Bash, In A NutShell

RUSTY MYERS - SYSTEMS ADMINISTRATOR, PENN STATE
CHRIS LAWSON - IT CONSULTANT/SUPPORT SPECIALIST, PENN STATE ALTOONA
BASH, IN A NUTSHELL

• What is bash?
  • Brian Fox 1989
  • Bourne Again Shell (bash)
  • Command Interpreter
    • A shell program that interprets commands and executes them.
      • Kernel: Core of the OS (Mac OS X Mach kernel)
      • Shell: Outer layer of OS that interacts with the user, sending requests to the kernel for execution.
  • Binary at /bin/bash
  • Responsible for spawning subshells
Follow Along!

Open Terminal.app & Play Around.
BASH, IN A NUTSHELL

• Terminal.app
  • /Applications/Utilities/
• Lots of various commands available.
• Examples
  • man – Help manual.
  • cp – Copy one or more files to another location.
  • mkdir – Create a new directory/folder
  • mv – Move or rename files or directories
  • chmod – Change access permissions
  • install – Copy files and set attributes
  • shutdown – Shutdown or restart OS
TAKING GUI PROCESSES TO TERMINAL
SYNTAX OF A COMMAND

• `ls -la ~/Desktop/`
  • `ls` = Command or Utility (List files/directories)
  • `-la` = Options or Flags (-l = List in long format / -a = Include directory/hidden entries)
  • `~/Desktop` = Argument (Which directory to pull the information from)  * ~ = Home directory

* Options or flags can also have arguments depending on the command.
USEFUL TIPS/SHORTCUTS

• ~ = Shortcut for Home directory
• tab complete = tap tab to complete text. Type ~/Des and hit tab to complete ~/Desktop/
• ↑↓ = Toggle through previous commands in that session.
• !(Command) = Will read your history and run the most recent run of the matched command.
• !! = Runs the last command
• Control + L (or Command+K) = Clear Terminal window.
• Control + C = Stop the current process.
USEFUL COMMANDS

# Bring up manual for ping command.
$ man ping

# Send packets over the network to test connectivity.
$ ping -c 5 www.apple.com

# Watch for packets transmitted and received and packet loss.
# Check network status.
$ ifconfig
# Print working directory.
$ pwd

# Change to another.
$ cd /Users/\$USER/Desktop/

# Print working directory.
$ pwd

# Should be /Users/\$USER/Desktop/
# Creating directories

# Create a single directory on the desktop.
$ mkdir ~/Desktop/Test1/

# Create a directory structure including all parent directories.
$ mkdir -p ~/Desktop/Test1/Test2/Test3/

# Create a directory structure including all parent directories setting permissions to read, write, and execute for the owner, the group and everyone else for the Test3 directory.
$ mkdir -pm 777 ~/Desktop/Test1/Test2/Test3/
# Create the test.txt file in the Test1 directory.
$ touch ~/Desktop/Test1/test.txt

# Read the file.
$ cat ~/Desktop/Test1/test.txt

#Append “Hello World” in test.txt
$ echo “Hello World” >> ~/Desktop/Test1/test.txt

# Display file. Hit “q” to quit.
$ less ~/Desktop/Test1/test.txt
MOVING FILES/DIRECTORIES

# Moving a single file to another directory
$ mv ~/Desktop/Test1/test.txt ~/Desktop/Test1/Test2/Test3/

# Moving a single Directory.
$ mv ~/Desktop/Test1/Test2/Test3/ ~/Desktop

# Rename a directory (or file) using the “mv” command.
$ mv ~/Desktop/Test1/Test2/ ~/Desktop/Test1/Test4/
DELETING OLD FILES

# Delete One File
$ rm /Users/$USER/Library/Preferences/com.apple.Safari.plist

# Delete One File From All Homes
$ sudo rm /Users/*/Library/Preferences/com.apple.Safari.plist

# Delete All Files in Caches
$ rm -R /Users/$USER/Library/Caches/
USING GOOGLE TO FIND THE WAY

• Search Terms
  • +terminal
  • +osx
  • +mac

• Additional Resources
  • http://www.defaults-write.com
  • http://apple.stackexchange.com
  • http://hints.macworld.com
RUN SOFTWARE UPDATES

# List all available updates.
$ softwareupdate -l

# Each update specified by args is downloaded but not installed.
$ softwareupdate -d

# Each update specified by args is downloaded and installed.
$ softwareupdate -i

# Enable or Disable Scheduled Check of Software Updates
$ softwareupdate --schedule [on|off]
# Install Pkg On Desktop

$ installer -pkg ~/Desktop/VLC-2.2.4.pkg -target /

# Print the last 100 lines of the install log to confirm success

$ tail -100 /var/log/install.log

# Waiting and monitoring the install process.

$ tail -f /var/log/install.log
# List profile information
$ system_profiler

# List all data types available.
$ system_profiler -listDataTypes

# Get computer serial number
$ system_profiler SPHardwareDataType | awk -F ': ' '/Serial Number/ {print $2} 
C04N51BFG9QK
# Find files matching case insensitive term
$ find ~/* -iname "*.sh"

