GeekBrains

Linux и облачные вычисления

Работа с текстовыми данными

Способы выбора данных из файла. Команды awk, grep, sed. Регулярные выражения.

Оглавление

Способы выбора данных из файла

Команда awk

Команда **AWK** предоставляет возможность работать с утилитой для преобразования строк. Например, **AWK** может извлечь только четвертый столбец из текстового файла с данными в формате таблицы. **AWK** — не просто утилита, а цельная реализация языка программирования. Утилиты **AWK** и **GAWK** (**GNU awk**) являются одинаковыми в плане функций.

Практическое применение

AWK позволяет как принимать данные от программы на вход, так и читать их из стороннего файла. Правила **AWK** передаются утилите посредством командной оболочки (например **Shell**, **sh**, **zsh**, **dsh**), либо переносятся в текстовый файл, имя которого должно сообщаться утилите после параметра **–f**. Стандартно данные на входе читаются из файлов, имена которых передаются утилите в качестве аргументов после ключей, а если этого не происходит, то данные поступают из **STDIN**.

Рассмотрим два примера. В одном правила для **AWK** передаются программе посредством командной оболочки, а данные на входе читаются из файла. В следующем примере для получения входных параметров используется сторонняя программа. То есть данные передаются утилите **AWK** при помощи программного канала. Все правила помещаются во внешнем файле сценария.

```
$ awk '{ print $4; }' inputfile
$ makedata | awk -f scriptmy.awk
```

Сценарии **AWK** в виде отдельных файлов становятся исполняемыми при помощи размещения правильной последовательности «шебанг» в их начальных строках:

```
#!/bin/awk -f
```

Важное замечание: если утилита **AWK** расположена не в директории для бинарных файлов **/bin/awk**, то нужно писать правильный путь к утилите. Его можно посмотреть, выполнив команду **which awk**.

Приведем ниже примеры использования утилиты **AWK**.

Выполним объединение текстовых файлов с пропуском определенных столбцов.

```
cat >myfile_1<<exp
O11 0.0181
O12 0.3441
O13 0.3243
exp
cat >myfile_2<<exp
O14 0.1045
O15 0.4515
O16 0.3227
exp
paste file_1 file_2 | awk '{print $2" "$3" "$4}'
0.0181 O14  0.1045
0.3441 O15  0.4515
0.3243 O16  0.3227
```

При помощи команды:

```
paste file_1 file_2 | awk '{print $2" "$3" "$4}'
```

Мы выполнили объединение двух файлов, используя только второй, третий и четвертый столбцы. **AWK** позволяет производить фильтрацию из полей стандартного вывода. Стандартно **AWK** разделяет поля пробелами.

Команда grep

grep — это утилита, которой требуются регулярные выражения для работы с файлами. Команда читает текст из файла и показывает на терминале те строки, которые совпадают с введенным ранее выражением в конструкции утилиты. **Man** утилиты приведен ниже:

```
grep [key] PATTERN [MY_FILE...]
```

Здесь **PATTERN** — регулярное выражение, а **MY_FILE** — файлы, к содержимому которых оно будет применено.

Если файл не задать, то утилита считает текст со стандартного ввода **STDin**. При помощи **options** можно задавать ключи утилиты **grep**. К примеру, ключ **'–v'** приводит к выводу все строки, не совпадающие с заданным регулярным выражением.

Рассмотрим примеры использования утилиты **grep** и ее выражений. Команда **ls (list)** показывает список файлов в директории. Команда **'ls /bin'** покажет список файлов из директории **/bin**. Команда **ls (list)** выполняет отображение на стандартный поток ввода **STDIN**. Допустим, нам нужны только те программы (файлы) из **/bin**, которые содержат буквы **'zip'**. Этой строке соответствует простое регулярное выражение **'zip'**. Перенаправляем вывод из **list** в рассматриваемую утилиту и получаем:

```
$ ls /bin | grep 'zip'
bunzip2
bzip2
bzip2recover
```

```
gunzip
gzip
```

Заданное выражение ставится в одинарные кавычки **''**, которые говорят оболочке **bash**, что внутри них — простая строка. Это правило разрешает использовать в этом выражении пропуски, и его требуется писать во многих случаях. Например, регулярное выражение **'a b'** описывает шаблон для строк, содержащих последовательно **'a'**, пробел и **'b'**. Если его указать **grep** без кавычек, то есть **'grep a b'**, то командный интерпретатор оболочки **bash**, разобрав строку, вызовет утилиту с двумя параметрами, и **grep** будет выполнять поиск по строкам с буквами **'a'** в файле **'b'**. При использовании кавычек командный интерпретатор будет считать выражение **'a b'** одним параметром и передаст его **grep** целиком, вместе с пробелом внутри.

