
3.5.3 Расширение параметров оболочки

Символ ‘$’ обозначает расширение параметров, замену команд или арифметическое расширение. Имя параметра или символ, подлежащий расширению, может быть заключен в фигурные скобки, которые необязательны, но служат для защиты расширяемой переменной от следующих сразу за ней символов, которые могут быть интерпретированы как часть имени.

При использовании фигурных скобок соответствующая завершающая скобка является первой "}", которая не экранируется обратной косой чертой или в строке, заключенной в кавычки, а не во встроенном арифметическом расширении, замене команд или расширении параметров.

Основной формой расширения параметров является ${*parameter*}. Подставляется значение *parameter*. *parameter* Это параметр оболочки, как описано выше (см. Параметры оболочки) или ссылка на массив (см. Массивы). Фигурные скобки требуются, когда *parameter* позиционный параметр содержит более одной цифры или когда *parameter* за ним следует символ, который не следует интерпретировать как часть его имени.

Если первый символ *parameter* является восклицательным знаком (!), а *parameter* не является ссылкой на имя, это вводит уровень косвенности. Bash использует значение, сформированное путем расширения остальной части, *parameter* как новое *parameter*; затем оно расширяется, и это значение используется в остальной части расширения, а не в расширении оригинала *parameter*. Это известно как `indirect expansion`. Значение может быть расширено с помощью тильды, расширения параметров, замены команд и арифметического расширения. Если *parameter* это nameref, оно расширяется до имени переменной, на которую ссылается *parameter*, вместо выполнения полного косвенного расширения. Исключением из этого правила являются расширения ${!*prefix**} и ${!*name*[@]} описано ниже. Восклицательный знак должен следовать сразу за левой фигурной скобкой, чтобы ввести косвенность.

В каждом из приведенных ниже случаев *word* требуется расширение тильды, расширение параметров, замена команд и арифметическое расширение.

Если расширение подстроки не выполняется, используя форму, описанную ниже (например, ‘:-’), Bash проверяет параметр, который не установлен или равен нулю. Пропуск двоеточия приводит к тестированию только для параметра, который не задан. Другими словами, если включено двоеточие, оператор проверяет как существование

${*parameter*:-*word*}

Если *parameter* значение не задано или равно нулю, заменяется расширение *word*. В противном случае подставляется значение *parameter*.

```
$ v= 123
$ echo ${v-unset}
123
```

${*parameter*:=*word*}

Если *parameter* значение не задано или равно нулю, то расширение *word* присваивается *parameter*. Затем подставляется значение *parameter*. Позиционные параметры и специальные параметры не могут быть назначены таким образом.

```
$ var=
$ : ${var:=ПО УМОЛЧАНИЮ}
$ echo $var
По умолчанию
```

${*parameter*:?*word*}

Если *parameter* значение равно нулю или не задано, расширение *word* (или соответствующее сообщение, если *word* его нет) записывается в стандартную ошибку, и оболочка, если она не интерактивна, завершает работу. В противном случае подставляется значение *parameter*.

```
$ var=
$ : ${var:?значение var не задано или равно нулю}
bash: var: значение var не задано или равно нулю
```

${*parameter*:+*word*}

Если *parameter* равно нулю или не задано, ничего не заменяется, в противном случае заменяется расширение *word*.

```
$ var=123
$ echo ${var:+значение var задано, а не равно null}
значение var установлено, а не равно null
```

${*parameter*:*offset*}

${*parameter*:*offset*:*length*}

Это называется расширением подстроки. Оно расширяется до *length* символов со значением *parameter*, начинающимся с символа, указанного *offset*. Если *parameter*

length это значение опущено, оно расширяется до подстроки значения *parameter*, начинающейся с символа, указанного *offset*, и продолжающейся до конца значения. *length* и *offset* являются арифметическими выражениями (см. арифметику оболочки).

Если *offset* значение меньше нуля, значение используется как смещение в символах от конца значения *parameter*. Если *length* значение меньше нуля, это интерпретируется как смещение в символах от конца значения *parameter*, а не как количество символов, и расширение представляет собой символы между *offset* и этим результатом. Обратите внимание, что отрицательное смещение должно быть отделено от двоеточия по крайней мере одним пробелом, чтобы избежать путаницы с расширением ‘:-’.

