Shell Scripting with Bash

Reindert-Jan Ekker
http://nl.linkedin.com/in/rjekker/
@rjekker

pluralsight
hardcore developer training

Overview

- Output
  - echo, printf
- Input
  - read
- Standard streams
- Input and Output redirection

echo

- Echo
  - Prints its arguments to standard output, followed by a newline
  - -n suppresses the newline
  - -e allows use of escape sequences
    - \t: tab
    - \b backspace
  - These options are not portable to non-bash shells

printf

- printf
  - Can do more sophisticated output than echo
  - Uses a format string for formatting
  - Will not append a newline by default
- For help:
  - http://goo.gl/ZThKCj
  - help printf
  - man printf
  - man 3 printf

- -v will send output to a variable

read revisited

- read
  - Reads input into a variable
  - “read x”
  - No variable specified? Will use REPLY
  - -n or -N specifies number of characters to read
  - -s will suppress output (useful for passwords)
  - -r disallows escape sequences, line continuation
  - Good Habit: always use -r
  - Several more useful options (see help)

- read can read words in a line into multiple variables
  - read x y
  - input “1 2 3”: x=1, y=”2 3”
  - Uses IFS variable for delimiters

Standard Streams

- Three standard streams
  - input, output, error
  - Represented by number (file descriptor), or special file
- 0: Standard Input (stdin)
  - /dev/stdin
- 1: Standard Output (stdout)
  - /dev/stdout
- 2: Standard Error (stderr)
  - /dev/stderr
  - Used for diagnostic or error message
- /dev/null discards all data sent to it

Redirection 1

- Get input from somewhere else, send output or errors somewhere else
- Input redirection: <
  - grep milk < shoppingnotes.txt
- Output redirection: >
  - ls > listing.txt
  - Will overwrite existing files!
  - (although this can be customized with the set command)
  - >> appends to the end of a file
- Pipes
  - ls | grep x

Redirection 2

- Redirect a specific stream with N>
  - “cmd 2> /dev/null” discards all errors
- Redirect to a specific stream with >&N
  - >&2 sends output to stderr (equivalent to 1>&2)
  - 2>&1 redirects stderr into stdout
- Sending both error and output to a single file
  - cmd > logfile 2>&1
  - Don’t do this: cmd > logfile 2> logfile
  - Don’t use &> or >&
- Allowed anywhere on the command line, but order matters
  - cmd < inputfile > outputfile
  - >outputfile cmd < inputfile
  - cmd >logfile 2>&1 (sends errors to the logfile)
  - 2>&1 >logfile cmd (sends errors to stdout)

Summary

- Advanced read usage
- echo and printf
- Three standard streams
  - stdin
  - stdout
  - stderr
- Input redirection: <
- Output redirection: >, >>
- Redirecting a specific stream: 2>
- Redirecting into another stream: 2>&1