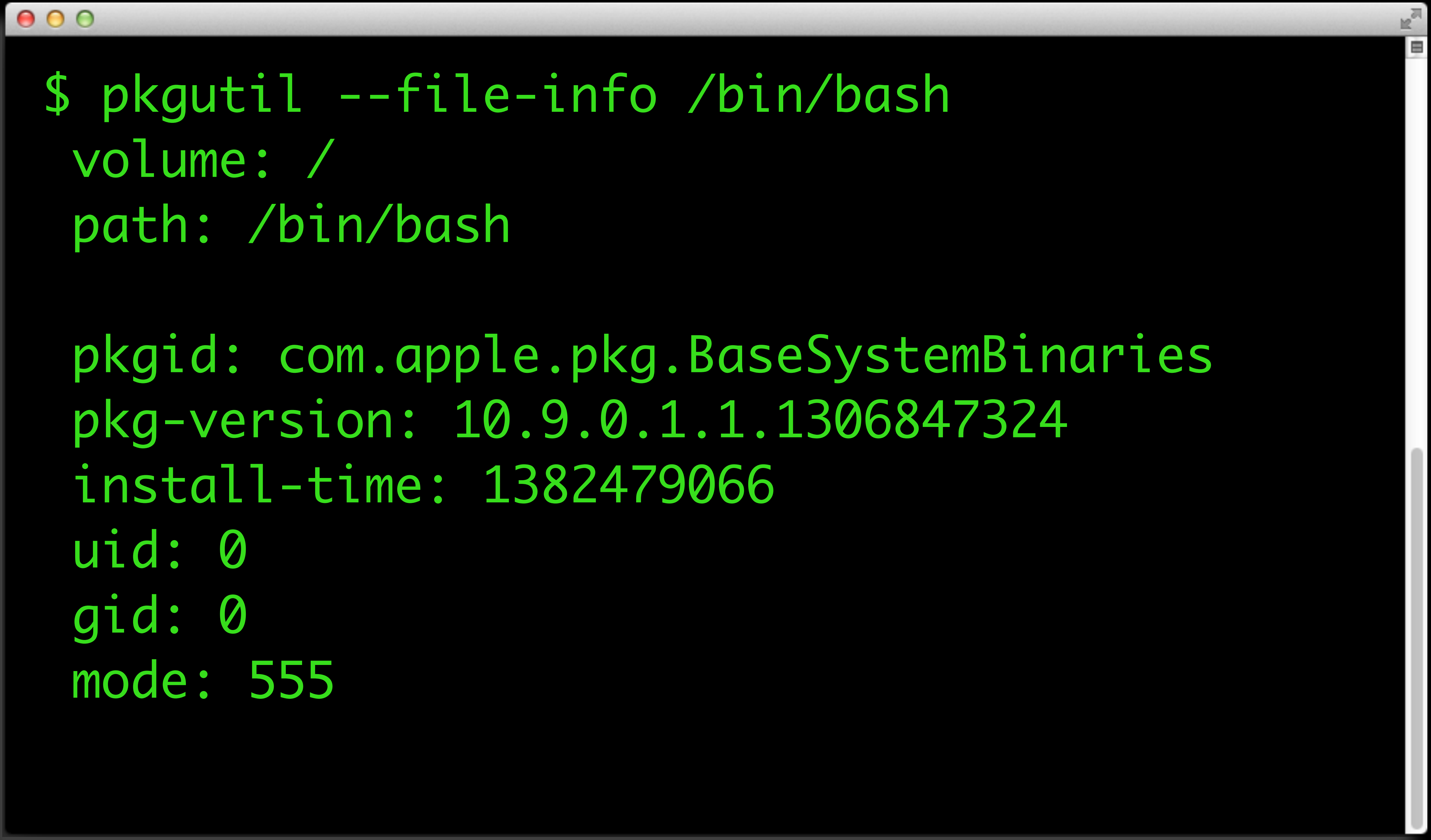
```
$ NewDir() {
   mkdir $1
   cd 1
}
$ pwd
 /
$ NewDir Fun
$ pwd
 /Fun
```

# pkgutil

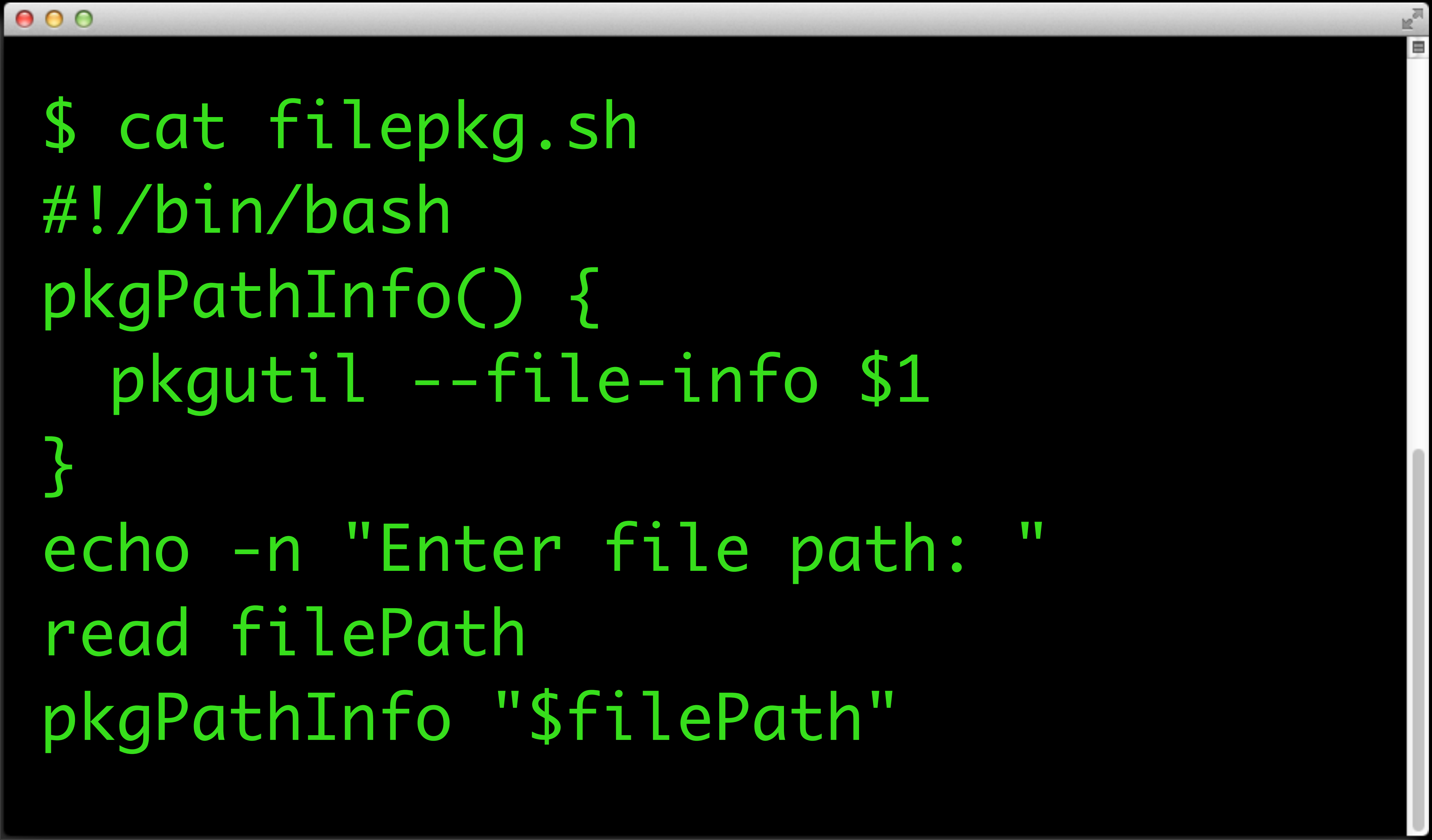
- Reads and Manipulates flat packages
- Accesses 'receipt database'
- `pkgutil --pkgs` = List all installed pkgs
- `pkgutil --file-info /file/path`



