Grep, awk, sed

Общий синтаксис для sed и awk: command [options] script filename

Если скрипт определяется в команде, то одинарные кавычки.

-f filename имя скрипта

Каждая инструкция скрипта состоит из блока шаблона и процедуры. Шаблон отделяется /. Строки скрипта интерпретируются последовательно, если найдено соответствие шаблону - исполняется процедура. Применяется каждая строка скрипта. Sed выводит строку, поведение awk нужно определять в скрипте. В скриптовом файле шаблон пишется без кавычек

```
cat sedscr
s/ MA/, Massachusetts/
s/ PA/, Pennsylvania/
s/ CA/, California/
s/ VA/, Virginia/
s/ OK/, Oklahoma/
```

grep

Регулярки пишутся в двойных кавычках

grep "[PV]A" testone.txt

grep "10\{1,2\}1" testone.txt

grep по умолчанию не понимает спецсимволы типа \d,... \d интерпретируется как символ d. Чтобы понимал:

```
echo "123 abc" | grep -P '\d'
```

Опция	Описание
-P	Регулярка в стиле Perl
-o	Показать только совпадения

sed

Правила применяются последовательно, поэтому одно правило может сделать строку, соответствующую второму правилу. И результат замены может быть неожиданным.

Опция	Описание
-e	Многострочный скрипт в командной строке, sed -e ’s/ MA/ Massachusetts/’ -e ’s/ PA/ Pennsylvania/’ list Но проще через ; sed ’s/ MA/ Massachusetts/; s/ PA/ Pennsylvania/’ list

Правила для скрипта

Правило	Описание
s	Замена, файл меняется sed 's/ MA/ Massachusetts/' testone.txt ! случай 's/ MA/, Massachusetts/' не сработал, получилось без запятой. Какой-то нюанс.
-n 's/.../.../p'	Выводится только соответствующие шаблону, файл не меняется.
's/.../.../g'	Все вхождения в строке
's/что/на что/число'	Заменить каждое вхождение с номером Число. 's/foo/bar/2' - второе вхождение foo
sed 's/что/на что/w имяфайла'	сохранить после замены в имяфайла
/Sebastopol/s/CA/California/g	Замена CA на California если есть слово Sebastopol
'/.../d'	Удаление

awk

Интерпретирует каждую строку как запись и каждое слово как поле. Переменные

$0 полная строка

$1, $2, ... номера полей. Вне awk скрипта $ - параметр bash!

Различные форматы применяются последовательно, например

```
awk '/VA/ { print $1}' testone.txt
```

Для строк, в которых есть VA, будет выведено первое поле.

Можно создавать переменные и затем использовать их в сравнении. Переменные создаются внутри {}, используются вне. Изначально все переменные - пустые строки, автоматически инициализируются.

```
{LastState = $1}
```

Параметр	Описание
-F	Разделитель, например awk -F, '{ print $1}' testone.txt

Правила для скрипта формата {}

Правило	Описание
print	Вывод данных awk '{ print $1 }' testone.txt awk '{ print "Name: " $1}' testone.txt
;	Разделитель команд внутри awk '{ print $1; print $2 }' testone.txt

Правила для скрипта формата //

Правило	Описание
/some/	Поиск с выводом соответствия шаблону awk ’/some/’ list

Пример совмещения задач

```
! /bin/sh
awk -F, '{print $4 ", " $0}' $* |
sort |
awk -F, '
$1 == LastState {print "\t" $2}
```

```
$1 != LastState {LastState = $1; print $1; print "\t" $2}
'
```

Вывод

```
CA
        Amy Wilde
Massachusetts
        Eric Adams
        John Daggett
        Sal Carpenter
OK
        Orville Thomas
PA
        Terry Kalkas
VA
        Alice Ford
        Hubert Sims
```

Revision #5
Created 29 August 2025 04:45:12 by Admin
Updated 30 August 2025 07:39:52 by Admin