

Команда Linux bc с примерами

12 января 2023 г.

БАШ КОМАНДЫ ЛИНУКС

[Главная](#) » [DevOps и разработка](#) » Команда Linux bc с примерами

Введение

Команда Linux **bc** (сокращение от **b**asic **c**alculator) – это утилита командной строки, которая действует как научный калькулятор. Команда интерпретирует язык bc и выполняет арифметические операции произвольной точности с интерактивным выполнением операторов.

Используйте **bc** команду как интерактивную математическую оболочку, принимающую стандартный ввод, или как математический язык сценариев.

В этом уроке вы научитесь использовать **bc** команду в Linux и увидите практические примеры ее использования.



Предпосылки

- Система под управлением [Linux](#) .
- Доступ к терминалу (**Ctrl + Alt + T**).

Синтаксис команды Linux bc

Синтаксис команды **bc** похож на язык программирования C. Общий синтаксис команды:



- Доступные варианты **[options]** описаны в разделе ниже.
- Укажите **a**, **[file]** чтобы прочитать его содержимое и выполнить операторы по мере их чтения. Если не указать файл, откроется интерактивный режим и будет ожидать ввода данных от пользователя.

Параметры команды Linux bc

Используйте **bc** параметры команды, чтобы настроить режим работы и указать способ обработки входных данных.


Доступны следующие варианты:

Вариант	Длинная форма	Описание
-c	/	Опция -c скомпилирует [file] параметр без вызова dc команды. bc Команда является препроцессором для dc , вызывая его автоматически, если -c опция не указана. Это приводит -c к отправке вывода на стандартный вывод.
-h	--help	Выводит на экран использование команды и завершает работу.
-i	--interactive	Принудительно включает интерактивный режим.
-l	--mathlib	Определяет библиотеку математических функций и устанавливает масштабную переменную на 20. Значение масштаба по умолчанию — 0.
-w	--warn	Выдает предупреждения для расширений POSIX bc.
-s	--standard	Обрабатывать в точности язык POSIX bc.
-q	--quiet	Запускает команду без вывода приветствия GNU bc .
-v	--version	Выводит номер версии программы, сведения об авторских правах и завершает работу.

Как работает команда Linux bc?

Команда **bc** работает, обрабатывая код из всех файлов, указанных в командной строке в порядке перечисления файлов. После обработки файлов **bc** начинает чтение из стандартного ввода, выполняя весь код по мере его чтения. Команда не читает из стандартного ввода, если файл содержит команду остановки процессора.

При указании входных файлов убедитесь, что это текстовые файлы, содержащие последовательность команд, операторов или определений функций, которые команда может прочитать и выполнить **bc**. Команда позволяет пользователям определять математическую библиотеку перед обработкой файлов с помощью **-l** параметра.



Важно: Хотя **bc** команда работает с произвольной точностью, по умолчанию она имеет ноль цифр после десятичной точки, если только не **-l** указан флаг или **scale** переменная не задана вручную. **-l** флаг устанавливает **scale** переменную на 20 цифр после десятичной точки.

При работе с входными значениями **bc** позволяет пользователям указывать входную и выходную базу для математических операций в десятичных, восьмеричных или шестнадцатеричных значениях. Пользователи также могут работать с переменными или

- Арифметические операторы.
- Операции [увеличения и уменьшения](#) .
- Операторы присваивания.
- Операции сравнения и отношения.
- Логические и [булевы](#) операции.
- Математические функции.
- Условные высказывания.
- Итеративные утверждения.
- Комментарии в стиле C, начинающиеся `/*` и заканчивающиеся на `*/`.

Некоторые основные и специальные выражения, которые **bc** поддерживает:

Функция	Описание
<code>length (expression)</code>	Значение функции длины – это количество значащих цифр в выражении.
<code>read ()</code>	Функция read считывает число из стандартного ввода, независимо от того, где находится функция. Используйте с осторожностью, так как это может вызвать проблемы из-за смешивания данных и программы в стандартном вводе.
<code>scale (expression)</code>	Количество цифр после десятичной точки в выражении представляет собой значение функции scale .
<code>sqrt (expression)</code>	Квадратный корень выражения является значением функции sqrt .
<code>++ var</code>	Увеличивает переменную на единицу, и новое значение является результатом выражения.
<code>var ++</code>	Результатом выражения является значение переменной, а затем переменная увеличивается на единицу.
<code>-- var</code>	Уменьшает переменную на единицу, а затем сохраняет новое значение переменной.
<code>var --</code>	Результатом выражения является значение переменной, а затем переменная уменьшается на единицу.
<code>(expr)</code>	Скобки изменяют стандартный приоритет и вызывают принудительное вычисление выражения.
<code>var = expr</code>	Переменной var присваивается значение выражения.

