Beginners Shell Scripting

for

Batch Jobs

Evan Bollig

-and-

Geoffrey Womeldorff

SCS
Florida State University
School of Computational Science

Before we begin...

- Everyone please visit this page for example scripts and grab a crib sheet from the front
  - http://www.scs.fsu.edu/~bollig/TechSeries
- Also please sign in on the clipboard being passed around

Outline

- Introduction to Shells
  - History
  - What is the *nix “ENV”?
  - What is Shell Scripting?
- Novice Scripting
  - HelloWorld (HelloWorld.sh)
  - Variables (HelloWorld2.sh)
  - \`...\`, '...' and “...” (Example3.sh)

Outline (2)

- Intermediate Scripting
  - File and Output Control (Example4.sh)
  - Loops, Conditionals and Utilizing Available Programs (Example5.sh)
- Advanced Scripting
  - Substrings and Case-Statements for Machine and Architecture Specific Tasks (Example6.sh)
  - Restart Files, Integers and Avoiding Redundant Tasks (Example7.sh)
  - Calling Matlab for Batch Processing (Example8.sh)

Introduction to Shells

What are *nix Shells?

- Unix was the first OS to separate the command interpreter (shell) from the operating system (kernel)
- Shells abstract the complexities of systems and provide standard commands for all machines
- A shell has built in commands, plus allows us access to other Unix commands
  - history, export, cd --> built-in
  - cp, ls, grep --> Unix commands (/bin, /usr/bin, etc.)
- http://www.phys.ualberta.ca/~gingrich/research/shells/shells.html

SCS
Florida State University
School of Computational Science

A General History

- Bourne Shell (/bin/sh):
  - Bell Laboratories, by Stephen Bourne
  - Basis for all shells, but provided limited functionality
- C-Shell (/bin/csh):
  - UC Berkley, by Bill Joy (creator of "*vi*")
  - Expanded Bourne Shell to mimic C-language and provide interactivity (in reality, it was too buggy to use for scripts)
  - TENEX-csh (/bin/tcsh) was later developed to get rid of bugs but Unix manufacturers stuck with csh
- http://www.softlab.ntua.gr/facilities/documentation/unix/shelldiff.html#3

SCS
Florida State University
School of Computational Science

History (2)

- Korn Shell (/bin/ksh):
  - AT&T, by David Korn
  - Hybrid of sh and csh using sh language with csh interactivity
  - Although it was the best of both worlds, it was still from AT&T => ksh was not free.
- Bourne Again Shell (/bin/bash):
  - GNU's POSIX compatible Bourne Shell
  - Equivalent to Korn but free implementation
  - Perfect fit in Linux which resulted in mass popularity
- rc, zsh and others are still emerging...

Understanding the “ENV”

- The “ENV” is the Unix environment
  - Key=value pairs describe current configuration
  - Every running application, user, etc that interacts with the operating system has an environment
  - A process inherits its parent's environment unless the parent overrides/specifies other properties.
    - Users' initial ENV is provided by the shell and based on /etc/profile.d/*, /etc/*shrc and ~/.*shrc files.
  - Changes to child's environment only affect child's children and are lost when the subtree exits.

Controlling the “Env”

- Shells allow us to control the environment with built-in commands; Unix commands can also be used:
  - %> env
    - Lists your current environment
  - %> export <key>=”<value>”
    - Set a new key or update an existing one
    - NOTE: this is equivalent to setenv <key> “<value>” in the C-shell family
  - %> echo $<key>
    - Print current value of variable

Understanding Shell Scripts

- Shell scripts are files that contain commands as you would run them on the command prompt.
- Scripts commonly use variables, conditionals and loops.
  - These are all available on the command prompt
- What makes a script so special?
  - It has the executable bit set (chmod +x <script>)
  - It executes until it encounters an error (interpreted)
  - Scripts can use any available executables; no need to worry about libraries, headers or APIs

Why Script?

- Quick and dirty solutions
  - Mostly you use pre-existing binaries
  - Changes to script have immediate effects
  - no need to write/(re-)compile code
- Inter-program communication is performed through standard file I/O pipes
  - No worries about communication standards
- Shell Scripts use shells that are standard on all *nix platforms
  - NOTE: only the shells are standard. This is not always true for the commands a script contains.

Novice Shell Scripting

HelloWorld

- Example 1: HelloWorld.sh
- Key features:
    - Sh-bang (#!/bin/sh)
        - Every shell script must start with a sh-bang on the first line. This tells the executing shell what type of shell should interpret it (i.e. sh, bash, csh, perl, python, tcl, etc.)
    - Comments
        - Comments begin with the hash (#) and proceed to the end of the line

Variables

- Example2: HelloWorld2.sh; Hello.sh
- Key Features:
  - Local and Global Variables
    - <key>=<value> is a local variable that is not passed to children
    - export <key>=<value> is a global variable that is passed to children but is not visible to parent of this script

`...`, '...' and “...”

- Example3: Example3.csh
- Key Features:
  - Use of different shell (see csh in sh-bang)
  - `...`: execute program performing substitution on variables
  - '...': strings without variable substitution
  - “...”: strings with variable substitution

Intermediate Shell Scripting

File and Output Control

- Example4: Example4.sh; HandsFreeCharmm.sh
- Key Features:
  - cat << EOF ...
    - Print everything between “<< EOF” and the next occurrence of “EOF” to stdout.
    - NOTE: the > example4.out to the right of “<< EOF” redirects stdout to a file
  - Redirecting both stdout and stderr
    - Use ([command] > [stdoutFile]) > [stderrFile] to store output separately
    - Use [command] &> [outFile] to store together

Loops, Conditionals and Utilizing Available Programs

- Example5: Example5.sh
- Key Features:
  - For-loop executes on list of filenames
  - Conditional (if-then-else) tests for a lock file to avoid extra work
  - We take advantage of standard Unix commands like mkdir, pwd, grep and echo but also use non-standard Imagemagick to perform batch image conversion

Advanced Shell Scripting

Substrings and Case-Statements for Machine and Architecture Specific Tasks

- Example6: Example6.sh
- Key Features:
  - Case-statement to compare machine's name (hostname)
  - Case-statement to compare architecture of machine
  - Substrings from variables using ${var_name#*/} and ${var_name%/*}

Restart Files, Integers and Avoiding Redundant Tasks

- Example7: Example7.sh
- Key Features:
  - Restart file informs script what tasks it has already completed
  - Integers require unique handling (we cannot compare like strings)
  - Locates ALL jpegs in ALL subdirectories and rebuilds subdirs in output folder (important to keep batch jobs organized)

Calling Matlab as for Batch Processing

- Example8: Example8.sh; testMatlab.m; mandrill_cvt.m
- Key Features:
  - Per machine tasks
    - different compression for each cluster: Class*, Prism*, Hallway*, and Vislab machines
  - Executes Matlab behind the scenes with a complex clustering algorithm for data compression (easily coded in Matlab and easy to visualize results)

Conclusion

- You now have example shell scripts to get started.
- Use the resources on the crib sheet and on the SCS TechSupport Twiki to help you write your own scripts.
- Use logging, locks and restart files to help accelerate batch jobs.
- Remember: don't reinvent the wheel! If a command exists and it functions correctly, don't waste your time re-writing it from scratch. If it has bugs then its another story...

SCS
Florida State University
School of Computational Science

Special Thanks...

- to TSG for the excuse to play with shell scripts
- to Gordon Erlebacher for the extra time allotment away from WATT
- to SCS Custodial for making life interesting during the wee-hours of the morning
- And of course! Thank you all for attending...

SCS
Florida State University
School of Computational Science