Бухвалова Варвара Александровна,
Бухвалова Вера Вацлавовна

УДК 004.434

ПРИМЕНЕНИЕ ЯЗЫКА AWK ДЛЯ ОПЕРАТИВНОЙ ОБРАБОТКИ ЭКОНОМИЧЕСКОЙ ИНФОРМАЦИИ

Аннотация

В статье описана практика применения языка AWK для обработки и анализа экономической информации. В отличии от других работ, посвященных программированию на этом языке, в качестве примеров рассматриваются реальные задачи анализа и приводятся очень компактные их решения на языке AWK. Эти программы при минимальном изменении могут быть использованы в задачах анализа и обработки больших объемов данных в других областях.

Ключевые слова: анализ и обработка больших объемов информации, язык AWK, язык скриптов, формат CSV, регулярное выражение, шаблон.

1. ВВЕДЕНИЕ

Если перед нами стоит задача в одном или нескольких файлах сначала найти необходимую информацию, а потом выполнить с ней несложные вычисления, то возникает естественное желание использовать для этого какое-либо из доступного программного обеспечения. Авторы этой статьи и их ученики часто в подобных случаях используют язык программирования AWK [4, 3], хотя нельзя сказать, что этот язык является популярным. AWK удивительный язык программирования с несколько необычным синтаксисом. Чтобы проиллюстрировать это, начнем с двух программ на языке AWK, которые должны произвести впечатление на тех, кто впервые знакомится с этим языком.

Предположим у нас имеется текстовый файл со списком FT Global 500 за 2012 г.[1], и мы хотим узнать, какие компании из России вошли в этот список.

Программа на языке AWK, которая поможет нам получить ответ на этот вопрос, состоит всего из одной короткой строки:

```
$4~/Russia/{print $3}
```

А следующая программа, которая вычисляет, сколько компаний из России вошло в этот список, состоит из двух строк:

```
$4~/Russia/ {nr+=1}
     END {print nr}
```


[1] Газета «Financial Times», начиная с 1997 г., ежегодно составляет список FT Global 500 – 500 крупнейших по капитализации компаний в мире.

Никак не будем здесь комментировать эти программы (мы сделаем это позже, в разделе 3), а отметим только, что компактность программ связана с тем, что AWK *специально предназначен* для решения задач по обработке структурированных текстов.

Язык AWK не является единственным языком программирования, на котором эквиваленты программ, приведенных выше, и тех, которые мы будем обсуждать потом, также будут состоять из нескольких строк. Например, языки Perl и Python также поддерживают работу с регулярными выражениями и образцами и богаче (это нельзя отрицать) языка AWK по возможностям. Но они, без сомнения, уступают языку AWK *по читаемости* кода и компактности интерпретаторов (есть практически для всех платформ и их размер не более 400K). Поэтому обычно вам достаточно иметь с собой флэшку с интерпретатором, чтобы выполнять AWK-программы на любом компьютере. Образно говоря, AWK – *это язык, который всегда с тобой*.

Побудительных мотивов для написания этой статьи было два. Во-первых, уже много лет мы очень активно используем AWK для обработки экономической информации. Причем делается это с различными целями: подготовка информации для лекций по финансам, исследованию операций и информатике; подготовка данных для подстановки в различные модели; подготовка данных для подстановки в пакеты (например, MATLAB, SAS) при проведении дальнейших вычислений. В этих вопросах мы накопили определенный опыт, которым хотели бы поделиться. Во-вторых, имеющиеся многочисленные введения в AWK (например, [7]), в своем рассмотрении ограничиваются примитивными примерами применения, которые вряд ли подвигнут к использованию языка непрограммистов. Нам же представляется, что среди пользователей этого языка должны быть, по крайней мере, преподаватели широкого круга дисциплин.

Статья имеет следующую структуру. Раздел 2 посвящен краткому обзору языка AWK: история возникновения языка, характеристика имеющихся интерпретаторов, структура программы и основные элементы языка. Информации, имеющейся в этом разделе, должно быть достаточно для понимания программ из следующих разделов. В разделе 3 на примере файла со списком FT Global 500 за 2012 г. продемонстрировано, как выполнять поиск информации и ее простейшую обработку. Некоторые проблемы, которые могут возникнуть при подготовке файлов с данными, обсуждаются в разделе 4. В разделе 5 даются рекомендации по хранению коллекции программ на языке AWK. В заключении (раздел 6) приведен пример еще одного класса задач [1], для которых использование языка AWK также естественно, как и для задач, о которых идет речь в этой статье.

