> «Деньги при их разумном использовании помогут вам стартовать быстрее. Но их отсутствие никогда не удержит вас от старта». Майк Михаловиц

sed в linux - примеры использования

⌚ марта 2022 | Категория: Bash

Утилита *sed* это мощный потоковый редактор текста с поддержкой регулярных выражений. С помощью *sed* вы можете заменять шаблоны текста (причем непосредственно в файле!), удалять строки (элементы массива), выводить подходящие по маске строки (подобно *grep*). Редактор *sed* поддерживает применение нескольких команд и расширенный синтаксис регулярных выражений (при котором не нужно экранировать спец. символы).

Важно!

В sed нет поддержки опережающих и ретроспективных проверок в регулярках! Для замены с использованием расширенного синтаксиса regex используйте:

```
find . -type f -name '*.blade.php' -exec perl -
```

Внимание!

В sed довольно проблемно работать с символом перевода строки! Самое удобное решение - это:

```
echo "text" | perl -pe 's/\n/_/'
```

Шаблон:

```
sed [-opt] 's/regex/replace/flag' input-file

sed 's/regex/replace/flag'  # замена найденных п
sed '1,5s/regex/replace/gi' # замена только в ук
sed -r 's/regex/replace/g'  # расширенный синтак
sed 's/regex//g'            # удалить найденные
sed '/regex/d'              # удалить строки под
sed -n 2p                   # вывести 2ю строку
sed -n '/composer/p'        # вывести только стр
sed 's/1-9/&/p'             # & при замене означ
```

В качестве разделителей можно использовать любые символы (напрмиер: #, @). *Match* части (которые внутри круглых скобок) доступны как \1, \2, \n.

Опции утилиты:

```
-p   вывести на экран
-d   удалить
-i   выполнять изменения непосредственно в файле
-n   не выводить результат замены/поиска на экра

-e   указывает на передачу инструкции (команда з

-E   расширенный regex, ближе к JavaScript, Go.
-r   расширенный regex синтаксис. Спец символы а
-P   perl-совместимый regex синтаксис

-s   consider files as separate rather than as a
```

Флаги строки-команды (указывать в конце маски):

```
g    глобальный поиск/замена, а не только первое
i,-l регистро-независимый поиск
p,   печать найденных подстрок
d    удалить строки
```

Примеры

Фильтрация строк

Вывести строки 1-5:

```
sed '1,5p'
head -5
```

Вывести файлы соответствующие маске:

```
ls | sed -n '/composer/p'
```

Строки длиннее 80 символов:

```
sed -n '/^.{80}/p'
sed -n '/^.{80}/!p' # короче 80 символов
```

Замена по шаблону

Заменить строки начинающиеся с:

```
sed 's/^line_start=.*$/line_start="replacement"/
```

Вывести вхождения (*matches*) через табуляцию:

```
sed -r 's/^ +([^ ]+) +(.+)$/\1\t\2/'
```

Заменить названия файлов (*composer* на *composer-dev*):

```
ls | sed -r 's/(composer)/\1-dev/g'
```

Заменить символы (regex):

```
echo 'aa,bb,xx' | sed "s/xx/cc/g" # aa,bb,cc
```

Заменить URL в файле (штука в разделителях |, и -i для замены в файле):

```
sed -i "s|$old_site_url|$new_site_url|g" file.ym
```

Заменить параметр в конфиге:

```
sed -ie '/project_file_path *=/ s|=.*$|=/home/pi
```

Заменить значение в `xml`-конфиге:

```
sed -i -r 's/(name="width" value=")[^"]+"/\148KP
```

Удалить начальные пробелы (аналог *`ltrim`*):

```
echo "   some string" | sed 's/^ *//g'
```

Удалить конечные пробелы:

```
echo "test " | sed 's/ *$//'
```

Альтернатива `trim`:

```
echo "  test  " | xargs
echo "  test  " | sed -e 's/^[[:space:]]*//' -e
```

Удалить `line breaks`:

```
tr -d '\n'
```

Удалить часть приглашения командной строки:

```
xsel -p -o | sed 's/^[^\$|#]*[\$|#] //'
```

Удаление строк

Удалить из файла строку подходящую шаблону:

```
sed '/regex/d' /path/file
```

Удалить первую строку вывода:

```
sed 1d        # удалить первую строку
sed '5, 10d'  # удалить строки с 5-й по 10-ю
```

Удалить строки от первой до соответствующей *`regex`*:

```
sed "1,/end string pattern/d"
```

Заменить подстроку:

```
echo '--some string' | sed 's/\(Some\)/New \1/i'
```

Примечание

По умолчанию необходимо экранировать все спец. символы в `regex`'ах, что крайне затрудняет чтение масок. Для того, чтобы экранировать спец.символы только в случае описания в тексте их самих - включите расширенный режим *`regex`* выражений с помощью опции `-r`.

Удалить пустые строки:

```
sed '/^$/d'
```

Удалить последние `N=2` символа:

```
echo "latest" | sed "s/..$//" # late
```

Извлечение подстрок

Вырезать / запомнить последние `N=4` символа:

```
echo "latest" | sed "s/.*\(....$\)/\1/" # test
```

#sed #regexp #trim #replace

⌚ марта 2022 | категория: Bash

Сообщения

Пройти капчу и отправить сообщение

1