Лабораторная работа
Базовые регулярные выражения. Редактор sed.

1 Регулярные выражения

Регулярные выражения предоставляют механизм для выбора определенных строк из множества заданных. Регулярные выражения имеют контекстнонезависимый синтаксис, позволяющий им обрабатывать различные наборы символов с различными правилами сортировки опираясь на текущие установки локали. Большинство приложений поддерживает базовые регулярные выражения (BRE). Некоторые приложения поддерживают расширенные регулярные выражения (ERE). Далее в этом разделе будут использоваться следующие термины:
Полное регулярное выражение — объединенное множество из одного или более BRE (ERE) определяющее шаблон для выбора строк.
Соответствие. Последовательность из нуля или более символов называется соответствующей BRE или ERE, если символы в последовательности соответствуют тем, которые указаны в шаблоне.

2 Базовые регулярные выражения

2.1 BRE соответствующие единственному символу или сопоставимому элементу

В BRE обычный символ, специальный символ с предшествующей обратной чертой или точка соответствуют одному символу. Выражение в квадратных скобках соответствует единственному символу или сопоставимому элементу. Обычный символ в BRE соответствует сам себе. Это любой символ из используемого кодового набора, за исключением специальных символов. Специальные символы имеют особые свойства в некотором контексте. Вне этого контекста, либо предваренные обратной чертой, специальные символы соответствуют сами себе. Специальные символы BRE и контексты в которых они имеют специальное значение следующие:

`.[\` специальные, кроме случаев когда используются в выражениях с квадратными скобками.

`*` специальный, за исключением использования в квадратных скобках, в качестве первого символа полного BRE (возможно после `^`), первым символом подвыражения (возможно после `^`).

`^` специальный когда является первым символом полного BRE или выражения в квадратных скобках.

`$` специальный когда является первым символом полного BRE.

Точка, когда используется вне выражения в квадратных скобках, соответствует любому символу из поддерживаемого набора кроме символа с кодом ноль.

2.2 Выражения в квадратных скобках

Выражение в квадратных скобках соответствует единственному сопоставимому элементу содержащемуся в непустом множестве сопоставимых элементов определенном выражением внутри скобок. К выражениям в квадратных скобках применимы следующие правила и определения:

1. Выражение в квадратных скобках может быть либо списком соответствия, либо списком несоответствия. Они состоят из одного или более выражений: подходящие элементы, подходящие символы, классы эквивалентности, классы символов или выражения диапазонов. Специальные символы: `.` `*` `[` `\` теряют свое специальное значение внутри скобок.

Последовательности символов `[.` `[=` `[:` имеют специальное значение внутри скобок и используются для отделения подходящих символов, выражений классов эквивалентности, выражений классов символов. За такими последовательностями символов должны следовать правильные выражения, заканчивающиеся, соответственно, символами `.]` `=]` `:]` .

2. Список соответствия определяет список соответствующий любому имеющемуся в нем выражению. Напр., `[abc]` соответствует любому из символов a, b, c.

3. Список несоответствия начинается с уголка (`^`), и определяет список соответствующий любому символу или подходящему элементу, исключая выражения содержащиеся в списке. Напр., `[^abc]` соответствует любому символу или сопоставимому элементу за исключением a, b, c.

4. Сопоставимый символ это сопоставимый элемент ограниченный символами `[.` и `.]`.

5. Класс эквивалентности определяет набор сопоставимых элементов. Класс эквивалентности указывается внутри ограничителей `[=` и `=]`.

6. Класс символов определяет набор символов. Название класса символов указывается внутри ограничителей `[:` и `:]`. Следующие классы символов определены для всех локалей: alnum, alpha, blank, cntrl, digit, graph, lower, print, punct, space, upper, xdigit.

7. Выражение диапазона соответствует любому символу находящемуся между двумя указанными включительно, с учетом установленного порядка сортировки. При записи начальный и конечный элементы диапазона разделяются знаком минус.

2.3 BRE соответствующие нескольким символам

Следующие правила могут использоваться для построения базовых регулярных выражений соответствующих нескольким символам из BRE соответствующих одному символу:

1. Объединение BRE соответствует объединению строк соответствующих каждому из компонентов BRE.

2. Могут определяться подвыражения, путем заключения BRE между парами символов `\(` и `\)`. Такие выражения соответствуют тому же, чему и без символов `\(` `\)`.

3. Выражение обратной ссылки
   n соответствует той же строке (возможно пустой), что и подвыражение предшествующее n. Символ n является цифрой от 1 до 9 включительно и определяет номер подвыражения в строке.

