

[Каталог документации](#) / [Раздел "Программирование, языки"](#) / [Оглавление документа](#)

## Advanced Bash-Scripting Guide: Искусство программирования на языке сценариев командной оболочки

[Назад](#)

Глава 9. К вопросу о переменных

[Вперед](#)

## 9.2. Работа со строками

Bash поддерживает на удивление большое количество операций над строками. К сожалению, этот раздел Bash испытывает недостаток унификации. Одни операции являются подмножеством операций [подстановки параметров](#), а другие -- совпадают с функциональностью команды UNIX - [expr](#). Это приводит к противоречиям в синтаксисе команд и перекрытию функциональных возможностей, не говоря уже о возникающей путанице.

### Длина строки

```

${#string}
expr length $string
expr "$string" : '.*'

stringZ=abcABC123ABcabc

echo ${#stringZ}           # 15
echo `expr length $stringZ` # 15
echo `expr "$stringZ" : '.*'` # 15

```

### Пример 9-10. Вставка пустых строк между параграфами в текстовом файле

```

#!/bin/bash
# paragraph-space.sh

# Вставка пустых строк между параграфами в текстовом файле.
# Порядок использования: $0 <FILENAME

MINLEN=45      # Возможно потребуется изменить это значение.
# Строки, содержащие количество символов меньше, чем $MINLEN
#+ принимаются за последнюю строку параграфа.

```

```

while read line # Построчное чтение файла от начала до конца...

```

```
do
    echo "$line"    # Вывод строки.

    len=${#line}
    if [ "$len" -lt "$MINLEN" ]
    then echo      # Добавление пустой строки после последней строки параграфа.
    fi
done

exit 0
```

**Длина подстроки в строке (подсчет совпадающих символов ведется с начала строки)**

```
expr match "$string" '$substring'
```

где *\$substring* -- регулярное выражение.

```
expr "$string" : '$substring'
```

где *\$substring* -- регулярное выражение.

```
stringZ=abcABC123ABCabc
#      |-----|
```

```
echo `expr match "$stringZ" 'abc[A-Z]*.2'`    # 8
echo `expr "$stringZ" : 'abc[A-Z]*.2'`        # 8
```

## Index

```
expr index $string $substring
```

Номер позиции первого совпадения в *\$string* с первым символом в *\$substring*.

```
stringZ=abcABC123ABCabc
echo `expr index "$stringZ" C12`              # 6
                                              # позиция символа C.
```

```
echo `expr index "$stringZ" 1c`               # 3
# символ 'c' (в #3 позиции) совпал раньше, чем '1'.
```

Эта функция довольно близка к функции *strchr()* в языке C.

## Извлечение подстроки

## `${string:position}`

Извлекает подстроку из *\$string*, начиная с позиции *\$position*.

Если строка *\$string* -- "\*" или "@", то извлекается **позиционный параметр** (аргумент), [1] с номером *\$position*.

## `${string:position:length}`

Извлекает *\$length* символов из *\$string*, начиная с позиции *\$position*.

```
stringZ=abcABC123ABCAbc
```

```
#      0123456789.....
```

```
#      Индексация начинается с 0.
```

```
echo ${stringZ:0}           # abcABC123ABCAbc
```

```
echo ${stringZ:1}           # bcABC123ABCAbc
```

```
echo ${stringZ:7}           # 23ABCAbc
```

```
echo ${stringZ:7:3}         # 23A
```

```
# Извлекает 3 символа.
```

```
# Возможна ли индексация с "правой" стороны строки?
```

```
echo ${stringZ:-4}          # abcABC123ABCAbc
```

```
# По-умолчанию выводится полная строка.
```

```
# Однако . . .
```

```
echo ${stringZ:(-4)}        # CAbc
```

```
echo ${stringZ: -4}         # CAbc
```

```
# Теперь выводится правильно.
```

```
# Круглые скобки или дополнительный пробел "экранируют" параметр позиции.
```

```
# Спасибо Dan Jacobson, за разъяснения.
```

