IT 240
Shell Scripting for Administrators

Chapter 1
Introducing Shells

Stan J. Senesy
IT Program/CCS
New Jersey Institute of Technology

Introduction

- Today's agenda:
  - Course Requirements
  - Introduction to Shells

Course Requirements

- Although there are no formal prerequisites for this course, it is expected that students will have completed the first year IT cores, in particular CS 115 & IT 102.
- If you have not had any previous programming courses, you will be at a significant disadvantage.

Course Requirements

- We will be using a cloud-based virtualization service called vCloud during the course
- In order to access vCloud, you need to have the campus VPN software installed and configured and have installed the Firefox browser

What is a Shell?

- Simply put - a shell is a process (program in execution) running on the computer
- It receives commands that the user types and the calls and them passes them to the operating system for execution

What is a Shell?

- Shells display a prompt ($ for user, # for root) and await commands
- When you type a command at the prompt and press return, the shell looks for a matching program in your current path
- The shell receives the results of the program and displays them on the screen

Why Use Shells?

- Before the GUI, shells were the only way to interact with a computer
- Unix was the first popular operating system to allow multiple simultaneous shells (users)

Why Use Shells?

- The Unix philosophy is that commands are small and functional and can be linked together to complete more complex tasks
- Shells facilitate this functionality by providing a built-in programming language
- Programs in this language are called shell scripts

Shell Types

- Bourne Shell (sh)
  - The original Unix shell
  - Widely available
  - Limited functionality
- C Shell (csh)
  - Syntax parallels the C language
  - Added functionality to sh
  - Heavily involved with BSD

Shell Types

- Korn Shell (ksh)
  - AT&T's response to the popularity of BSD
  - Backwards compatible with sh
  - Standard shell used with System V Unix
  - Created problems in the Unix/Linux community because it is not Open Source
  - Linux does not run ksh (see previous)

Shell Types

- Bourne Again Shell (bash)
  - Created as an alternative to ksh
  - The most popular shell currently in use
  - The default shell on Linux and post 10.3.x OS X systems (although it will be still be called sh on Linux)

Shell Types

- Ash Shell (ash)
  - Default shell in Cygwin for Windows
- Z Shell (zsh)
  - Focuses on interactive usage
- RC shell (rc)
  - Standard on Plan 9 OS

Changing Your Shell

- To change and set your default shell, run the *chsh* command as follows:
  - $chsh senesy /bin/bash
- Linux users will need to use the following syntax:
  - $chsh -s /bin/bash senesy
- Although the above commands work on OS X, Mac users may also set their terminal preferences

Using Windows

- By default, the cmd shell available in Windows is not sufficient for the projects we'll be completing in the course
- Cygwin or Microsoft's Windows Services for Unix will add additional functionality if you do not wish to use Linux, but this is not recommended

Which Shell Is Running?

- Entering the command below will show which shell is currently running on your system:
  - $echo $SHELL
- Remember that some shells will masquerade as sh or others

Fun Shell Commands

- To show the OS name:
  - $uname
- To show more detail:
  - $uname -o
- To show hardware platform:
  - $uname --hardware-platform
  - $uname -p

Fun Shell Commands

- Combining it all together:
  - $uname -a
  - $uname --all
- To get more info:
  - $man uname

Command Editing

- The *backspace* key deletes from the end of the command
- *Ctrl-C* cancels the command
- *!!* allows a previous command to be repeated
- *!$* repeats the last argument in the previous command

Command Editing

- The *up* and *down* arrow keys may be used to cycle through the command history buffer
- *history* may be used to view the entire contents of the command buffer
- *!3* will recall the third command in the buffer, etc

Using an Editor

- To set the program used for editing:
  - $set -o vi
  - $set -o emacs

Filename Completion

- The *tab* key may be used to automatically complete the current line that is being typed
- *Esc-?* may be used to view possible arguments

Wildcards

- * matches one or more characters
- ? matches exactly one character

Background Execution

- By default, programs run in the foreground and will not return the command prompt till execution is finished
- To run a program in the background, put the & after the program name