Арифметическое расширение

```
$(( <ВЫРАЖЕНИЕ> ))

$[ <ВЫРАЖЕНИЕ> ]
```

Вычисляется арифметическое выражение `<EXPRESSION>` и расширяется до результата. Результат арифметического расширения гарантированно будет состоять из одного слова и цифры в Bash.

Пожалуйста, **не используйте вторую форму `$[ … ]`** ! Оно устарело. Предпочтительная и стандартизированная форма `$(( ... ))`!

Пример

```
функция printSum {
     typeset-имя набора аргументов

для имени в первую секунду; do
        [[ -t  0 ]] && printf' Enter %s положительное целое число: '
"$name" >&2
         read -r  $ {BASH_VERSION+-e} "аргументы [$name]"
        [[ ${args[$name]} == +([[:digit:]]]) ]] || return  1 # Провер
ка чрезвычайно важна всякий раз, когда пользовательский ввод использу
ется в арифметике.
    выполнено
     printf ' Сумма равна %d.' $((${args[first]} + ${аргументы [вторы
e]}))
}
```

Обратите внимание, что в Bash вам не нужно арифметическое расширение для проверки логического значения арифметического выражения. Это можно сделать с помощью составной команды арифметической оценки:

```
% printf s 'Введите число: ' >&2
 считайте -r число
, если ((число == 1234)); затем
     echo' Good угадайте, или

     echo' Xaxa... :-P'
fi
```

Переменные, используемые внутри арифметического расширения, как и во всех арифметических контекстах, могут использоваться с расширением переменных или без него:

```
x = 1

 echo $((x))       # Хорошо.
echo $(($ x))      # Хорошо. Избегайте расширений в арифметике. Используйте переменные напрямую.
Ошибка echo $(("$ x"))    #. В арифметических контекстах нет удаления кавычек. Оно расширяется до $(("1")), что является недопустимым арифметическим выражением.
echo $((x[0]))    # Хорошо.
echo $(($ {x[0]})) # Хорошо. Снова вложенное расширение.
echo $ (($ {x [$(($ {x [! $ x]}- $ x))]}))  # То же, что и выше, но более нелепо.
echo $(($x[0]))   # Ошибка. Это расширяется до $((1[0])), недопустимое выражение.
```

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.