Linux, Bash, and System Administration Cheat Sheet

by beersj02 via cheatography.com/89657/cs/20407/

Bash Scripting Basics

#!/bin/env bash — the 'shebang' used to tell the operating system the path it should use to interpret the file

bash *file-name*.sh — run the bash script in terminal

./ *file-name*.sh — run the bash script in terminal if set to executable

<*parameter*> — use in documentation to specify if a parameter is required when running script

[*parameter*] — use in documentation to specify is a parameter is optional when running script

— used to make comments throughout script

|| — logical OR

&& — logical AND

$# — resolved to the number of arguments that have been passed to the script

$0 — refer back to the script name

$1, $2, etc. — refer to user input (parameters) that user can add when running script, separated by a space

exit [0-255] — exit script and return number from 0 to 255. 0 means everything worked as intended, but other values can be used to denote errors that the script ran into

Bash Loops and Conditions

if fi — basic structure of all if-then-exit, if-then-else, or if-elif-else statements

if *condition* ; then *do-something* — if condition is met, do something

if *condition* ; then *do-something* else *do-something-else* — if condition is met, do something, otherwise do something else

if *condition* ; then *do-something* elif *condition2* ; then *do-something-else* else *do-final-thing* — if condition is met, do something; however if a different condition is met, then do something else; otherwise do the final thing

while *condition-is-true* ; do *action* done — perform the action as long as the condition is true

until *condition-is-true* ; do *action* done — opposite of while loop, perform the action until the condition becomes true

sleep *time* — sleep or wait for a specified number of second before continuing through script, usually performed within loops

Bash Loops and Conditions (cont)

for *value* in *list-of-values* ; do *thing-with-value* done — iterate over a list of values

for ((*counter=number* ; *counter<=number* ; *counter++*)); do *something* done — start at counter is equal to a number, then do something and increment the counter by 1 until the counter is greater than another number

for counter in {*starting-value*..*ending-value*}; do *something* done — brace expansion that iterates over a number range or character range from starting value to the ending value

{*starting-value*..*ending-value*..*increment-value*} — specify the increment value in a for loop, otherwise the default is 1

for ((; ;)); do *something* done — infinite loop

break — can add to while or for loops to exit from the loop but continue the rest of the script

continue — used to skip current iteration of a loop and continue to the next iteration of the loop

cut — cut different parts of a string

basename *path* — get the filename from a given path

Bash Arrays and Functions

***array*=(*"elements" "of" "array"*)** — create an array of strings

${*array*[0]} — get the first element of the array

${*array*[*]} — get all values in the array

${*array*[-1]} — get the last value in the array

${*array*[@]} — expand all of the array elements

declare -A *associative-array* — declare an associative array that allows string indices, similar to a dictionary in Python

***associative-array*=([*"association"*]=*"string"*)** — add an association to an associative array

***array*+=(*"new" "elements"*)** — append elements to the end of an array

shift — move argument $2 to $1

***function()* { *content-of-function* }** — define a function

alias — list all aliases defined in the current session

alias *alias*='*bash-command* ' — define an alias

type -a *command* — tells us if command is an alias