Шаблоны и сопоставление шаблонов

Шаблон - это **строковое описание**. Bash использует их по-разному:

- Расширение имени пути (сопоставление с глобальными именами файлов)
- Сопоставление шаблонов в условных выражениях
- Удаление подстроки и поиск и замена в Расширение параметров
- Ветвление на основе шаблонов с помощью команды case

Язык описания шаблонов относительно прост. Любой символ, который не упоминается ниже, соответствует самому себе. `NUL` Символ может не встречаться в шаблоне. Если специальные символы заключены в кавычки, они сопоставляются буквально, то есть без их особого значения.

Не путайте шаблоны с ***регулярными выражениями***, потому что они разделяют некоторые символы и выполняют аналогичную работу по сопоставлению.

Обычный язык шаблонов

Последовательность	Описание
`*`	Соответствует **любой строке**, включая нулевую строку (пустую строку)
`?`	Соответствует любому **отдельному символу**
`X`	Соответствует символу `X` , который может быть любым символом, не имеющим особого значения
`\X`	Соответствует символу `X` , где особое значение символа удаляется обратной косой чертой
`\\`	Соответствует обратной косой черте
`[…]`	Определяет **выражение в скобках** шаблона (см. Ниже). Соответствует любому из вложенных символов в этой позиции.

Выражения в скобках

Выражение в скобках `[…]` , упомянутое выше, имеет несколько полезных применений:

Выражение в

Выражение в скобках	Описание
Выражение в скобках	Описание
`[XYZ]`	"Нормальное" выражение в скобках, соответствующее либо `X`, `Y` либ
`[X-Z]`	Выражение диапазона: сопоставление всех символов от `X` до `Y` (ваш определяет, как **сортируются** символы!)
`[[:class:]]`	Соответствует всем символам, определенным классом символов PO (https://pubs.opengroup.org/onlinepubs/009696899/basedefs/xbd_chap(`alnum`, `alpha`, `ascii`, `blank`, `cntrl`, `digit`, `graph`, `lower`, p `space`, `upper`, `word` и `xdigit`
`[^…]`	Отрицающее выражение: оно соответствует всем символам, которы скобках
`[!…]`	Эквивалентно `[^…]`
`[]...]` или `[-…]`	Используется для включения символов `]` и `-` в набор, они должны б символами после открывающей скобки
`[=C=]`	Соответствует любому символу, эквивалентному весу сопоставления
`[[.SYMBOL.]]`	Соответствует символу сопоставления `SYMBOL`

Примеры

Несколько простых примеров с использованием обычного сопоставления с образцом:

- `"Hello world"` Совпадения шаблонов
  - `Hello world`
- `[Hh]"ello world"` Совпадения шаблонов
  - ⇒ `Hello world`
  - ⇒ `hello world`
- `Hello*` Совпадения шаблонов (например)
  - ⇒ `Hello world`
  - ⇒ `Helloworld`
  - ⇒ `HelloWoRlD`
  - ⇒ `Hello`
- `Hello world[[:punct:]]` Совпадения шаблонов (например)
  - ⇒ `Hello world!`
  - ⇒ `Hello world.`
  - ⇒ `Hello world+`
  - ⇒ `Hello world?`
- `[[.backslash.]]Hello[[.vertical-line.]]world[[.exclamation-mark.]]` Совпадение шаблонов (с использованием символов сопоставления (https://pubs.opengroup.org/onlinepubs/009696899/basedefs/xbd_chap07.html#tag_
  - ⇒ `\Hello|world!`

Расширенный язык шаблонов

Если вы установите параметр оболочки `extglob` , Bash понимает некоторые мощные шаблоны. А `<PATTERN-LIST>` - это один или несколько шаблонов, разделенных символом канала (`PATTERN|PATTERN`).

`?(<PATTERN-LIST>)`	Соответствует **нулю или одному** появлению заданных шаблонов
`*(<PATTERN-LIST>)`	Соответствует **нулю или более** вхождений заданных шаблонов
`+(<PATTERN-LIST>)`	Соответствует **одному или нескольким** вхождениям заданных шаблонов
`@(<PATTERN-LIST>)`	Соответствует **одному** из заданных шаблонов
`!(<PATTERN-LIST>)`	Соответствует чему угодно, **кроме** одного из заданных шаблонов

Примеры

Удалить все, кроме одного конкретного файла

```
рм -ф ! (survivior.txt )
```

Конфигурация сопоставления с образцом

Связанные параметры оболочки

опция	классификация	Описание
`dotglob`	глобализация	см. раздел Настройка расширения имени пути
`extglob`	глобальный	включить / отключить расширенный язык сопоставления с образцами, как описано выше
`failglob`	глобализация	см. раздел Настройка расширения имени пути
`nocaseglob`	глобализация	см. раздел Настройка расширения имени пути

опция	классификация	Описание
`nocasematch`	сопоставление шаблонов и строк	выполняйте сопоставление с образцом без учета регистра отдельных букв
`nullglob`	глобализация	см. раздел Настройка расширения имени пути
`globasciiranges`	глобализация	см. раздел Настройка расширения имени пути

Проблемы с ошибками и переносимостью

* Вопреки интуиции, `[!chars]` POSIX определяет только синтаксис для отрицания символьного класса для сопоставления с шаблоном оболочки. `[^chars]` это просто широко поддерживаемое расширение. Даже dash поддерживает `[^chars]` , но не шикарно.