Вот несколько примеров, иллюстрирующих расширение подстрок в параметрах и массивах с подпиской:

```
$ string=01234567890abcdefgh
$ echo ${строка:7}
7890abcdefgh
$ echo ${строка:7:0}

$ echo ${строка:7:2}
78
$ echo ${строка:7:-2}
7890abcdef
$ echo ${строка: -7}
bcdefgh
$ echo ${строка: -7:0}

$ echo ${строка: -7:2}
bc
$ echo ${строка: -7:-2}
bcdef
$ set -- 01234567890abcdefgh
$ echo ${1:7}
7890abcdefgh
$ echo ${1:7:0}

$ echo ${1:7:2}
78
$ echo ${1:7:-2}
7890abcdef
$ echo ${1: -7}
bcdefgh
$ echo ${1: -7:0}

$ echo ${1: -7:2}
bc
$ echo ${1: -7:-2}
```

```
7890abcdefgh
$ echo ${массив[0]:7:0}

$ echo ${массив[0]:7:2}
78
$ echo ${массив[0]:7:-2}
7890abcdef
$ echo ${массив[0]: -7}
bcdefgh
$ echo ${массив[0]: -7:0}

$ echo ${массив[0]: -7:2}
bc
$ echo ${массив[0]: -7:-2}
bcdef
```

Если *parameter* равно ‘@’ или ‘*’, результатом будут *length* позиционные параметры, начинающиеся с *offset*. Принимается отрицательное значение *offset* относительно на единицу большего, чем наибольший позиционный параметр, поэтому смещение на -1 соответствует последнему позиционному параметру. Это ошибка расширения, если *length* значение меньше нуля.

Следующие примеры иллюстрируют расширение подстроки с использованием позиционных параметров:

```
$ set -- 1 2 3 4 5 6 7 8 9 0 a b c d e f g h
$ echo ${@:7}
7 8 9 0 a b c d e f g h
$ echo ${@:7:0}

$ echo ${@:7:2}
7 8
$ echo ${@:7:-2}
bash: -2: выражение подстроки < 0
$ echo ${@: -7:2}
b c
$ echo ${@:0}
./bash 1 2 3 4 5 6 7 8 9 0 a b c d e f g h
$ echo ${@:0:2}
./bash 1
$ echo ${@: -7:0}
```

Если *parameter* имя индексированного массива подписано на ‘@’ или ‘*’, результатом будут *length* элементы массива, начинающиеся с `${parameter[offset]}`. Принимается отрицательное значение *offset* относительно единицы, превышающей максимальный индекс указанного массива. Это ошибка расширения, если *length* значение меньше нуля.

```
$ array=(0 1 2 3 4 5 6 7 8 9 0 a b c d e f g h)
$ echo ${массив[@]:7}
7 8 9 0 a b c d e f g h
$ echo ${array[@]:7:2}
7 8
$ echo ${array[@]: -7:2}
b c
$ echo ${array[@]: -7:-2}
bash: -2: выражение подстроки < 0
$ echo ${array[@]:0}
0 1 2 3 4 5 6 7 8 9 0 a b c d e f g h
$ echo ${array[@]:0:2}
0 1
$ echo ${array[@]: -7:0}
```

Расширение подстроки, примененное к ассоциативному массиву, приводит к неопределенным результатам.

Индексация подстрок основана на нуле, если не используются позиционные параметры, и в этом случае индексация начинается с 1 по умолчанию. Если *offset* равно 0 и используются позиционные параметры, `$0` добавляется префикс к списку.

${!*prefix}**

${!*prefix*@}

Расширяется до имен переменных, имена которых начинаются с *prefix*, разделенных первым символом `IFS` специальной переменной. Когда используется ‘@’ и расширение заключено в двойные кавычки, имя каждой переменной расширяется до отдельного слова.

${!*name*[@]}

${!*name*[*]}

Если *name* является переменной массива, расширяется до списка индексов (ключей) массива, назначенных в *name*. Если *name* не является массивом, расширяется до 0, если *name* задано, и null в противном случае. Когда используется ‘@’ и расширение заключено в двойные кавычки, каждая клавиша расширяется до отдельного слова.