Если *\$string* -- "\*" или "@", то извлекается до *\$length* позиционных параметров (аргументов), начиная с *\$position*.

```
echo ${*:2}                 # Вывод 2-го и последующих аргументов.
```

```
echo ${@:2}                 # То же самое.
```

```
echo ${*:2:3}               # Вывод 3-х аргументов, начиная со 2-го.
```

```
expr substr $string $position $length
```

Извлекает *\$length* символов из *\$string*, начиная с позиции *\$position*.

```
stringZ=abcABC123ABCabC
```

```
#      123456789.....
```

```
#      Индексация начинается с 1.
```

```
echo `expr substr $stringZ 1 2`           # ab
```

```
echo `expr substr $stringZ 4 3`           # ABC
```

```
expr match "$string" '\($substring\)
```

Находит и извлекает первое совпадение *\$substring* в *\$string*, где *\$substring* -- это [регулярное выражение](#).

```
expr "$string" : '\($substring\)
```

Находит и извлекает первое совпадение *\$substring* в *\$string*, где *\$substring* -- это регулярное выражение.

```
stringZ=abcABC123ABCabC
```

```
#      =====
```

```
echo `expr match "$stringZ" '\([b-c]*[A-Z]..[0-9]\)` # abcABC1
```

```
echo `expr "$stringZ" : '\([b-c]*[A-Z]..[0-9]\)` # abcABC1
```

```
echo `expr "$stringZ" : '\(.....\)` # abcABC1
```

```
# Все вышеприведенные операции дают один и тот же результат.
```

```
expr match "$string" '.*\($substring\)
```

Находит и извлекает первое совпадение *\$substring* в *\$string*, где *\$substring* -- это регулярное выражение. Поиск начинается с конца *\$string*.

```
expr "$string" : '.*\($substring\)
```

Находит и извлекает первое совпадение *\$substring* в *\$string*, где *\$substring* -- это регулярное выражение. Поиск начинается с конца *\$string*.

```
stringZ=abcABC123ABCabC
```

```
#      =====
```

```
echo `expr match "$stringZ" '.*\([A-C][A-C][A-C][a-c]*\)` # ABCabc
```

```
echo `expr "$stringZ" : '.*\(.....\)` # ABCabc
```

## Удаление части строки

### `${string#substring}`

Удаление самой короткой, из найденных, подстроки *\$substring* в строке *\$string*. Поиск ведется с начала строки

### `${string##substring}`

Удаление самой длинной, из найденных, подстроки *\$substring* в строке *\$string*. Поиск ведется с начала строки

```
stringZ=abcABC123ABCAbc
```

```
#      |----|
```

```
#      |-----|
```

```
echo ${stringZ#a*C}      # 123ABCAbc
```

```
# Удаление самой короткой подстроки.
```

```
echo ${stringZ##a*C}     # abc
```

```
# Удаление самой длинной подстроки.
```

### `${string%substring}`

Удаление самой короткой, из найденных, подстроки *\$substring* в строке *\$string*. Поиск ведется с конца строки

### `${string%%substring}`

Удаление самой длинной, из найденных, подстроки *\$substring* в строке *\$string*. Поиск ведется с конца строки

```
stringZ=abcABC123ABCAbc
```

```
#                      ||
```

```
#      |-----|
```

```
echo ${stringZ%b*c}      # abcABC123ABCa
```

```
# Удаляется самое короткое совпадение. Поиск ведется с конца $stringZ.
```

```
echo ${stringZ%%b*c}     # a
```

```
# Удаляется самое длинное совпадение. Поиск ведется с конца $stringZ.
```

