Shell Script: Print Hexadecimal Representation of String

Finnbarr P. Murphy
(fpm@fpmurphy.com)

The IEEE 1003.1 (POSIX.1) standard contains the following normative text in the extended description for *printf()*:

> The argument operands shall be treated as strings if the corresponding conversion specifier is b , c , or s , and shall be evaluated as if by the strtod() function if the corresponding conversion specifier is a , A , e , E , f , F , g , or G . Otherwise, they shall be evaluated as unsuffixed C integer constants, as described by the ISO C standard, with the following extensions:
>
> * **If the leading character is a single-quote or double-quote, the value shall be the numeric value in the underlying codeset of the character following the single-quote or double-quote.**
> ....

The embolding is mine.

I never took much notice of this sentence until recently when I happened to come across this extension being used to print out the numeric value of a character in the underlying codeset.

A simple example of what this means might clarify things for you:

```
$ printf "%X\n" "A"
0
$ printf "%X\n" A
0
$ printf "%X\n" \"A
41
$ printf "%X\n" \'A
41
$
```

Note that both the Korn shell and the Bash shell also support trailing single and double quotes.

```
$ printf "%X\n" \"A\"
41
$ printf "%X\n" \'A\'
41
```

The Korn shell also supports the following format

```
$ printf "%X\n" L\'A
41
```


One use for this functionality is to print out the ASCII codeset values in hexadecimal format for the characters in a string as shown in the following example:

```
#!/bin/ksh93
str="Hi there Vi Castillo!"
for (( i=0; i < ${#str}; i++ ))
do
   c=${str:$i:1}
   if [[ $c == ' ' ]]
   then
      printf "[%s] 0x%X\n" " " \'\ \'
   else
      printf "[%s] 0x%X\n" "$c" \'$c\'
   fi
done
```

Note that spaces must be escaped with a slash otherwise *ksh93* emits a warning about an invalid string constant. Bash is silent but outputs two zeros.

Here is the output from this example:

```
[H] 0x48
[i] 0x69
[ ] 0x20
[t] 0x74
[h] 0x68
[e] 0x65
[r] 0x72
[e] 0x65
[ ] 0x20
[V] 0x56
[i] 0x69
[ ] 0x20
[C] 0x43
[a] 0x61
[s] 0x73
[t] 0x74
[i] 0x69
[l] 0x6C
[l] 0x6C
[o] 0x6F
[!] 0x21
```

This example also works in the Bash shell.

Please let me know if you find any other interesting applications of this functionality.