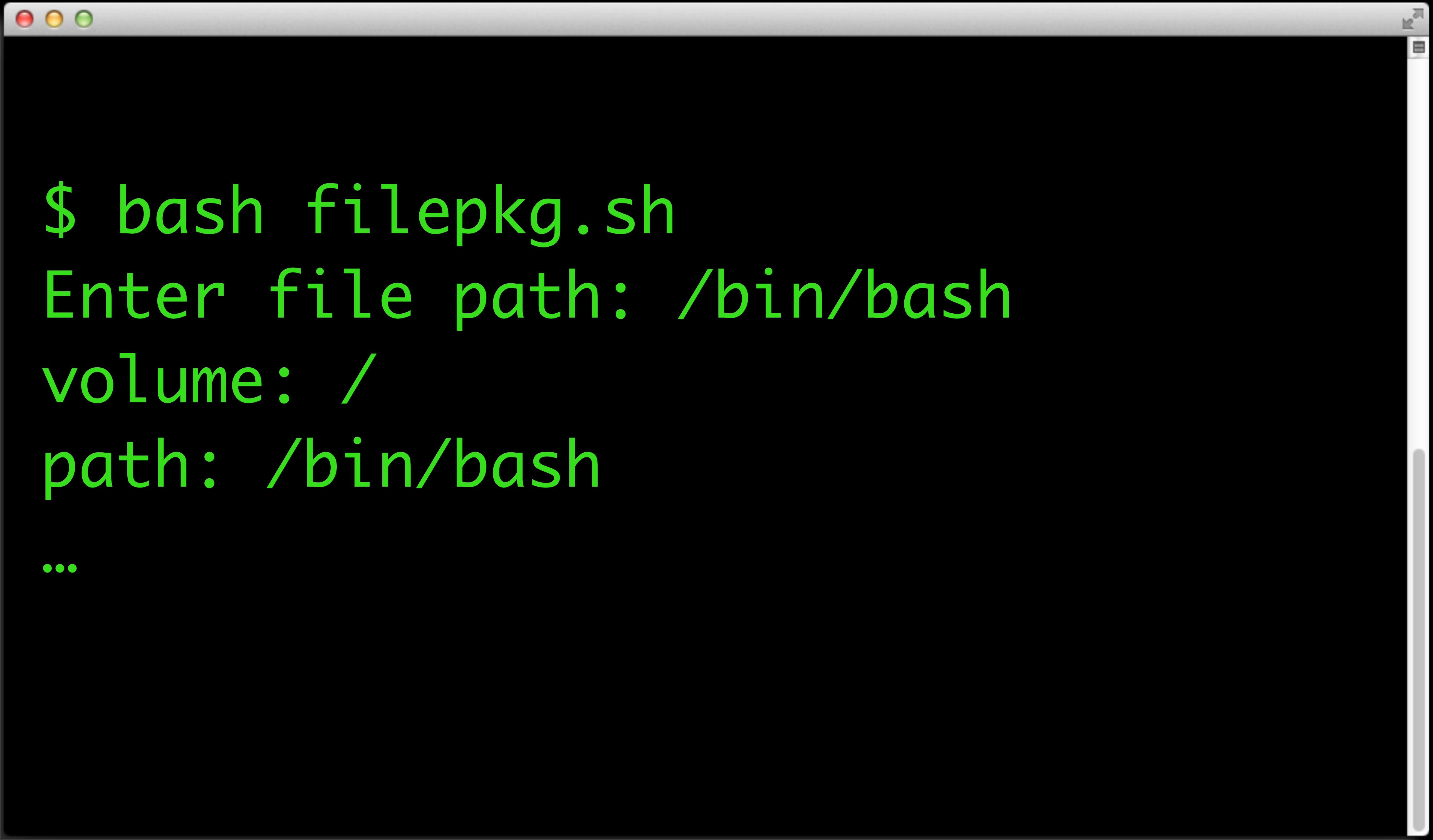
```
$ pkgutil --pkgs
…
com.apple.pkg.BSD
com.apple.pkg.Safari7.0.4Mavericks
com.apple.pkg.update.os.10.9.3.13D65.patch
…
```



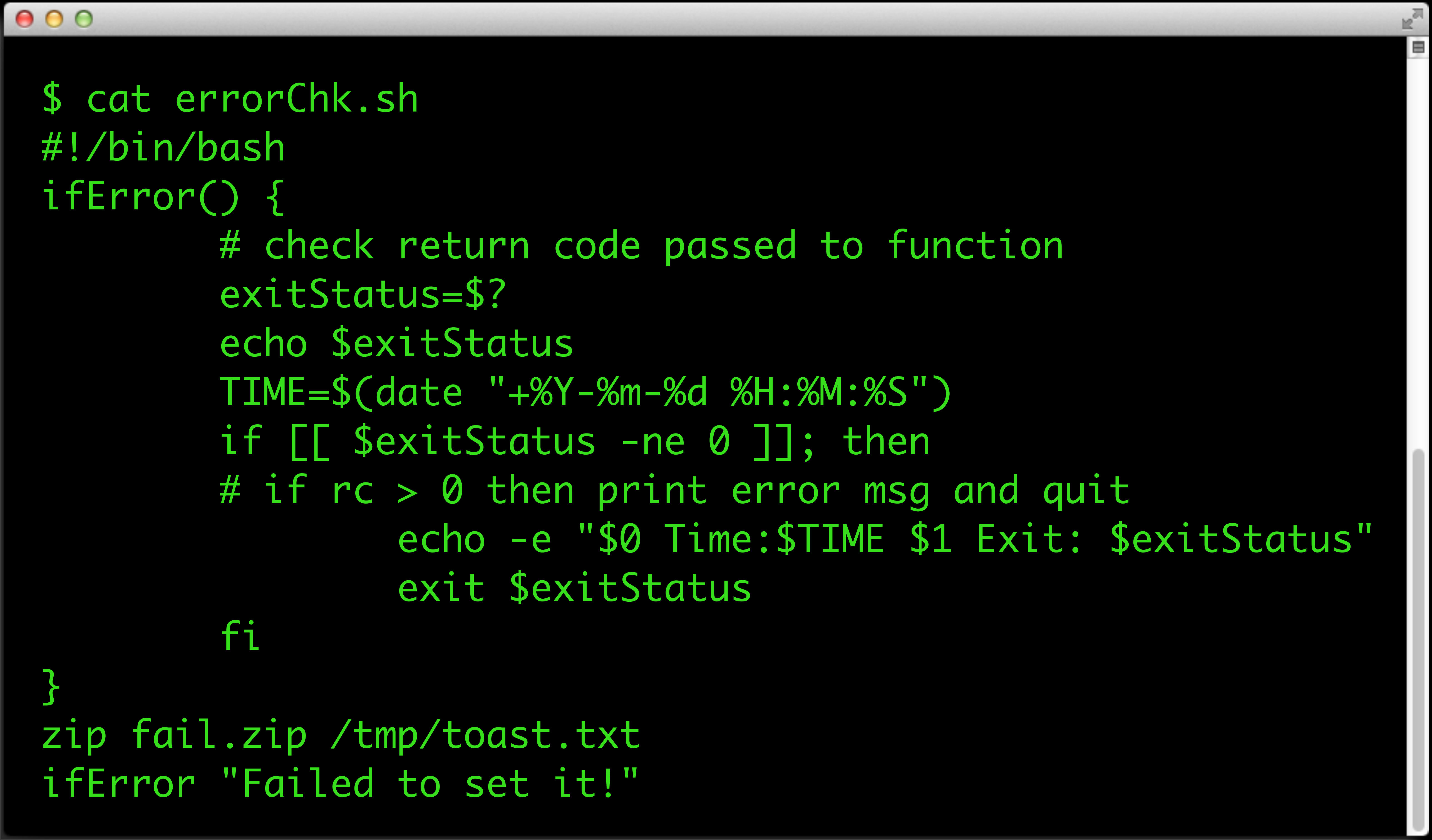
```
$ pkgutil --file-info /bin/bash
 volume: /
 path: /bin/bash

 pkgid: com.apple.pkg.BaseSystemBinaries
 pkg-version: 10.9.0.1.1.1306847324
 install-time: 1382479066
 uid: 0
 gid: 0
 mode: 555
```



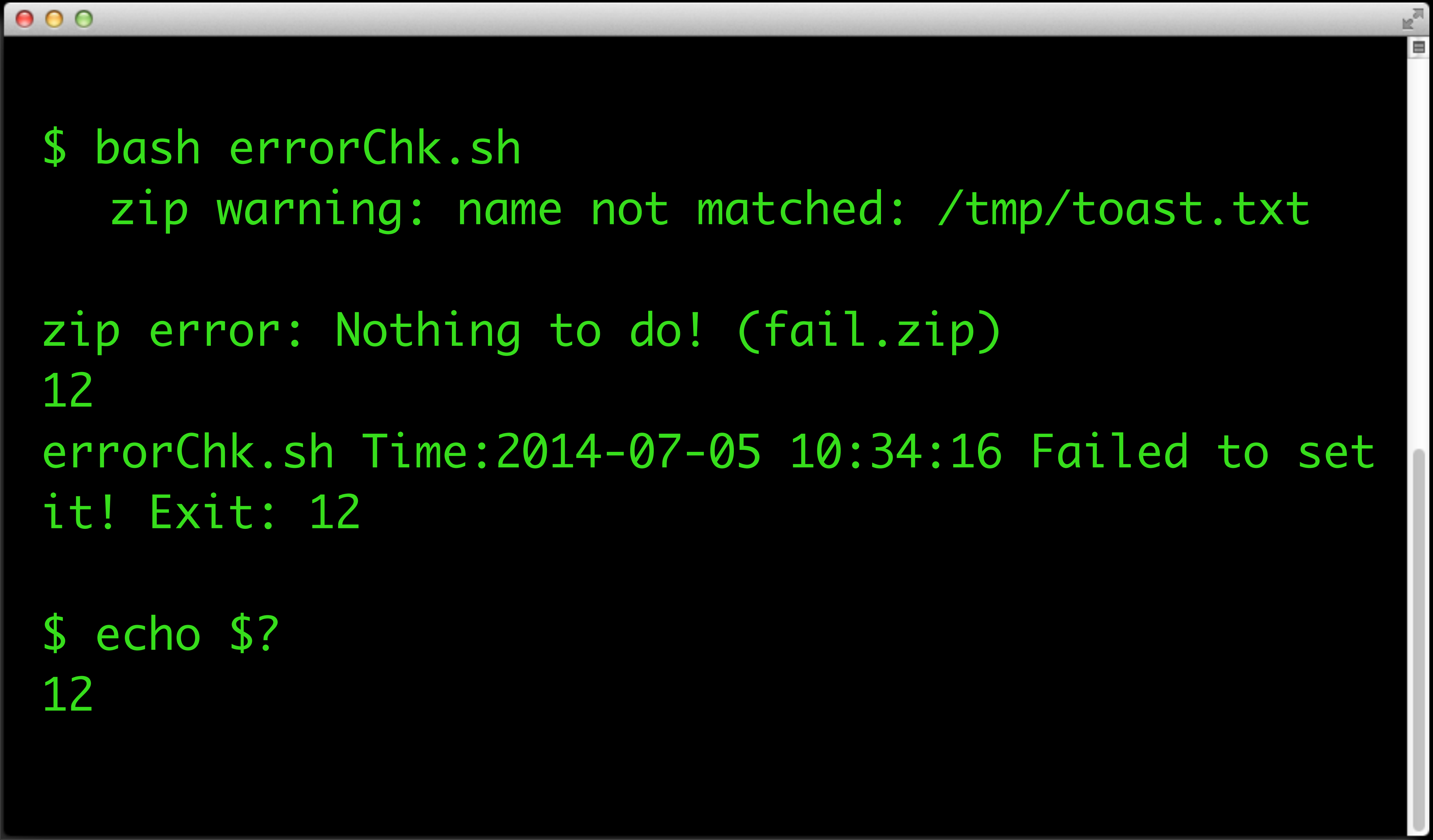
```
$ cat filepkg.sh
#!/bin/bash
pkgPathInfo() {
  pkgutil --file-info $1
}
echo -n "Enter file path: "
read filePath
pkgPathInfo "$filePath"
```



```
$ bash filepkg.sh
Enter file path: /bin/bash
volume: /
path: /bin/bash
…
```



```
$ cat errorChk.sh
#!/bin/bash
ifError() {
        # check return code passed to function
        exitStatus=$?
        echo $exitStatus
        TIME=$(date "+%Y-%m-%d %H:%M:%S")
        if [[ $exitStatus -ne 0 ]]; then
        # if rc > 0 then print error msg and quit
                echo -e "$0 Time:$TIME $1 Exit: $exitStatus"
                exit $exitStatus
        fi
}
zip fail.zip /tmp/toast.txt
ifError "Failed to set it!"
```



```
$ bash errorChk.sh
   zip warning: name not matched: /tmp/toast.txt

zip error: Nothing to do! (fail.zip)
12
errorChk.sh Time:2014-07-05 10:34:16 Failed to set
it! Exit: 12

$ echo $?
12
```