**Пример 9-11. Преобразование графических файлов из одного формата в другой, с изменением имени файла**

```
#!/bin/bash
# cvt.sh:
# Преобразование всех файлов в заданном каталоге,
#+ из графического формата MacPaint, в формат "pbm".

# Используется утилита "macstopbm", входящая в состав пакета "netpbm",
#+ который сопровождается Brian Henderson (bryanh@giraffe-data.com).
# Netpbm -- стандартный пакет для большинства дистрибутивов Linux.

OPERATION=macstopbm
SUFFIX=pbm          # Новое расширение файла.

if [ -n "$1" ]
then
    directory=$1      # Если каталог задан в командной строке при вызове сценария
else
    directory=$PWD     # Иначе просматривается текущий каталог.
fi

# Все файлы в каталоге, имеющие расширение ".mac", считаются файлами
#+ формата MacPaint.

for file in $directory/* # Подстановка имен файлов.
do
    filename=${file%.*c} # Удалить расширение ".mac" из имени файла
                        #+ ( с шаблоном '*.c' совпадают все подстроки
                        #+ начинающиеся с '.' и заканчивающиеся 'c',
    $OPERATION $file > "$filename.$SUFFIX"
                        # Преобразование с перенаправлением в файл с новым именем
    rm -f $file         # Удаление оригинального файла после преобразования.
    echo "$filename.$SUFFIX" # Вывод на stdout.
done

exit 0

# Упражнение:
# -----
# Сейчас этот сценарий конвертирует *все* файлы в каталоге
# Измените его так, чтобы он конвертировал *только* те файлы,
#+ которые имеют расширение ".mac".
```

**Замена подстроки**

```
${string/substring/replacement}
```

Замещает первое вхождение *\$substring* строкой *\$replacement*.

**`${string//substring/replacement}`**

Замещает все вхождения *\$substring* строкой *\$replacement*.

```
stringZ=abcABC123ABCabc
```

```
echo ${stringZ/abc/xyz}      # xyzABC123ABCabc
                             # Замена первой подстроки 'abc' строкой 'xyz'.
```

```
echo ${stringZ//abc/xyz}    # xyzABC123ABCxyz
                             # Замена всех подстрок 'abc' строкой 'xyz'.
```

**`${string/#substring/replacement}`**

Подстановка строки *\$replacement* вместо *\$substring*. Поиск ведется с начала строки *\$string*.

**`${string/%substring/replacement}`**

Подстановка строки *\$replacement* вместо *\$substring*. Поиск ведется с конца строки *\$string*.

```
stringZ=abcABC123ABCabc
```

```
echo ${stringZ/#abc/XYZ}    # XYZABC123ABCabc
                             # Поиск ведется с начала строки
```

```
echo ${stringZ/%abc/XYZ}    # abcABC123ABCXYZ
                             # Поиск ведется с конца строки
```

## 9.2.1. Использование awk при работе со строками

В качестве альтернативы, Bash-скрипты могут использовать средства **awk** при работе со строками.

### Пример 9-12. Альтернативный способ извлечения подстрок

```
#!/bin/bash
# substring-extraction.sh
```

```
String=23skidoo1
#      012345678   Bash
#      123456789   awk
```

# Обратите внимание на различия в индексации:

# Bash начинает индексацию с '0'.

# Awk  начинает индексацию с '1'.

```
echo ${String:2:4} # с 3 позиции (0-1-2), 4 символа
                  # skid
```

# В эквивалент в awk: substr(string,pos,length).

```
echo | awk '
{ print substr("'"${String}"'",3,4)      # skid
}
'
```

# Передача пустого "echo" по каналу в awk, означает фиктивный ввод,  
#+ деля, тем самым, ненужным предоставление имени файла.

```
exit 0
```

## 9.2.2. Дальнейшее обсуждение

Дополнительную информацию, по работе со строками, вы найдете в разделе [Section 9.3](#) и в [секции](#), посвященной команде [expr](#). Примеры сценариев:

1. [Пример 12-6](#)
2. [Пример 9-15](#)
3. [Пример 9-16](#)
4. [Пример 9-17](#)
5. [Пример 9-19](#)

## Примечания

[1] Применяется к аргументам командной строки или входным параметрам [функций](#).

---

[Назад](#)

К вопросу о переменных

[К началу](#)

[Наверх](#)

[Вперед](#)

Подстановка параметров

Партнёры:



При поддержке  
**inferno solutions\***



[Закладки на сайте](#)  
[Проследить за страницей](#)

Created 1996-2024 by [Maxim Chirkov](#)  
[Добавить](#), [Поддержать](#), [Вебмастеру](#)