# Find Files Modified > 3 Days Ago (DISCLAIMER: DO NOT RUN THIS COMMAND. It will find and erase all files modified or created in the last 3 days.)
$ find ~/* -mtime +3d -print0 | xargs -0 rm

# Use metadata search
$ mdfind -onlyin ~/* -name .sh

# Find PDFs in my Home
$ mdfind -onlyin ~/* "kMDItemContentType == 'com.adobe.pdf'"

# List kMDItemContentType
$ mdls mcxToProfile.py
kMDItemContentType = "public.python-script"

$ mdfind -onlyin ~/* "kMDItemContentType == 'com.adobe.pdf'"
# Watch the file system for changes.
sudo fs_usage -w | grep -i TextWrangler| grep -i plist

18:50:39.330969    WrData[AT2]     D=0x04d77428  B=0x2000   /dev/disk1  /Users/rzm102/Library/Preferences/com.barebones.textwrangler.plist.TDKJ8ly

# File System Writing Data: "WrData[AT2]"
# To a Plist: "/Users/rzm102/Library/Preferences/com.barebones.textwrangler.plist.TDKJ8ly"
# Domain: com.barebones.textwrangler

$ defaults read com.barebones.textwrangler UserName
Rusty Myers

$ defaults write com.barebones.textwrangler UserName "TheSpider"
# Create a profile! https://github.com/timsutton/mcxToProfile
COMBINING CODE INTO A SCRIPT
EDITING A SCRIPT

- Script Editors
- Nano
- Vim (Vi)
- Atom
- TextWrangler
#!/bin/bash
# Basic script
echo "Hello World"
RUNNING A SCRIPT

$ chmod u+x script.sh
$ chmod 755 script.sh
$ /bin/bash script.sh
Hello World
$ ./script.sh
Hello World
$ /Users/rzm102/script.sh
Hello World
AUTOMATING THROUGH BASH SCRIPTING
#!/bin/bash
# Time Zone Setup
systemsetup -settimezone America/New_York
# Set to use Network Time Server clock.psu.edu
systemsetup -setusingnetworktime on
systemsetup -setnetworktimeserver clock.psu.edu
echo "Network Time Configured"
INSTALL SOFTWARE

#!/bin/bash


installer -pkg /tmp/Office2016.Company.pkg -target /

curl -o /tmp/PerfectSoftware.Company.pkg http://company.com/pkgs/PerfectSoftware.Company.pkg


for myPKG in /tmp/* . pkg; do
    installer -pkg "$myPKG" -target /
done
#!/bin/bash

# Skip First Run for All Apps

$ for i in $domains ; do
defaults write $i kSubUIAppCompletedFirstRunSetup1507 -bool true
done

https://macops.ca/disabling-first-run-dialogs-in-office-2016-for-mac
VARIABLES
DOUBLE QUOTES

• Double Quotes " " = Escape all except:
  $  `  \\
• \ = Escapes next character

SINGLE QUOTES

• Single Quotes ' ' = Escapes all

INTERNAL FILE SEPARATOR IS WHITESPACE
$ echo $PWD
/Users/student

$ echo "$PWD"
/Users/student

$ echo '$PWD'
$PWD

$ echo \$PWD
$PWD
# Create new variable using alphanumeric
$message = "Hello"
$ending = "World"
$echo "$message" "$ending"
Hello World

# List all built in variables
$env

# List the user name and home folder.
$echo "$USER - $HOME"
# Create new variable using $( command )

$ date
$ today=$(date)
$ echo $today
# Decide what to do based on model type

macModel=$(system_profiler SPHardwareDataType | awk '/Model Identifier/ { print $3 }')

$ echo $macModel
MacBookPro11,3

$ if [[ "$macModel" == *Book* ]]; then
  echo "This is a portable"
fi

This is a portable
$
CATCHING ERRORS
EXIT STATUS

$ cat script.sh
#!/bin/bash
echo "ERROR"
exit 2

$ ./script.sh
ERROR

$ echo $?
2

EXIT STATUS
#!/bin/bash

diskutil eraseDisk HFS+ Untitled disk1

exitValue=$(echo $?)

echo "Disk Utility Exit Value: $exitValue"

if [ $exitValue -gt 0 ]; then
  echo "Disk Utility did not complete successfully"
else
  echo "Disk Utility completed successfully"
fi
# Make a Beep if eraseDisk fails
$ diskutil eraseDisk disk1 || osascript -e 'beep'

# Copy new profile to MDM server
$ ./mcxToProfile.py --plist com.barebones.textwrangler.plist
--identifier rzm102Textwrangler \
&& scp rzm102Textwrangler rzm102@profileserver.com:/MDM/Profiles/
ShellCheck
finds bugs in your shell scripts.
You can cabal, apt-get, yum or brew install it right now.
Paste a script to see what it will be like:

Your Editor (Ace)

Your Terminal

$ shellcheck myscript
Line 6:
if [[ $errorMessage == Disk* ]]; then
  ~- SC2154: errorMessage is referenced but not assigned.
$
# Questions?
# What do you want to script?
MacAdmins2016$ /bin/bash /Users/Rusty&Chris/BashInANutShell.sh
Thanks For Coming!

MacAdmins2016$ echo $? 0