2. ЯЗЫК AWK: КРАТКИЙ ОБЗОР

Первоначально язык программирования AWK возник и был реализован в операционной системе UNIX. В этой системе имеются специальные программы – *программируемые фильтры*, которые читают входной поток (файлы аргументы или стандартный входной поток), выполняют простые операции над ним и записывают результат в выходной поток. Эти фильтры могут соединяться друг с другом в конвейеры (pipelines), использование которых может быть очень удобно. Одним из таких программируемых фильтров и является AWK. Имя программы составлено из первых букв фамилий ее создателей: А. Ахо (Alfred Aho), П. Вайнбергер (Peter Weinberger), Б. Керниган (Brian Kernighan)[1]. Программа AWK просматривает входной файл в поисках строк, которые соответствуют какому-либо из образцов, указанных в программе. Язык, на котором программируется поведение программы AWK, больше всего похож на язык программирования *C*: имеются аналогичные по форме записи условные операторы, циклы, переменные и функции пользователя. Сходство с языком *C* порождает

[1] Все три автора AWK считаются гуру программирования. Подробные сведенья о них приведены в [2]. Там же имеются их интервью, в которых они излагают свои взгляды на теорию и практику программирования.

некие трудности, но только для тех, кто привык программировать на *C*: некоторые конструкции в языке AWK отсутствуют, а некоторые отличаются от соответствующих конструкций языка *C*.

2.1. ИНТЕРПРЕТАТОРЫ

Язык для команды AWK (его также стали называть AWK) был по достоинству оценен программистами, и в 90-е годы прошлого века появились его интерпретаторы. Одним из первых в рамках проекта GNU был создан интерпретатор *mawk* (автор M. Brennan). Интерпретатор *mawk* имеет много ограничений и мы не советуем использовать его сейчас. В настоящее время наиболее популярны следующие версии языка AWK и интерпретаторы для них: *GNU AWK (gawk)* и *One True Awk (nawk, awk95)*. Все эти интерпретаторы имеются в свободном доступе в интернете. Для выполнения программ из этой статьи мы использовали сначала интерпретатор awk95 (написан в 2000 г. самим Б. Керниганом), а затем проверяли совпадение полученных результатов с результатами при использовании интерпретаторов, входящих пакет GnuWin32, v. 3.1.6-1.

В системе Windows запуск любого интерпретатора осуществляется командой

```
<интерпретатор> -f awkfile data1file data2file . . .
```

Текст программы читается из файла awkfile, а данные из файлов **data1file**, **data2file** и т. д.[1] Результат работы программы выводится в выходной поток. Можно переадресовать выходной поток в файл **resfile** обычным способом, например:

```
awk95 -f awkfile datafile > resfile
```

До начала сканирования первого входного потока интерпретатор проверяет программу (файл **awkfile**) на наличие в ней синтаксических ошибок. Если обнаруживается ошибка, то выдается диагностическое сообщение.

2.2. ЛИТЕРАТУРА О ЯЗЫКЕ AWK И ЕГО РЕАЛИЗАЦИЯХ

Для тех, кто только начинает программировать на AWK, лучшей по-прежнему остается книга авторов языка [6]. Дальнейшее обучение можно продолжить по книге [8]. Грамотному программированию на конкретных примерах с использованием разных языков программирования, в частности AWK, учит книга [5].

Много информации, связанной с программированием на языке в AWK (в частности, тексты руководств), имеется в интернете. Так как адреса такой информации быстро устаревают, приведем только два из них (домашняя страница Б. Кернигана и все про AWK из проекта GNU): http://cm.bell-labs.com/who/bwk/; http://www.gnu.org/software/gawk/.