* Все кванторы extglob, поддерживаемые bash, поддерживались ksh88. Набор кванторов extglob, поддерживаемых ksh88, идентичен тем, которые поддерживаются Bash, mksh, ksh93 и zsh.

* mksh не поддерживает классы символов POSIX. Таким образом, такие диапазоны символов, как `[0-9]` , несколько более переносимы, чем эквивалентный класс POSIX, например `[:digit:]` .

* Bash использует пользовательский интерпретатор среды выполнения для сопоставления с образцом. (по крайней мере) ksh93 и zsh переводят шаблоны в регулярные выражения, а затем используют компилятор регулярных выражений для генерации и кэширования оптимизированного кода сопоставления с образцом. Это означает, что Bash может быть на порядок или более медленнее в случаях, связанных со сложным обратным отслеживанием (обычно это означает вложение квантора extglob). Возможно, вы захотите использовать поддержку регулярных выражений Bash (`=~` оператор), если производительность является проблемой, потому что Bash будет использовать вашу реализацию регулярных выражений библиотеки C, а не свой собственный сопоставитель шаблонов.

ЗАДАЧА: описать ошибку, связанную с выходом из шаблона https://gist.github.com/ormaaj/6195070 (https://gist.github.com/ormaaj/6195070)

ksh93 дополнительно

ksh93 поддерживает некоторые очень мощные функции сопоставления с образцом в дополнение к описанным выше.

* ksh93 поддерживает произвольные кванторы, как и ERE, используя `{from,to}(pattern-list)` синтаксис. `{2,4}(foo)bar` соответствует 2-4 буквам "foo", за которыми следует "bar". `{2,}(foo)bar` соответствует 2 или более буквам "foo", за

которыми следует "bar". Вероятно, вы можете разобраться с остальным. Пока ни одна из других оболочек не поддерживает этот синтаксис.

* В ksh93 a `pattern-list` может быть разделено либо `&` или `|`. `&` означает "все шаблоны должны быть сопоставлены" вместо "любой шаблон". Например,

```
[[ fo0bar == @(fo[0-9]&+([[:alnum:]])) bar ]]
```

было бы верно, если

```
[[ f00bar == @(fo[0-9]&+([[:alnum:]])) bar ]]
```

является ложным, потому что все члены и-списка должны быть удовлетворены. Пока ни одна другая оболочка не поддерживает это, но вы можете имитировать некоторые случаи в других оболочках, используя двойное отрицание extglob. Вышеупомянутый шаблон ksh93 эквивалентен в Bash:

```
[[ fo0bar == !(!(fo[0-9])|!(+([[: alnum:]])))бар ]]
```

, который технически более переносим, но уродлив.

* встроенная функция printf в ksh93 может переводить шаблоны оболочки в ERE и обратно, используя спецификаторы `%R` и `%P` format соответственно.

ЗАДАЧА: `~()` (и регулярное `.sh.match` выражение), обратные ссылки, особое `${var/…/…}` поведение, `%()`

Обсуждение

Амит Верма, 2010/09/16 12:30 ()

Что, если мы хотим сопоставить определенный шаблон n раз??

Ян Шампера, 2010/09/17 04:47 ()

Это невозможно с шаблонами (или расширенными шаблонами). Подобные шаблоны не являются заменой регулярных выражений (но на самом деле обычно шаблоны могут выполнять самую "обычную" работу, иногда люди (ab) используют регулярные выражения для очень простых задач в кодировании оболочки). Возможно, в будущем в Bash появится новая функция, подобная этой, для расширенных шаблонов.

В зависимости от ваших потребностей вы можете либо использовать какую-либо внешнюю программу (grep, awk, sed, ...), либо условное выражение с `=~` оператором. Расширение имени пути с помощью регулярных выражений невозможно.

Тим Джонс, 2012/07/11 08:32 ()

Почему в bash должен быть включен extglob, чтобы включить эти глобальные "расширения", если они были доступны в ksh десятилетиями, по умолчанию?

Это вызывает много проблем, поскольку скрипты не могут быть совместимы как с ksh, так и с bash: (

Ян Шампера, 07.07.2012 / 12 21:45 ()

Я думаю, что это вопрос, который вы должны задать сопровождающему. Я не думаю, что есть способ сделать разные оболочки совместимыми на 100%, даже в теории.

Andi, 2015/09/11 17:53 ()

Что, если мы хотим сопоставить +? Это должно быть экранировано, как /+ , правильно?

syntax/pattern.txt Последнее изменение: 2021/10/21 00:52 автор fgrouse