Shell Scripting with Bash

Variables

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

pluralsight
hardcore developer training

Overview

- **Variables**
  - Create
  - Assign value
  - Use value
- **Variable names**
- **Good habits when using variables**

Variables

- **Used to store data by name**
- **To create: just assign a value**
  - x=10
  - If x already existed, it is assigned the new value
  - filenames="notes.txt picture.jpg movie.mov"
  - Values containing spaces: use quotes
  - Don't use whitespace around =
- **To get the value**
  - Prefix with $
  - echo $x
- **Bash variables have no type**
  - Basically just store a string

Variable Names

- **Names:**
  - Only letters, numbers, and underscore are allowed
  - First character should be a letter or an underscore
  - Variable names are case-sensitive

- **Uppercase variables:**
  - Bash has many pre-defined variables
  - PATH, HOME, SECONDS, IFS, etc.
  - You don't want to override them by mistake

- **Good Habit:**
  - Use lowercase names for your variables

Using Variables

- **Good habit: surround your variables with quotes**
  - Use "$x" instead of $x
  - Prevent surprises when it contains spaces
  - Use double quotes: keep meaning of dollar sign intact

- **Braces**
  - Where does your variable name end?
  - echo "${foo}bar"
  - prints value of var "foo" followed by string "bar
  - echo "$foobar" prints value of "foobar"
  - Using braces a lot is a Good Habit

- **Another good habit**
  - Use $HOME instead of ~

Reading Input

- **read**
  - Reads a line of input into a variable
  - read var
  - Is a shell builtin
  - “help read”
  - “man builtins”
  - read -p “Type your name: “ name

Summary

- **Variables**
  - Assign value
  - Get value ($)
  - No whitespace around =
- **Variable names**
  - Use lowercase names
- **Using variables**
  - Quotes
  - Braces
- **Reading input**
  - read
- **Debugging**
  - Use -x option in hashbang line
  - Or use “set -x” to enable and “set +x” to disable