2.3. СТРУКТУРА ПРОГРАММЫ И ЕЕ ЭЛЕМЕНТЫ

Программа на языке AWK представляет собой набор конструкций вида:

```
BEGIN {начальные действия}
шаблон {действия}
шаблон {действия}
.................
шаблон {действия}
END {конечные действия}
```

После того как интерпретатор языка AWK проверит программу на отсутствие синтаксических ошибок, выполняются начальные действия, приписанные к шаблону **BEGIN**. Далее

[1] При этом следует соблюдать следующее правило: если имя какого-либо файла в командной строке содержит пробелы, его необходимо заключить в двойные кавычки.

последовательно читаются все строки всех файлов с данными (если где-либо в действиях не предусмотрено изменение или прекращение этого процесса). В каждый момент доступна только одна строка – текущая.

В текущей строке ищутся шаблоны в том порядке, в котором они перечислены в программе. Если шаблон в строке обнаружен (*шаблон соответствует строке*), то со строкой выполняются действия, которые соответствуют этому шаблону.

После чтения и обработки всех строк выполняются конечные действия, приписанные к шаблону **`END`**. Шаблоны **`BEGIN`** и **`END`** могут быть опущены. Допускается опускать одну из частей конструкции шаблон действия. Действия без шаблона выполняется с каждой строкой входного потока. Если не указаны действия, то строки, соответствующие шаблону, копируются в выходной поток.

Комментарий может быть вставлен в конце любой строки. Он начинается с символа **`#`** и заканчивается в конце строки.

Поля. Каждая входная строка автоматически разбивается на поля. По умолчанию поля это последовательности символов без пробелов, разделенные пробелами и/или символами табуляции. Разделитель полей контролируется встроенной переменной **`FS`**. Присваивание в шаблоне **`BEGIN`** переменной **`FS`** любого символа, отличного от пробела, делает этот символ разделителем полей. Входные строки читаются по одной и сохраняются в переменной **`$0`**. Поля текущей строки от начала к концу присваиваются встроенным переменным **`$1`**, **`$2`**, ..., **`$NF`**, где **`NF`** – *встроенная* переменная, значение которой устанавливается равным числу полей в строке.

Типы данных. В языке AWK есть только два базовых типа данных: числа и строки символов. Способы записи числовых констант и набор операций с числами такие же как в популярных языках программирования. Имеется еще *строково-числовой* тип, данные которого имеют одновременно и числовые, и строковые значения. Какой именно тип использовать при выполнении операций, определяется контекстом. Переменные, не являющиеся встроенными, определяются самим фактом их использования. По умолчанию они инициируются строково-числовым значением **`null`**, числовым значением **`0`** и строковым значением **`""`** (пустая строка). Преобразование типов, когда это нужно, выполняется автоматически. Явного преобразования типов можно добиться конкатенацией с пустой строкой или прибавлением числа **`0`**. Рекомендуется избегать неявных преобразований типов.

Шаблоны. Наиболее типичное использование шаблонов в программах на языке AWK сводится к задачам простой проверки данных. Большинство из них немногим сложнее, чем поиск строк, не удовлетворяющих какому-либо критерию. Для таких программ отсутствие выходного потока свидетельствует о том, что все данные удовлетворяют указанному критерию.

В ряде случаев удобно использовать *составные шаблоны* выражения, которые образуются из других шаблонов с помощью скобок и логических операторов. Составной шаблон соответствует текущей входной строке, если выражение, его задающее, истинно (имеет значение **`true`**).

Ассоциативные массивы. Обычно мы представляем себе индекс массива как целое число, а в языке AWK в качестве индекса можно использовать любое значение (строку). Имеется специальная форма цикла **`for`** для перебора всех индексов ассоциативного массива. Такой цикл устанавливает значение переменной индекса равным каждому значению индекса поочередно. Однако порядок перебора индексов непредсказуем, поэтому может возникнуть необходимость в сортировке.

Встроенные функции. Как и в других языках программирования, в языке AWK имеются встроенные функции, предназначенные, прежде всего, для работы со строками и файлами. Далее в тексте будут примеры применения некоторых из них.

Приведенную в этом разделе информацию нельзя рассматривать как полноценное «введение в язык AWK». Более полную информацию можно найти в литературе из списка в конце статьи.

