Bash XOR A String

Finnbarr P. Murphy

(fpm@fpmurphy.com)

Here is an example of how to XOR a string variable using the Bash shell. Each character in the *plaintext* string is XOR'ed with decimal 90.

```
#!/usr/local/bin/bash
plaintext="abcdefg"
echo "Plaintext: $plaintext"
cyphertext=""
for ((i=0; i < ${#plaintext}; i++ ))
do
   ord=$(printf "%d" "'${plaintext:$i:1}")
   tmp=$(printf \$(printf '%03o' $((ord ^ 90)) ))
   ciphertext="${ciphertext}${tmp}"
done
echo "Ciphertext: $ciphertext"
# now XOR again and we should get the original string back
plaintext=""
for ((i=0; i < ${#ciphertext}; i++ ))
do
   ord=$(printf "%d" "'${ciphertext:$i:1}")
   tmp=$(printf \$(printf '%03o' $((ord ^ 90)) ))
   plaintext="${plaintext}${tmp}"
done
echo "Plaintext: $plaintext"
```

The key lines are the *ord=* and the *tmp=* lines. The first line converts the character returned by *${string:$index:1}* into an ordinal number matching the ASCII chart. Note the use of the *"'..."* syntax. You need that leading single quote! The second line XORs the value of the *ord* variable with 90 and then converts the result back into a character.

This also works for *ksh93* and should work for the *zsh* shell but I have not tested it.

