Глобализация файлов

Глобирование файлов может звучать как необычный термин, но концепция глобализации, вероятно, знакома. Подобно джокеру в колоде карт, которую вы можете использовать в качестве любой карты («подстановочный знак»), символы глобуса - это специальные символы, которые можно использовать для сопоставления имен файлов или путей.

Для получения подробной информации о том, как работает globbing, вы можете просмотреть справочную страницу по **glob** в разделе 7 руководства, выполнив:

```
man 7 glob
```

С этой страницы руководства:

Строка является шаблоном подстановки, если она содержит один из символов «?», «*» Или «[». Глобирование - это операция, которая расширяет шаблон подстановки в список имен путей, соответствующих шаблону.

Оболочка - это то, что выполняет расширение подстановочных знаков, поэтому использование глобирования не ограничивается конкретной командой. Глобализация особенно полезна для управления файлами, поскольку ее можно использовать для сопоставления путей к файлам, которыми вы хотите управлять.

*Символ **

Подстановочный знак * можно использовать для сопоставления с любой строкой, включая пустую строку.

```
echo *
```

Команда **echo** используется для просмотра файлов, которые будут соответствовать шаблону glob. В этом случае отображается имя каждого файла в текущем каталоге пользователя.

Пустая строка - это строка, в которой ничего нет. Это обычно упоминается как "".

Чтобы ограничить подстановочный знак *, чтобы он соответствовал меньшему количеству имен файлов, его можно комбинировать с другими символами. Например, шаблон **D*** будет соответствовать только именам файлов, которые начинаются с символа **D**:

```
echo D*
```

Символ * может появиться в любом месте шаблона; это может даже появляться несколько раз. Можно найти файл, который не только начинается с буквы D, но также содержит букву n, используя шаблон D* n*:

```
echo D*n*
```

Символ ?

? символ в строке будет соответствовать ровно одному символу. Например, чтобы увидеть все файлы в каталоге **/usr/bin**, которые имеют только один символ в имени файла, используйте шаблон **/usr/bin /?:**

```
cd /usr/bin
echo ?
```

Добавляя больше символов ? можно сопоставить с дополнительными символами. Чтобы увидеть, какие файлы имеют два или три символа в качестве имен файлов, выполните следующие две команды echo:

```
echo ??
echo ???
```

Включение других символов в шаблоны с помощью символа ?, включая другие символы подстановки, позволяют осуществлять более точный поиск. Например, `w??` будет соответствовать файлам, которые начинаются с буквы w и имеют длину ровно три символа:

```
echo w??
```

Хотя **w??*** будет соответствовать файлам, которые начинаются с буквы w, но имеют длину не менее трех символов:

```
echo w??*
```

Символы [и]

Используя [и] (квадратные скобки) можно заключить набор символов, который будет использоваться точно для соответствия одному символу. Вы можете думать о квадратных скобках

как о подстановочном знаке **?**, но ограниченного символами, которые указаны внутри квадратных скобок.

Например, используя шаблон **[abcd]??** будет соответствовать именам файлов, которые начинаются с символов a, b, c или d и имеют длину три символа. В качестве альтернативы, буквы, которые являются последовательными, также могут быть выражены в виде диапазона, например **[a-d]??:**

```
cd /usr/bin
echo [a-d]??
```

```
root@L-FW:~# cd /usr/bin/
root@L-FW:/usr/bin# echo [a-d]??
apt awk cal cmp col cut
```

Когда символ - используется в квадратных скобках, он называется диапазоном. Например: [a-z] будет соответствовать любому отдельному символу, который находится в диапазоне символов от a до z.

Этот диапазон определяется текстовой таблицей ASCII. Просто используйте символ более низкого значения из таблицы ASCII в качестве первого символа и более высокое значение в качестве второго.

Чтобы увидеть, какие значения имеют символы в текстовой таблице ASCII, либо выполните поиск «текстовой таблицы ASCII» в Интернете, либо просмотрите таблицу с помощью команды ascii.

Команда **touch** в следующем примере используется для создания файлов, которые будут сопоставлены. Одиночные кавычки использовались при указании имени файла, создаваемого с помощью команды **touch**, потому что внутри одинарных кавычек глобусные символы теряют свое особое значение для оболочки.

```
cd ~
touch '*' '**' '***' '?' '??' '???' '[][][]'
ls
```

Помещенные в квадратные скобки символы подстановки теряют значение подстановки и совпадают только с собой. Таким образом, шаблон, подобный **[*]***, будет соответствовать имени файла, которое начинается с символа ***. Шаблон **[?]*** Будет соответствовать имени файла, которое начинается с **?**.

```
echo [*]*
```

Даже использование квадратных скобок внутри квадратных скобок будет соответствовать квадратным скобкам, поэтому [[]] * будет соответствовать имени файла, которое начинается с квадратных скобок.

Обычно требуется избегать использования специальных символов, таких как ***,?, [И]** в именах файлов, потому что эти символы, когда они используются в имени файла в командной строке, могут непреднамеренно совпадать с другими именами файлов.

Как и в случае с другими символами подстановки, квадратные скобки могут появляться в шаблоне несколько раз. Например, чтобы сопоставить файл в каталоге /usr/bin, который содержит как минимум две цифры в имени файла, используйте шаблон **/etc/*[0-9][0-9]*:**

```
cd /usr/bin
echo *[0-9][0-9]*
```

```
root@L-FW:/usr/bin# echo *[0-9][0-9]*
base32 base64 cpan5.24-x86_64-linux-gnu i386 linux32
a224sum sha256sum sha384sum sha512sum x86_64
```

Квадратные скобки также поддерживают отрицание набора символов (также называемое дополнением). Если первый символ внутри квадратных скобок либо **!** или ^, тогда этот первый символ имеет особое значение, а не следующие символы, то есть соответствует одному символу, который не входит в набор символов, следующих за ним.

Другими словами, шаблон **[!a-v]*** будет соответствовать имени файла, которое **не начинается** с буквы от **a до v**:

```
echo [!a-v]*
```

```
root@L-FW:/usr/bin# echo [!a-v]*
[ 2to3 2to3-2.7 2to3-3.5 w wall watch watchgnupg wc
procps write x86_64 xargs xauth xsubpp xxd xz xzcat
```