3. АНАЛИЗИРУЕМ СПИСОК FT GLOBAL 500

В этом разделе мы покажем, как с помощью языка AWK можно получать ответы на вопросы, которые часто возникают у экономистов, аналитиков и людей других специальностей при подготовке учебных курсов и различных материалов к ним. В качестве примера источника информации мы выбрали список FT Global 500 за 2012 г., в котором имеется информация о 500 крупнейших по рыночной капитализации компаний в мире. Мы будем считать, что имеется текстовый файл, в котором эта информация уже представлена в удобном для работы с языком AWK виде: 500 строк, в которых содержится информация о странах в порядке убывания рейтинга; нет лишних строк (например, строк с заголовками); каждая строка состоит из 14 полей, которые разделены символом `";"`; пустой графе в таблице соответствует поле пустая строка (`""`). На самом деле, часто приходится преобразовывать файл с данными, чтобы привести его к такому виду. О том, как это можно сделать мы расскажем в следующем разделе.

Напомним, что интерпретатор использует в качестве разделителя полей значение (строка) встроенной переменной **`FS`**. Поэтому, чтобы программы из этого раздела работали корректно, необходимо сообщить интерпретатору в шаблоне **`BEGIN`**, что разделителем полей в файле с данными является строка `";"`:

```
BEGIN {FS = ";"}
```

Для того чтобы представлять, какая информация о компаниях хранится в файле приведем таблицу с исходными названиями граф:

Column	Name	Column	Name
1	Global rank 2012	8	Net Income $m
2	Global rank 2011	9	Total assets $m
3	Company	10	Employees
4	Country	11	Price $
5	Sector	12	P/E ratio
6	Market value $m	13	Dividend yeild (%)
7	Turnover $m	14	Year End

Анализ информации мы организовали в форме «вопрос-ответ», где ответом будет программа на AWK. Так как во всех примерах речь идет о списке FT Global 500 в 2012 г., мы далее, в целях экономии места, будем ссылаться на него как на FT Global 500. И начнем мы с вопроса из введения к этой статье.

Вопрос 1. Какие компании из России вошли в FT Global 500 и каковы их рейтинги?

Решение. Российским компаниям соответствуют строки, у которых в поле 4 (Country) написано **`Russia`**. Названия компаний записаны в поле 3 (Company), а их рейтинги (места в порядке возрастания) в поле 1 (Global rank 2012). Поэтому ответ на поставленный вопрос будет получен после выполнения следующей программы на AWK:

```
$4~/Russia/ {print $1 ": " $3}
```

В результате выполнения этой программы будет создан файл, состоящий из 10 строк (10 компаний). Для экономии места запишем его содержимое в 3 строки, разделив названия компаний запятыми:

```
31: Gazprom, 79: Rosneft, 86: Sberbank of Russia, 132: Lukoil,
180: Novatek, 182: Surgutneftegas, 222: Norilsk Nickel,
343: Gazprom Neft, 366: VTB Bank, 374: Uralkali
```

Так как мы уверены, что строки, содержащие шаблон **/Russia/**, относятся только к российским компаниям, то возможен сокращенный вариант записи шаблона:

```
/Russia/ {print $1 ": " $3}
```

Для того, чтобы ответить на этот же вопрос о Норвегии, внесем изменение в программу:

```
/Norway/ {print $1 ": " $3}
```

и получим список из 3 компаний:

```
65: Statoil, 278: Telenor, 430: DNB
```

Замечание. Приведенный нами текст программы не является единственно возможным. Даже в некоторых руководствах по AWK подобные примеры предлагается записывать так:

```
{if ($4=="Russia") {print $1 ": " $3}}
```

Однако, такая форма записи, будучи синтаксически верной, нарушает стиль AWK-программирования. Она менее наглядна и не использует имеющуюся возможность возможность писать шаблоны, выделяя сразу строки, с которыми будут выполняться действия.

Так как интересна динамика изменения рейтинга для России, получим ответ на следующий вопрос.

Вопрос 2. Каких компаний из России не было в FT Global 500 в 2011 г. и каков их рейтинг в 2012 г.?