4. Если за BRE соответствующим одному символу, подвыражением или обратной ссылкой следует символ звездочки, то такое выражение (вместе с символом звездочки) соответствует нулю или более последовательных включений соответствующих данному BRE.

5. Если за BRE соответствующим одному символу, подвыражением или обратной ссылкой следует интервальное выражение вида `\{m\}`, `\{m,\}` или `\{m,n\}`, то такое выражение (вместе с интервальным выражением) соответствует определенному интервальным выражением числу включений соответствующих данному BRE. Выражение `\{m\}` соответствует точно m включениям, выражение `\{m,\}` соответствует не менее чем m включениям и выражение `\{m,n\}` соответствует числу включений от m до n включительно.

2.4 Приоритет BRE

```
[= =] [: :] [. .]
\<специальный символ>
[ ]
\( \) \n
* \{m,n\}
^ $
```

2.5 Привязки

BRE могут быть привязаны к началу или концу строк. Правила привязки следующие:

1. Уголок (^), когда является первым символом всего регулярного выражения, соответствует началу строки.
2. Знак доллара, когда является последним символом всего регулярного выражения, соответствует концу строки.
3. BRE ограниченное символами ^ и $ соответствует только целой строке.

3 Расширенные регулярные выражения

3.1 Расширенные регулярные выражения соответствующие единственному символу или сопоставимому элементу

В ERE обычный символ, специальный символ с предшествующей обратной чертой или точка соответствуют одному символу. Выражение в квадратных скобках соответствует единственному символу или сопоставимому элементу. Обычный символ в ERE соответствует сам себе. Это любой символ из используемого кодового набора, за исключением специальных символов ERE.

3.2 Специальные символы ERE

Специальные символы ERE имеют особые свойства в некотором контексте. За пределами этого контекста или будучи предварены обратной косой чертой, такие символы соответствуют сами себе. Специальные символы и контексты в которых они имеют специальное значение следующие:

`. [\(` имеют специальное значение за исключением использования внутри квадратных скобок.

`)` имеет специальное значение когда соответствует открывающей скобке.

`* + ? { |` имеют специальное значение за исключением случаев использования внутри квадратных скобок.

`^` имеет специальное значение когда используется в качестве символа привязки, либо является первым символом выражения в квадратных скобках.

`$` имеет специальное значение когда используется в качестве символа привязки.

Точка является ERE соответствующим любому символу из поддерживаемого множества за исключением NULL.

3.3 Выражения в квадратных скобках

Правила построения выражений в квадратных скобках не отличаются от BRE.

3.4 ERE соответствующие нескольким символам

Следующие правила используются при построении ERE соответствующих нескольким сим-волам:

1. Объединение ERE соответствует объединению последовательностей символов соответствующих каждой компоненте ERE. Объединение ERE заключенное в круглые скобки соответствует тому же, чему и без круглых скобок.

2. Когда за ERE соответствующим одному символу или ERE заключенным в фигурные скобки следует знак +, то такое выражение соответствует одному или большему числу последовательных включений соответствующих данному ERE.

3. Когда за ERE соответствующим одному символу или ERE заключенным в фигурные скобки следует знак *, то такое выражение соответствует нулю или большему числу последовательных включений соответствующих данному ERE.

4. Когда за ERE соответствующим одному символу или ERE заключенным в фигурные скобки следует знак ?, то такое выражение соответствует нулю или одному включению соответствующему данному ERE.

5. Когда за ERE соответствующим одному символу или ERE заключенным в фигурные скобки следует интервальное выражение вида `{m}`, `{m,}` или `{m,n}`, то такое выражение соответствует определенному интервальным выражением числу последовательных включений соответствующих данному ERE.

3.5 Альтернативные ERE

Два ERE разделенные символом вертикальной черты (|) соответствуют строке, соответствующей одному из данных ERE.

3.6 Приоритеты ERE

```
[= =] [: :] [. .]
\<special character>
[ ]
( )
* + ? {m,n}
^ $
|
```

3.7 Привязки ERE

ERE могут быть привязаны к началу и концу строки. Для привязки используются символы `^` и `$`.

1. Знак `^`, за пределами выражения в квадратных скобках, привязывает начинающееся с него выражение или подвыражение к началу строки.

2. Знак `$`, за пределами выражения в квадратных скобках, привязывает заканчивающееся им ERE к концу строки.