Названия файлов из **/bin**, которые заканчиваются на **'2'**:

```
$ ls /bin | grep '2$'
bash2
bunzip2
bzip2
```

Названия файлов из **/bin**, которые начинаются на **'b'**:

```
$ ls /bin | grep '^b'
basename
bash
bash2
bunzip2
bzcat
bzip2
bzip2recover
```

Названия файлов из **/bin**, начинающиеся на **'b'** и содержащие в своем имени букву **a**:

```
$ ls /bin | grep '^b.*a'
basename
bash
bash2
bzcat
```

В данном регулярном выражении указано, что оно:

- должно совпадать с началом строки — ^;
- в начале строки должна быть буква 'b' — ^b;
- дальше может быть любой символ — ^b;
- и таких символов может быть сколько угодно — 0 или больше— ^b.*;
- а дальше должна быть буква 'a' — ^b.*a.

Команда sed

Команда **grep** производит фильтрацию строк и отображает на консоль найденные результаты в виде, в котором они и были. Иногда требуется не только выполнить поиск текста, но и скорректировать его.

Для этого можно воспользоваться утилитой **SED** — редактором вывода потока (**StreamEDitor**). SED нужен для основных корректировок текста, который выводится из файла или поступает со стандартного потока ввода. **STDin** совершает преобразование над вводом за операцию. Общий формат исполнения утилиты:

```
SED [key] COMMANDS [name FILE...]
```

Из многочисленных приемов утилиты SED покажем только поиск и замену. Их представление — **'s/SED1/SED2/'**, выполняется поиск в каждой из строк текста регулярного выражения **SED1**. Результаты совпадения заменяются на выражение **SED2**. Результирующий текст выводится на стандартный поток вывода **STDout**. Покажем использование команды замены текста в **SED** на практике. В простом случае заменим один фрагмент текста на другой:

```
~ $ ls -1 /var/cache
apt
fontconfig
man
$ ls /var/cache/ | sed 's/apt/APT/'
APT
fontconfig
man
```

В директории **/var/cache** есть файлы (выведены выше) — список можно получить из консоли командой **ls (list)**. Выражение **'apt'** совпадает с одной из строк вывода, и мы меняем это выражение на **'APT'**.

```
$ ls /var/cache/ | sed 's/a/A/'
Apt
fontconfig
mAn
```

На этот раз произвели замену в выводе **ls** буквы **'a'** на **'A'.** SED выполняет свои команды для каждой из строк вывода, поэтому в них, где была буква **'a'**, она была заменена.

Команда **uptime** предоставляет характеристики по работе системы, ее стабильности, времени без перезагрузок:

```
~ $ uptime
08:41:43 up 123 days, 12:14, 5 user, load average: 1.24, 5.20, 4.31
```

Чтобы показать из этого вывода текущее число пользователей, залогиненных в системе, используем утилиту. Число пользователей — это определенное количество целых чисел — **'[0-9]\+'**, за которыми после пробела (или нескольких пробелов в общем случае) — **'[0-9]\+ \+'** — следует слово **user** (или **users**). Нам нужно узнать только число — выберем его в подвыражении: **'\([0-9]\+\) \+user'**. В начале строки есть текст, отделенный от числа пользователей пробелом: **'^.* \([0-9]\+\) \+user'**. Конец строки может быть любым: **'^.* \([0-9]\+\) \+user.*'**.

Указанное выражение совпадает со всей строкой и выделяет в подстроку **\1** число пользователей. Выполнив замену целиком на **\1**, получим в результате только это число:

```
$ uptime | sed 's/^.* \([0-9]\+\) \+user.*/\1/'
5
```

Также можно получить время работы системы (строку вида **'123 days, 12:14'**):

```
$ uptime | sed 's/^.* up \+\(.\+\), \+[0-9]\+ \+user.*/\1/'
123 days, 12:14
```

Практическое задание

1. Выбрать из домашней директории пользователя **ubuntu** файлы с расширением **.py**, название которых начинается на букву **t**.
2. Из всех файлов с расширением **.py**, расположенных в домашней директории пользователя **ubuntu**, выбрать строки, содержащие команду **print**, и вывести их на экран.
3. Из результатов работы команды **uptime** выведите число дней, которое система работает без перезагрузки.

Дополнительные материалы

1. Bash-скрипты, часть 8: язык обработки данных awk.
2. Маленький учебник по Sed и Awk.
3. Команда grep: опции, регулярные выражения и примеры использования.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. Использование awk в Linux.
2. Поиск текста в файлах Linux.
3. Изучаем команды Linux: sed.