Arithmetic expansion

```
$(( <EXPRESSION> ))

$[ <EXPRESSION> ]
```

The arithmetic expression `<EXPRESSION>` is evaluated and expands to the result. The output of the arithmetic expansion is guaranteed to be one word and a digit in Bash.

Please **do not use the second form `$[ … ]`** ! It's deprecated. The preferred and standardized form is `$(( ... ))` !

Example

```
function printSum {
    typeset -A args
    typeset name
    for name in first second; do
        [[ -t 0 ]] && printf 'Enter %s positive integer: ' "$name" >&
2
        read -r ${BASH_VERSION+-e} "args[$name]"
        [[ ${args[$name]} == +([[:digit:]]) ]] || return 1 # Validati
on is extremely important whenever user input is used in arithmetic.
    done
    printf 'The sum is %d.' $((${args[first]} + ${args[second]}))
}
```

Note that in Bash you don't need the arithmetic expansion to check for the boolean value of an arithmetic expression. This can be done using the arithmetic evaluation compound command:

```
printf %s 'Enter a number: ' >&2
read -r number
if ((number == 1234)); then
    echo 'Good guess'
else
    echo 'Haha... :-P'
fi
```

Variables used inside the arithmetic expansion, as in all arithmetic contexts, can be used with or without variable expansion:

```
x=1

echo $((x))       # Good.
echo $(($x))      # Ok. Avoid expansions within arithmetic. Use variables directly.
echo $(("$x"))    # Error. There is no quote-removal in arithmetic contexts. It expands to $(("1")), which is an invalid arithmetic expression.
echo $((x[0]))    # Good.
echo $((${x[0]})) # Ok. Nested expansion again.
echo $((${x[$((${x[!$x]}-$x))]})) # Same as above but more ridiculous.
echo $(($x[0]))   # Error. This expands to $((1[0])), an invalid expression.
```

Bugs and Portability considerations

- The original Bourne shell doesn't have arithmetic expansions. You have to use something like `expr(1)` within backticks instead. Since `expr` is horrible (as are backticks), and arithmetic expansion is required by POSIX, you should not worry about this, and preferably fix any code you find that's still using `expr`.

See also

- arithmetic expressions
- arithmetic evaluation compound command
- Introduction to expansion and substitution
- POSIX definition (http://pubs.opengroup.org/onlinepubs/9699919799/utilities/V3_chap02.html#tag_18_

Discussion

Jochen, 2012/07/17 07:59 ()

The line

read -p "Enter a number: "

in the second example should read

read -p "Enter a number: " number

Jan Schampera, 2012/08/12 07:05 ()

Fixed, thx

Yclept Nemo, 2012/11/27 01:51 (), 2012/11/30 19:40 ()

Should mention that `$(())` form doesn't accept quoted variable names.