4 Утилита grep

Формат командной строки:
`grep [-Ecinlv] [-e` *шаблоны*`...] [-f` *файл_шаблонов* `]...` *файл*`...`

Производит поиск в указанных файлах или во входном потоке, отбирая строки в которых имеется соответствие одному или более из указанных шаблонов. Аргумент *шаблоны* является списком регулярных выражений разделенных символами новой строки. Аргумент *файл_шаблонов* определяет файл содержащий список шаблонов. Значения прочих параметров следующие:

`-E` использовать расширенные регулярные выражения

`-c` вывести только количество строк удовлетворяющих условию

`-i` не различать регистр символов

`-n` предварить каждую выводимую строку ее номером в файле

`-l` вывести только имена файлов содержащих строки удовлетворяющие условию

`-v` вывести строки не удовлетворяющие ни одному из шаблонов

5 Потоковый редактор sed

Формат командной строки:
`sed [-n]` *сценарий* `[` *файл* `...]`
`sed [-n] [-e` *сценарий* `]... [ -f` *файл_сценария* `]... [` *файл* `...]`

Sed — потоковый редактор считывающий строки из текстовых файлов или со стандартного ввода, изменяющий их, в зависимости от команд редактирования указанных в параметре *сценарий* или в файле *файл_сценария* и выводящий результат на стандартный вывод. Ключ `-n` подавляет вывод всех строк файла (производимый по умолчанию). Каждый сценарий (параметр *сценарий*) состоит из команд редактирования (по одной в строке) следующего вида:

```
[адрес![,адрес3]]команда[аргументы]
```

допускаются пробельные символы перед первым адресом и перед командой. В нормальном режиме `sed` циклически копирует входную строку в рабочую область (в случае, если там ничего не осталось от предыдущей команды), применяет к ней последовательно все команды сценария, чьи адресные выражения содержат данную строку, выводит результат на стандартный вывод (если не указан ключ `-n`) и очищает рабочую область. Некоторые команды используют область сохранения, чтобы запомнить всю рабочую область или ее часть для последующего использования. Рабочая область и область сохранения способны хранить не менее 8192 байт. Адрес либо пустое поле, либо десятичное число, означающее номер входной строки, либо знак доллара (`$`), соответствующий последней строке, либо контекстный адрес (который состоит из регулярного выражения предваренного и заканчивающегося некоторым ограничителем, обычно косой чертой. Командная строка не содержащая адреса применяется к каждой строке. Командная строка содержащая один адрес применяется к каждой строке соответствующей адресу. Командная строка содержащая два адреса применяется ко всем строкам начиная с соответствущей первому адресу и заканчивая соответствующей второму. Начиная с первой строки, следующей за указанным диапазоном `sed` снова проверяет строки на соответствие первому адресу и, в случае нахождения соответствия, процесс повторяется.

Команды могут применяться к строкам не входящим в диапазон при помощи команды отрицания `!`.

Sed поддерживает базовые регулярные выражения со следующими дополнениями:

- В контекстных адресах, конструкция `cREc`, где `c` любой символ, отличный от обратной косой черты и символа новой строки, идентична `/RE/`.
- Последовательность `\n` соответствует символу новой строки.

В следующем списке команд, максимальное число адресов для команды определено последовательностями `[0addr]`, `[1addr]` и `[2addr]`. Аргумент `text` состоит из одной или более строк. Каждый символ новой строки должен быть предварен обратной косой чертой. Прочие символы новой строки удаляются.

`[2addr]command-list`
: Выполнять command-list, для указанных строк.

`[1addr]a\text`
: Вывести текст перед обработкой указанной строки.

`[2addr]b [метка]`
: Перейти к команде `:`, содержащей указанную метку. Если метка пуста, перейти в конец сценария.

`[2addr ]c\text`
: Удалить рабочую область. При пустом или одном адресе или в конце двухадресного диапазона вывести текст.

`[2addr]d`
: Удалить рабочую область.

`[2addr]D`
: Удалить начало рабочей области, до первого перевода строки включительно.

`[2addr]g`
: Заменить содержимое рабочей области содержимым области сохранения.

`[2addr]G`
: Добавить к содержимому рабочей области символ новой строки с последующим содержимым области сохранения.

`[2addr]h`
: Заменить содержимое области сохранения содержимым рабочей области.

`[2addr]H`
: Добавить к содержимому области сохранения содержимое рабочей области.

`[1addr]i\text`
: Вывести текст.

