Bash Scripting Part 1

P. TenHoopen - WMLUG

What is the shell?

The shell is a command interpretor.

Bash is the most common shell in use on Linux. It's a wordplay of “Bourne-Again shell” because it is based on another shell called Bourne.

What is a script?

A script is a shell program.

A script can consist of just a file listing commands that you could issue at the command line. For example, this is a simple script named simplescript.sh and consists of two lines:

simplescript.sh

```
cd /var/log
ls *.log
```

Running scripts

You can run scripts two ways.

If the script file is not flagged as executable:

```
sh simplescript.sh
```

If it is flagged as executable, and is in the command path:

```
simplescript.sh
```

You can flag the file as executable by using:

```
chmod +x simplescript.sh
```

Bash scripts

If you use actual bash commands (discussed next), then the file needs to be setup with the first line containing:

```
#!/bin/bash
```

This indicates that this file is a script and that it should be fed into the bash shell.

Comments

The # (pound sign) is used to indicate comments and any text following it on that line is ignored.

However, even though the `#!/bin/bash` line begins with #, it is not a comment.

Some examples:

```
# Pound signs indicate comment lines and are
# ignored by the script processor
```

Script Output

You can use the echo command to write text to the terminal window.

Enclose the text in double-quotes for best results.

To create a blank line, you can have an echo command with no text.

Some examples:

```
echo “Hello”    # this writes Hello
echo            # this line writes a blank line
```

Escaping special characters

When outputting special characters using an echo command, you must prefix the characters with a \ (back slash).

Special characters include $, ", and \.

Escaping special characters

For example:

```
echo "System variable, \$USER, is the current user."
```

This outputs:

```
System variable, $USER, is the current user.
```

Without the \, it would output:

```
System variable, ptenhoopen, is the current user.
```

Variables

Variables are used to hold information. They can contain text or numbers.

```
echo “Hello, $USER”
```

This outputs the text “`Hello, `” followed by the name of the current user, using a system variable named `USER`.

System variables

Useful system variables:

`HOME`	Current user's home directory
`HOSTNAME`	Hostname of the computer running the script
`USER`	Current user
`PWD`	Current directory

Creating variables

On the left, specify the name of your variable, follow it with an equals sign and give the value. There can be no spaces before or after the equals sign.

For example:

```
# creating custom variables

# a text/string variable
MyVar=“Some text”

# a numeric variable
aNum=17
```

Using variables

When you create a variable, you don't prefix it with the $ but when you go to use it, you need to add the $.

Using your variables in a script:

```
echo $MyVar
# prints: Some text

echo “Variable aNum = $aNum”
# prints: Variable aNum = 17
```

Script Input

You can read input from the user using the read command.

```
echo “What is your favorite color?”
read favcolor
```

The above commands ask the user what their favorite color is and stores it in the favcolor variable.

Putting it all together

```
#!/bin/bash

# Advanced Script

# lines starting with a # are comments and are ignored by the script processor

# clear the terminal window
clear

# demonstrate using system variables with echo output
echo "The system variable, \$USER, holds the current user name."
echo "So, \$USER=$USER"
echo
echo "Hello, $USER"

echo "Your current directory is $PWD"
echo

echo "The files in $PWD, are:"
ls -l $PWD

# demonstrate reading in user input
echo
echo
echo -n "What text are you looking for? "
read srchtext
echo
echo "You are searching for \"$srchtext\"."
echo

# demonstrate using the inputted text in a command
echo "Conducting search in the current directory..."
echo
grep $srchtext $PWD/*.*
echo
echo Search complete.  There were `grep $srchtext $PWD/*.* | wc -l` occurrences found.
```

References

For more information, see the Advanced Bash-Scripting Guide at

http://tldp.org/LDP/abs/html/index.html