Решение. У тех компаний, которые не входили в список в 2011 г., поле 2 (Global rank 2011) является пустой строкой:

```
($4~/Russia/)&&($2=="") {print $1 ": " $3}
```

После выполнения этой программы мы установим, что в FT Global 500 в 2011 г. не было «Уралкалия», который в рейтинге 2012 г. занимает 374 место:

```
374: Uralkali
```

Что касается Норвегии, то, выполнив аналогичную программу, мы узнаем, что все три норвежские компании были в рейтинге оба года (файл с результатами содержит 0 строк).

Для оценки информации в целом, определим, страны каких компаний вошли в рейтинг и сколько компаний у этих стран в списке.

Вопрос 3. Какие страны вошли в FT Global 500 и сколько их компаний в этом списке?

Решение. Мы воспользуемся для ответа на этот вопрос *ассоциативным массивом*, которому мы присвоим имя **ncom**. Индексами в этом массиве будут сроки названия стран. Значения элементов массива количество компаний в списке из страны-индекса. Для перебора всех значений индекса (в программе он обозначен **cntr**) ассоциативного массивом (к сожалению, в неизвестном порядке) существует специальная форма оператора цикла **for.** Текст программы, которая выводит необходимую информацию, гораздо короче текста этих пояснений:

```
   {ncom[$4]+=1} # в строке компания из страны $4
END{for(cntr in ncom) print cntr, ncom[cntr]}
```

После выполнения этой программы получаем файл из 38 строк (38 стран). Приведем только первые и последние 3 страны, чтобы показать, что порядок перебора индекса, действительно, непредсказуемый:

```
Finland 2, Denmark 2, Taiwan 5,
........
India 12, South Africa 7, Netherlands 5
```

Для того чтобы убедиться, что полученный последней программой список имеет правильный размер, найдем ответ на следующий вопрос.

Вопрос 4. Компании скольких стран вошли в FT Global 500?

Решение. Ответ на этот вопрос получим, выполнив следующую программу:

```
ncom[$4]==0 {ncom[$4]=1; n+=1}
# компания из страны $4 встретилась первый раз
END{print "Количество стран в FT Global 500 (2012) = " n}
```

Выходной поток этой программы состоит из одной строки:

```
Количество стран в FT Global 500 (2012) = 38
```

Ответы на следующие два вопроса не только содержат полезную информацию, но и демонстрируют использование еще двух операторов AWK: next и exit.

Вопрос 5. Какая компания из FT Global 500 масимально улучшила свой рейтинг в 2012 г.?

Решение. Для того чтобы не писать длинные шаблоны и ускорить поиск ответа на поставленный вопрос, будем пропускать строки, относящиеся к компаниям, которые не входили в рейтинг в 2011 г. или которые не улучшили свой рейтинг в 2012 г. Если будут выполнены условия соответствующего составного шаблона, будем сразу переходить к анализу следующей строки (оператор **next**). Среди оставшихся строк выберем строку с максимальным ростом: **max ($2-$1).** Мы вывели для найденной компании TJX Companies (в таблице указан ее торговый код TJX Cos)[1] информацию о стране, секторе и величине роста рейтинга:

```
($2==""")||($2 <=$1){next} # ненужные строки
   ($2 - $1)> max {max=$2-$1; com = $3; cntr = $4; sec = $5}
        END{print "Лучший результат: "
           com " (" cntr ", " sec ", изменение = " max ")"}
```

Выходной поток этой программы состоит из одной строки:

```
Лучший результат: TJX Cos (US, General retailers, изменение = 223)
```

Оператор **exit** используется, когда необходимая информация найдена и требуется прервать просмотр оставшихся строк файла с данными. Определим теперь, какой банк занимает самую высокую позицию в рейтинге.

Вопрос 6. Какой банк из FT Global 500 имеет самый высокий рейтинг?

Решение. Так как компании в списке перечислены в порядке убывания рейтинга, то необходимо определить первый встретившийся в списке банк. После этого просмотр файла следует остановить и вывести полученную информацию:

```
$5~/Banks/{com = $3; cntr = $4; r = $1; exit}
END{print "Лучший банк (2012): " com " (" cntr ", rank=" r ")"}
```

Теперь мы точно знаем, что лучший банк в 2012 г. это китайский банк «Industrial & Commercial Bank of China» и он занимает шестую позицию в списке:

```
Лучший банк (2012): Industrial & Commercial Bank of China (China, rank=6)
```

Просматривая исходные таблицы, мы обнаружили, что, несмотря на всемирную борьбу с курением, в списке имеются табачные компании.

