Команда Bash bc в Linux [с примерами]

Последнее обновление: 6 декабря 2023 г. | Linuxopsys

Содержание

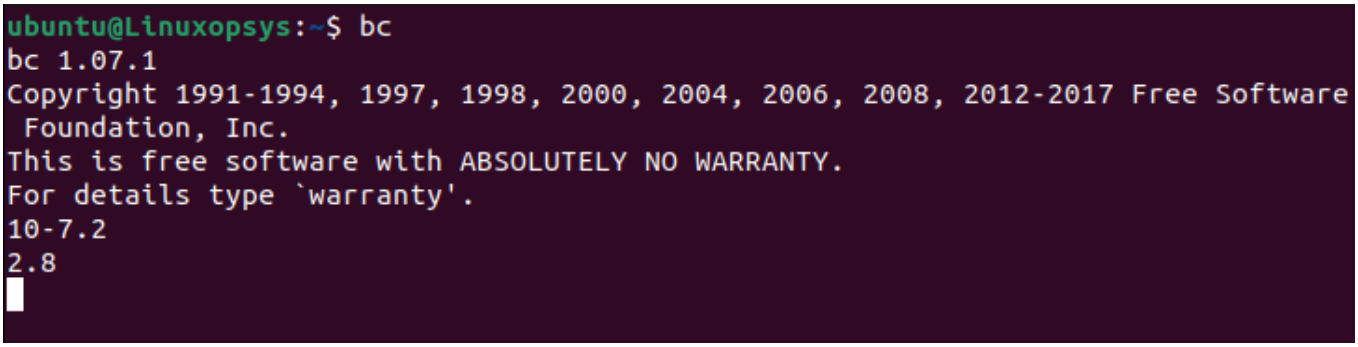
Команда bc (базовый калькулятор) используется для математических вычислений, включая основные арифметические операции и высокоточные вычисления. Она может обрабатывать **как целочисленную, так и арифметику с плавающей точкой** и подходит для интерактивного использования в терминале, а также для выполнения вычислений **в скриптах** .

Кроме того, bc поддерживает переменные, операторы присваивания, математические функции, управляющие структуры и т. д. По сравнению с expr , bc — более многофункциональный и точный калькулятор.

Базовое использование

Чтобы запустить калькулятор bc, просто введите bc в терминале:

```
bc
```



Вы войдете в интерактивный режим, где вы можете выполнять вычисления построчно. Чтобы выйти, нажмите **Ctrl + D** или введите **quit, а затем нажмите ENTER .**

В большинстве дистрибутивов Linux пакет bc установлен по умолчанию.

Если вы получили сообщение об ошибке «bc: команда не найдена», вам необходимо продолжить установку пакета bc на сервере.

```
###Debian/Ubuntu
sudo apt install bc
###RHEL/CentOS Поток

sudo yum install bc
###Fedora

sudo dnf install bc
```

Выполнение элементарных арифметических действий

Вы можете выполнять основные арифметические операции с bc, используя следующие символы:

- +для дополнения.
- -для вычитания.
- *для умножения.
- /для разделения.
- %для модуля.
- ^для экспоненты.

Пример:

```
$ bc
3 + 5
8
20 * 4
80
10 / 2
5
15 % 7
1
```

Вы также можете выполнять **вычисления, не вводя интерактивную подсказку bc,** используя подстановку команд и команду echo. Этот метод особенно полезен, когда вы хотите вычислить определенное выражение в скрипте или как часть конвейера команд.

Пример:

```
$ echo "10.5+5" | bc
5.5
```

Точность и масштаб

По умолчанию bc использует масштаб 0, что означает, что он выполняет целочисленную арифметику. Вы можете настроить масштаб, чтобы указать желаемый уровень точности для вычислений с плавающей точкой.

Поведение по умолчанию (масштаб 0 — целочисленная арифметика) :

```
$ bc
10 / 3
3
```

В приведенном выше примере результат равен 3, поскольку по умолчанию он работает со шкалой 0, фактически выполняя целочисленное деление.

Настройка масштаба (арифметика с плавающей точкой) :

```
$ bc
scale=2
10 / 3
3.33
```

Здесь мы устанавливаем масштаб на 2, указывая, что нам нужно два десятичных знака в результате. В результате bc выполняет деление с плавающей точкой, и результат равен 3,33.

Присвоение значений переменной

Вы можете присваивать значения переменным в bc с помощью `=`оператора. Переменные могут содержать числовые значения или даже выражения.

