Bash программирование: Условный оператор if.

Category: bash & sh,bash программирование,GNU/Linux
2025-06-03

Bash. Условный оператор if.

Описание.

Условный оператор if в Bash позволяет выполнять определённые команды или действия в зависимости от истинности заданного условия. Это конструкция для контроля потока выполнения скриптов, которая позволяет принимать решения на основе значений переменных, результатов команд и других условий.

Если кратко, то условный оператор if позволяет проверить условие и от полученного результата выполнить один или другой блок кода.

Синтаксис.

Синтаксис примерно такой:

```
if [[ условие ]]; then
команды
elif [[ условие ]]; then
команды
else
команды
fi
```

Можно использовать несколько elif, или убрать elif и else если они не требуются.

Помимо двойных квадратных скобок можно применять одинарные двойные и двойные круглые. Но двойные квадратные скобки — более универсальны, и позволяют делать все тоже самое что и одинарные квадратные и двойные круглые. Поэтому легче запомнить однотипные конструкции с двойными квадратными скобками.

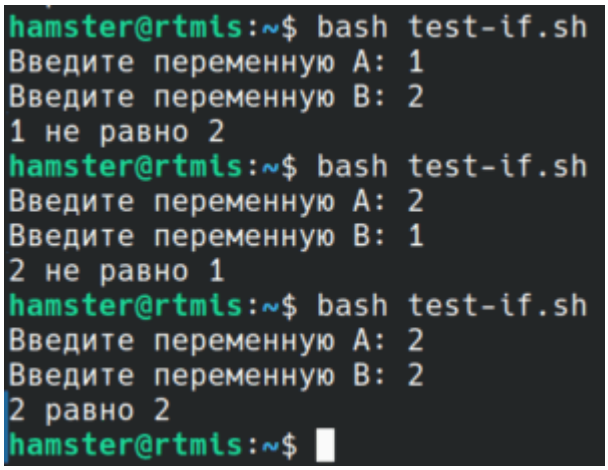
Сравнение чисел.

В таком сравнении используются следующие операторы:

- **-eq** – равно;
- **-ne** – не равно;
- **-gt** – больше;
- **-ge** – больше или равно;
- **-lt** – меньше;
- **-le** – меньше или равно.

Рассмотрим пример – скрипт на **bash**, который считывает переменные a и bc консоли, принимает решение о том, что равно (-eq) или не равно (-ne) и выводит соответствующие запросы:

```
$ cd ~

$ mcedit test-if.sh

$ cat test-if.sh


#!/bin/bash

# Запрос на ввод переменной A
read -p "Введите переменную A: " a

# Запрос на ввод переменной B
read -p "Введите переменную B: " b

# Проверка условий
if [[ "$a" -eq "$b" ]]; then
   echo "$a равно $b"
elif [[ "$a" -ne "$b" ]]; then
   echo "$a не равно $b"
else
   echo "A и B равны"
fi


$ bash ~/test-if.sh

Ответ:
```



Как это работает:

- **read -p "Введите переменную A: " a** – выводит сообщение “Введите переменную A:” и ожидает ввода пользователя. Введенное значение сохраняется в переменную a.
- **read -p "Введите переменную B: " b** – аналогично, выводит сообщение “Введите переменную B:” и сохраняет введенное значение в переменную b.
- Далее скрипт проверяет условия и выводит соответствующее сообщение в зависимости от значений переменных a и b.

Сравнение строк.

Строки сравниваются оператором <, >, =.

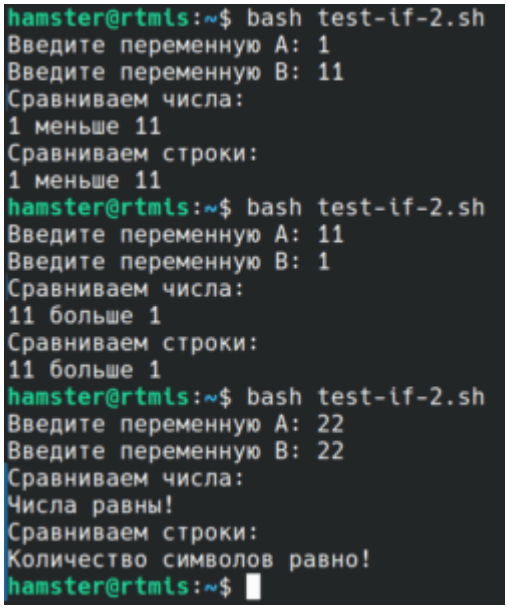
Например:

```
$ cd ~

$ mcedit test-if-2.sh

$ cat test-if-2.sh


#!/bin/bash

# Запрос на ввод переменной A
read -p "Введите переменную A: " a

# Запрос на ввод переменной B
read -p "Введите переменную B: " b

# сравниваем числа
echo "Сравниваем числа:"
if [[ "$a" -gt "$b" ]]; then
   echo "$a больше $b"
elif [[ "$a" -lt "$b" ]]; then
```

```
   echo "$a меньше $b"
else
   echo "Числа равны!"
fi

# сравниваем строки
echo "Сравниваем строки:"
if [[ "$a" > "$b" ]]; then
   echo "$a больше $b"
elif [[ "$a" < "$b" ]]; then
   echo "$a меньше $b"
else
   echo "Количество символов равно!"
fi
```

```
$ bash ~/test-if-2.sh
```

Ответ:



Сравнение чисел и сравнение строк.

