**7.3. Операции сравнения**

**сравнение целых чисел**

-eq

равно

**if [ "$a" -eq "$b" ]**

-ne

не равно

**if [ "$a" -ne "$b" ]**

-gt

больше

**if [ "$a" -gt "$b" ]**

-ge

больше или равно

**if [ "$a" -ge "$b" ]**

-lt

меньше

**if [ "$a" -lt "$b" ]**

-le

меньше или равно

**if [ "$a" -le "$b" ]**

<

меньше (внутри [двойных круглых скобок](http://www.opennet.ru/docs/RUS/bash_scripting_guide/x4862.html) )

**(("$a" < "$b"))**

<=

меньше или равно (внутри двойных круглых скобок)

**(("$a" <= "$b"))**

>

больше (внутри двойных круглых скобок)

**(("$a" > "$b"))**

>=

больше или равно (внутри двойных круглых скобок)

**(("$a" >= "$b"))**

**сравнение строк**

=

равно

**if [ "$a" = "$b" ]**

==

равно

**if [ "$a" == "$b" ]**

Синоним оператора =.

[[ $a == z\* ]] # истина, если $a начинается с символа "z" (сравнение по шаблону)

[[ $a == "z\*" ]] # истина, если $a равна z\*

[ $a == z\* ] # имеют место подстановка имен файлов и разбиение на слова

[ "$a" == "z\*" ] # истина, если $a равна z\*

# Спасибо S.C.

!=

не равно

**if [ "$a" != "$b" ]**

Этот оператор используется при поиске по шаблону внутри [[[ ... ]]](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c2171.html#DBLBRACKETS).

<

меньше, в смысле величины ASCII-кодов

**if [[ "$a" < "$b" ]]**

**if [ "$a" \< "$b" ]**

Обратите внимание! Символ "<" необходимо экранировать внутри **[ ]**.

>

больше, в смысле величины ASCII-кодов

**if [[ "$a" > "$b" ]]**

**if [ "$a" \> "$b" ]**

Обратите внимание! Символ ">" необходимо экранировать внутри **[ ]**.

См. [Пример 25-6](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c12790.html#BUBBLE) относительно применения этого оператора сравнения.

-z

строка "пустая", т.е. имеет нулевую длину

-n

строка не "пустая".

|  |  |
| --- | --- |
| Caution | Оператор **-n** требует, чтобы строка была заключена в кавычки внутри квадратных скобок. Как правило, проверка строк, не заключенных в кавычки, оператором **! -z**, или просто указание строки без кавычек внутри квадратных скобок (см. [Пример 7-6](http://www.opennet.ru/docs/RUS/bash_scripting_guide/x2565.html#STRTEST)), проходит нормально, однако это небезопасная, с точки зрения отказоустойчивости, практика. *Всегда* заключайте проверяемую строку в кавычки. [[1]](http://www.opennet.ru/docs/RUS/bash_scripting_guide/x2565.html" \l "FTN.AEN2722) |

**Пример 7-5. Операции сравнения**

#!/bin/bash

a=4

b=5

# Здесь переменные "a" и "b" могут быть как целыми числами, так и строками.

# Здесь наблюдается некоторое размывание границ

#+ между целочисленными и строковыми переменными,

#+ поскольку переменные в Bash не имеют типов.

# Bash выполняет целочисленные операции над теми переменными,

#+ которые содержат только цифры

# Будьте внимательны!

echo

if [ "$a" -ne "$b" ]

then

echo "$a не равно $b"

echo "(целочисленное сравнение)"

fi

echo

if [ "$a" != "$b" ]

then

echo "$a не равно $b."

echo "(сравнение строк)"

# "4" != "5"

# ASCII 52 != ASCII 53

fi

# Оба варианта, "-ne" и "!=", работают правильно.

echo

exit 0

**Пример 7-6. Проверка -- является ли строка *пустой***

#!/bin/bash

# str-test.sh: Проверка пустых строк и строк, не заключенных в кавычки,

# Используется конструкция if [ ... ]

# Если строка не инициализирована, то она не имеет никакого определенного значения.

# Такое состояние называется "null" (пустая) (это не то же самое, что ноль).

if [ -n $string1 ] # $string1 не была объявлена или инициализирована.