# Try It!

- Update/Create Script
- Add a function
- Try a Log function with date

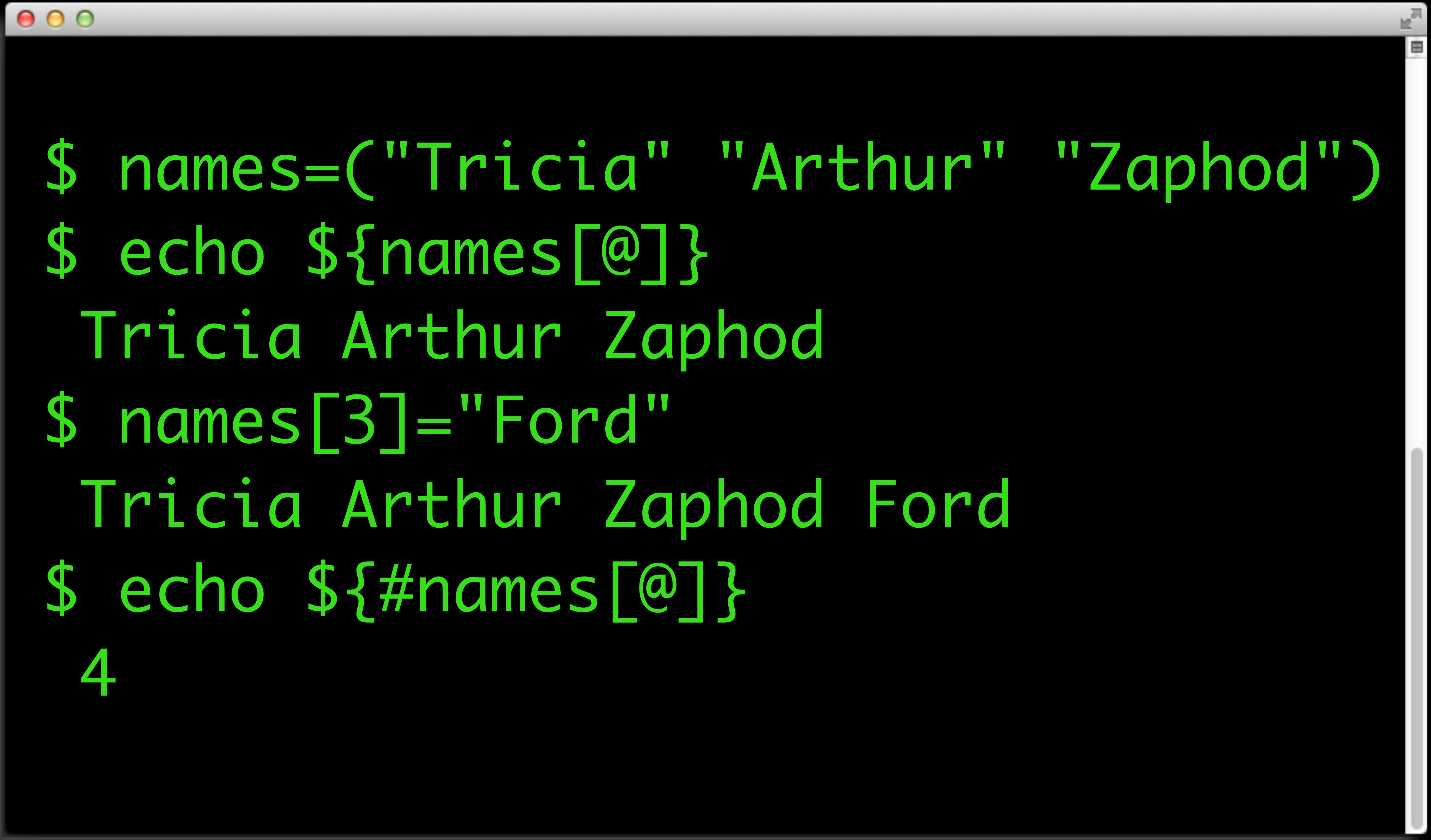
## Need More?

## source

- Read File for
  - Variables
  - Functions
- Keep Functions in separate Files and Source them as needed

## arrays

- Numbered list of strings
- 0 based
- `array=("string1" "string2")`
- All Elements = `${array[@]}`
- Element N = `${array[N]}`
- Number of Elements = `${#array[@]}`



```
$ names=("Tricia" "Arthur" "Zaphod")
$ echo ${names[@]}
 Tricia Arthur Zaphod
$ names[3]="Ford"
 Tricia Arthur Zaphod Ford
$ echo ${#names[@]}
 4
```