Вопрос 7. Сколько табачных компаний имеется в FT Global 500?

[1] TJX Companies одна из крупнейших торговых сетей в мире (одежда и предметы интерьера эконом класса). Компания имеет магазины в США, Канаде, Европе и владеет известными брендами T.J. Maxx, Marshalls, Winners, HomeSense.

Решение. Для того, чтобы ответить на этот вопрос, достаточно знать, что табачным компаниям соответствует сектор **Tobacco**:

```
$5~/Tobacco/{nt+=1}
      END{print "Табачные компании (2012): " nt " (" nt/500*100 "%)"}
```

Теперь мы точно знаем, что в списке 8 табачных компаний, что составляет 1.6 % от общего числа компаний:

```
Табачные компании (2012): 8 (1.6%)
```

Так как нам было интересно, какова динамика, мы проанализировали список FT Global 500 в 2011 г. и получили ответ:

```
Табачные компании (2011): 7 (1.4%)
```

Какой неожиданнй результат: несмотря на борьбу во всем мире с курением, количество табачных компаний в списке FT Global 500 в 2012 г. увеличилось!

Заметим для сравнения, что количество банков за это же время уменьшилось на 4. Проверим теперь, как изменилась за эти два года общая капитализация табачных компаний.

Вопрос 8. Какова доля общей капитализации табачных компаний в FT Global 500 в 2011 и 2012 гг.?

Решение. Для того, чтобы ответить на этот вопрос, напомним, что капитализация компаний (**Market value**) записана в шестом поле:

```
            {sum=+$6}
$5~/Tobacco/{sum_tbc+=$6}
        END{
print "Доля капитализации табачных компаний (2011): " sum_tbc/sum*100 " %" }
```

Выполнив эту программу с двумя файлами данных, мы установим, что и доля общей капитализации табачных компаний также заметно увеличилась:

```
Доля капитализации табачных компаний (2011): 1.42%
Доля капитализации табачных компаний (2012): 1.95%
```

Для более обстоятельного анализа часто приходится одновременно использовать информацию, которая хранится в нескольких файлах. Чтобы показать,что и в этих случаях программы на языке AWK обычно занимают несколько строк, ответим на следующий вопрос.

Вопрос 9. Какие компании из FT Global 500 в 2011 г. не вошли в этот список в 2012 г.?

Решение. Для ответа на этот вопрос нам понадобятся два файла с данными, которые имеют равное число строк (500). Чтобы различать, из какого файла текущая строка, будем использовать встроенную переменную **NR** порядковый номер текущей строки. Если **NR <= 500**, то текущая строка из FT Global 500 в 2011 г., при **NR > 500** из FT Global 500 в 2012 г. При просмотре первого файла с данными составляем список компаний с их рейтингами, а при просмотре второго файла отмечаем в нем компании, которые вошли в оба списка:

```
BEGIN {
print "Какие компании из FT500 Global в 2011 не вошли в него в 2012?"}
NR<=500{rank[$3]=$1} # список 2011
NR >500{delete rank[$3]} #список 2012
END{for(com in rank)
    rank[com] ": " com}
```

Мы не приводим полученный список только из-за экономии места (в нем 54 компании). Большинство этих компаний располагались с нижней части списка 2011 г. Исключение составила французская фармацевтическая компания Sanofi-Aventis, которая занимала 59 место в 2011 г. Но оказалось, что исключение таковым не является: в мае 2011 г. компания поменяла имя на Sanofi и занимает все то же 59 место в FT Global 500 в 2012 г.

4. ПОДГОТОВКА ФАЙЛОВ С ДАННЫМИ

В этом разделе мы на примере покажем, какие могут возникнуть сложности при подготовке файлов с данными для программ на языке AWK и как их преодолевать.

