PSU Mac 2012
Intro to Scripting

I really do mean “intro”

Nick McSpadden
Client Systems Manager
Schools of the Sacred Heart, San Francisco

Scripting: Why?

Anything you do more than once, you can find a way to do it more than once **faster.**

Repetitive tasks are annoying. Scripts take out the tedium from the tasks by automating them, and only bother you when you want them to. Set it, forget it!

Laziness tomorrow is the mother of invention today.

You can impress everyone at your high school reunion.

Scripting: When?

Simple logic is best

If it's easy to document so someone else can do it

If it feels like a waste of time...

If the task is repetitive

If something needs to be done regularly that shouldn't take up your time

Scripting: How?

- There's tons of different ways!
  - Shell (commonly bash)
  - Perl or Python
  - PHP
  - Ruby
  - AppleScript/Automator

Scripting: How?

There are a lot of programming elements involved.

The good news is, most of the hard work is done by the environment already; all you have to do is provide logic.

You don't have to be a brilliant programmer to write scripts to make your life or job easier.

Shell Scripting

- What is the shell?
  - It's your command line environment.
  - It's your interpreter.
    - It takes input from stdin (your input) or a file and figures out what to do with it.
    - Examples: sh, csh, tcsh, bash, ksh
    - By default, Mac OS X uses bash.

Shell Scripting

- What's a shell script?
  - Like any program, it's a series of instructions.
  - It can have input and output of different kinds.
  - Most environment details are already configured by the shell itself, so all you need to do is provide logic.
  - Bash makes it easy to jump right in.

Shell Scripting

- Main parts of bash scripting
  - #!/bin/bash
  - echo Text
  - read
  - if then elif else fi
  - loops
  - exit

Shell Scripting: Where?

- It's probably easiest from the Terminal.
- Make a new file ending in .sh
- First: #!/bin/bash
- echo "Hello World!"
- exit 0

Run it!
sh file.sh
./file.sh

Why did the last one fail?

Shell Scripting: Run!

- sh file.sh
  - Invokes the shell to interpret the contents of the file
- ./file.sh
  - Reads the interpreter from the top of the file
  - Requires execution privileges! chmod +x
- http://askubuntu.com/questions/22910/what-is-the-difference-between-and-sh-to-run-a-script

Shell Scripting: User input

- echo “What is your favorite iOS app?”
- read NAME
- Repeat the user’s input back to them:
  - echo “Your favorite app is $NAME”
- Run it!

Shell Scripting: Variables

- variable=value
  - No spaces!
  - = assigns variables, **but** can also be test operator in the right context, so **be careful**.
- Variables have no type. You can overwrite them with a completely different kind of value at any point, so **be aware**.

Shell Scripting: Using Variables

- echo $variable
  - $ must proceed the variable for the interpreter to pull out the **value** instead of the **name**

variable=10

echo variable
variable

echo $variable
10

Shell Scripting: If

if then elif else fi

```
if [conditional]
then
do something
elif [other condition]
then
do something else
else
do something different
fi
```

Shell Scripting: File Test Operators

- -e (file exists)
- -f (it's a regular file, not a directory nor device)
- -h (file is a symlink)
- -s (file is not empty)
- -r (file can be read)
- -w (file can be written)
- -x (file can be executed)

Shell Scripting: File Test Operators

- -O (user is owner of file)
- -G (group-id of file is same as user's)
- -N (file modified since last read)
- -nt (newer than another file)
- -ot (older than another file)
- ! (negation)

Shell Scripting: File Test Operators

```
if [-e file1]
then
do something to file1
elif [ file2 -nt file3 ]
suggest the user use file 2 instead
fi
```

Shell Scripting: Loop de Loop

- for loops
  - for **arg** in [list]
- These are **very** different from other programming loop structures.
- Takes an argument and assigns it the value of each variable in the list

Shell Scripting: Loop de Loop

```
for planet in “Mercury Venus Earth Mars Jupiter Saturn Neptune”

do
   echo $planet
done
```

Shell Scripting: Loop de Loop

- while loops
  - while [condition] is true, do something
- until loops
  - until [condition] is true, do something

Shell Scripting: Ending

- exit
  - Exits with the status of last command run.
- exit 0
  - Exits as “successful.”
- exit 1-255
  - Exits “unsuccessful.” Return value is relevant for error checking.

Shell Scripting: Piping

- Command A | Command B
- Command A output -> Command B input
- lsof | grep <something>
  - Use grep to search for something in the list of open files

Shell Scripting: DEMO

Please work, please work, please work...

Scripting: Other languages

- Perl, PHP, Python, Ruby... even C
- Many of them are invoked the same way - a script that has a different interpreter at the top, or manually invoked via the correct environment:
- i.e. /usr/bin/perl perlscript.pl

Scripting: AppleScript

- AppleScript is built into Mac OS X.
  - /Applications/Utilities/AppleScript Editor.app
    - You can run shell scripts in AppleScript with “do shell script”
- Automator
  - /Applications/Automator

Scripting: AppleScript

- AppleScript Editor -> Open Dictionary... is a great way to find out scripting possibilities for applications.
- Most of the Apple dictionaries are also available online.
- These provide a nice, very readable interface to otherwise clunky scripts.
- There isn't really a better way to interact with the GUI.

Scripting: Repeat it Repeat it again

- Other *NIX systems: Cron
  - Easy to use it, provides simple time-based execution of commands.
- Mac OS X: launchd
  - Execution based on numerous conditions.
  - Harder to get into than cron, but extremely powerful.

Scripting: launchd

- Launchd is a powerful beast.
- Pid 1 - it starts the OS at boot.
- Rather than using a single command like crontab to manage, you create plists that contain your launch parameters, and store them in (~)/Library/LaunchAgents/ or .../LaunchDaemons/.
- There's a lot to learn about launchd, and it's worth reading tutorials online.

Learn by Doing

The best way to learn is by doing
it yourself.

Make a test machine or VM and
go crazy on it!

Resources

- http://tldp.org/LDP/abs/html/ -- Advanced Bash-Scripting Guide
- http://steve-parker.org/sh/sh.shtml -- Steve's Bourne / Bash shell scripting tutorial
- http://www.afp548.com/article.php?story=20050620071558293
- http://www.devdaily.com/mac-os-x/launchd-examples-launchd-plist-file-examples-mac
- http://www.macenterprise.org/

Thanks for coming!