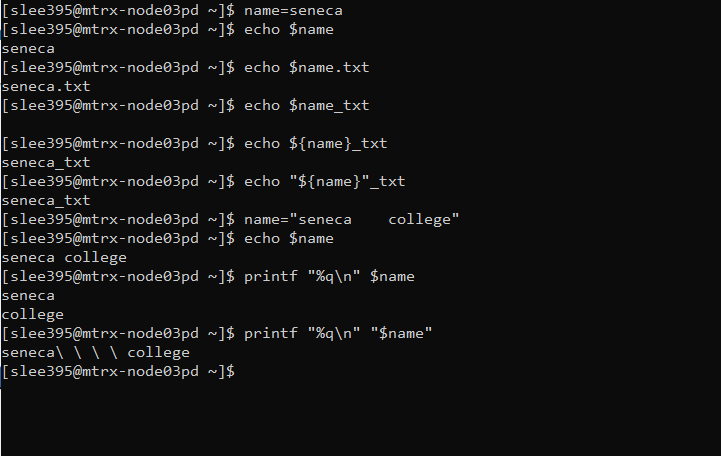
**UNX510 – Act15**

**Sanghyuk Lee(129405171)**

p.4

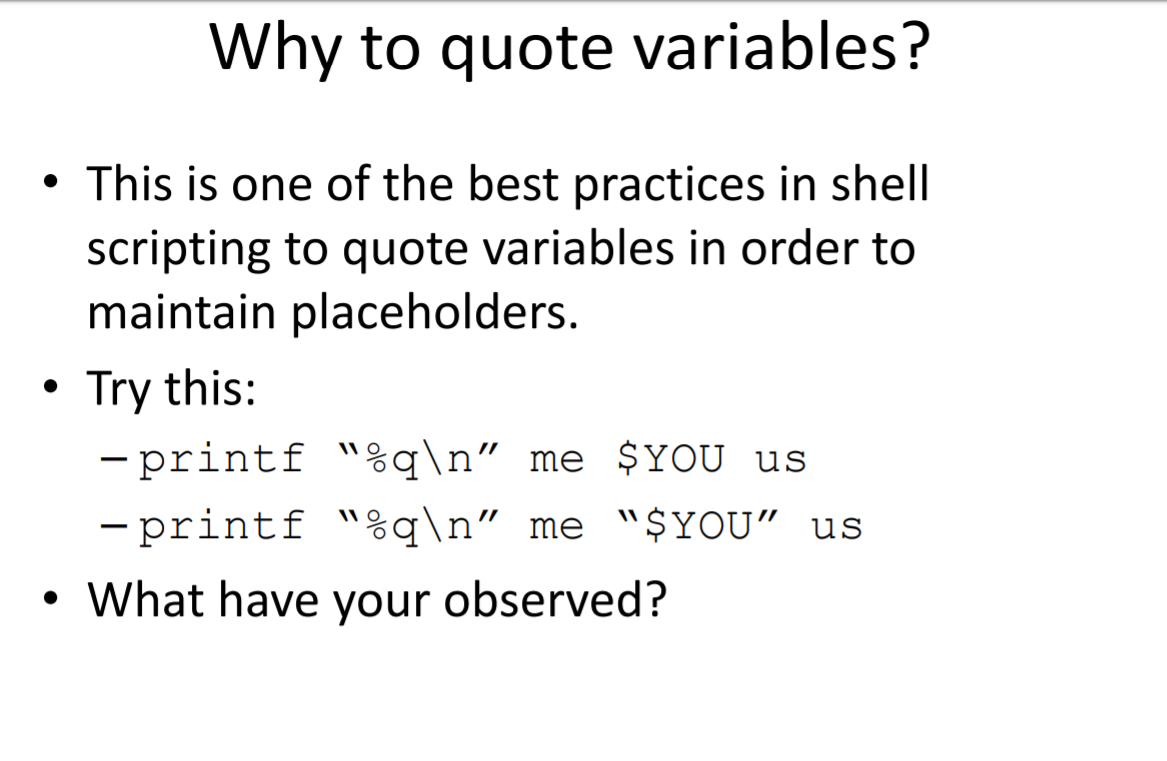


What have you observed?