then

echo "Строка \"string1\" не пустая."

else

echo "Строка \"string1\" пустая."

fi

# Неверный результат.

# Выводится сообщение о том, что $string1 не пустая,

#+не смотря на то, что она не была инициализирована.

echo

# Попробуем еще раз.

if [ -n "$string1" ] # На этот раз, переменная $string1 заключена в кавычки.

then

echo "Строка \"string1\" не пустая."

else

echo "Строка \"string1\" пустая."

fi # Внутри квадратных скобок заключайте строки в кавычки!

echo

if [ $string1 ] # Опустим оператор -n.

then

echo "Строка \"string1\" не пустая."

else

echo "Строка \"string1\" пустая."

fi

# Все работает прекрасно.

# Квадратные скобки -- [ ], без посторонней помощи определяют, что строка пустая.

# Тем не менее, хорошим тоном считается заключать строки в кавычки ("$string1").

#

# Как указывает Stephane Chazelas,

# if [ $string 1 ] один аргумент "]"

# if [ "$string 1" ] два аргумента, пустая "$string1" и "]"

echo

string1=initialized

if [ $string1 ] # Опять, попробуем строку без ничего.

then

echo "Строка \"string1\" не пустая."

else

echo "Строка \"string1\" пустая."

fi

# И снова получим верный результат.

# И опять-таки, лучше поместить строку в кавычки ("$string1"), поскольку...

string1="a = b"

if [ $string1 ] # И снова, попробуем строку без ничего..

then

echo "Строка \"string1\" не пустая."

else

echo "Строка \"string1\" пустая."

fi

# Строка без кавычек дает неверный результат!

exit 0

# Спвсибо Florian Wisser, за предупреждение.

**Пример 7-7. zmost**

#!/bin/bash

#Просмотр gz-файлов с помощью утилиты 'most'

NOARGS=65

NOTFOUND=66

NOTGZIP=67

if [ $# -eq 0 ] # то же, что и: if [ -z "$1" ]

# $1 должен существовать, но может быть пустым: zmost "" arg2 arg3

then

echo "Порядок использования: `basename $0` filename" >&2

# Сообщение об ошибке на stderr.

exit $NOARGS

# Код возврата 65 (код ошибки).

fi

filename=$1

if [ ! -f "$filename" ] # Кавычки необходимы на тот случай, если имя файла содержит пробелы.

then

echo "Файл $filename не найден!" >&2

# Сообщение об ошибке на stderr.

exit $NOTFOUND

fi

if [ ${filename##\*.} != "gz" ]

# Квадратные скобки нужны для выполнения подстановки значения переменной

then

echo "Файл $1 не является gz-файлом!"

exit $NOTGZIP

fi

zcat $1 | most

# Используется утилита 'most' (очень похожа на 'less').

# Последние версии 'most' могут просматривать сжатые файлы.

# Можно вставить 'more' или 'less', если пожелаете.

exit $? # Сценарий возвращает код возврата, полученный по конвейеру.

# На самом деле команда "exit $?" не является обязательной,

# так как работа скрипта завершится здесь в любом случае,

**построение сложных условий проверки**

-a

логическое И (and)

*exp1 -a exp2* возвращает true, если *оба* выражения, и exp1, и exp2 истинны.

-o

логическое ИЛИ (or)

*exp1 -o exp2* возвращает true, если хотябы одно из выражений, exp1 *или* exp2 истинно.

Они похожи на операторы Bash **&&** и **||**, употребляемые в [двойных квадратных скобках](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c2171.html#DBLBRACKETS).