В таком сравнении используются следующие операторы:

- **-ne** – не равно, если вы сравниваете числа;
- **!=** – не равно, если вы сравниваете строки.

При сравнении по символам надо иметь в виду, что:

- Оператор -ne сравнивает числа, а != сравнивает строки.
- Если нужно сравнить длину строк, надо использовать операторы

${#a} и ${#b}.
- Строки с числами “01” и “02” не равны по содержимому, но их длина одинакова.

Сравнение чисел (-ne).

Когда используется оператор -ne, **bash** интерпретирует переменные a и b как числа.

В этом случае:

a=01 интерпретируется как число 1.
b=02 интерпретируется как число 2.

Поскольку 1 и 2 — это разные числа, условие [[“$a” -ne “$b”]] возвращает true.

Ответ:

> *01 не равно по величине с 02*

Сравнение строк (!=).

Когда вы используете оператор !=, Bash интерпретирует переменные a и b как строки.

В этом случае:

a=01 — это строка, состоящая из символов 0 и 1.
b=02 — это строка, состоящая из символов 0 и 2.

Поскольку строки “01” и “02” не равны (они содержат разные символы), условие [[“$a” != “$b”]] возвращает true.

Ответ:

> *01 не равно по символам с 02*

Сравнение символов в строках (${#a} и ${#b}).

Вопрос:

> *Почему не выводится “Количество символов равно!”, если ввести a=01, а b=02, хотя в a — это 2 символа и в b — это 2 символа?*

Ответ:

> *Потому что операторы **-ne** и != не проверяют количество символов в строках. Они сравнивают строки по их содержимому.*

Если нужно проверить, равны ли строки по длине, нужно использовать другую конструкцию.

Например:

```
$ cd ~
```

```
$ mcedit test-if-dlina.sh
```

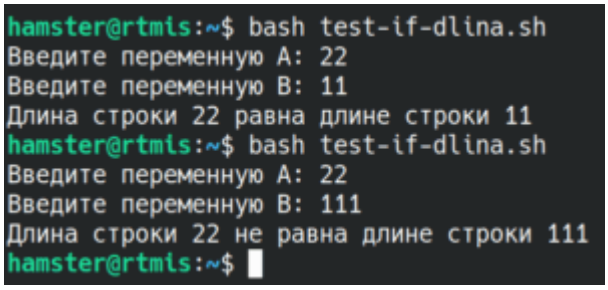
```
$ cat test-if-dlina.sh
```

```
#!/bin/bash

# Запрос на ввод переменной A
read -p "Введите переменную A: " a

# Запрос на ввод переменной B
read -p "Введите переменную B: " b

# Сравниваем длину строк
if [[ ${#a} -ne ${#b} ]]; then
echo "Длина строки $a не равна длине строки $b"
else
echo "Длина строки $a равна длине строки $b"
fi
```

Здесь ${#a} и ${#b} возвращают длину строк a и b соответственно.

```
$ bash test-if-dlina.sh
```

Ответ:



Итоговый демонстрационный скрипт.

Посмотрим как это всё работает:

```
$ cd ~
```

```
$ mcedit test-if-3.sh
```

```
$ cat test-if-3.sh
```

```
#!/bin/bash

# Запрос на ввод переменной A
read -p "Введите переменную A: " a

# Запрос на ввод переменной B
read -p "Введите переменную B: " b

# Сравниваем числа
echo "Сравниваем числа:"
if [[ "$a" -ne "$b" ]]; then
echo "$a не равно по величине с $b"
else
echo "Числа равны!"
fi

# Сравниваем строки
echo "Сравниваем строки:"
if [[ "$a" != "$b" ]]; then
echo "$a не равно по символам с $b"
else
echo "Строки равны!"
fi

# Сравниваем длину строк
echo "Сравниваем длину строк:"
if [[ ${#a} -ne ${#b} ]]; then
echo "Длина строки $a не равна длине строки $b"
else
echo "Длина строки $a равна длине строки $b"
fi

$ bash test-if-3.sh
```