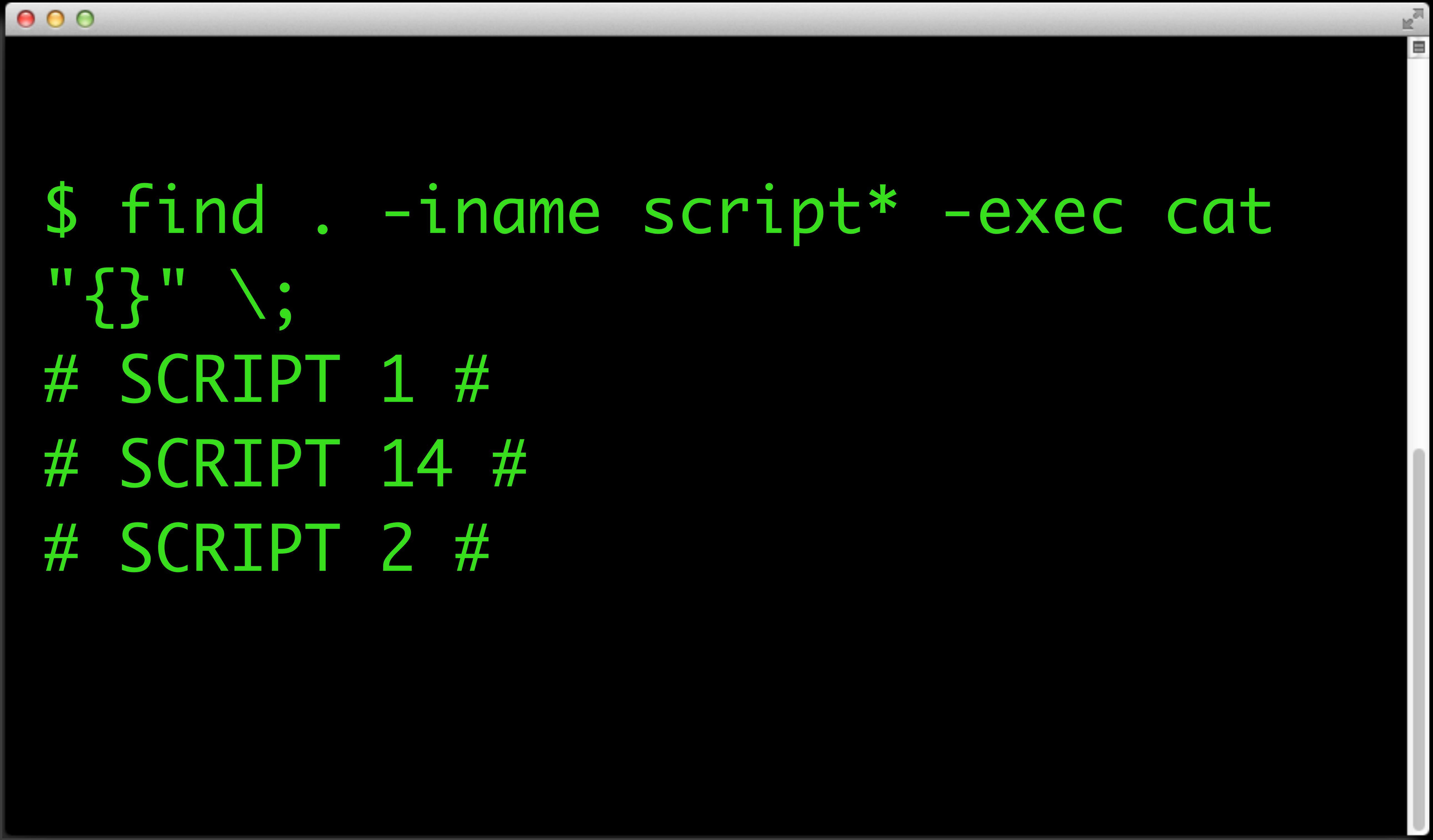
Feedback: http://j.mp/psumac13

## IFS

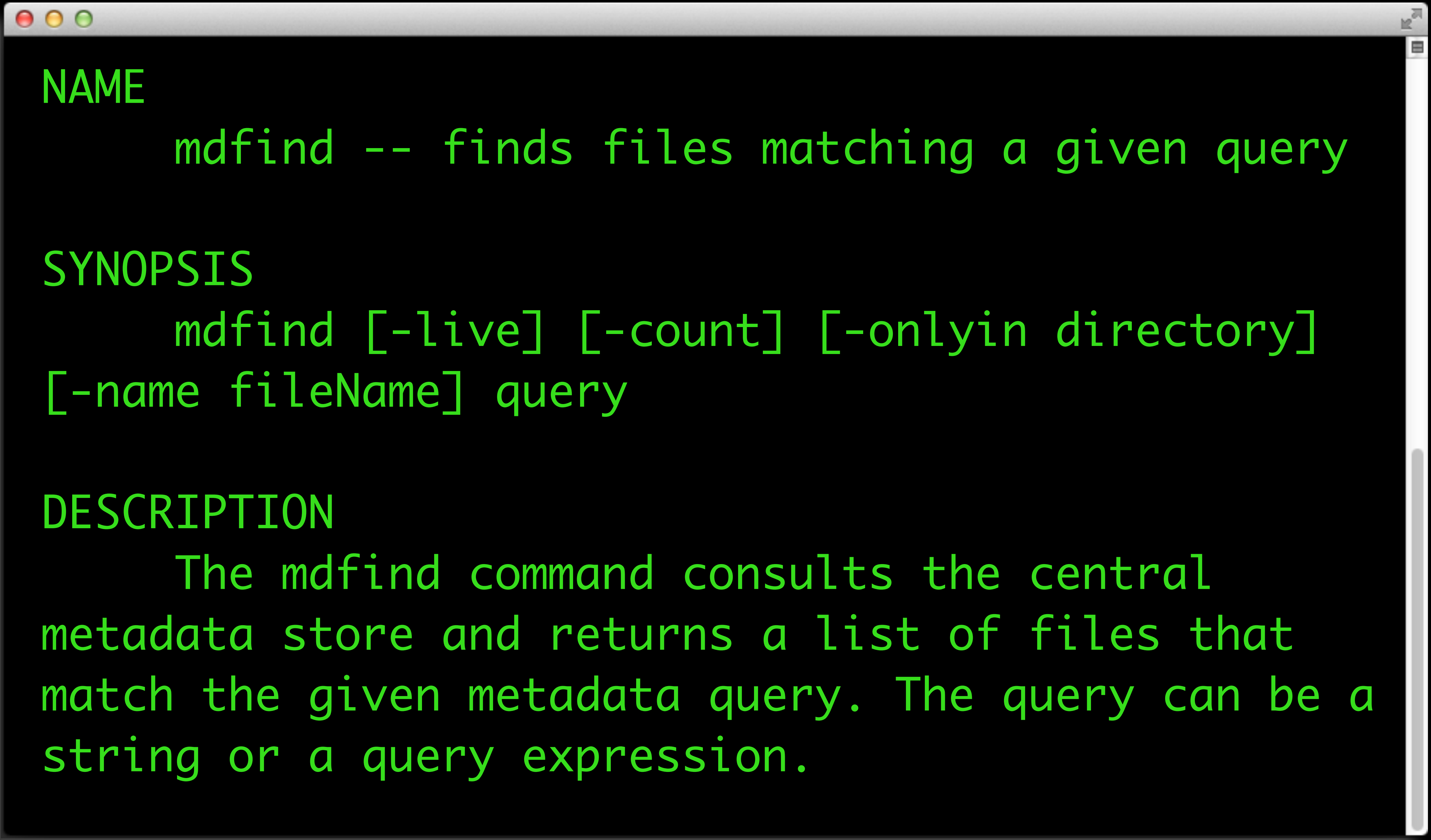
- Internal File Separator
- Splits fields with whitespace
- Can be changed
- IFS=',' = For CSV Files

