UPDATED | COMMENTS

Performing Math calculation in Bash

I use math in bash scripts a lot, from simple crontab reports to Nagios monitoring plugins… Here is few small examples on how to do some maths in Bash with integers or float.

Integer Math

First way to do math with integer (and only integer) is to use the command "*expr — evaluate expression*".

```
Mac-n-Cheese:~ nicolas$ expr 1 + 1
2
Mac-n-Cheese:~ nicolas$ myvar=$(expr 1 + 1)
Mac-n-Cheese:~ nicolas$ echo $myvar
2
Mac-n-Cheese:~ nicolas$ expr $myvar + 1
3
Mac-n-Cheese:~ nicolas$ expr $myvar / 3
1
Mac-n-Cheese:~ nicolas$ expr $myvar \* 3
9
```

When doing a "multiply by" make sure to backslash the "asterisk" as it's a wildcard in Bash used for expansion.

Another alternative to *expr*, is to use the bash builtin command *let*.

```
Mac-n-Cheese:~ nicolas$ echo $myvar
6
Mac-n-Cheese:~ nicolas$ let myvar+=1
Mac-n-Cheese:~ nicolas$ echo $myvar
7
Mac-n-Cheese:~ nicolas$ let myvar+1
Mac-n-Cheese:~ nicolas$ echo $myvar
7
Mac-n-Cheese:~ nicolas$ let myvar2=myvar+1
Mac-n-Cheese:~ nicolas$ echo $myvar2
8
```

Also, you can simply use the parentheses or square brackets :

```
Mac-n-Cheese:~ nicolas$ echo $myvar
3
Mac-n-Cheese:~ nicolas$ echo $((myvar+2))
5
Mac-n-Cheese:~ nicolas$ echo $[myvar+2]
5
Mac-n-Cheese:~ nicolas$ myvar=$((myvar+3))
```

This allow you to use C-style programming :

```
Mac-n-Cheese:~ nicolas$ echo $myvar
3
Mac-n-Cheese:~ nicolas$ echo $((myvar++))
3
Mac-n-Cheese:~ nicolas$ echo $myvar
4
Mac-n-Cheese:~ nicolas$ echo $((++myvar))
5
Mac-n-Cheese:~ nicolas$ echo $myvar
5
```

Floating point arithmetic

You can't do floating point arithmetic natively in bash, you will have to use a command line tool, the most common one being "*bc (http://www.gnu.org/software/bc/) - An arbitrary precision calculator language*".

```
Mac-n-Cheese:~ nicolas$ bc
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
3*5.2+7/8
15.6
15.6+299.33*2.3/7.4
108.6
```

Of course you can use the STDIN to send your formula to "*bc (http://www.gnu.org/software/bc/)*" then get the output on STDOUT.

```
Mac-n-Cheese:~ nicolas$ echo "3.4+7/8-(5.94*3.14)" | bc
-15.25
```

or by using the here-doc (http://tldp.org/LDP/abs/html/here-docs.html) notation: `bash Mac-n-Cheese:~ nicolas$ bc <<< "3.4+7/8-(5.94*3.14)" -15.25`

I encourage you too take a look at the man pages to get more detail on how it works (*man bc (http://www.gnu.org/software/bc/manual/bc.html)*).

> There are four special variables, scale, ibase, obase, and last. scale defines how some operations use digits after the decimal point. The default value of scale is 0. ibase and obase define the conver-
> sion base for input and output numbers. The default for both input and output is base 10. last (an extension) is a variable that has the value of the last printed number.

The "scale" variable is really important for the precision of your results, especially when using integers only (Note: you can also use "bc -l" to use mathlib and see the result at max scale) .

```
Mac-n-Cheese:~ nicolas$ echo "2/3" | bc
0
Mac-n-Cheese:~ nicolas$ echo "scale=2; 2/3" | bc
.66
Mac-n-Cheese:~ nicolas$ echo "(2/3)+(7/8)" | bc
0
Mac-n-Cheese:~ nicolas$ echo "scale=2;(2/3)+(7/8)" | bc
1.53
Mac-n-Cheese:~ nicolas$ echo "scale=4;(2/3)+(7/8)" | bc
1.5416
Mac-n-Cheese:~ nicolas$ echo "scale=6;(2/3)+(7/8)" | bc
1.541666
Mac-n-Cheese:~ nicolas$ echo "(2/3)+(7/8)" | bc -l
1.54166666666666666666
```

Another way to do floating point arithmetic is to use *AWK (http://www.gnu.org/software/gawk/manual/gawk.html)*:

```
Mac-n-Cheese:~ nicolas$ awk "BEGIN {print 100/3}"
33.3333
```

You can use printf to adjust the precision of the results:

```
Mac-n-Cheese:~ nicolas$ awk "BEGIN {printf \"%.2f\n\", 100/3}"
33.33
```

When using negative values, make sure to leave a white space between signs.

```
Mac-n-Cheese:~ nicolas$ awk "BEGIN {print -8.4--8}"
awk: cmd. line:1: BEGIN {print -8.4--8}
awk: cmd. line:1:                    ^ syntax error
```

```
Mac-n-Cheese:~ nicolas$ awk "BEGIN {print -8.4 - -8}"
-0.4
```

If you want to go further, you can check my post Advanced Math Calculation in Bash using GNU bc (/2015/01/01/advanced-math-calculation-in-bash-using-gnu-bc/).

Posted by Nicolas Brousse Updated ShellTips

Tweet G+ Like Share 28 people like this. Be the first of your friends.

« Graphing Java JMX Object values with Ganglia and Python using JPype (/2010/05/31/graphing-java-jmx-object-values-with-ganglia-and-python-using-jpype/)

Linux sysctl configuration and tuning script » (/2010/09/13/linux-sysctl-configuration-and-tuning-script/)

Comments

26 Comments **Shell Tips!** 1 **Login**