Ответ:

```
hamster@rtmis:~$ bash test-if-3.sh
Введите переменную A: 11
Введите переменную B: 22
Сравниваем числа:
11 не равно по величине с 22
Сравниваем строки:
11 не равно по символам с 22
Сравниваем длину строк:
Длина строки 11 равна длине строки 22
hamster@rtmis:~$ bash test-if-3.sh
Введите переменную A: 01
Введите переменную B: 02
Сравниваем числа:
01 не равно по величине с 02
Сравниваем строки:
01 не равно по символам с 02
Сравниваем длину строк:
Длина строки 01 равна длине строки 02
hamster@rtmis:~$ bash test-if-3.sh
Введите переменную A: 001
Введите переменную B: 02
Сравниваем числа:
001 не равно по величине с 02
Сравниваем строки:
001 не равно по символам с 02
Сравниваем длину строк:
Длина строки 001 не равна длине строки 02
hamster@rtmis:~$ bash test-if-3.sh
Введите переменную A: 03
Введите переменную B: 03
Сравниваем числа:
Числа равны!
Сравниваем строки:
Строки равны!
Сравниваем длину строк:
Длина строки 03 равна длине строки 03
hamster@rtmis:~$
```

Обработка пустой строки.

В сравнении строк применяются не только бинарные операторы, где сравниваются две строки, а также унарные, где смотрится является ли строка пустой или нет.

Обратите внимание! Здесь обрабатывается только одна строка, поэтому оператор называется унарным:

- **[[-z "$a"]]** – является ли строка пустой;
- **[[-n "$a"]]** – является ли строка не пустой.

Пример с пустой строкой:

```
$ cd ~

$ mcedit test-if-empty.sh

$ cat test-if-empty.sh


#!/bin/bash

# Запрос на ввод переменной A
read -p "Введите переменную A (ENTER чтобы ввести пустое значение):
" a

if [[ -z "$a" ]]; then
   echo "Строка пустая!"
elif [[ -n "$a" ]]; then
   echo "Строка не пустая!"
fi

$ bash test-if-empty.sh
```

Ответ:

```
  hamster@rtmis:~bash test-if-empty.sh
Введите переменную A (ENTER чтобы ввести пустое значение):
Строка пустая!
hamster@rtmis:~$ bash test-if-empty.sh
Введите переменную A (ENTER чтобы ввести пустое значение): 33
Строка не пустая!
hamster@rtmis:~$
```

Булевы операторы.

Булевы операторы ! (нет), && (и), || (или) также можно использовать

c if:

[[условие]] && [[условие]]

Например:

$ cd ~

$ mcedit test-if-boolean.sh

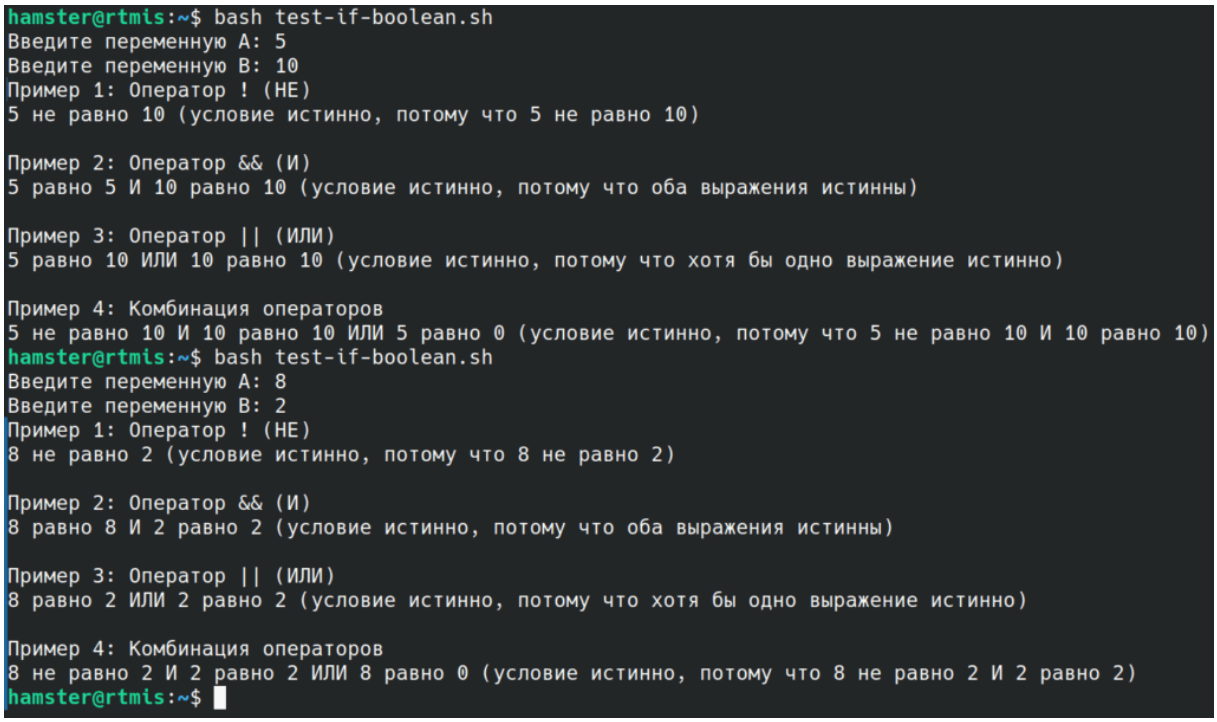
$ cat test-if-boolean.sh

```
#!/bin/bash

# Запрос на ввод переменной A
read -p "Введите переменную A: " a

# Проверка, что введенное значение — число
if ! [[ "$a" =~ ^[0-9]+$ ]]; then
    echo "Ошибка: переменная A должна быть числом."
    exit 1
fi

# Запрос на ввод переменной B
read -p "Введите переменную B: " b

# Проверка, что введенное значение — число
if ! [[ "$b" =~ ^[0-9]+$ ]]; then
    echo "Ошибка: переменная B должна быть числом."
    exit 1
fi

# Демонстрация булевых операторов с if

# Пример 1: Оператор ! (НЕ)
echo "Пример 1: Оператор ! (НЕ)"
if ! [[ "$a" -eq "$b" ]]; then
    echo "$a не равно $b (условие истинно, потому что $a не равно
$b)"
else
    echo "$a равно $b (условие ложно)"
fi

# Пример 2: Оператор && (И)
echo -e "\nПример 2: Оператор && (И)"
if [[ "$a" -eq "$a" ]] && [[ "$b" -eq "$b" ]]; then
    echo "$a равно $a И $b равно $b (условие истинно, потому что оба
выражения истинны)"
```

```
else
    echo "Условие ложно (хотя бы одно из выражений ложно)"
fi

# Пример 3: Оператор || (ИЛИ)
echo -e "\nПример 3: Оператор || (ИЛИ)"
if [[ "$a" -eq "$b" ]] || [[ "$b" -eq "$b" ]]; then
    echo "$a равно $b ИЛИ $b равно $b (условие истинно, потому что
хотя бы одно выражение истинно)"
else
    echo "Условие ложно (оба выражения ложны)"
fi

# Пример 4: Комбинация операторов
echo -e "\nПример 4: Комбинация операторов"
if ! [[ "$a" -eq "$b" ]] && [[ "$b" -eq "$b" ]] || [[ "$a" -eq "0"
]]; then
    echo "$a не равно $b И $b равно $b ИЛИ $a равно 0 (условие
истинно, потому что $a не равно $b И $b равно $b)"
else
    echo "Условие ложно"
fi
```

```
$ bash test-if-boolean.sh
```

Ответ:



Проверка файлов.

С помощью if можно проверить существует ли файл, является ли файл файлом или каталогом и многое другое.

Унарные операторы:

- **-e** – объект существует (файл или каталог).
- **-f** – объект существует и является файлом.
- **-s** – файл не пуст (имеет не нулевой размер).
- **-d** – файл является каталогом.
- **-b** – файл является блочным устройством.
- **-c** – файл является символьным устройством.
- **-p** – файл является каналом.
- **-h** – файл является символической ссылкой.
- **-S** – файл является сокетом.
- **-t** – является ли дескриптор файла открытым и связанным с терминалом. Например для проверки stdin [-t 0] или stdout [-t 1].
- **-r** – файл доступен для чтения (пользователю, запустившему сценарий).
- **-w** – файл доступен для записи (пользователю, запустившему сценарий).
- **-x** – файл доступен для исполнения (пользователю, запустившему сценарий).
- **-O** – вы являетесь владельцем файла.
- **-G** – вы принадлежите к той же группе, что и файл.

Бинарные операторы:

- **f1 -nt f2** – файл f1 более новый, чем f2.
- **f1 -ot f2** – файл f1 более старый, чем f2.
- **f1 -ef f2** – файлы f1 и f2 являются «жесткими» ссылками на один и тот же файл.

Использование команды вместо условия.

В этом случае используется такая конструкция:

if команда; then

Если команда завершилась с кодом выхода 0, то значит условие выполнилось.

Например:

```
$ cd ~
$ mcedit fruits.txt
```

```
$ cat fruits.txt

яблоко
слива
груша
вишня

$ mcedit test-if-fruits.sh

$ cat test-if-fruits.sh

#!/bin/bash

if grep вишня fruits.txt; then
echo "Вишня в тексте есть!"
fi

$ bash test-if-fruits.sh
```

Ответ:

```
hamster@rtmis:~$ bash test-if-fruits.sh
вишня
«Вишня в тексте есть!»
hamster@rtmis:~$
```

На этом все!

Оригиналы источников информации.

1. telegra.ph «Bash. Условный оператор if».