## More Find

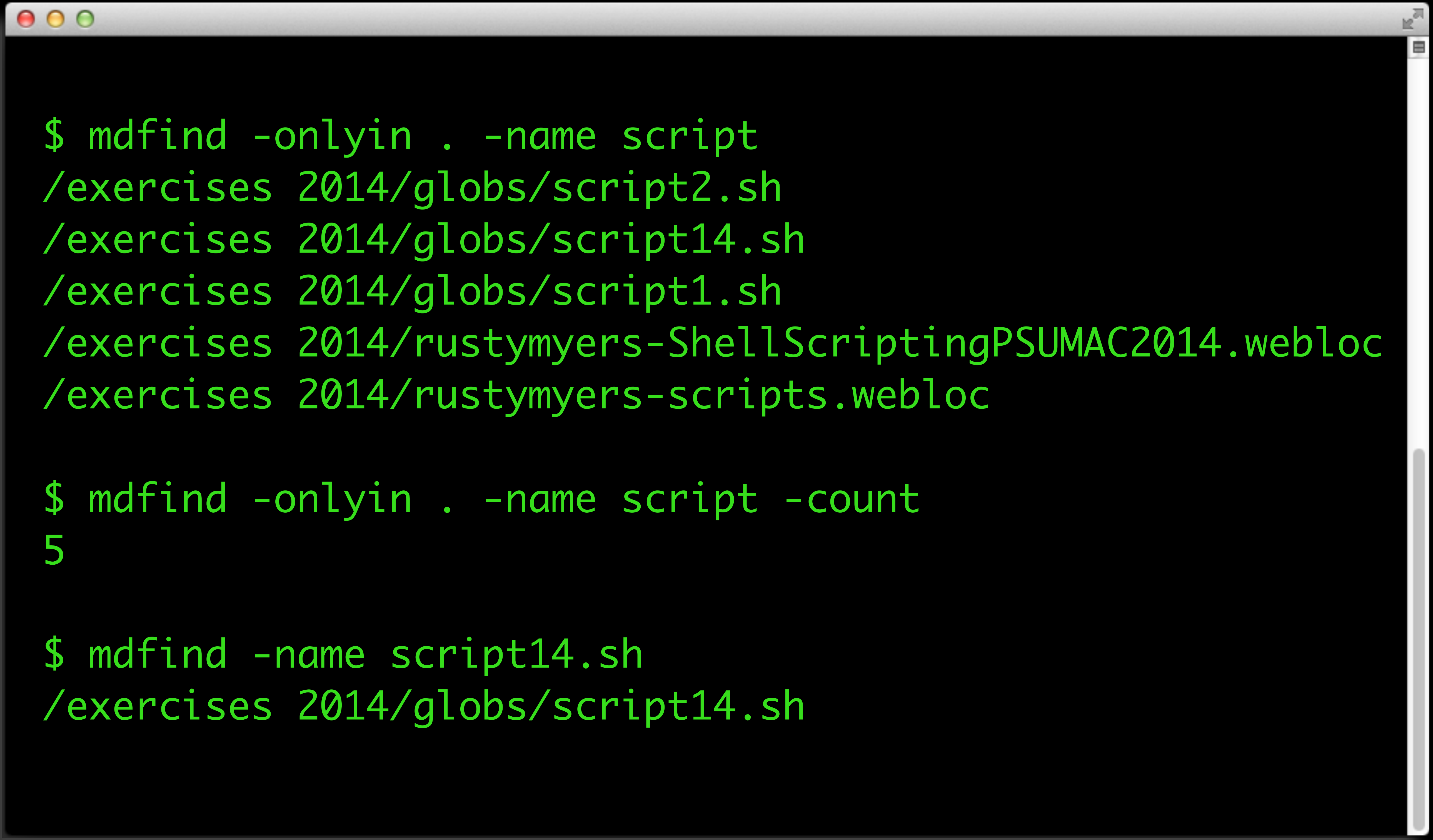
- Exec command with find



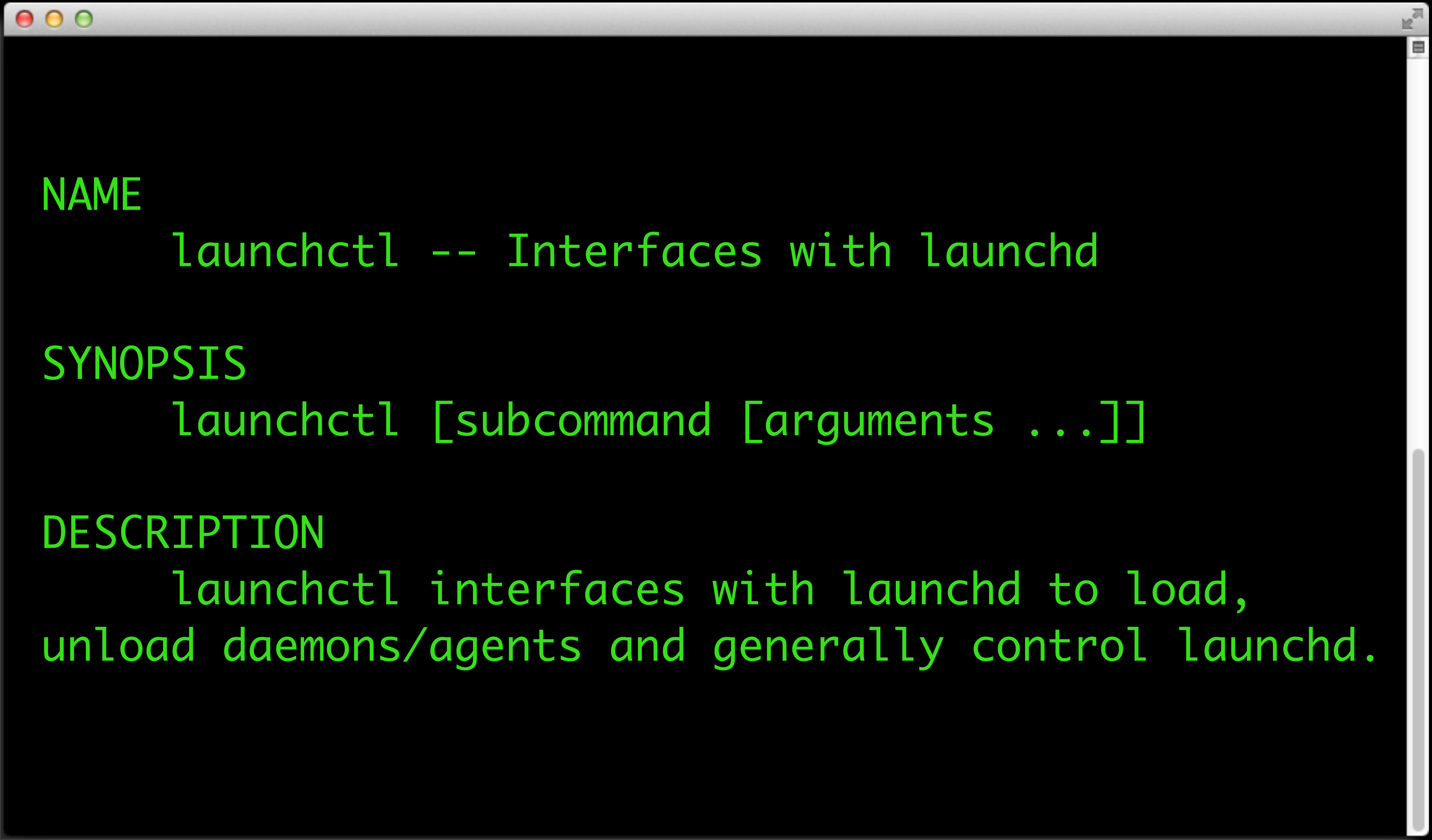
```
$ find . -iname script* -exec cat
"{}" \;
# SCRIPT 1 #
# SCRIPT 14 #
# SCRIPT 2 #
```



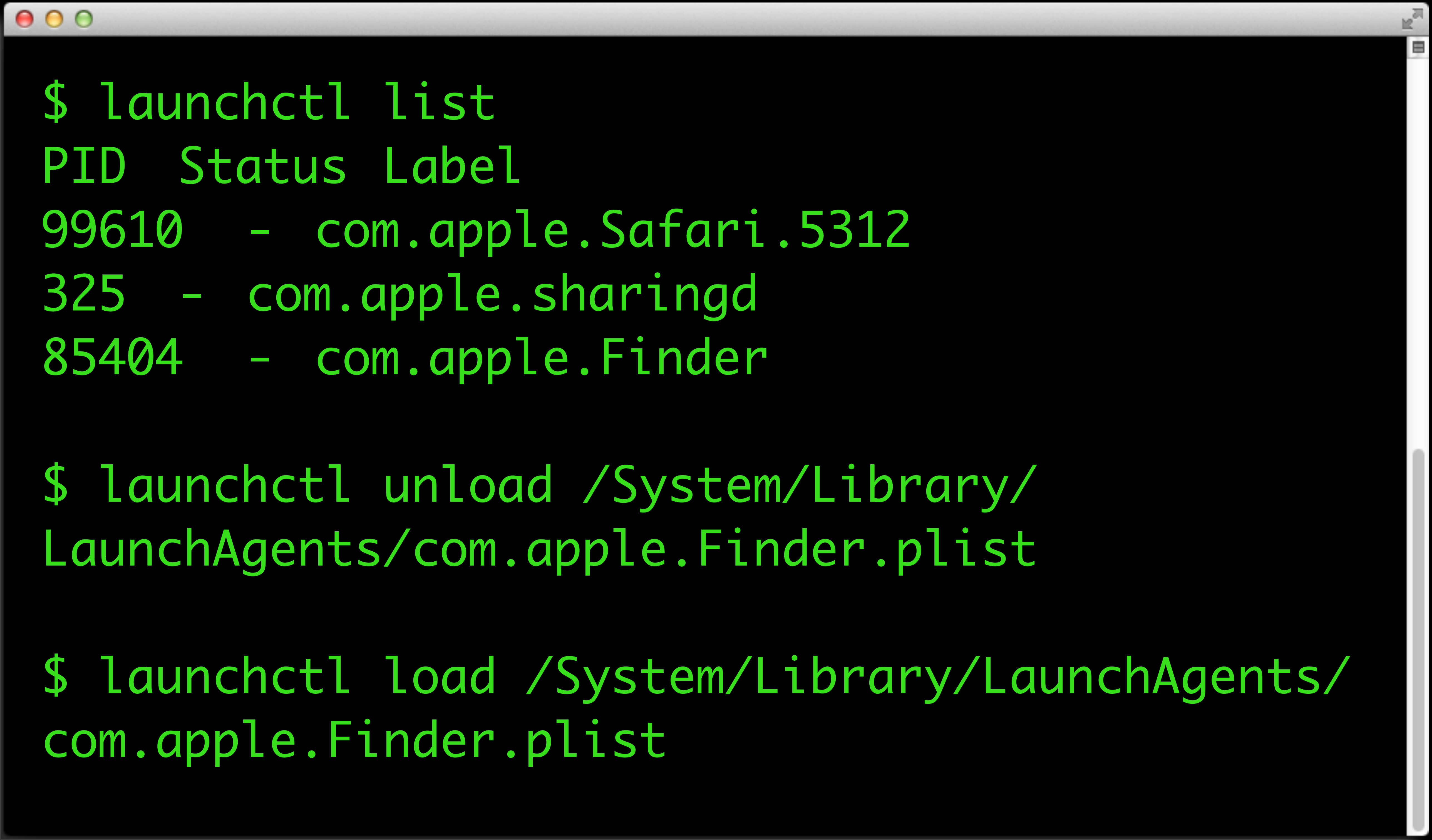
```
NAME
     mdfind -- finds files matching a given query

SYNOPSIS
     mdfind [-live] [-count] [-onlyin directory]
[-name fileName] query

DESCRIPTION
     The mdfind command consults the central
metadata store and returns a list of files that
match the given metadata query. The query can be a
string or a query expression.
```



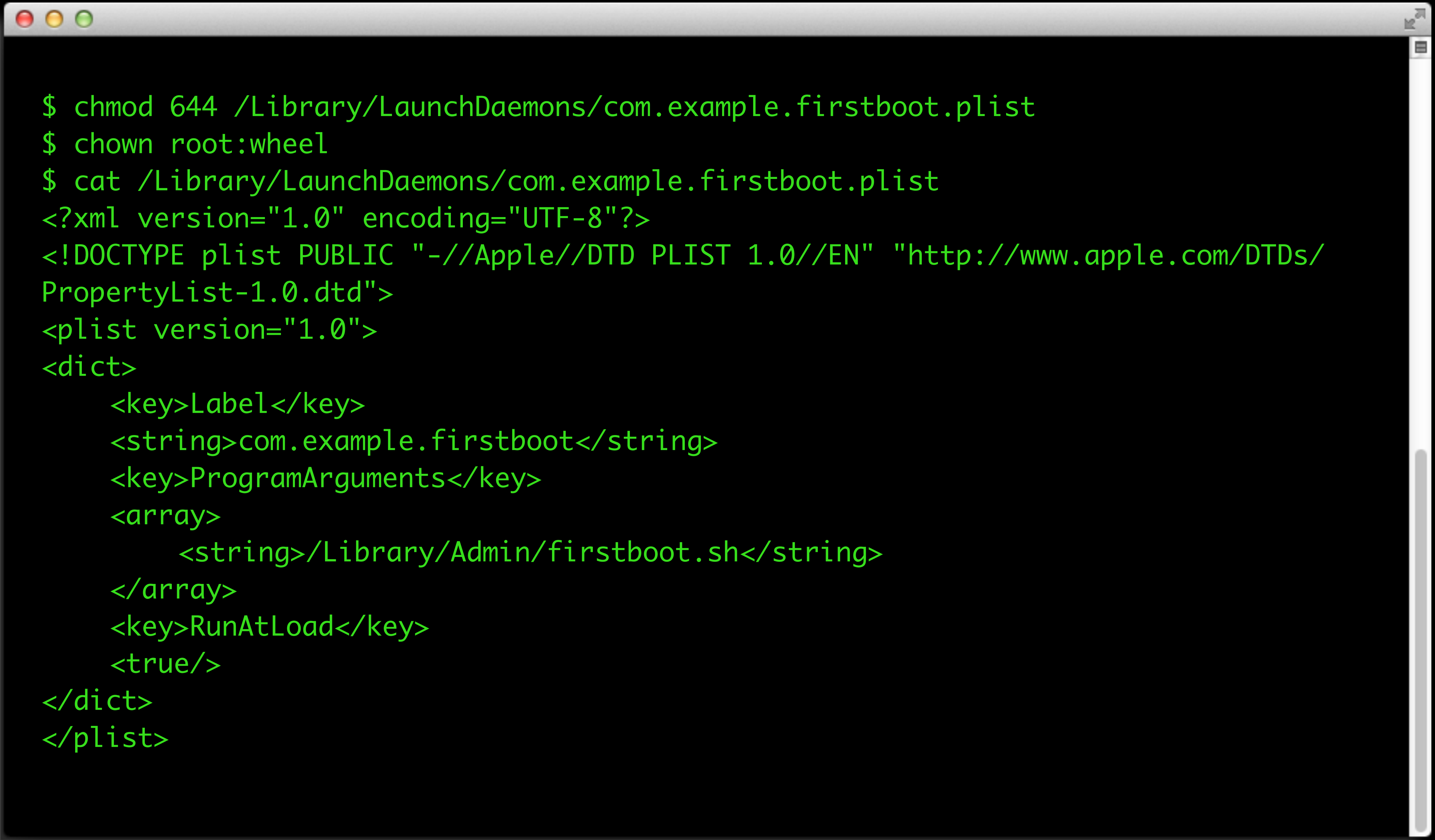
```
$ mdfind -onlyin . -name script
/exercises 2014/globs/script2.sh
/exercises 2014/globs/script14.sh
/exercises 2014/globs/script1.sh
/exercises 2014/rustymyers-ShellScriptingPSUMAC2014.webloc
/exercises 2014/rustymyers-scripts.webloc

$ mdfind -onlyin . -name script -count
5

$ mdfind -name script14.sh
/exercises 2014/globs/script14.sh
```