${#*parameter*}

Подставляется длина в символах расширенного значения *parameter*. Если *parameter* равно ‘*’ или ‘@’, подставляемое значение представляет собой количество позиционных параметров. Если *parameter* это имя массива, подписанное ‘*’ или ‘@’, подставляемое значение представляет собой количество

большей максимального индекса *parameter*, поэтому отрицательные индексы отсчитываются от конца массива, а индекс, равный -1, ссылается на последний элемент.

${*parameter*#*word*}

${*parameter*##*word*}

word Расширяется для создания шаблона и подбирается в соответствии с правилами, описанными ниже (см. Сопоставление с шаблоном). Если шаблон совпадает с началом расширенного значения *parameter*, то результатом расширения является расширенное значение *parameter* с удалением самого короткого совпадающего шаблона (регистр '#') или самого длинного совпадающего шаблона (регистр '##'). Если *parameter* равно '@' или '*', операция удаления шаблона применяется к каждому позиционному параметру по очереди, и расширение является результирующим списком. Если *parameter* это переменная массива, подписанная '@' или '*', операция удаления шаблона применяется к каждому элементу массива по очереди, и расширение является результирующим списком.

${*parameter*%*word*}

${*parameter*%%*word*}

word Расширяется для создания шаблона и подбирается в соответствии с правилами, описанными ниже (см. Сопоставление с шаблоном). Если шаблон соответствует конечной части расширенного значения *parameter*, то результатом расширения является значение *parameter* с удаленным самым коротким совпадающим шаблоном (регистр '%') или самым длинным совпадающим шаблоном (регистр '%%'). Если *parameter* равно '@' или '*', операция удаления шаблона применяется к каждому позиционному параметру по очереди, и расширение является результирующим списком. Если *parameter* это переменная массива, подписанная '@' или '*', операция удаления шаблона применяется к каждому элементу массива по очереди, и расширение является результирующим списком.

${*parameter*/*pattern*/*string*}

${*parameter*//*pattern*/*string*}

${*parameter*/#*pattern*/*string*}

${*parameter*/%*pattern*/*string*}

pattern Расширяется для создания шаблона точно так же, как при расширении имени файла. *Parameter* расширяется, и самое длинное совпадение *pattern* с его значением заменяется на *string*. *string* выполняется расширение тильды, расширение параметров и переменных, арифметическое расширение, замена команд и процессов и удаление кавычек. Сопоставление выполняется в соответствии с правилами, описанными ниже (см. Сопоставление с шаблоном).

совпадения *pattern* заменяются на *string*. Если *pattern* ему предшествует '#' (третья форма выше), оно должно совпадать с началом расширенного значения *parameter*. Если *pattern* перед ним стоит '%' (четвертая форма выше), оно должно совпадать с развернутым значением *parameter*. Если расширение *string* равно нулю, совпадения с *pattern* удаляются. Если *string* равно null, совпадения с *pattern* удаляются, а '/' следующий *pattern* символ может быть опущен.

Если `patsub_replacement` опция оболочки включена с помощью `shopt`, любые экземпляры '&' без кавычек в *string* заменяются соответствующей частью *pattern*. Это предназначено для дублирования распространенной `sed` идиомы.

Цитирование любой части *string* запрещает замену при расширении указанной части, включая строки замены, хранящиеся в переменных оболочки. Обратная косая черта будет экранировать '&' в *string*; обратная косая черта удалена, чтобы разрешить использование литерала '&' в строке замены. Пользователи должны быть осторожны, если *string* заключен в двойные кавычки, чтобы избежать нежелательного взаимодействия между обратной косой чертой и двойными кавычками, поскольку обратная косая черта имеет особое значение в двойных кавычках. Замена шаблоном выполняет проверку на '&' без кавычек после расширения *string*, поэтому пользователи должны убедиться, что они правильно указывают любые вхождения '&', которые они хотят понимать буквально при замене, и убедиться, что все экземпляры '&', которые они хотят заменить, не заключены в кавычки.

Например,

```
var=abcdef
rep='& '
echo ${var/abc/& }
echo "${var/abc/& }"
echo ${var/ abc/$ rep}
echo "${var/abc/$rep}"
```

отобразит четыре строки "abc def", в то время как

```
var=abcdef
rep='& '
echo ${var/abc/\& }
echo "${var/abc/\& }"
echo ${var/abc/"& "}
echo ${var/abc/"$ rep"}
```

отобразит четыре строки "& def". Подобно операторам удаления шаблона, двойные кавычки, окружающие строку замены, заключают в себя расширенные символы, в

внимание никакие заключающие двоиные кавычки.