Как уже отмечалось выше, язык AWK предназначен, прежде всего, для обработки структурированных текстовых файлов. При этом предполагается, что текстовый файл это файл, содержащий информацию в виде последовательности текстовых символов, разделенных символами конца строки. В системах Windows признаком конца строки являются два идущих подряд *непечатаемых* символа: **<LF>** (ASCII код 10) и **<CR>** (ASCII код 13). В системе Unix признаком конца строки является один символ: **<LF>**. По соглашению, заимствованному из языка C, символ перевода строки в программах на языке AWK изображается как **\n**. В текстовых файлах применяются также символы табуляции (ASCII код 9) и перевода страницы (ASCII код 12).

Самый распространенный текстовый формат, предназначенный для представления табличных данных, называется CSV (Comma Separated Values). Например, в таком виде доступна большая часть информации, распространяемой Организацией Экономического Сотрудничества и Развития (ОЭСР). В формате CSV каждой строке таблицы соответствует строка файла. Значения отдельных граф таблицы разделяются специальным символом. Раньше в качестве этого символа использовалась только запятая, но в настоящее время стандарт CSV допускает использование и других символов в качестве разделителя. В частности, если запятая используется в таблице, то в качестве разделителя в файле часто используется точка с запятой.

В разделе 3 мы анализировали информацию из списка FT Global 500 за 2012 г.

На сайте www.ft.com мы получили свободный доступ к этой информации в виде pdf-файла, состоящего из 14 страниц мелкого шрифта. Каждая из этих страниц – таблица, в которой 14 граф с заголовками. Количество строк в этих таблицах несколько меняется от страницы к странице.

Начали мы с того, что использовали программу ABBYY FineReader для перевода pdf-файла в CSV-формат[1]. Так как знак «,» используется в таблице для разделения групп разрядов, то в качестве разделителя полей мы использовали знак «;» (предлагается по умолчанию).

Далее мы выполнили дополнительное преобразование полученного CSV-файла, использовав для этого все тот же язык AWK. Так как каждая из 14 страниц pdf-файла начиналась с заголовка страницы, а каждая таблица имела заголовки граф, то эта информация попала и в CSV-файл. Мы оставили только содержательные строки таблиц, у которых первое поле – это целое число из интервала [1; 500]. В том случае, если содержимое графы таблицы состояло из нескольких слов (например, Sberbank of Russia), то после перевода соответствующее поле строки заключалось в кавычки («Sberbank of Russia», но Gazprom). Так как эти кавычки являются ненужными, можно удалить их с помощью встроенной функции **gsub(s_del,s_in)** (**s_del** – удаляемый образец, **s_in** – его замена).

[1] Тех, кто решит конвертировать pdf-файл в txt-файл с помощью программы Adobe Acrobat предупредим, что возникнут сложности с пустыми графами таблицы. Они будут проигнорированы и установить, какая именно графа была пустой, будет невозможно.

Одновременное удаление лишних строк и кавычек было выполнено с помощью следующей программы на языке AWK:

```
($1>=1)&&($1<=500){gsub(/"/, ""); print}
```

Именно это преобразование файла с данными способствовало тому, что AWK-программы из раздела 3 такие компактные и не содержат дополнительных проверок.

5. ХРАНЕНИЕ КОЛЛЕКЦИИ ПРОГРАММ

В этом разделе мы опишем простейший способ хранения коллекции программ на языке AWK, относящихся к обработке одних и тех же данных.

На примере анализа списка FT Global 500 (раздел 3) мы показали, что программа ответа на типичный вопрос занимает не более 5 строк, а вопросов, на которые захочется получить ответ, много. Если хранить каждую такую программу в отдельном файле, то придется придумывать правило, по которому можно будет по имени файла определять, какому вопросу он соответствует.

