Лабораторная работа №4

Утилита gawk

Цель работы: *изучению утилиты gawk - мощного инструмента ОС Linux; ее функциональным возможностям и синтаксису, созданию shell-сценариев с использованием этой утилиты, ее команд и управляющих структур.*

Продолжительность работы - 4 ч.

Теоретические сведения

Утилита gawk

Утилита gawk предназначена для нахождения информации в текстовых файлах по заданным критериям выбора. Она снабжена мощными средствами обработки текста в больших текстовых файлах. Утилиту **gawk** можно отнести к программируемым фильтрам, настроенным на выполнение конкретной задачи.

Gawk одновременно является утилитой, программируемым фильтром и средой программирования, с помощью которой можно создавать другие фильтры. Поэтому она занимает особое место в операционных системах ОС Unix и ОС Linux и является мощным и гибким инструментом. Gawk является версией утилиты awk, используемой в UNIX и разработанной одним из создателей языка С Брайеном Керниганом и другими.

Любая среда программирования имеет свой язык программирования, это относится и к утилите **gawk**. Многие операторы взяты из языка программирования С и имеют такой же синтаксис. Фильтры, создаваемые с использованием языка программирования **gawk**, могут использоваться в среде любого интерпретатора ОС Unix и ОС Linux, например, в BASH, shell, TCSH. Созданные фильтры считывают информацию из файла или стандартного потока ввода, анализируют, выделяют по определенным заданным критериям или изменяют эту информацию, сохраняют выходные данные или направляют в стандартный поток вывода.

Таким образом, можно создавать не только фильтры, но и собственные команды ОС Unix и ОС Linux, что особенно актуально в больших системах распределенных ресурсов с нестандартной структурой организации. Утилиту **gawk** можно использовать для создания отчетов, поиска заданных текстовых фрагментов, выполнения вычислений на основе вводимых данных, для работы со структурами, напоминающими базы данных, т.е. состоящих из записей, подразделяющихся на поля, разделенные специальными символами.

Утилиту **gawk** можно вызвать непосредственно из командной строки или из shell-сценария с помощью ключевого слова **gawk**. Если сценарий включает gawk как составляющую, то его можно рассматривать как новый фильтр, утилиту или команду.

Синтаксис команды **gawk** следующий:

$ 'Шаблон или условие {действие}' имена_файлов

Командой **gawk** используется шаблон поиска такой, какой применяется в командах-фильтрах **grep, fgrep, egrep**. Шаблон выделяется символами косой черты (знаками слеш).

Пример 1.

$ gawk '/интерфейс Gnome/ {print}' /home/info/*

Ищутся все файлы в каталоге ***/home/info***, в которых встречается последовательность из слов ***интерфейс Gnome*** и результаты направляются на стандартное устройство вывода - действие ***{print}***. В результатах выводятся имена файлов и строки, содержащие шаблон.

Действие направления на стандартное устройство вывода ***{print}*** обычно задается как действие по умолчанию, поэтому команду примера можно переписать так:

$ gawk '/интерфейс Gnome/' /home/info/*

Результат при этом не изменится.

Пример 2. *Найти по указанному пути файлы исходных текстов на языке С, включающие слова, обозначающие имя переменной, например, SHELLDATA или SHDATA:*

$ gawk '/ SHELLDATA|SHDATA***/' /home/include/****

Переменные и константы

Утилита gawk позволяет определять **переменные и константы**. Существует три типа переменных: переменные для обозначения полей, специальные переменные и пользовательские переменные. Пользовательские переменные определяет пользователь, остальные утилита определяет автоматически. Существует два типа констант: арифметические и строковые. Арифметические константы состоят из цифр, строковые константы заключаются в двойные кавычки и состоят из любых символов, включая символы цифр. **Переменные опредеделяются после их первого использования.**

Поля

Утилита gawk может работать с переменными, называемыми **полями**. Поле представляет любой набор символов, ограниченный разделителями полей (первое и последнее поля могут иметь по одному разделителю). По умолчанию в качестве разделителя используется пробел или знак табуляции. Утилита gawk нумерует поля строк текстового файла, начиная с 1, а также автоматически определяет переменную для каждого поля файла. Имя переменной, обозначающей

поле, состоит из знака $ и номера поля. Например: $1, $2, $3 и т.д. Переменная - специальная переменная, определяющая всю строку.

Пример 3. *Предположим, что у нас есть текстовый файл /home/tabl/list_students, отсортированный по алфавиту, каждая строка которого включает: фамилию, имя, факультет, курс, рейтинговая оценка, и состоящий из следующих строк-записей:*

Аринин Иван МП 1 4
Бетинов Евгений ЭКТ 2 5
Кошин Леонид МП 1 4
Кошкин Владимир ЭКТ 1 5
Липин Федор МП 2 3
Пенов Николай МП 1 4
Яшин Петр ЭКТ 2 4

Требуется вывести список, состоящий из фамилий, имен и рейтинговых оценок.

