Shell Scripting with Bash

Variables 2

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

pluralsight
hardcore developer training

Overview

- **Variable attributes**
  - declare
- **Integer variables**
- **Arithmetic expressions**
- **Read-only variables**
- **Exporting variables**
- **Arrays**

Variable attributes

- **Variables hold simple string values**
  - But can also have extra attributes
- **Turn these on/off with declare**
  - You can also use “typeset” (but that is deprecated)
- **Print attributes for a variable**
  - declare -p var

Integer Variables

- **Integer variables**
  - declare -i num
  - Now $num can only hold numbers
  - Trying to set it to something else will NOT give an error
  - Instead, this will set a value of 0

- **Unset an attribute with +**
  - declare +i num

- **Triggers arithmetic evaluation**

Arithmetic Expressions

- **C-like syntax for doing calculations**
- **let command**
  - let n=100/2
- **((..))**
  - ((++x))
  - ((p=x / 100))
  - ((p= $(ls | wc -l) * 10))
  - This is a command equivalent to let
- **$((..))**
  - This a substitution, not a command
  - p=$((x / 100))
- **With a variable declared as an integer**
  - num="30 % 8"

Arithmetic Expressions 2

- **No need to quote variables**
- **((..)) can be used in if, while**
  - 0 is false, anything greater than 0 is true
  - ((0)) || echo “false”
- **Pitfall: numbers with leading zeros are interpreted as octal**
  - So 010 = 8
- **((..; ..; ..)) syntax in for loop is NOT an arithmetic expression**
  - but the three expressions separated by ; are

- **More information:**
  - http://goo.gl/HnPkiq
  - Or the bash man page

Read-only variables

- **Read-only variables**
  - declare -r constant=”some value”
  - Cannot give $constant another value
  - Bash will report an error

Exporting variables

- **By default, variables are local to your script**
  - Or terminal session
- **Export a variable**
  - To make it available to subprocesses
  - You cannot pass a variable to the program that runs your script
- **export var**
  - export var=”value”
- **declare -x var**
  - declare -x var=”value”
- **Attributes are not exported**

Arrays

- **An array can hold multiple values**
  - Stored and retrieved by index
- **Storing a value**
  - x[0]=”some”
  - x[1]=”word”
- **Retrieving a value**
  - ${x[0]} : some
  - ${x[1]} : word
  - ${x[@]} or ${x[*]} retrieve all values (quoting works like $*, $@)
- **declare -a x**
  - Or simply assign with an index like above
- **Initializing an array:**
  - ar=(1 2 3 a b c)

Arrays 2

- **Count the number of elements in $array**
  - ${#array[@]}
- **The indices in $array**
  - ${!array[@]}
  - There can be gaps in the indices
- **You cannot export an array**
- **Bash 4 supports associative arrays**
  - Where elements are stored and retrieved by a name, not an index
  - declare -A array
- **More information**
  - http://goo.gl/g6xtca

Summary

- **Integer variables**
  - declare -i
- **Arithmetic expressions**
  - ((..))
  - $((..))
- **Read-only variables**
  - declare -r
- **Exporting variables**
  - declare -x
  - export
- **Arrays**