Полный список поддерживаемых выражений и функций см. в [руководстве по команде bc](#) .

Команда Linux bc как интерактивная математическая оболочка

Войдите в интерактивную математическую оболочку, выполнив **bc** команду без указания файлов:

bc

```
bosko@pnap:~$ bc
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software
Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
```

quit

Пример 1: использование bc в качестве калькулятора

Команда **bc** может работать как калькулятор на основе CLI для простых математических операций, таких как сложение, вычитание, деление и умножение. Она также поддерживает различные продвинутое математические функции, такие как синус, косинус, тангенс и натуральные логарифмы.

Калькулятор превосходит Bash, поскольку Bash не может выполнять некоторые сложные арифметические операции, такие как сравнение чисел с плавающей точкой.

Например, запустите команду и протестируйте некоторые математические операции:

```
7+3
3*5
```

```
7+3
10
3*5
15
```

Результат отображается под каждой операцией по мере ввода данных пользователем.

Пример 2: Работа с десятичными дробями

При работе с десятичными числами программа по умолчанию использует ноль цифр после запятой. Однако вы можете настроить поведение, если определите **scale** значение переменной или укажете **-l** опцию при запуске команды.

Например, попробуйте разделить восемь на три, не задавая **scale** переменную. Затем попробуйте снова, задав ей два знака после запятой:

```
8/3
2
scale = 2 /* setting the scale value to 2 keeps 2 decimal places */
8/3
2.66
```

Во-первых, программа не показывает никаких цифр после десятичной точки, что является поведением по умолчанию. После установки **scale** значения переменной в 2 программа сохраняет два десятичных знака. Комментарий в стиле C в примере выше начинается с **/*** и заканчивается ***/** и не нарушает работу программы.

Установите **scale** значение, равное количеству знаков после запятой, которое вы хотите иметь в выводе. Запуск **bc** команды с **-l** опцией сохраняет 20 знаков после запятой:

```
bosko@pnap:~$ bc -l
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software
Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
8/3
2.66666666666666666666
```

Пример 3: Входной сигнал трубы от эха

Команда **bc** позволяет пользователям выполнять математические операции без входа в интерактивную оболочку. Один из способов сделать это — перенаправить вывод команды **echo** в **bc**.

Например:

```
echo "6+17" | bc
```

```
bosko@pnap:~$ echo "6+17" | bc
23
bosko@pnap:~$
```

```
bosko@pnap:~$ echo 'scale=5;8/3' | bc
2.66666
bosko@pnap:~$
```

Результат выводится с пятью десятичными знаками.

Пример 4: Преобразование десятичного числа в шестнадцатеричное

Используйте **bc** для преобразования значений из одной системы счисления в другую. Команда достигает этого с помощью двух специальных переменных - **ibase**(входная база) и **obase**(выходная база). Переменные определяют базу преобразования для входных и выходных чисел. Допустимые **obase**значения находятся в диапазоне от 2 до 999, в то время как допустимые **ibase** значения находятся в диапазоне от 2 до 16.

Например, следующая команда преобразует число 255 из системы счисления с основанием 10 в систему счисления с основанием 16:

```
echo 'obase=16;255' | bc
```

```
bosko@pnap:~$ echo 'obase=16;255' | bc
FF
bosko@pnap:~$
```

Пример 5: Преобразование десятичного числа в двоичное

Использование - **ibase** и **obase**, **bc** позволяет пользователям преобразовывать десятичные числа в двоичные. Например, следующая команда преобразует число 12 из десятичной системы счисления в двоичную:

```
echo 'obase=2;12' | bc
```

```
bosko@pnap:~$ echo 'obase=2;12' | bc
1100
bosko@pnap:~$
```

Команда позволяет пользователям преобразовывать значения между другими поддерживаемыми системами счисления – десятичной, шестнадцатеричной, двоичной и восьмеричной.



Примечание: При преобразовании из двоичных в десятичные значения обязательно задайте выходное **obase** значение с использованием шестнадцатеричных значений. Например, 10 в шестнадцатеричном формате – это 16.

Команда Linux bc как язык математических сценариев

bc помогает преодолеть ограничения языков сценариев оболочки, которые ограничены целочисленной арифметикой. Таким образом, команда часто встраивается в существующие сценарии оболочки с помощью конвейера или [документа here](#).

Приведенные ниже примеры демонстрируют некоторые варианты использования **bc** в сценариях оболочки.

Пример 1: объявление переменных

Используйте переменные оболочки для **bc** сохранения значения в переменной, что полезно при написании сценариев оболочки.

