

Донник А. М.

Операционные системы

Практикум

САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Г. ЧЕРНЫШЕВСКОГО

Донник А. М.

Операционные системы

Практикум

*Учебное пособие для студентов, обучающихся
по направлениям подготовки бакалавриата
09.03.03 «Прикладная информатика»,
38.03.05 «Бизнес-информатика»,
02.03.01 «Математика и компьютерные науки»*

Саратов
ООО Издательский Центр «Наука»
2016

УДК 004.451(075.8)
ББК 32.973–018.2я73
Д67

Донник А.М.

Д67 **Операционные системы. Практикум:** Учебное пособие для студентов, обучающихся по направлениям подготовки бакалавриата 09.03.03 «Прикладная информатика», 38.03.05 «Бизнес-информатика», 02.03.01 «Математика и компьютерные науки» /А.М. Донник. — Саратов: ООО Издательский Центр «Наука», 2016. — 40 с.

ISBN 978-5-9999-2727-9

Учебное пособие представляет собой набор практических работ по дисциплине «Операционные системы». Практические работы направлены на знакомство с управлением операционной системой Linux посредством консольных команд от наиболее простых и распространённых до уровня написания bash-скриптов средней сложности.

Для студентов, обучающихся по направлениям подготовки бакалавриата 09.03.03 «Прикладная информатика», 38.03.05 «Бизнес-информатика», 02.03.01 «Математика и компьютерные науки», также может быть полезно студентам других направлений подготовки бакалавриата для базовой подготовки в области информационных технологий и администрирования компьютерных систем.

Рецензенты:

Кафедра математического и компьютерного моделирования
Саратовского национального исследовательского государственного
университета имени Н.Г. Чернышевского

Зав. кафедрой, д.ф.-м.н., профессор **Ю.А. Блинков**

УДК 004.451(075.8)
ББК 32.973–018.2я73

Работа издана в авторской редакции

ISBN 978-5-9999-2727-9

© А.М. Донник, 2016

Содержание

Практическая работа 1. Знакомство с терминалом Linux.....	4
Практическая работа 2. Файлы	7
Практическая работа 3. Фильтрация вывода.....	9
Практическая работа 4. Использование перенаправлений и конвейера (программного канала)	17
Практическая работа 5. Текстовый процессор awk.....	34
Список литературы	40

Практическая работа 1. Знакомство с терминалом Linux

Выполнение работы

Данная практическая работа не подразумевает индивидуальных вариантов заданий, выполняется под управлением преподавателя.

Запустить программу терминал Linux. В графической оболочке Linux вы можете сделать это командой `konsole` (для KDE). Для операционных систем семейства Windows потребуется утилита `putty` (или иной SSH-клиент) и компьютер с ОС Linux, предоставляющий доступ посредством SSH. Будет выдано приглашение для работы:

```
student@r111wslin01 ~ $
```

Здесь символ `~` (тильда) – указывает, что текущий каталог является домашним каталогом пользователя; `$` (доллар) – означает что данный пользователь не является администратором системы `root`, т.к. для `root` приглашение выглядит иначе, вместо знака `$` (доллар) отображается знак `#` (решётка).

По каждой команде можно прочитать справочную информацию, набрав следующие команды

```
имя_команды --help  
man имя_команды
```

В первом случае вы получите краткую справку, которая будет распечатана в консоль, во втором – документ с полной справкой по команде (документ встроенной системы документирования команд Linux).

С помощью стрелок вверх и вниз на клавиатуре, можно, не набирая заново команды, использовать уже ранее набранные, это позволяет делать оболочка `bash`, так как она имеет возможность сохранять историю команд.

1. Создание каталогов в оболочке `bash`. Команда `mkdir` – позволяет создать каталог

```
student@r111wslin01:~$ mkdir Ivanov
```

Чтобы создать каталог в каталоге:

```
student@r111wslin01:~$ mkdir Ivanov/hobby
```

Настоятельно рекомендуется первой командой создать каталог, соответствующий своей фамилии и дальнейшие действия выполнять уже в нём.

2. Команда `pwd` — позволяет узнать имя текущего каталога.
3. Команда `cd` — позволяет осуществить переход в другой каталог

```
student@r111wslin01:~$ cd /usr/bin  
student@r111wslin01:/usr/bin$
```

Чтобы удостовериться в том, что действительно выполнено перемещение в заданный каталог, использовать команду `pwd`.

`cd ..` – позволяет подняться на уровень выше.

`cd /` – переход в корневой каталог.

4. Команда `ls` – позволяет просмотреть список файлов, которые находятся в выбранном для просмотра каталоге.

`ls` – выдает список файлов текущего каталога

`ls ~` – выдает содержание домашнего каталога пользователя

Обратите внимание:

`.bash-history` – файлы, начинающиеся с точки – это скрытые файлы.

`.` – одна точка – ссылка на текущий каталог

`..` – две точки – ссылка на каталог на уровень выше

Любая команда очень часто имеет дополнительные ключи.