Пример:

```
$ bc
x = 5
y = 3
x + y
8
result = x * y
result
15
```

Операторы присваивания

В bc вы можете использовать операторы присваивания для присвоения значений переменным и выполнения различных операций над переменными. Давайте рассмотрим несколько примеров:

Оператор присваивания (`=`) : присваивает значение переменной.

```
$ echo "val = 10; val" | bc
10
```

Оператор сложения и присваивания (+=) : добавляет значение к переменной и присваивает результат обратно переменной.

```
$ echo "val = 5; val += 2; val" | bc
7
```

-=Оператор вычитания и присваивания ( ) : вычитает значение из переменной и присваивает результат обратно переменной.

```
$ echo "val = 10; val -= 3; val" | bc
7
```

/=Оператор деления и присваивания ( ) : делит переменную на значение и присваивает результат обратно переменной.

```
$ echo "val = 10; val /= 2; val" | bc
5
```

^=Оператор возведения в степень и присваивания ( ) : возводит переменную в степень и присваивает результат обратно переменной.

```
$ echo "val = 5; val ^= 2; val" | bc
25
```

%=Оператор модуля и присваивания ( ) : вычисляет модуль переменной и значения и присваивает результат обратно переменной.

```
$ echo "val = 10; val %= 3; val" | bc
1
```

Использование операторов инкремента (++) и декремента (--), как и в других языках программирования, но напрямую не поддерживается. Обычно для выполнения таких операций используются операторы присваивания. Давайте рассмотрим несколько примеров.

Эквивалент предварительного приращения в bc:

```
$ echo "var=6; var = var + 1; var" | bc
7
```

В этом примере мы сначала присваиваем значение переменной var, а затем увеличиваем его на 1 перед печатью результата.

Пост-инкрементный эквивалент в bc:

```
$ echo "var=5; result = var; var = var + 1; result" | bc
5
```

Здесь мы сначала присваиваем var значению result, затем увеличиваем var на 1 и, наконец, выводим result.

Предварительно декрементный эквивалент в bc:

```
$ echo "var=15; var = var - 1; var" | bc
14
```

Аналогично мы вычитаем 1 из var, чтобы выполнить операцию предварительного декремента.

Постдекрементный эквивалент в до н.э.:

```
$ echo "var=15; result = var; var = var - 1; result" | bc
15
```

В этом случае мы сначала присваиваем var значению result, затем уменьшаем var на 1 и, наконец, выводим result.

Реляционные операторы

В bc вы можете использовать реляционные операторы для сравнения двух чисел или переменных, и результатом будет либо 1 (истина), либо 0 (ложь). Вот как работают эти реляционные операторы с примерами:

<**Оператор «Меньше чем» ( )** : возвращает 1, если левый операнд меньше правого операнда, в противном случае – 0.

```
$ echo "10 < 5" | bc
0
```

<=**Оператор «Меньше или равно» ( )** : возвращает 1, если левый операнд меньше или равен правому операнду, в противном случае – 0.

```
$ echo "10 <= 10" | bc
1
```

>**Оператор «больше, чем» ( )** : возвращает 1, если левый операнд больше правого операнда, в противном случае – 0.

```
$ echo "10 > 5" | bc
1
```

>=Оператор «Больше или равно» ( ) : возвращает 1, если левый операнд больше или равен правому операнду, в противном случае – 0.

```
$ echo "5 >= 10" | bc
0
```

==Оператор равенства ( ) : возвращает 1, если левый операнд равен правому операнду, в противном случае – 0.

```
$ echo "5 == 5" | bc
1
```

!=Оператор неравенства ( ) : возвращает 1, если левый операнд не равен правому операнду, в противном случае – 0.

```
$ echo "1 != 2" | bc
1
```

Логические или булевы операторы

Вы можете использовать логические или булевы операторы для оценки логических выражений. Вот как работают эти операторы с примерами:

Логический &&оператор И ( ) : возвращает 1, если оба выражения не равны нулю, в противном случае – 0.

```
$ echo "26 && 8" | bc
1
```

В этом примере и 26, и 8 не равны нулю, поэтому результат равен 1.

Логический ||оператор ИЛИ () : возвращает 1, если хотя бы одно из выражений не равно нулю, в противном случае – 0.

```
$ echo "0 || 0" | bc
0
```

В этом примере ни один из них не 0является 0ненулевым, поэтому результат равен 0.

Логический !оператор НЕ () : возвращает 1, если выражение равно нулю, в противном случае – 0.

```
$ echo "! 0" | bc
1
```

Здесь 0равно нулю, поэтому логический оператор НЕ !возвращает 1.