Нужно вывести 1, 2, 5 поля. Поля перечисляются через запятую и пробел, команда выглядит следующим образом:

$ gawk '{print $1, $2, $5}' /home

Скорее всего, Вы находитесь в каталоге ***/home*****,** поэтому имя файла достаточно описать так: ***/tabl/list_students***. В результате выполнения на экран выводится следующая таблица, поля которой будут выровнены.

Аринин Иван 4
Бетинов Евгений5
Кошин Леонид 4
Кошкин Владимир 5
Липин Федор 3
Пенов Николай 4
Яшин Петр 4

Пример 4. Теперь распечатаем весь файл в виде таблицы, т.е. чтобы все поля были выровнены. Для этого используем переменную **$0**, иначе выведенные данные не выровняются и будут плохо читаемы:

$ gawk '{print $0}' /home/tabl/list_students

Результат вывода на экран будет следующий:

Аринин	Иван	МП	1	4
Бетинов	Евгений	ЭКТ	2	5
Кошин	Леонид	МП	1	4
Кошкин	Владимир	ЭКТ	1	5
Липин	Федор	МП	2	3
Пенов	Николай	МП	1	4
Яшин	Петр	ЭКТ	2	4

Пример 5. Выберем из исходного файла */home/tabl/list_students* студентов с рейтинговой оценкой 5 и распечатаем все значения полей для выбранных записей:

$ gawk '/5/ {print $0}' /home/tabl/list_students

Бетинов	Евгений	ЭКТ	2	5
Кошкин	Владимир	ЭКТ	1	5

В случае, если 5 встречается в других полях, например, курс 5, то условия выборки недостаточны.

Операторы сравнения и логические операции

Для формирования логических выражений, выполняющих проверку условий используются **операторы сравнения**. В последнем примере для поиска нужно использовать в шаблоне оператор сравнения, а именно, оператор проверки на равенство (==):

$ gawk '/$5==5/ {print $0}' /home/tabl/list_students

Другой оператор сравнения называется оператор проверки на неравенство (**!**=) и в качестве примера может быть использован для поиска записей с рейтинговой оценкой, отличной от 5:

$ gawk '/$5!=5/ {print $0}' /home/tabl/list_students

Утилита gawk использует операторы сравнения такие же, как в языкеС: <, >, >=, <=, но, в отличие от С, может сравнивать строковые значения. Сравнение выполняется в соответствии с алфавитом. Утилита gawk использует логические операторы:

&& - логическое И (AND);
||- логическое ИЛИ (OR);
! - логическое отрицание (NOT).

Они дают возможность использовать сложные логические условия, при этом условия записываются в скобках.

Пример. Выбрать из списка студентов факультета ЭКТ, 1 курса:

$ gawk '($3 == "ЭКТ") && ($4 == 1) {print}' list_student

Функция lenght

В условии можно использовать функцию lenght, определяющую длину поля. Например, нужно выбрать записи студентов факультета ЭКТ. Можно записать так:

$ gawk '/ЭКТ/ {print}' /home/tabl/list_students

Используя функцию ***lenght*** и замечая, что третье поле (факультета) в наших данных состоит из 2 или 3 символов,

а нужно выбрать состоящее из трех, можно записать так:

$ gawk '/length ($3 == 3) / {print}' /home/tabl/list_students

В качестве упражнения можно выбрать все фамилии с длиной > 5 символов.

Специальные переменные языка утилиты gawk

В утилите gawk определяются специальные переменные:

NR - хранит номер текущей просматриваемой или обрабатываемой записи, с ее помощью удобно нумеровать строки;

NF - количество полей в текущей записи;

$0 - все поля текущей записи;

$n - номер поля (см. раздел "Поля");

FS - разделитель поля ввода;

FILENAME - имя файла, в котором работаем.

Предположим, нам нужно вывести данные исходного файла в виде таблицы и пронумеровать строки:

$ gawk '{print NR, $0}' /home/tabl/list_students

Результат вывода на экран будет следующий:

1	Аринин	Иван	МП	1	4
2	Бетинов	Евгений	ЭКТ	2	5
3	Кошин	Леонид	МП	1	4
4	Кошкин	Владимир	ЭКТ	1	5
5	Липин	Федор	МП	2	3
6	Пенов	Николай	МП	1	4
7	Яшин	Петр	ЭКТ	2	4

Использование слов BEGIN и END

Для описания действий до или после просмотра и обработки всех записей используются ключевые слова **BEGIN** и **END** соответственно.