[[ condition1 && condition2 ]]

Операторы **-o** и **-a** употребляются совместно с командой **test** или внутри одинарных квадратных скобок.

if [ "$exp1" -a "$exp2" ]

Чтобы увидеть эти операторы в действии, смотрите [Пример 8-3](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c2792.html#ANDOR) и [Пример 25-11](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c12790.html#TWODIM).

**Примечания**

|  |  |
| --- | --- |
| [[1]](http://www.opennet.ru/docs/RUS/bash_scripting_guide/x2565.html" \l "AEN2722) | Как указывает S.C., даже заключение строки в кавычки, при построении сложных условий проверки, может оказаться недостаточным. **[ -n "$string" -o "$a" = "$b" ]** в некоторых версиях Bash такая проверка может вызвать сообщение об ошибке, если строка $string пустая. Безопаснее, в смысле отказоустойчивости, было бы добавить какой-либо символ к, возможно пустой, строке: **[ "x$string" != x -o "x$a" = "x$b" ]** (символ "x" не учитывается). |

|  |  |  |
| --- | --- | --- |
| [Назад](http://www.opennet.ru/docs/RUS/bash_scripting_guide/x2389.html) | [К началу](http://www.opennet.ru/docs/RUS/bash_scripting_guide/index.html) | [Вперед](http://www.opennet.ru/docs/RUS/bash_scripting_guide/x2770.html) |
| Операции проверки файлов | [Наверх](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c2171.html) | Вложенные условные операторы if/then |

# 7.2. Операции проверки файлов

**Возвращает true если...**

-e

файл существует

-f

обычный файл (не каталог и не файл устройства)

-s

ненулевой размер файла

-d

файл является каталогом

-b

файл является блочным устройством (floppy, cdrom и т.п.)

-c

файл является символьным устройством (клавиатура, модем, звуковая карта и т.п.)

-p

файл является каналом

-h

файл является символической ссылкой

-L

файл является символической ссылкой

-S

файл является сокетом

-t

файл ([дескриптор](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c11620.html#FDREF)) связан с терминальным устройством

Этот ключ может использоваться для проверки -- является ли файл стандартным устройством ввода stdin (**[ -t 0 ]**) или стандартным устройством вывода 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)

-g

set-group-id (sgid) флаг для файла или каталога установлен

Если для каталога установлен флаг sgid, то файлы, создаваемые в таком каталоге, наследуют идентификатор группы каталога, который может не совпадать с идентификатором группы, к которой принадлежит пользователь, создавший файл. Это может быть полезно для каталогов, в которых хранятся файлы, общедоступные для группы пользователей.

-u

set-user-id (suid) флаг для файла установлен

Установленный флаг suid приводит к изменению привилегий запущенного процесса на привилегии владельца исполняемого файла. Исполняемые файлы, владельцем которых является root, с установленным флагом set-user-id запускаются с привилегиями root, даже если их запускает обычный пользователь. [[1]](http://www.opennet.ru/docs/RUS/bash_scripting_guide/x2389.html" \l "FTN.AEN2488) Это может оказаться полезным для некоторых программ (таких как **pppd** и **cdrecord**), которые осуществляют доступ к аппаратной части компьютера. В случае отсутствия флага suid, программы не смогут быть запущены рядовым пользователем, не обладающим привилегиями root.

-rwsr-xr-t 1 root 178236 Oct 2 2000 /usr/sbin/pppd

Файл с установленным флагом suid отображается с включенным флагом s в поле прав доступа.

-k

флаг sticky bit (бит фиксации) установлен

Общеизвестно, что флаг "sticky bit" -- это специальный тип прав доступа к файлам. Программы с установленным флагом "sticky bit" остаются в системном кэше после своего завершения, обеспечивая тем самым более быстрый запуск программы. [[2]](http://www.opennet.ru/docs/RUS/bash_scripting_guide/x2389.html" \l "FTN.AEN2508) Если флаг установлен для каталога, то это приводит к ограничению прав на запись. Установленный флаг "sticky bit" отображается в виде символа t в поле прав доступа.

drwxrwxrwt 7 root 1024 May 19 21:26 tmp/

Если пользователь не является владельцем каталога, с установленным "sticky bit", но имеет право на запись в каталог, то он может удалять только те файлы в каталоге, владельцем которых он является. Это предотвращает удаление и перезапись "чужих" файлов в общедоступных каталогах, таких как /tmp.

