Week 1 – Lesson 2: Creating Shell Scripts, Linux Commands

Chapter Objectives

- In this chapter, you will:
  - Learn how to create Shell Scripts
  - Commenting / Making Portable Shell Scripts
  - Making Shell Scripts Executable
  - “Learning How to Learn”

Creating Shell Scripts

- As mentioned in the previous lesson, you will need to learn how to automate routine tasks sometimes when you are an IT professional.

- One way to do this is to create a Shell Script. This is an interpreted language that uses an interpreter to read and execute commands within a file.

- The file contains commands (the term “script” refers to “text” – the commands themselves)

Creating Shell Scripts

- When creating shell scripts, make certain that you select a name for that script that will NOT be confused with other command names already known by the shell.
    - For example, creating a script called “ls” would not be a good name for your shell script, since the “ls” command is already known to the shell
    - A good idea to avoid this confusion is to add an extension the name corresponding to the shell that will be running this shell script.

Creating Shell Scripts

Examples of shell script extensions:

filename.**bash** – script to be run in the Bash Shell

filename.**csh** - script to be run in the C Shell

filename.**sh** - script to be run in the Bourne Shell

filename.**ksh** - script to be run in the Korn Shell

Creating Portable Shell Scripts

- As you may have learned in a previous Linux course, Shells have evolved over the years, each one introducing an newer feature or syntax.

- The problem is, that newer shell script syntax may not be compatible when executed in older shells, and may actually may not allow the shell script to properly run!

- A method should be used to “force” the shell script to run in that specified shell. If that particular shell is NOT available on that machine, the shell script will simply fail to run at all! (This prevents the program from crashing while running…)

Creating Portable Shell Scripts

- In shell scripts, the # symbol can be placed BEFORE text to allow the interpreter to IGNORE that text for the remainder of the line.

- A special comment can be placed at the BEGINNING of the FIRST LINE of the shell script to force that shell script to run in a specific shell…

  For Example:

```
#!/bin/bash      <- Run in Bash Shell
#!/usr/bin/csh   <- Run in C Shell
#!/usr/bin/ksh   <- Run in Korn Shell
#!/usr/bin/sh    <- Run in Bourne Shell
```

Creating Portable Shell Scripts

There are a few important fact to know about this “special comment”:

- This special comment is referred to as the she-bang line (The symbol # is called “she”, the symbol ! is called bang)
- This special comment MUST be in the first line of the shell script, and the #! symbol MUST be the first two characters on the line (otherwise, it is treated like a REGULAR comment…)
- The pathname follows #! to a valid pathname of the shell (which really is just a command). If the pathname is invalid or non-existant, then the shell script will NOT run. You can use the which or whereis command to locate the location(s) of the shell…

Creating Portable Shell Scripts

- When creating Shell Scripts, it is important to be able to use a text editor to create these files containing Linux commands.

- There are many text editors available such as nled, pico, emacs, and vi.

- **USEFUL TIP:**

  In the Resources section, there are useful online tutorials regarding how to use the vi editor, and how to issue common Linux commands. It is recommended that you perform these tutorials prior to you starting this week's lab…

Making Shell Scripts Executable

You cannot simply type the shell script by name to run the shell script. There is an important factor to consider first:

- **Does shell script have execute permissions?**
  - By default, the user-mask does NOT create execute permissions for newly-created regular files. You must either run the shell name with the shell script name as an argument (if it does not have execute permissions)

    eg. `bash  myShellScript.bash`

    or add or assign execute permissions:

    ```
    chmod +x myShellScript.bash

    ./myShellScript.bash
    ```

    Note: you can use ./ to force the shell script to be run from the current directory, or you can add the directory pathname to the PATH variable – more on that later…

Learning How to Learn

- By now, you should realize that there are many commands in Unix and Linux (over 1500 in Unix and over 2500 in Linux).
- Instead of trying to memorize all commands, it is better to develop a strategy:
  - Create a reference sheet of common commands and methods for quizzes, midterms and final exam
  - Learn to get online help for commands and their options as you practice the labs, and work on assignments or projects…

Learning How to Learn

- The normal rule for creating a reference sheet to help on tests, and exams is use just one 8 ½ by 11 inch sheet of paper. Commands and methods can be hand-written on both sides, but cannot be machine generated (i.e. not by a printer – why? To help the memorization process + not permitted on tests/exams).

- Refer to Resources for a listing of common Linux Commands…

- For labs and assignments, you can use the man or info commands in Linux (or perform a net search on the Web)

  Eg. `man –k copy` <- helps to search for commands for "copy"

Chapter Summary

- Use a script name that is not confused with an existing Linux command (use shell name as extension).
- A she-bang line at the top of the script forces (ensures) that the shell script runs in the shell it was designed for, or does not allow the shell script to run if shell is missing or pathname to shell is incorrect (common error message: “Bad Interpreter”
- You need to give shell script filename executable permissions prior to running by script filename only.

Chapter Summary

- You can either add the directory pathname into the PATH environment variable to run the shell script, or you can force the shell script to be run from the current directory by using the relative pathname ./
- It is highly recommended to create a reference sheet (8 ½ by 11 inch, both sides, handwritten) containing commands and methods to assist you when writing OPS435 quizzes, midterms and final exams…
- In order to be a good Linux system administrator and/or shell-scripter, you need to "learn how to learn", and use the man, info, or Net searching skills to obtain information on how to properly use Linux commands and see if there are any useful options available…