`ls -a` (all) – позволяет увидеть все файлы в каталоге

`ls -l` (long) – позволяет увидеть файлы с дополнительными сведениями

`ls -la` можно использовать дополнительные ключи, совместно распечатается таблица. Внимательно изучите сведения, предоставляемые этой таблицей

5. Команда `clear` – очистка экрана

6. Команда `ps` – даёт возможность посмотреть, какие процессы в данный момент выполняются.

```
student@r111wslin01:~$ ps
```

```
student@r111wslin01:~$ ps -aux
```

– расширенный список

Внимательно изучите назначение каждой колонки в таблице, выдаваемой командой `ps -aux`.

7. Команда `cat` – позволяет посмотреть файл на экране, например:

```
student@r111wslin01:~$ cat 5.txt
```

8. Создать пустой файл в оболочке `bash` можно с помощью команды `dd`:

```
student@r111wslin01:~$ dd if=/dev/null of=646.txt
```

```
0+0
```

```
0+0
```

```
0 bytes transferred in 0.103982 seconds
```

– значит файл был успешно создан

Выясните, какие ещё возможности имеются у команды `dd`, а также, что такое `/dev/null`.

9. Команда `rm` – используется для удаления файла

```
student@r111wslin01:~$ rm 646.txt
```

удаляет ранее созданный в домашнем каталоге файл, если файл находится в другом каталоге, необходимо полностью указать путь к нему либо перейти сначала в нужный каталог. Выясните, как удалить несколько файлов списком или по шаблону.

10. Команда `rmdir` – используется для удаления каталогов

```
student@r111wslin01:~$ rmdir Ivanov
```

С помощью `rmdir` нельзя удалить файл, появляется следующее сообщение `Not a directory` (каталог не найден). Выясните, как удалить непустой каталог.

11. Команда `echo` текст — распечатывает любой текст. Если использовать `echo текст >1.txt`

введённое слово будет занесено в файл `1.txt`. Команда `echo $SHELL` — позволяет узнать какая оболочка запущена.

12. Команда `man` — вызывает страницы руководства (интерактивная система помощи). `q` — выход из программы просмотра
`man ls`

Справку можно получить также, используя ключ `--help`
`student@r111wslin01:~$ ln -help`

13. Команда `cp` — позволяет осуществлять копирование
`student@r111wslin01:~$ cp <источник> <назначение>`
`student@r111wslin01:~$ cp 646.txt viva.txt`

14. Команда `grep` — поиск в файлах и каталогах
`student@r111wslin01:~$ grep <шаблон> <файл>`
`student@r111wslin01:~$ grep perl 646.txt`

на экране будет распечатано искомое содержимое в файле

15. Команда `ln` — создание жёстких связей с файлом того же раздела
`student@r111wslin01:~$ ln <источник> <назначение>`
`student@r111wslin01:~$ ln 2323.txt 3333.txt`

Команда `ln` с ключом `s` — создание символических ссылок. Выясните разницу между жёсткими и символическими ссылками.

16. Команда `mv` — переименование файлов и каталогов
`student@r111wslin01:~$ mv Ivanov Petrov`

17. Команда `df` — выводит на экран информацию о свободном дисковом пространстве.

18. Команда `du` — выводит на экран информацию о занятом дисковом пространстве.

19. Команда `free` — выводит на экран информацию об использовании памяти.

20. Команда `who` — выводит список пользователей, подключённых к системе.

21. Команда `whoami` — выводит имя пользователя.

22. Команда `id` — выводит детальную информацию о пользователе.

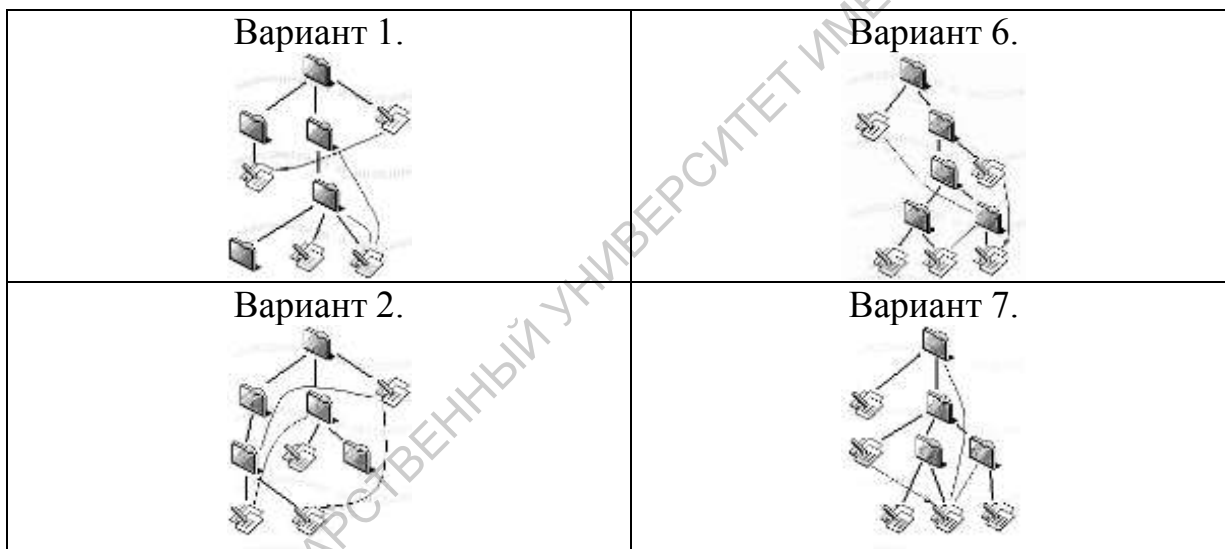
23. Команда `which` — позволяет определить месторасположение любой программы, например, `which pwd`.