Математические функции

Математическая библиотека предварительно загружена, а масштаб по умолчанию настроен, если bc запущен с **опцией -l** . Масштаб, установленный во время вызова, используется математическими функциями для вычисления их результатов.

Вот некоторые из наиболее часто используемых математических функций:

- s(x): Вычисляет синус x, выраженный в радианах.
- c(x): Вычисляет косинус x, выраженный в радианах.
- l(x): Вычисляет натуральный логарифм (по основанию «e») числа x.
- e(x): Вычисляет показательную функцию, возводя «e» в степень x.
- J(n,x): Вычисляет функцию Бесселя целого порядка n от x.

Дополнительные функции:

- length(x): Возвращает общее количество значащих десятичных цифр в выражении x.
- read(): Считывает входное значение.
- scale(expression): Устанавливает масштаб (количество десятичных знаков после запятой) на основе выражения.
- sqrt(): Вычисляет квадратный корень числа.

- `ibase`и `obase`: Указывает основание числа для преобразования ввода и вывода. Основание 10 является значением по умолчанию для ввода и вывода.

Некоторые примеры:

Функции синуса и косинуса :

```
$ echo "s($pi/6)" | bc -l
0.49999999999999999999

$ echo "c($pi/4)" | bc -l
0.70710678118654752440
```

Функция длины (`length(x)`) :

```
$ echo "length($pi)" | bc -l
21
```

Функция квадратного корня (`sqrt()`) :

```
$ echo "sqrt(4)" | bc
2
```

ibase и obase (системы счисления) :

```
$ echo "ibase=2;1111" | bc -l
15

$ echo "obase=8;10" | bc -l
12

$ echo "obase=8;ibase=2;101" | bc
5
```

Установка числа Пи с помощью`a(1)` **:**

```
$ pi=`echo "h=10;4*a(1)" | bc -l`
$ echo $pi
3.14159265358979323844
```

Файловый ввод/вывод

Вы можете выполнять операции ввода и вывода файлов, считывая входные данные из файлов и записывая результаты в файлы.

Чтение входных данных из файла:

```
$ cat inputfile.txt
3 + 5
20 * 4
10 / 2
15 % 7
```

Вы можете использовать bc для чтения и обработки этих выражений из файла следующим образом:

```
$ bc < inputfile.txt
8
80
5
1
```

Вы можете использовать >оператор для перенаправления вывода в файл.

```
$ bc < inputfile.txt > outputfile.txt
```

Чтобы добавить вывод в существующий файл вместо его перезаписи

```
$ bc < inputfile.txt >> existingfile.txt
```

Структуры управления

Условные операторы

В bc вы можете использовать условные операторы для принятия решений и выполнения операторов на основе оценки условия. Синтаксис условного оператора в bc следующий:

Синтаксис:

```
if(condition) {statement} else {statement}
```

Вот пример условного оператора в bc:

```
$ echo 'a=8;b=10;if(a==b) print "a equals to b" else print "a
not equals to b" ' | bc -l
a not equals to b
```

Петли

В bc вы можете использовать операторы цикла для повторения ряда инструкций или операторов определенное количество раз или до тех пор, пока не будет выполнено определенное условие. Два распространенных типа операторов цикла в bc – это цикл for и цикл while. Вот как они работают с примерами:

Синтаксис цикла:

```
для (инициализация; условие; приращение) { операторы }
```

Пример

```
$ echo "for (i = 1; i <= 5; i++) { i; }" | bc
1
2
3
4
5
```

Синтаксис цикла while выглядит следующим образом:

```
while (condition) { statements }
```

Пример

```
$ echo "i=0;while(i<5) {i; i+=1}" | bc
0
1
2
3
4
```

Приведенная выше команда выполняет цикл while в калькуляторе bc, который увеличивает значение переменной i от 0 до 4.

Если этот ресурс вам помог, дайте нам знать о своей заботе, написав благодарственный твит в Twitter.

Напишите твит с благодарностью

Была ли эта статья вам полезна?

Комментарии

Пожалуйста, оставьте комментарий ниже, чтобы предоставить автору свои идеи, благодарность и отзывы.

Оставить ответ

Имя

Электронная почта

Веб-сайт

Опубликовать комментарий

Содержание

Похожие посты

06 июля 2022 г.

Как использовать команду ethtool с примерами

03 июля 2023 г.

Объяснение команд Linux more [с примерами]

15 авг. 2023 г.

Объяснение команды Strings в Linux [с примерами]