Например:

```
VAR=10 ; echo "$VAR^2" | bc
```

```
bosko@pnap:~$ VAR=10 ; echo "$VAR^2" | bc
100
bosko@pnap:~$
```

Использование **bcwith files** позволяет пользователям повторять сложные вычисления несколько раз. Чтобы предоставить входные данные из файла или нескольких файлов, укажите путь к файлу при запуске команды **bc**. Файл должен быть текстовым файлом, читаемым **bc**. Поддерживается несколько файлов.

```
bosko@pnep:~$ cat calculation.txt
5+7
8*13
6-5
10/3
```

```
bc -l calculation.txt
```

```
bosko@npap:~$ bc -l calculation.txt  
bc 1.07.1  
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software  
Foundation, Inc.  
This is free software with ABSOLUTELY NO WARRANTY.  
For details type `warranty'.  
12  
104  
1  
3.333333333333333333333333
```

```
cat calculation.txt | bc -l
```

```
scale=2
print "\nConvert Fahrenheit degrees to Celsius\n\n"
print "Enter temperature in Fahrenheit: " ; fah = read()
print "\n"
print "The equivalent Temperature in Celsius is: "
(fah - 32.0) * 5.0 / 9.0
quit
```

```
bc -q [filename]
```

```
The equivalent temperature in Celsius is: 15.55
bosko@pnap:~$
```

Команда сначала запрашивает ввод в градусах Фаренгейта, а затем преобразует их в градусы Цельсия. Опция **-q** запускает команду молча, не отображая версию программы и информацию об авторских правах.

Пример 4: использование операторов if

Язык bc поддерживает многочисленные операторы управления, включая оператор **if/else**. Отличие от общих операторов **if/else** заключается в том, что **else** предложение заключено в фигурные скобки, а **then** предложение – нет. Однако оба заканчиваются точкой с запятой.

Чтобы создать скрипт для факториальной функции (**n!**), выполните следующие действия:

1. **Создайте скрипт** с помощью текстового редактора и вставьте следующие строки:

```
define f (x) {
    if (x <= 1) return (1);
    return (f(x-1) * x);
}
```

2. Сохраните файл.

3. Выполните скрипт, используя следующий синтаксис:

```
bc -q [filename]
```

Проверьте работоспособность скрипта, предоставив пример:

```
bosko@pnap:~$ bc -q fact.bc
f(5)
120
```

Команда вычисляет факториал числа 5 как 120 ($5*4*3*2*1=120$), что означает, что скрипт работает.

Заключение

В этой статье объясняется, как **bc** работает команда, и приводятся несколько примеров использования. **bc** Команда предлагает множество возможностей и облегчает написание скриптов автоматизации в Bash.

Далее прочитайте наше подробное руководство, чтобы узнать больше о [математических операциях Bash](#) и различных командах, которые можно использовать для арифметических операций.

Была ли эта статья полезной?

Да

Нет



Боско Мариан

Работая в качестве педагога и автора контента, в сочетании с его давней страстью ко всему, что связано с высокими технологиями, Боско стремится упростить сложные концепции и сделать их удобными для пользователя. Это привело его к техническому письму в PhoenixNAP, где он продолжает свою миссию распространения знаний.

Далее вам следует прочитать

Bash printf – как распечатать переменную в Bash

24 февраля 2022 г.

Команда printf выводит форматированный текст в терминале. В этом руководстве показано, как использовать команду printf для печати и форматирования переменного вывода...

ЧИТАТЬ ДАЛЕЕ

Системный администратор Оператор объявления Bash: синтаксис и примеры

23 декабря 2021 г.

Хотя встроенный оператор declare не обязательно использовать для явного объявления переменной в Bash, эта команда часто используется для более сложных задач с переменными...

ЧИТАТЬ ДАЛЕЕ

DevOps и , системный администратор разработки Как использовать команду Bash let {с примерами}

20 января 2022 г.

Команда Bash let – это встроенная утилита, используемая для оценки арифметических

новые экземпляры BMC : получите доступ по требованию к выделенным серверам с двумя графическими процессорами

Intel® MAX 1100!

Узнайте

ЧИТАТЬ ДАЛЕЕ

DevOps и , системный администратор разработки

Заявление по делу Bash

15 декабря 2021 г.

Оператор case проверяет входное значение, пока не найдет соответствующий шаблон и не выполнит команду, связанную с этим значением. Изучите основы оператора case в bash и как использовать его в скриптах оболочки.

ЧИТАТЬ ДАЛЕЕ



💬 Онлайн чат

✅ Получить предложение

🛠️ Поддержка | 1-855-330-1509

🛒 Продажи | 1-877-588-5918

Центр конфиденциальности Не продавайте и не передавайте мою личную информацию

Связаться с нами

- Юридический
- политика конфиденциальности
- Условия эксплуатации
- DMCA
- GDPR
- Карта сайта

©2024 Copyright phoenixNAP | Global IT Services. Все права защищены.