Например, если перед печатью всех пронумерованных записей мы хотим поставить заголовок, то можно написать так (считаем, что мы находимся в ***/home/tab***):

$ gawk 'BEGIN {print "student's list"} {print NR, $0}' list_students

Сначала будет выведен заголовок ***student's list***, далее пронумерованный по строкам текст таблицы данных.

Арифметические операторы и функции

Gawk поддерживает полный набор арифметических операторов, они такие же, как в языке С: *****, /, +, **-**, **%**.

Арифметические вычисления в gawk выполняются над **числовыми шаблонами**. Числовой шаблон представляет собой последовательность цифр. Это может быть арифметическая константа, переменная или поле, которые состоят из цифр. Сюда входят встроенные числовые переменные NR и NF (описаны выше).

Пример 1. *Вывести все четные записи текстового файла.*

$ gawk ' (NR % 2) {print NR, $0}' /home/tabl/list_students

$ gawk '{print NR, $0}' /home/tabl/list_students

Пример 2. *Посчитать суммарный балл оценок студентов (находимся в **/home/tabl**). Вывести на экран таблицу и суммарный итог.*

$ gawk '{print; sum += $5} END {print "summa"=, sum}' list_students

Пример 3. *Посчитать средний балл оценок студентов (находимся в **/home/tabl**). Вывести на экран заголовок таблицы, таблицу и среднее значение.*

Пример 4. *Вывести на экран заголовок таблицы, таблицу, количество записей и среднее значение. Количество записей определяется переменной **NR** после просмотра всех записей. В процессе обработки **NR** увеличивается на 1 при обращении к очередной записи файла. Нумерация начинается с 1.*

$ gawk 'BEGIN {print "exam results" }{print; sum += $5} END {print NR, "average"=, sum}' list_students

Пример 5. Пусть есть файл ***/home/plant***, состоящий из записей со следующими полями: имя предприятия, доход по 1, 2, 3, 4 кварталам. Найти итоговый доход d для каждого предприятия и распечатать записи, пронумеровав их и указав итоговый доход последним полем каждой записи.

$ gawk '{ d = $1 + $2 + $3 + $4; print NR, $0, d}' /home/plant

Из языка программирования С заимствовано несколько функций:

int(*число*) - округление числа с плавающей точкой до целого;

cos(*число*) - возвращает косинус аргумента *число*

sin(*число*) - возвращает синус аргумента *число*

log(*число*) - возвращает натуральный логарифм аргумента *число;*

exp(*число*) - возвращает экспоненту аргумента *число;*

sgrt(*число*) - возвращает квадратный корень аргумента *число;*

rand() - возвращает случайное число;

arand(*число*) - использует аргумент *число* в качестве нового начального значения функции rand().

Пример 6. Посчитать экспоненту значения 5-го поля второй записи.

$ gawk '(NR == 2) {print exp($5)}' /home/tabl/list_students

Поиск по шаблону

Для выполнения поиска по шаблону в полях текущей записи применяются специальные символы: ~, !~. С

помощью операции ~ можно проверить, присутствует ли конкретный шаблон в определенном поле (шаблон здесь часть поля или все поле). С помощью операции !~ можно проверить, в каких записях отсутствует конкретный шаблон в определенном поле.

Пример 1. Присутствует ли фамилия Леонов в списке файла ***list_students***?

$ gawk '($1 ~ /Леонов/) {print}' /home/tabl/list_students

Пример 2. Есть запись студента Иванова в списке файла ***list_students***? Причем мы не уверены, с прописной или строчной буквы ее могли написать.

$ gawk '($1 !~ /[Ии]ванов/) {print}' /home/tabl/list_students

Оформление инструкций gawk отдельным файлом

Если инструкция gawk большая в написании, то ее можно оформить отдельным файлом, например, выше указанный пример:

$ gawk 'BEGIN {print "exam results" }{print; sum += $5} END {print NR, "average"=, sum}' list_students

можно записать так. Создаем отдельный файл, пусть **f_instr**:

```
BEGIN
{print "exam results"}
{print; sum += $5}
 END        {
print NR, "average"=, sum}
```

Далее выполняем команду gawk с опцией -f, которая дает возможность читать инструкции из указанного файла, а не из командной строки. Тогда общая структура команды следующая:

$ gawk -f имя_файла_инструкции имя_обрабатываемого_файла

$ gawk -f f_instr list_students

Иногда файлы инструкций для удобства записывают с расширением gawk, т.е. имя файла тогда выглядит так: ***f_instr.gawk,*** *а команда так:*

$ gawk -f f_instr.gawk list_students

Использование массивов и ассоциативных массивов

Элемент массива, как и в языках программирования более высокого уровня, определяется по индексу, имеющему целочисленное значение. Элемент ассоциативного массива определяется по индексу, имеющему **строковое** значение.