What I could observed while I run this script, I found that when you do not use {} and just use $, it will retrieve the value but when you use {} it can retrieve the delineated value name so that it can get the result which is not available when you use $ only.

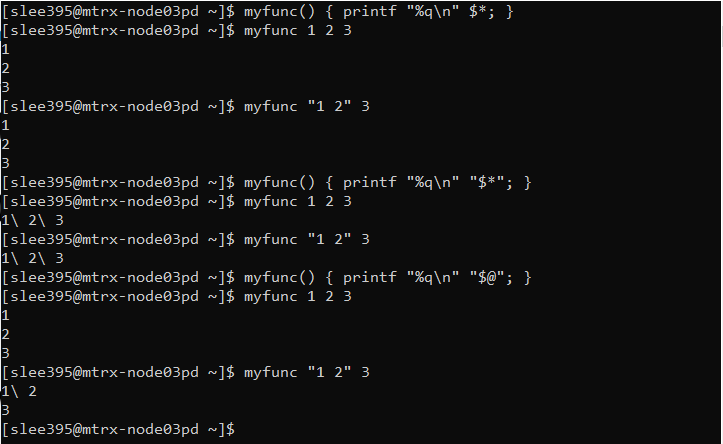
When you are trying to echo the value of a variable with spaces between two words, it will display one space no matter how many spaces are there. The next script with printf “%q\n” $name will divide the two words by spaces so that you can get two lines of result as in the picture. The last script is by using double quotation wrapping the $name, it will display one line result with the spaces by \.

p.5



Similar to the previous question, the first one will get multiple lines when you do have spaces between words, whereas the second one will get one line result with spaces in \.

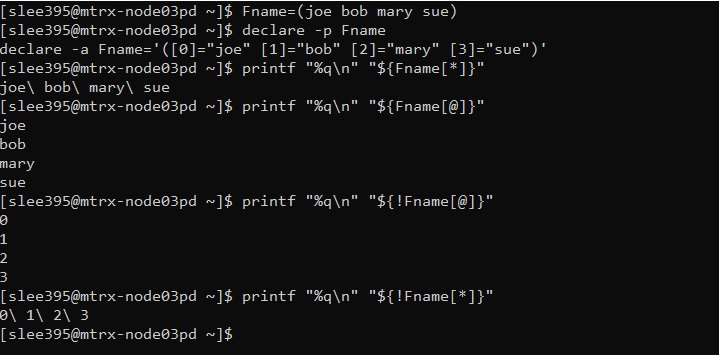
p.12



What have you observed?

What I observed in this script is the first myfunc function would consider and divide and make a new line no matter how it has a double quotation or not. The second myfunc with wrapped $\* will take the arguments and results in one line by displaying all the arguments divided by \ which stands for space. The third one take all the arguments but in this time, it consider the argument wrapped in double quotation as one so it divides 12\3 and divide 1 and 2 in \.

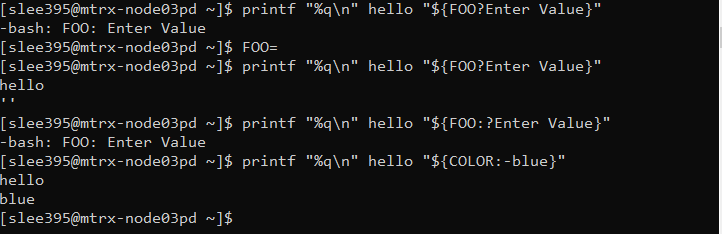
p.15



What you have observed?

After you declare the array, the first printf command will display all the elements with space(\), the second one is displaying four lines which does have each element which is divided by space. The third one, as it has used ! which stands for the index, it displays the index in a way it has displayed the element. The last one does the same as the first command but displaying the index like previous one.

p.16



What you have observed?

The first one is trying to display the value but as it does not have any assigned value to the variable FOO, it displays an error message. The second one display both hello and ‘’ because you declared a variable FOO with empty value. So, although it does not have any actual value it prints out. The third one, :? stands for sending error message to stderr if “var” is null or unset. So as the value of the variable FOO is null, it displays an error message. The last one is showing hello and blue because :- stands for if the value of the FOO is null, it displays the default value which is hardcoded by user. In this case, blue is the hardcoded default value so it prints out hello and blue and divided by space and display in different lines.