-O

вы являетесь владельцем файла

-G

вы принадлежите к той же группе, что и файл

-N

файл был модифицирован с момента последнего чтения

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

!

"НЕ" -- логическое отрицание (инверсия) результатов всех вышеприведенных проверок (возвращается true если условие отсутствует).

**Пример 7-4. Проверка "битых" ссылок**

#!/bin/bash

# broken-link.sh

# Автор Lee Bigelow <ligelowbee@yahoo.com>

# Используется с его разрешения.

#Сценарий поиска "битых" ссылок и их вывод в "окавыченном" виде

#таким образом они могут передаваться утилите xargs для дальнейшей обработки :)

#например. broken-link.sh /somedir /someotherdir|xargs rm

#

#На всякий случай приведу лучший метод:

#

#find "somedir" -type l -print0|\

#xargs -r0 file|\

#grep "broken symbolic"|

#sed -e 's/^\|: \*broken symbolic.\*$/"/g'

#

#но это не чисто BASH-евский метод, а теперь сам сценарий.

#Внимание! будьте осторожны с файловой системой /proc и циклическими ссылками!

##############################################################

#Если скрипт не получает входных аргументов,

#то каталогом поиска является текущая директория

#В противном случае, каталог поиска задается из командной строки

####################

[ $# -eq 0 ] && directorys=`pwd` || directorys=$@

#Функция linkchk проверяет каталог поиска

#на наличие в нем ссылок на несуществующие файлы, и выводит их имена.

#Если анализируемый файл является каталогом,

#то он передается функции linkcheck рекурсивно.

##########

linkchk () {

for element in $1/\*; do

[ -h "$element" -a ! -e "$element" ] && echo \"$element\"

[ -d "$element" ] && linkchk $element

# Само собой, '-h' проверяет символические ссылки, '-d' -- каталоги.

done

}

#Вызов функции linkchk для каждого аргумента командной строки,

#если он является каталогом. Иначе выводится сообщение об ошибке

#и информация о порядке пользования скриптом.

################

for directory in $directorys; do

if [ -d $directory ]

then linkchk $directory

else

echo "$directory не является каталогом"

echo "Порядок использования: $0 dir1 dir2 ..."

fi

done

exit 0

[Пример 28-1](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c13041.html#COOKIES), [Пример 10-7](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c4875.html#BINGREP), [Пример 10-3](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c4875.html#FILEINFO), [Пример 28-3](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c13041.html#RAMDISK) и [Пример A-2](http://www.opennet.ru/docs/RUS/bash_scripting_guide/a14477.html#MAILFORMAT) так же иллюстрируют операции проверки файлов.

### Примечания

|  |  |
| --- | --- |
| [[1]](http://www.opennet.ru/docs/RUS/bash_scripting_guide/x2389.html" \l "AEN2488) | С флагом suid, на двоичных исполняемых файлах, надо быть очень осторожным, поскольку это может быть небезопасным. Установка флага suid на файлы-сценарии не имеет никакого эффекта. |
| [[2]](http://www.opennet.ru/docs/RUS/bash_scripting_guide/x2389.html" \l "AEN2508) | В современных UNIX-системах, "sticky bit" больше не используется для файлов, только для каталогов. |

|  |  |  |
| --- | --- | --- |
| [Назад](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c2171.html) | [К началу](http://www.opennet.ru/docs/RUS/bash_scripting_guide/index.html) | [Вперед](http://www.opennet.ru/docs/RUS/bash_scripting_guide/x2565.html) |
| Проверка условий | [Наверх](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c2171.html) | Операции сравнения |