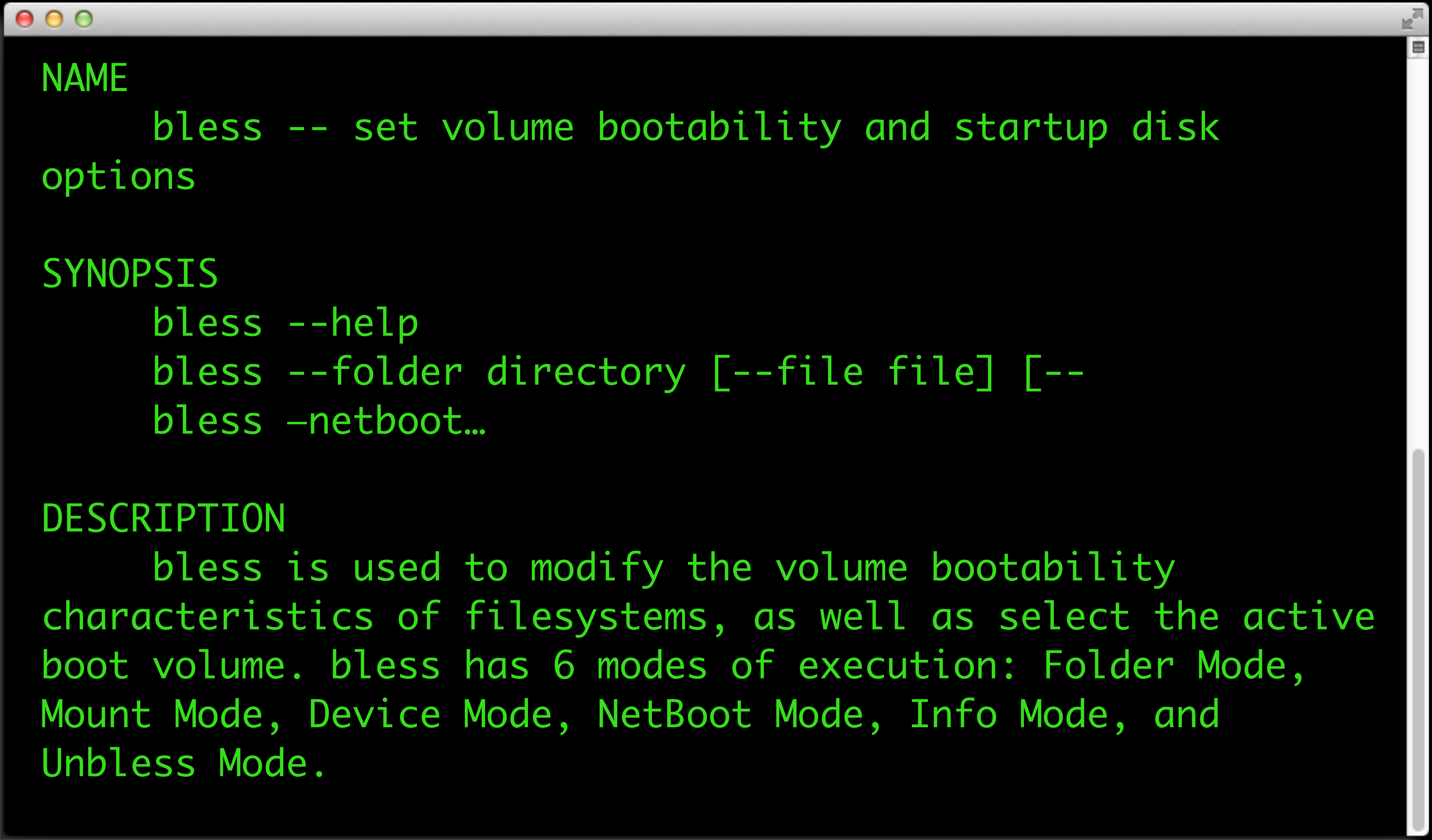
```
NAME
     launchctl -- Interfaces with launchd

SYNOPSIS
     launchctl [subcommand [arguments ...]]

DESCRIPTION
     launchctl interfaces with launchd to load,
unload daemons/agents and generally control launchd.
```



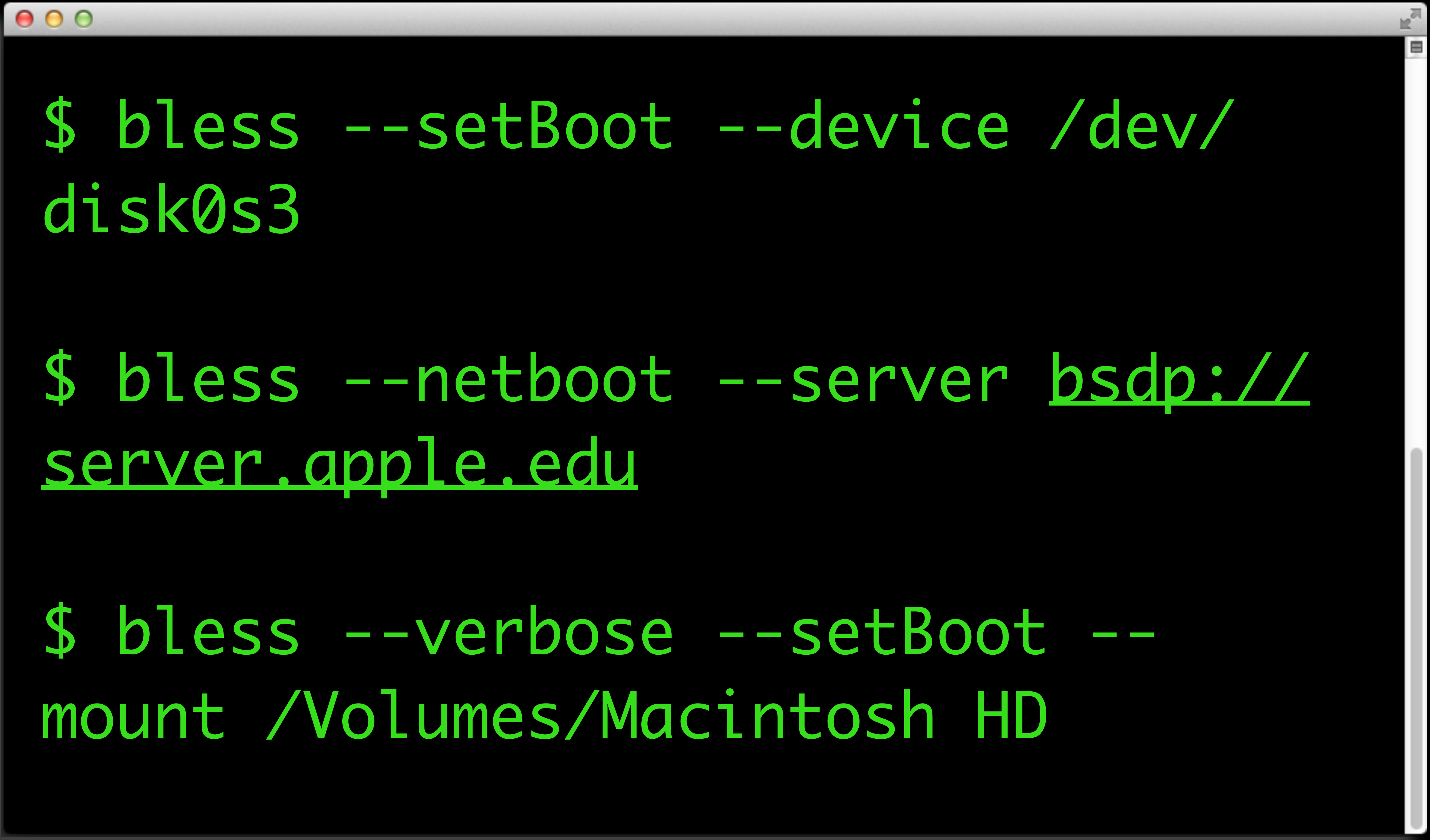
```
$ launchctl list
PID  Status Label
99610   -  com.apple.Safari.5312
325   -  com.apple.sharingd
85404   -  com.apple.Finder

$ launchctl unload /System/Library/
LaunchAgents/com.apple.Finder.plist

$ launchctl load /System/Library/LaunchAgents/
com.apple.Finder.plist
```