Гораздо удобнее хранить все программы в *одном* файле, разделяя программы строками комментариев. Такой список «вопрос-ответ» будет наглядным и его будет удобно редактировать (добавлять новые вопросы, корректировать и удалять старые). Такой файл должен начинаться с шаблона **BEGIN**, который относится ко всем программам из файла. Напомним, что для программ из раздела 3 в нем задается символ, который разделяет поля входных строк:

```
BEGIN {FS = ";"}
```

В каждую программу можно включить дополнительный шаблон **BEGIN** для печати текста вопроса. Для получения ответа на конкретный вопрос все строки программ, кроме тех которые соответствуют программе-ответу на этот вопрос, надо сделать комментариями. В качестве примера приведем фрагмент, который объединяет ответы на вопросы 3 и 4. Расстановка знаков комментариев соответствует печати вопроса 3 и получению ответа на него:

```
BEGIN {FS = ";"}
#3:
BEGIN {print "Какие страны вошли в список FT Global 500\n
          и сколько их компаний в этом списке?"}
      {ncom[$4]+=1} #следующая компания страны $4
END{for(cntr in ncom) print cntr, ncom[cntr]}

#4:
#BEGIN {print "Компании скольких стран вошли в список FT Global 500?"}
#ncom[$4]==0 {ncom[$4]=1; n+=1}# страна $4 первый раз
#END{print "Количество стран в FT 500 (2012) = " n}
```

Возможно получение одновременного ответа на два или более вопросов, но это потребует более сложного редактирования файла с программами.

6. ЗАКЛЮЧЕНИЕ

В статье описано только одно направление использование языка AWK – оперативная обработка экономической информации. На самом деле применение языка целесообразно во многих случаях, когда есть возможность получить информацию в виде структурированного текстового файла. Несколько лет назад авторы предприняли попытку обучить использованию языка AWK для обработки спутниковой альтиметрической информации. Целью было получение статистически обеспеченных параметров волн Россби [1]. В данном случае речь

шла о волнах Россби в океане, играющих важную роль в формировании крупномасштабных океанских циркуляций. К сожалению, хотя первый опыт показал удобство применения языка AWK для обработки такого рода информации, работа не была доведена до конца.

За рамками нашей статьи осталась технология программирования на языке AWK, когда информация расположена не в одном, а многих файлах. При этом имеется возможность запрограммировать не только процесс обработки информации, но и процесс ее получения из интернета. Этим вопросам авторы планируют посвятить отдельную статью.

Авторы выражают благодарность профессору И.В. Романовскому, от которого когда-то давно они узнали о существовании языка AWK и который помогал им в процессе работы над этой статьей своими советами и замечениями.

Литература

1. *Белоненко Т.В.* Наблюдения волн Россби в северо-западной части Тихого океана // Современные проблемы дистанционного зондирования земли из космоса. 2012. Т. 9, № 3. С. 209–215.

2. *Бьянкуцци Ф., Уорден Ш.* Пионеры программирования. Диалоги с создателями наиболее популярных языков программирования. Символ-Плюс, 2011.

3. *Бухвалова В.В.* Язык программирования AWK (методическое пособие для студентов математико-механического факультета). СПб: СПбГУ, 2001.

4. *Бухвалова В.В., Никандрова О.В.* Язык программирования AWK как инструмент обработки экономической информации // Математические модели и информационные технологии в менеджменте. Вып. 1. СПб: СПбГУ, 2001. С. 94–102.

5. *Керниган Б., Пайк Р.* Практика программирования СПб.: Невский Диалект, 2001.

6. *Aho A., Kernighan B., Weinberger P.* The AWK Programming Language Addison-Wesley Publishing, 1988.

7. *Capecchi R.* 6 Tricks with awk (18.01.2011) http://linuxaria.com/article/6-trucchi-con-awk (дата обращения 29.06.2013).

8. *Robbins A.* Effective AWK Programming. O'Reilly Media, 2001.

AWK AS A TOOL FOR PROCESSING ECONOMIC DATA

Abstract

The paper introduces simple methods for turning financial information into financial data using AWK. What sets this article apart from other articles on programming in AWK is the emphasis on specific and practical examples of data analysis. The examples happen to have very simple and concise solutions in AWK. These examples can be easily extended and used for data processing in other fields.

Keywords: Parsing Large Datasets, AWK, Script Language, CSV-files, Regular Expression, Pattern.

Бухвалова Варвара Александровна,
PhD, Associate Professor,
Department of Financial Economics,
BI Norwegian Business School,
barbara.bukhvalova@bi.no,

Бухвалова Вера Вацлавовна,
кандидат физико-математических наук, доцент кафедры исследования операций СПбГУ,
vera_cut@mail.ru.