Поскольку обратная косая черта может экранировать ‘&’, она также может экранировать обратную косую черту в строке замены. Это означает, что ‘\\’ вставит буквальную обратную косую черту в замену, поэтому эти две `echo` команды

```
var=abcdef
rep='\\&xyz'
echo ${var/abc/\\&xyz}
echo ${var/ abc/$ rep}
```

оба будут выводить ‘\abcxyzdef’.

Редко возникает необходимость заключать только *string* в двойные кавычки.

Если включена `nocasematch` опция оболочки (см. Описание `shopt` во встроенной программе Shopt), сопоставление выполняется без учета регистра буквенных символов. Если *parameter* равно ‘@’ или ‘*’, операция подстановки применяется к каждому позиционному параметру по очереди, и расширение является результирующим списком. Если *parameter* это переменная массива, подписанная ‘@’ или ‘*’, операция подстановки применяется к каждому элементу массива по очереди, и расширение является результирующим списком.

${*parameter*^*pattern*}

${*parameter*^^*pattern*}

${*parameter*,*pattern*}

${*parameter*,,*pattern*}

Это расширение изменяет регистр буквенных символов в *parameter*. *pattern* Расширяется для создания шаблона точно так же, как при расширении имени файла. Каждый символ в расширенном значении *parameter* проверяется на соответствие *pattern*, и, если он соответствует шаблону, преобразуется его регистр. Шаблон не должен пытаться сопоставить более одного символа.

Оператор ‘^’ преобразует совпадающие строчные буквы *pattern* в прописные; оператор ‘,’ преобразует совпадающие заглавные буквы в строчные. Расширения ‘^^’ и ‘,,’ преобразуют каждый совпадающий символ в расширенном значении; расширения ‘^’ и ‘,’ совпадают и преобразуют только первый символ в расширенном значении. Если *pattern* параметр опущен, он обрабатывается как ‘?’, который соответствует каждому символу.

Если *parameter* равно ‘@’ или ‘*’, операция изменения регистра применяется к каждому позиционному параметру по очереди, и расширение является результирующим списком. Если *parameter* это переменная массива, подписанная

${*parameter*@*operator*}

Расширение представляет собой либо преобразование значения *parameter*, либо информацию о *parameter* самом себе, в зависимости от значения *operator*. Каждый *operator* представляет собой одну букву:

U

Расширение представляет собой строку, представляющую собой значение *parameter* со строчными буквенными символами, преобразованными в верхний регистр.

u

Расширение представляет собой строку, представляющую собой значение *parameter* с первым символом, преобразованным в верхний регистр, если он буквенный.

L

Расширение представляет собой строку, представляющую собой значение *parameter*, с заглавными буквенными символами, преобразованными в строчные.

Q

Расширение представляет собой строку, представляющую собой значение *parameter*, заключенное в кавычки в формате, который может быть повторно использован в качестве входных данных.

E

Расширение представляет собой строку, которая является значением *parameter* с расширенными escape-последовательностями обратной косой черты, как с помощью `$'…'` механизма цитирования.

P

Расширение представляет собой строку, которая является результатом расширения значения *parameter*, как если бы это была строка приглашения (см. раздел Управление приглашением).

A

Расширение представляет собой строку в форме оператора присваивания или `declare` команды, которая, в случае вычисления, будет воссоздана *parameter* со своими атрибутами и значением.

K

ассоциативных массивов в виде последовательности пар ключ-значение в кавычках (см. Массивы).

a

Расширение представляет собой строку, состоящую из значений флагов, представляющих *parameter* атрибуты.

k

Аналогично преобразованию ‘K’, но расширяет ключи и значения индексированных и ассоциативных массивов до отдельных слов после разделения слов.

Если *parameter* равно ‘@’ или ‘*’, операция применяется к каждому позиционному параметру по очереди, и расширение является результирующим списком. Если *parameter* это переменная массива, подписанная ‘@’ или ‘*’, операция применяется к каждому элементу массива по очереди, и расширение является результирующим списком.

Результат расширения зависит от разделения слов и расширения имени файла, как описано ниже.

Далее: Замена команд, Предыдущее: Расширение тильды, Вверх: Расширения оболочки [Содержание] [Указатель]