

Linux User Group at UTA

Basic shell scripting – writing a
Wrapper for Markdown

Rohit Rawat

www.luguta.org

Redirection

- Direct the output of a command to a file
 - ls command prints the list of files in a directory
 - ls > list.txt
 - ls >> list.txt appends to the existing contents of list.txt
- Direct contents to a file to the input of a program
 - tr a-z A-Z will convert the input from lowercase to uppercase
 - tr a-z A-Z < list.txt

Markdown

- Markdown dumps HTML to the screen
- You can redirect it to a file
 - `markdown demoweb.md > index.html`
- Note that the output is missing the basic HTML structure

```
<html>
```

```
<head>
```

```
<title>Page title</title>
```

```
</head>
```

```
<body>
```

```
Body text and HTML goes here.
```

```
</body>
```

```
</html>
```

Markdown

- We can use redirection again:

```
# write header to a new index.html file
```

```
echo "<html><head><title>Home</title></head><body>" > index.html
```

```
# append markdown output
```

```
markdown demoweb.md >> index.html
```

```
# append footer
```

```
echo "</body></html>" >> index.html
```

Shell scripts – md2html.sh

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

```
echo "<html>
```

```
<head>
```

```
<title>$1</title>
```

```
</head>
```

```
<body bgcolor="pink">
```

```
$(markdown $2)
```

```
</body>
```

```
</html>
```

```
"
```

Shell scripts – md2html.sh

```
#!/bin/bash    - the program that will interpret this script
                (can be perl/python etc.)
echo "<html>    - the echo command prints the line to screen
                the double quotes allow us to span several lines
<head>
<title>$1</title> - $n refers to the n'th command line argument
</head>

<body bgcolor="pink"> - feel free to customize the page style

$(markdown $2) - $(command) executes the command and
                inserts the output in its place

</body>
</html>
"
```

Shell variables

- Assignment:

varname=value

- Do not use \$ sign on the left side
- Do not use spaces around the = sign

- Using a variable requires the \$ sign

\$varname

echo \$varname

- Example

var2=\$var1

Shell variable math

- Variables are treated as strings
- To do basic math:
 - Use let command

```
let a=5  
let b=6  
let c=a+b
```
 - Use expr command

```
c=`expr $a+$b`
```

If-then

```
if condition
then
    <stuff to do>
fi
```

Test command (short version)

```
if [ a test ]
then
    <stuff to do>
fi
```

The spaces around [and] are important!

Examples of numerical tests:

```
var1=5
var2=6
var3 = [ $var1 -eq $var2 ]
echo $var3
var3 = [ $var1 -ne $var2 ]
echo $var3
```

Other numerical tests: -gt (greater than) -lt (less than) -ge -le etc.

If-then

We can enhance our script by making sure the user enters the correct number of arguments:

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

```
if [ $# -ne 2 ]
```

```
Then
```

```
    # display usage and quit
```

```
    echo 'Invalid number of arguments.'
```

```
    Usage:
```

```
    md2html "Page Title" input.md'
```

```
    echo
```

```
    exit
```

```
fi
```

```
...
```

Fixing the `_` issue

- It is very common to use `_` in variable and file names. Markdown sees them as emphasis commands.
- Example
 - We copy `my_var` to `your_var`
becomes:
 - We copy *myvar* to *yourvar*
- Unless we replace `_` with `_`
 - We copy `my_var` to `your_var`
 - Which looks bad.

Solution

- Use only `*` for *emphasis* and replace all `_` with `_`
- We can do this by using the `sed` command.

sed

- sed – stream editor
- Many features, but mostly used for substitution
- Usage:
`sed 's/pattern/replacement/' filename`
- For fixing the underscores:
`sed 's/_/_/g' input.md`
 - The `\` needs to be escaped so that it is retained
 - The `g` (global) at the end changes the whole line instead of stopping after the first change in a line

P.S.: Please type the commands instead of copy/pasting them.
The PDF file encodes the ' symbol differently causing problems.

Pipes

- Our sed command can be used either as:

```
sed '/_/\_/_/g' $2 > fixed.md
```

...

```
$(markdown fixed.md)
```

...

```
rm fixed.md
```

Pipes

- Instead of creating the temporary 'fixed.md' file
- We can connect the output of one program to the input of another program with a pipe

```
$(sed '/_/\_/_/g' $2 | markdown)
```

Updated script

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

```
if [ $# -ne 2 ]
```

```
then
```

```
    echo "Invalid number of arguments.
```

```
    Usage:
```

```
    md2html.sh \"Page Title\" input.md"
```

```
    echo
```

```
    exit
```

```
fi
```

```
echo "<html>
```

```
<head>
```

```
<title>$1</title>
```

```
</head>
```

```
<body bgcolor="pink">
```

```
$(sed 's/_/\_ /g' $2 | markdown)
```

```
</body>
```

```
</html>
```

```
"
```