Массивы определеляются после первого использования индекса. Не требуется объявлять массив и задавать его границы. Массивы являются динамическими, их размерность увеличивается по мере добавления новых элементов.

Создадим массив фамилий нашего файла list_students, распечатаем его. Пока будем считать, что все фамилии разные. Размер массива равен количеству строк в файле.

$ gawk '{fam[n] = $1; print fam[n]; n++}'

А так можно сравнивать элемент массива с шаблоном:

$ gawk '(fam[n] = $1; fam[n] !~ /[Ии]ванов/) {print}' list_students

При работе с массивами и ассоциативными массивами используется функция split, последовательно разделяющая строку на элементы массива. В следующем примере каждая обрабатываемая строка разделяется на поля, которым присваиваются значения элементов массива ***massv***:

$ gawk '{split($0, massv); print massv[1]; massv[2], massv[3]}' list_students

Используя функцию ***split*** можно редактировать значения полей записей: ***$ gawk '{split("5", massv[5], " "); print massv[1]; massv[5]}' list_students***

Здесь в качестве аргументов функции ***split*** записаны по порядку: в двойных кавычках значение корректируемого поля, имя этого поля, знак символа-разделителя, пробел задается по умолчанию, его можно не указывать. Значений корректируемых полей может быть несколько, тогда они указываются через символ-разделитель. Например, мы хотим присвоить элеменам массива mass следующие значения: 3,4,5, указанные через символ-разделитель двоеточие, и распечатать первый и третий элементы массива:

$ gawk '{split("3:4:5", mass, ":"); print mass[1]; mass[3]}' filedata

При использовании строкового индекса его нужно указывать в двойных кавычках в элементе массива: {***print massiv["Bonal"]}***.

Управляющие структуры

Язык утилиты gawk дает возможность определять переменные, конструировать выражения, присваивать значения, включает управляющие структуры. Для выбора варианта предусмотрена управляющая структура **if**, для выполнения циклических действий: while, for, for-in. Разделителем действий этих структур является знак точки с запятой.

Управляющая структура **if** используется для выполнения действий, если выражение истинно, иначе производится действие, указанное после ключевого слова **else**. Синтаксис структуры **if** следующий:

```
if (условие)
{
действия 1
 }
else {
действия 2
 }
```

Структура может иметь сокращенную форму:
if (условие)
{
действия
}

```
if ($4 >= 256) $4=1.2 * $4
else $4=log($4);
print $4;
```

Управляющая структура while:
установка начального значения переменной цикла;
while (условие)
{
действия;
изменение переменной цикла;
}

Управляющая структура **for**:
for (инициализация; условие; приращение)
{
действия;
}
Пример.

```
for (i=1; ( i< 5); i++) {
    printf("%s\t", $1);
}
printf("\n");
```

Управляющая структура **for-in** предназначена для использования с ассоциативными массивами. В цикле for-in за ключевым словом for следует переменная, далее ключевое слово in, за ним - имя ассоциативного массива. Индексом такого массива является строковое значение.

Пример цикла в сценарии. *В ходе выполнения цикла переменной facultet последовательно присваиваются значения, являющиеся индексами массива familia. Распечатываютс элементы массива и текущий индекс.*

```
for (facultet in familia) {
    printf(familia[facultet],);
}
printf("\n");
```

Задание 1. Выполнить цикл, используя **while**.

Задание 2. Изучите материал работы, выполняя рекомендуемые задания. Для экспериментов создайте свои файлы, структурно похожие на приведенные в лабораторной работе для проверки соответствующих функциональных возможностей. Можно скопировать подходящие файлы, имеющие структуру таблиц из доступных системных каталогов, например, из /users или /tmp. Файлы, имеющие структуру таблиц - это файлы, состоящие из записей, разделенных на поля специальными символами: пробелами, табуляцией, двоеточиями или другими символами-разделителями. Ответить на контрольные вопросы преподавателя.

Задание 3. Просуммируйте длины первых слов в каждой строки текстового файла.

Лабораторное задание и порядок выполнения работы

Изучить материал, выполняя рекомендуемые примеры и задания.

1. Выполнить контрольное задание, описанное в конце работы, используя свои файлы и каталоги.
2. Кратко законспектировать материал по новым командам.
3. Оформить отчет и защитить работу.

Требования к отчету

Отчет должен содержать:

1. Описание последовательности команд создания своей файловой структуры (с Вашими именами файлов и каталогов).
2. Краткие сведения о работе и перечень опробованных в этой работе команд командного интерпретатора BASH.