Using Functions and Loops in Scripts

Andrew Mallett
LINUX AUTHOR AND CONSULTANT

@theurbanpenguin www.theurbanpenguin.com

Module Overview

Building functions to aid your script development and maintenance

Passing parameters and using return values

Implementing FOR loops

Iterating within WHILE loops

BASH Functions

Functions are modular blocks of code that can be referenced by the function name; helping make your code more readable and aiding future development

BASH Functions

List shell functions: $ declare -F

List a function definition: $ type quote

Using a function: $ quote bob

```
prompt_user () {
  read -p "Enter a username: " user_name
  read -sp "Enter a password: " user_password
  echo ""
  read -sp "Enter a password again: " user_password_check
  echo ""
}
```

Modular Code

Prompting the operator for a user name and password is spread over five lines of our script. Creating a function makes good sense as we can easily run the function more than once within the script

```
if [ $# -eq 0 ] ; then
  prompt_user
  if [ "$user_password" != "$user_password_check" ] ; then
    echo "${0}: Passwords do not match!"
    prompt_user
  fi
…
```

Implementing Functions

Rather than exiting the script now if the passwords don't match we can re-prompt for fresh input

Demo

We will now add the basic prompt_user function into the script.

Function Positional Parameters

Using parameters in the function can reduce other code: $1, $2 ... .

Similar to scripts: echo ${1:-"Enter New User Credentials"}

Execute with parameter: prompt_user "Passwords don't match"

Demo

We now add parameters to the function.

```
$ grep -q ^root: /etc/passwd && echo "root exists"
$ grep -q ^root: /etc/passwd || echo "root does not exist"
```

Simple Logic

For simple logic checks we do not need an IF statement, && can check for truth and || to check for false

```
check_user () {
  grep -q \^${1}\: /etc/passwd  && return 0
}

if check_user "$user_name" ; then
  prompt_user "The user already exists, choose a new user"
fi
```

Check User Exists

Although as yet, we are not adding the user account, we need to check that the user does not already exist. The -q option to grep only returns success or fail. If the user is there then grep will return 0 for success. We need to treat this as a negative and re-prompt for a unique name

Demo

We will now implement the user test within the script.

Iteration

Looping structures allows you to iterate through code either using FOR or WHILE constructs. Consider our passwords or user checks, we currently only check the first time and this should be a loop until correct

```
for u in $* ; do
  echo $u
done
```

FOR Loops

Great for iterating through a list

```
while [ "$user_password" != "$user_password_check" ] ; do
    prompt_user "Passwords Do Not Match!"
done
```

WHILE Loops

These are great to loop indefinitely until a condition is no longer met

Demo

Implementing WHILE loops to effect better checking of user input

Summary

Functions are used to create blocks of reusable code

Use **return** to leave a function with a given exit value

Loops iterate code blocks defined between **do** and **done** keywords

FOR loops iterate through a list of some kind

WHILE loops loop until a condition is no longer met

Next up:
Building Menus in BASH