`[2addr]l`
: Вывести содержимое рабочей области, заменяя символы табуляции, перевода строки, обратной косой черты и т.п. на пары символов \,\a ,\b ,\f ,\r ,\t ,\v. Прочие непечатные символы заменяются трехзначными восьмеричными числами предваренными обратной косой чертой.

`[2addr]n`
: Вывести содержимое рабочей области на стандартный вывод и считать в рабочую область следующую входную строку.

`[2addr]N`
: Добавить к содержимому рабочей области следующую входную строку. При этом изменяется текущий номер строки.

`[2addr]p`
: Вывести рабочую область на стандартный вывод.

`[2addr]P`
: Вывести начало рабочей области, до первого символа новой строки на стандартный вывод.

`[1addr]q`
: Перейти в конец сценария и выйти.

`[2addr]r rfile`
: Скопировать содержимое файла rfile на стандартный вывод до обработки следующей входной строки.

`[2addr]s/reg-exp/replacement/flags`
: Подставить replacement вместо подстрок рабочей области, соответствующих регулярному выражению reg-exp. Вместо косой черты можно использовать любой другой символ. Символ амперсенда & в replacement будет заменен строкой, соответствующей регулярному выражению. Символы `\n`, где `n` цифра, будут заменены соответствующей обратной ссылкой. Могут использоваться следующие флаги:

  `n` Заменять n-ое вхождение регулярного выражения.

  `g` Заменить все вхождения регулярного выражения.

  `p` Выводить содержимое рабочей области если произведена замена.

  `w wfile`
  : Добавить содержимое рабочей области в файл wfile если произведена замена.

`[2addr]t [label]`
: Перейти к команде :, содержащей метку label, если с момента последнего чтения входной строки или последнего выполнения команды t в буфере выполнялись подстановки. Если метка опущена, перейти в конец сценария.

`[2addr]w wfile`
: Добавить содержимое рабочей области к файлу wfile.

`[2addr]x`
: Обменять содержимое рабочей области и области сохранения.

`[2addr]y/string1/string2/`
: Заменить все включения символов содержащихся в строке string1 соответствующими символами строки string2. Строка string1 не должна содержать повторяющиеся символы, длины строк должны совпадать.

`[2addr]!command`
: Применить команду только к тем строкам, которые не принадлежат указанным адресам.

`[0addr]: label`
: Команда ничего не делает. Содержит метку, на которую можно перейти командами b и t.

`[1addr]=`
: Выводит номер текущей строки.

`[0addr]`
: Пустая команда.

`[0addr]#`
: Символ # и остальные, до конца строки, игнорируются (комментарий).

6 Практическое задание

1. Скопируйте в домашний каталог файл `/home/labs/text.txt`
2. Выведите в файл отчета все строки файла `text.txt` содержащие слово "Благозвон". Добавьте к файлу отчета количество найденных строк.
3. Добавьте к файлу отчета все строки файла `text.txt`, которые начинаются с буквы "Б" (возможно после нескольких пробелов).

4. Добавьте к файлу отчета все непустые строки файла `text.txt`, не содержащие русскую букву "а" (в любом регистре).

5. Добавьте в файл отчета имена файлов каталога `/home/labs`, которые содержат слово "include".

6. Используя редактор sed замените в файле `text.txt` все слова Снарк на СНАРК. Результат сохраните в файле `snark.txt`.

7. Проанализируйте сценарий `/home/labs/sed1.sh` и формат файла `/home/labs/rfc-index.txt`. Запустите сценарий несколько раз указывая в качестве аргумента целое число от 1 до 3000.

8. При помощи sed выведите из файла `/etc/passwd` идентификаторы пользователей и их имена, разделенные символом табуляции, в формате:

   ```
   0 root
   1 daemon
   2 operator
   ...
   ```

   Вывод должен быть отсортирован по идентификатору пользователя. Результат добавьте к файлу отчета.

9. Напишите сценарий для sed, который будет менять местами каждые две строки файла. Возможно sed прийдется запускать с ключом -n.

10. Если Вам не удалось выполнить предыдущее задание, обратите внимание на файл `/home/labs/rev.sed`.

11. Напишите сценарий для редактора sed, который будет добавлять:

    (a) В начало файла строку "<html><body>".

    (b) В начало всех строк начинающихся словом "Приступ" строку "<H2>", а в конец строку "</H2>".

    (c) В конец всех остальных строк "<br>".

    (d) К концу последней строки "</body></html>"

    Результат работы сценария сохраните в файле `snark.html` в каталоге `public_html` домашнего каталога (каталог потребуется создать). Сценарий сохраните под именем `sed1` в домашнем каталоге.