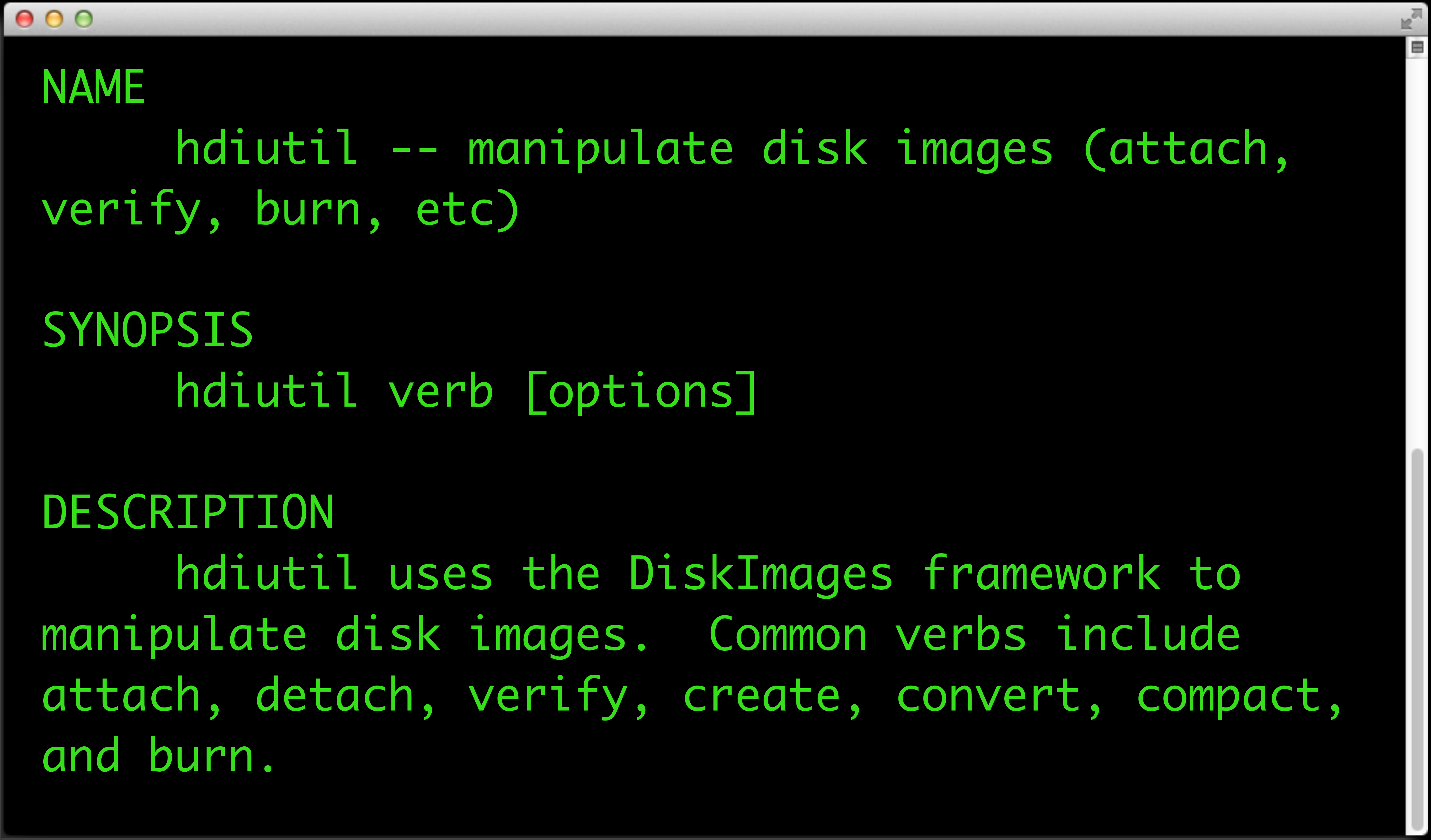
```
$ chmod 644 /Library/LaunchDaemons/com.example.firstboot.plist
$ chown root:wheel
$ cat /Library/LaunchDaemons/com.example.firstboot.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.example.firstboot</string>
    <key>ProgramArguments</key>
    <array>
        <string>/Library/Admin/firstboot.sh</string>
    </array>
    <key>RunAtLoad</key>
    <true/>
</dict>
</plist>
```



```
NAME
     bless -- set volume bootability and startup disk
options

SYNOPSIS
     bless --help
     bless --folder directory [--file file] [--
     bless –netboot…

DESCRIPTION
     bless is used to modify the volume bootability
characteristics of filesystems, as well as select the active
boot volume. bless has 6 modes of execution: Folder Mode,
Mount Mode, Device Mode, NetBoot Mode, Info Mode, and
Unbless Mode.
```

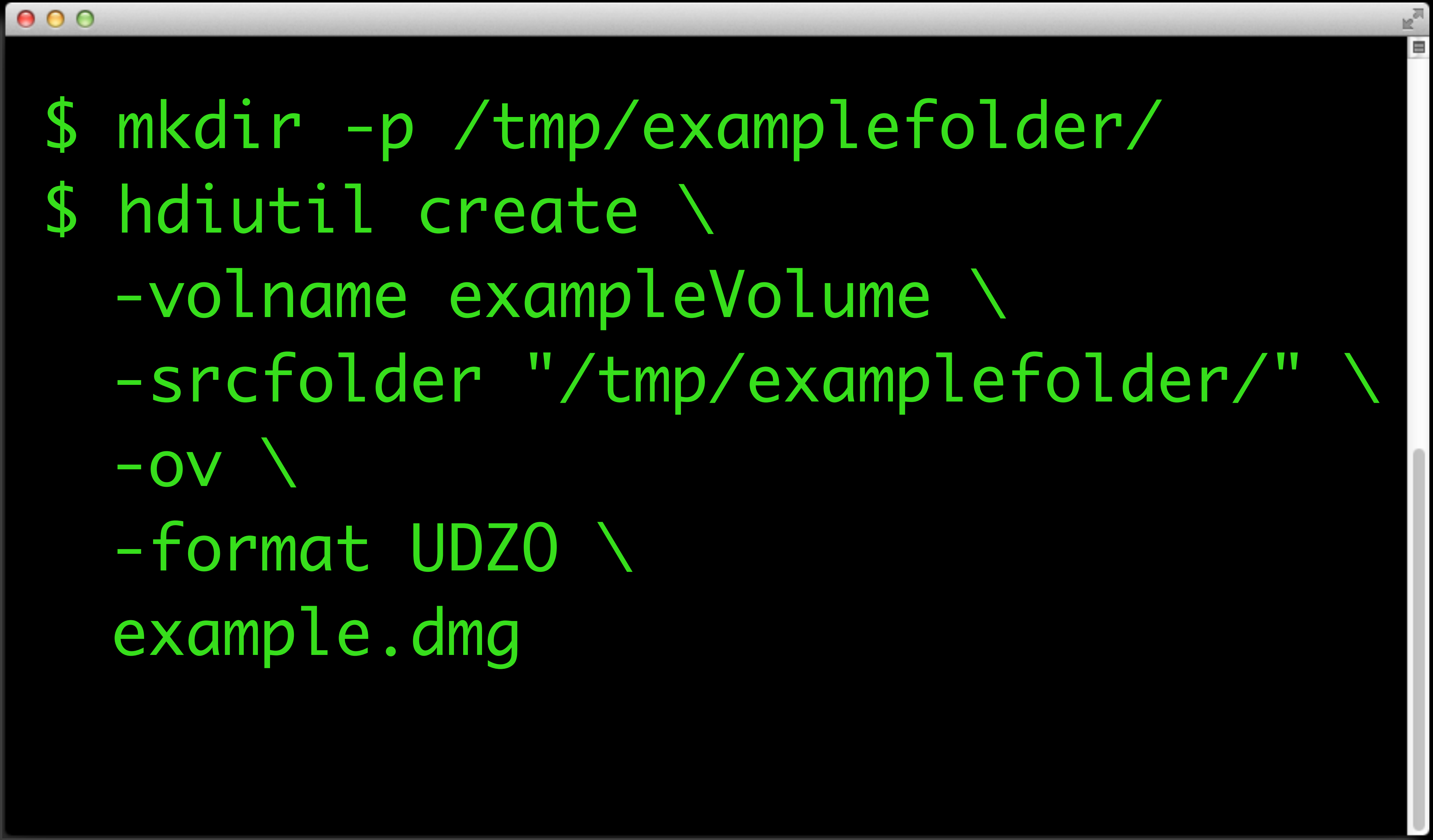


```
$ bless --setBoot --device /dev/
disk0s3

$ bless --netboot --server bsdp://
server.apple.edu

$ bless --verbose --setBoot --
mount /Volumes/Macintosh HD
```

