Лекция 4. Регулярные выражения

Александр Смаль

CS центр
11 марта 2013
Санкт-Петербург

Регулярные выражения

- Регулярные выражения (Regular Expressions, RegExp) — это формальный язык поиска и осуществления манипуляций с подстроками в тексте.
- Более простые аналоги — wildcards, файловые маски.
- Регулярные выражения реализованы в большинстве языков программирования.
- Различные реализации отличаются в деталях, но принципы создания регулярных выражений везде одинаковы.
- Две основные задачи: поиск и замена.

BRE

- Синтаксис BRE (basic regular expressions) на данный момент определён POSIX как устаревший.
- BRE широко распространён из соображений обратной совместимости.
- Многие UNIX-утилиты используют такие регулярные выражения по умолчанию.

ERE

- Синтаксис ERE (Extended Regular Expressions) определён в POSIX.
- Отличия от BRE:
  1. Отменено использование обратной косой черты для метасимволов `‘{}’` и `‘()’`.
  2. Обратная косая черта перед метасимволом отменяет его специальное значение.
  3. Добавлены метасимволы `‘+’`, `‘?’`, `‘|’`.
  4. Возможность использования символьных классов POSIX.

Основные операторы

- Логическое “или” ‘|’ — разделяет допустимые варианты:
  `black|block`
- Группировка ‘()’ — определяет область действия и приоритет операторов.
  `b(lo|a)ck`
- Произвольный символ ‘.’.
  `bl.ck`
- Список символов ‘[]’.
  `bl[ao]ck`
- При замене можно ссылаться на группы:
  `s/(black|white) (cat|dog)s/\2s are \1/`

Квантификация

- Замыкание Клини ‘*’ — любое количество повторений (в т.ч. 0).
  `noo*`
- “Плюс” ‘+’ — хотя бы одно повторение.
  `no+`
- “Вопрос” ‘?’ — не более одного повторения.
  `bl?ack`
- Явное указание количества ‘{}’.
  `wazzu{1,7}p  no{4,}`

Продвинутые операторы

- Список запрещённых символов ‘[^]’.
  `bl[^io]ck`
- Интервалы символов [-].
  `[A-Z][a-z]+ [1-9][0-9]*`
- Якоря: начало строки ^ и конец строки $.
  `^[A-Z][a-z]+ [1-9][0-9]*$`
- Классы символов POSIX: [:alnum:], [:cntrl:], [:lower:], [:space:], [:alpha:], [:digit:], [:print:], [:upper:], [:blank:], [:graph:], [:punct:], [:xdigit:]
  `[[:alpha:]]+ [[:digit:]]+`

PCRE

- Perl compatible regular expressions.
- Поставляется в виде отдельной библиотеки.
- Добавлено множество специальных символов и модификаторов.
  - Короткие синонимы для классов символов: \w, \W — буква, всё кроме букв; \d, \D — цифра, всё кроме цифр.
  - Атомарные группировки '(?:)'.
    b(?:lo|a)ck
  - Ограничители слов: \b
  - Модификаторы: i регистронезависимый поиск, m многострочный режим, s символ '.' совпадает и с переносом строки, A привязка к началу текста, E привязка к концу текста, U инвертирует "жадность".

Жадность квантификаторов

- Квантификаторы могут иметь один из трёх типов жадности:
  - жадный — квантификатор захватывает как можно больше;
  - ленивый — квантификатор захватывает как можно меньше;
  - сверхжадный (ревнивый) — квантификатор захватывает как можно больше и не отдаёт назад.
- Обозначение квантификаторов разной жадности:

Жадный	Ленивый	Сверхжадный
`*`	`*?`	`*+`
`?`	`??`	`?+`
`+`	`+?`	`++`
`{n,}`	`{n,}?`	`{n,}+`

grep

- Название представляет собой акроним английской фразы «search Globally for lines matching the Regular Expression, and Print them».
- Есть несколько синонимов: `egrep = grep -E` (ERE), `fgrep = grep -F`, `rgrep = grep -r`.
- `grep -P` использует PCRE.
- Пример использования:

```
$ grep -E '^[bcBC]at' heroes.txt
$ rgrep apache /etc
```

Sed

1. Потоковый редактор вроде ed.
2. Обрабатывает текст в один прогон.
3. Команды редактирования:
   `[ addr [ , addr ] ] cmd [ args ]`
4. Примеры:
   ```
   $ sed -e '/secret/d' text.txt
   $ sed -e 's/<[^>]*>//g;/^\s*$/d' index.html
   ```

Простые примеры:

- Слово с дефисами:
  `^[a-z0-9]+(-[a-z0-9]+)*$`
- Имя пользователя:
  `^[a-z0-9_-]{3,16}$`
- Пароль:
  `^[a-z0-9_!@$%^&*()+=-]{6,18}$`
- XML тег:
  `<([a-z]+)([^>]+)*(?:>(.*)<\/\1>|\s+\/>)`
- Email:
  `^([a-z0-9_\.-]+)@([a-z0-9_\.-]+)\.([a-z\.]{2,6})$`

Не такие уж простые...

Не такие уж простые...

1. Выражение для email соответствующее RFC 822.

```
(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t]
)+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:
                                    [78 строк]
|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*\>(?:(
?:\r\n)?[ \t])*))*)?;\s*)
```

Не такие уж простые...

1. Выражение для email соответствующее RFC 822.

```
(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t]
)+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:
                                    [78 строк]
|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*\>(?:(
?:\r\n)?[ \t])*))*)?;\s*)
```

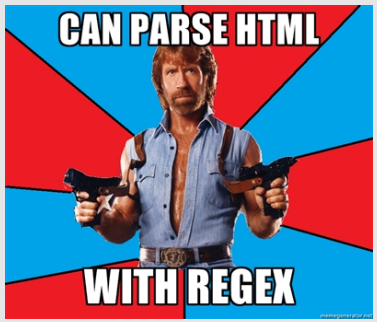
2. Регулярные выражения **не предназначены** для разбора сложноструктурированных текстов (вроде HTML и XML).

Не такие уж простые...

1. Выражение для email соответствующее RFC 822.

```
(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t]
)+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:
                                    [78 строк]
|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*\>(?:(
?:\r\n)?[ \t])*))*)?;\s*)
```

2. Регулярные выражения **не предназначены** для разбора сложноструктурированных текстов (вроде HTML и XML).



Спасибо за внимание!