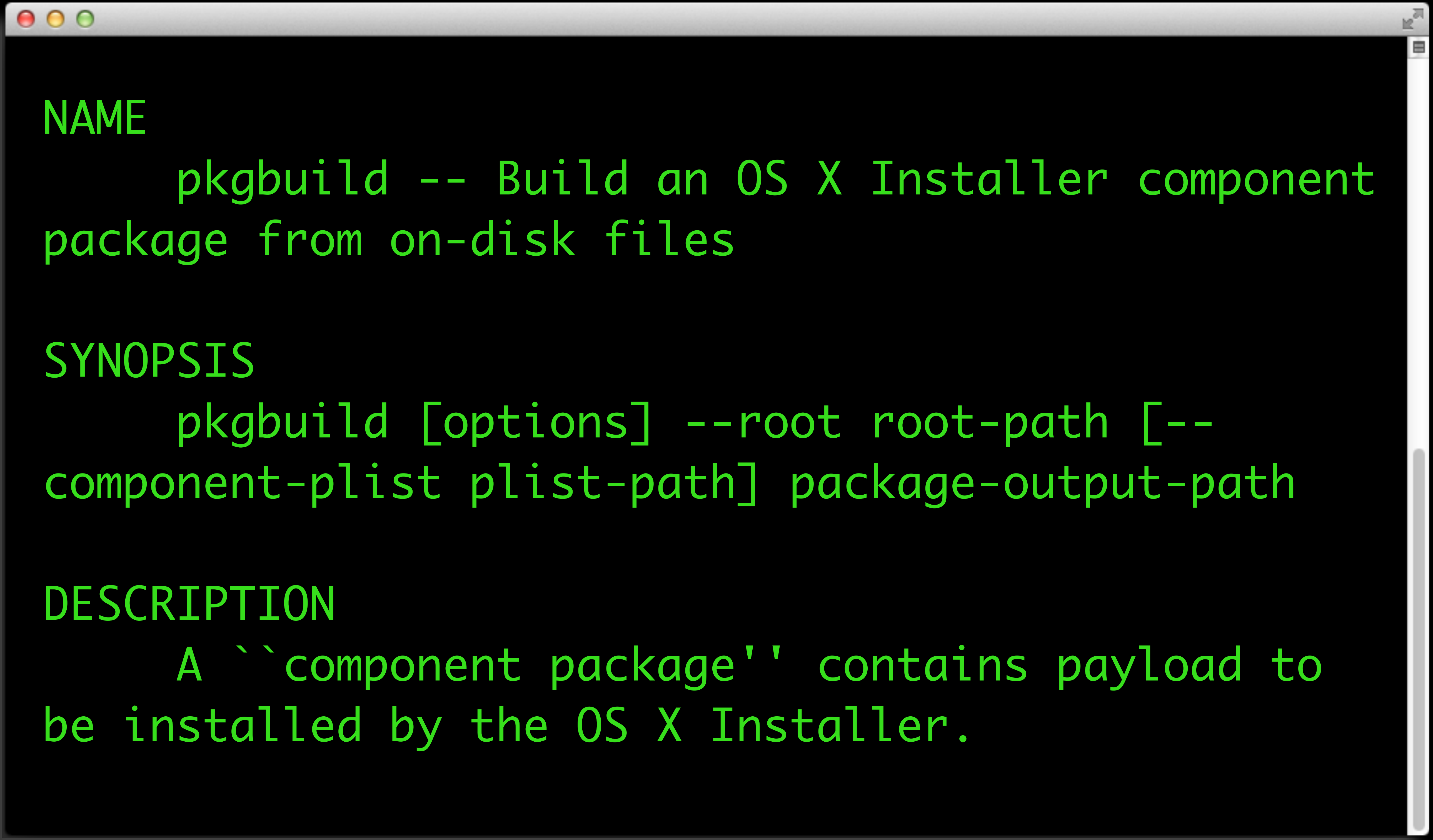


```
NAME
     hdiutil -- manipulate disk images (attach,
verify, burn, etc)

SYNOPSIS
     hdiutil verb [options]

DESCRIPTION
     hdiutil uses the DiskImages framework to
manipulate disk images.  Common verbs include
attach, detach, verify, create, convert, compact,
and burn.
```

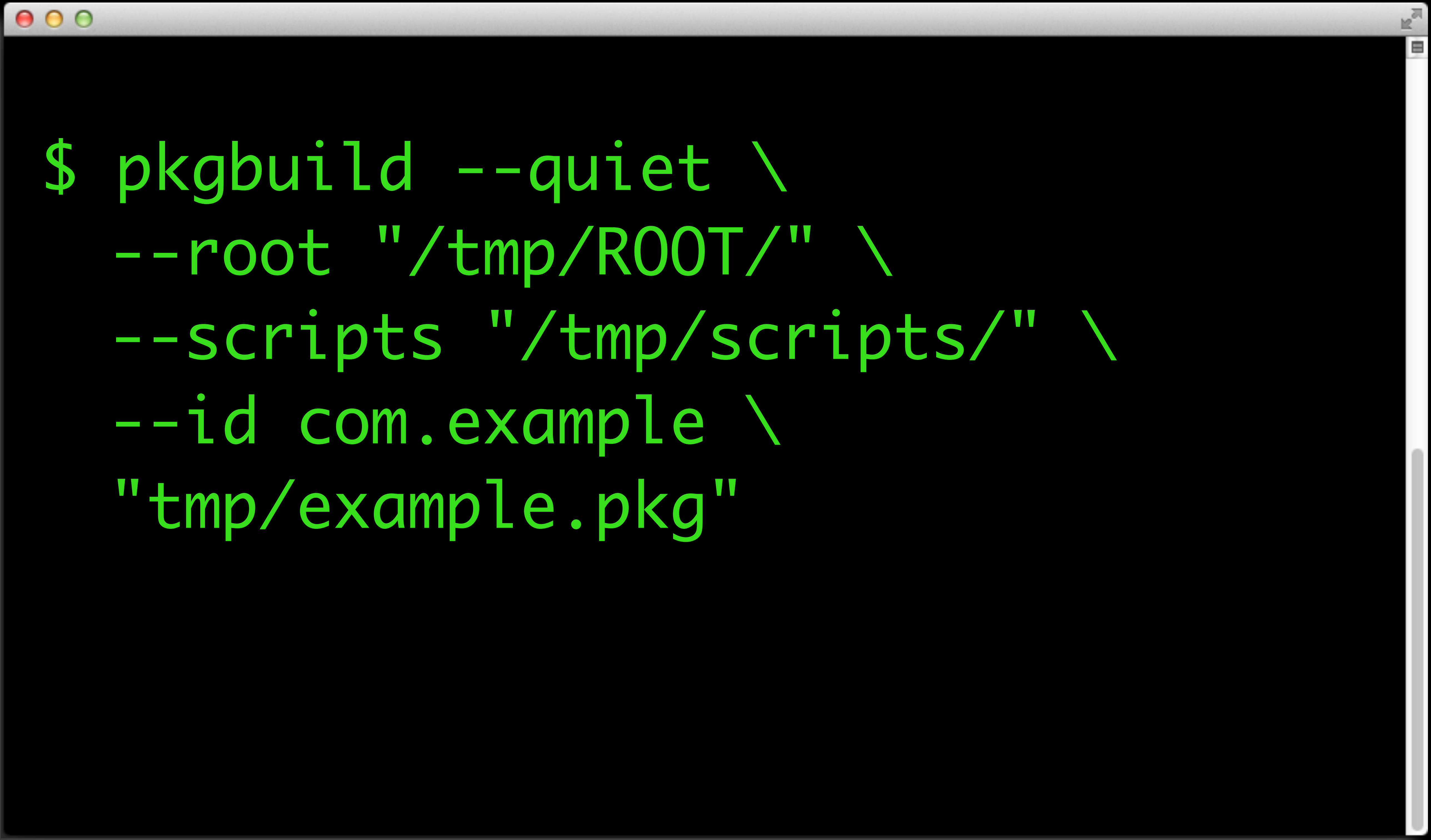
Feedback: http://j.mp/psumac13



```
$ mkdir -p /tmp/examplefolder/
$ hdiutil create \
  -volname exampleVolume \
  -srcfolder "/tmp/examplefolder/" \
  -ov \
  -format UDZO \
  example.dmg
```



```
NAME
     pkgbuild -- Build an OS X Installer component
package from on-disk files

SYNOPSIS
     pkgbuild [options] --root root-path [--
component-plist plist-path] package-output-path

DESCRIPTION
     A ``component package'' contains payload to
be installed by the OS X Installer.
```



```
$ pkgbuild --quiet \
  --root "/tmp/ROOT/" \
  --scripts "/tmp/scripts/" \
  --id com.example \
  "tmp/example.pkg"
```

## What's Your Project?

## DEV TIME!

# Show & Tell

# What Now?

- Run Scripts With:
  - Apple Remote Desktop
  - Payload Free Package
  - LaunchD plist
- Scripting For Better Package Deployment: How to tame 3rd party updates - BYOD
  Room 107, Wednesday, July 9 • 1:30pm - 2:45pm
- Unix The Command Line
  Room 106, Thursday, July 10 • 9:00am - 10:15am
- Extending OS X Management Systems with Scripting
  Room 106, Friday, July 11 • 9:00am - 10:15am

## Help!

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

http://mywiki.wooledge.org/BashGuide
http://guide.bash.academy

http://developer.apple.com/library/mac/documentation/OpenSource/Conceptual/ShellScripting/ShellScripting.pdf

http://www.shellcheck.net

# Thank You!

Many thanks to everyone at the
#bash channel on Freenode for guidance.
Shout out to greybot. YTMND

## Feedback!

http://j.mp/psumac13