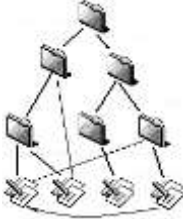
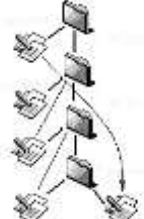
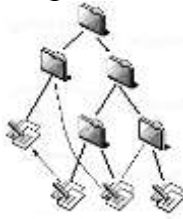
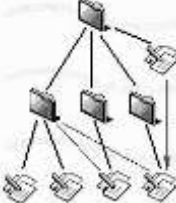
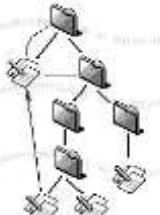
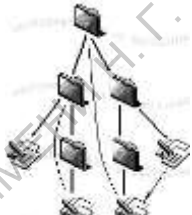
Практическая работа 2. Файлы

Выполнение работы

Создайте структуру каталогов, ссылки и символические ссылки, соответствующие варианту индивидуального задания. Выполните указанные в задании действия над каталогами, файлами, ссылками. Изучите команды изменения прав доступа и владения `chmod` и `chown`. Проведите изменения прав доступа к файлам и каталогам. Освойте работу с файловым менеджером `Midnight Commander` (запускается командой `mc` в терминале).

Создайте структуру каталогов в соответствии с вариантом. Черными линиями представлена вложенность файлов/подкаталогов в каталоги. Синими линиями представлены ссылки. Красными линиями – символические ссылки. Стрелка на красной линии указывает на целевой файл ссылки. Файлы создаются копированием ранее созданного файла командой `cp` с внесением в копии некоторых изменений. Ссылки создаются командой `ln`, символические ссылки – ей же, но с ключом `-s`:



<p>Вариант 3.</p> 	<p>Вариант 8.</p> 
<p>Вариант 4.</p> 	<p>Вариант 9.</p> 
<p>Вариант 5.</p> 	<p>Вариант 10.</p> 

Для всех вариантов выполнить следующие действия:

1. Создать ссылки (синие линии).
2. Создать символические ссылки (красные линии).
3. Сохранить протокол выполненных действий.
4. Провести ряд экспериментов, иллюстрирующих доступ к файлам по основным именам, по ссылкам и по символическим ссылкам. Для доступа использовать команду `cat` или редактор `vi`.
5. Провести ряд экспериментов, иллюстрирующих реакцию системы на удаление файла, на который имеются ссылки, и файла, на который имеются символические ссылки. Проверять результаты командой `ls -la`.
6. Уничтожить созданные подкаталоги и файлы в них, используя команды `rmdir` и `unlink`, сохранив, однако, файл, созданный в пункте 5.1.7 и одну его рабочую копию в домашнем каталоге.
7. Изучить справку к командам `chmod` и `chown`.
8. Открыть для своей группы доступ к своему домашнему каталогу – для поиска в каталоге и к рабочей копии файла в домашнем каталоге – для чтения и записи.
9. Послать своему партнеру сообщение об открытии доступа, указав в нем имя своего каталога и файла в нем.
10. Получив от своего партнера аналогичное сообщение, выполнить попытку чтения файла в каталоге партнера, а затем – внесения изменений в этот файл.

11. Послать своему партнеру сообщение о том, что в его файл внесены изменения.
12. Получив от партнера аналогичное сообщение, прочитать свой файл и найти в нем изменения, сделанные партнером.
13. Закрывать доступ к своему домашнему каталогу.
14. Изучить справку к файловому менеджеру Midnight Commander, запустить его, изучить перечень доступных команд, сочетания клавиш для выполнения часто применяемых команд, особенности встроенного текстового редактора.

Практическая работа 3. Фильтрация вывода

Выполнение работы

Возможность обрабатывать информацию с использованием регулярных выражений представляют собой одно из наиболее полезных свойств программ операционных систем семейства *nix. Регулярные выражения являются языком описания текстовых шаблонов, который используется во многих системных утилитах для выполнения операций поиска и отбора при разнообразных обработках текстовых строк. Мы начинаем изучать регулярные выражения с применения их в утилите поиска grep.

В каталоге <http://nto.immpu.sgu.ru/p3/> имеются пять текстовых файлов с именами query1, query2, query3, query4 и query5. Файлы содержат структурированный текст. Структура файлов следующая:

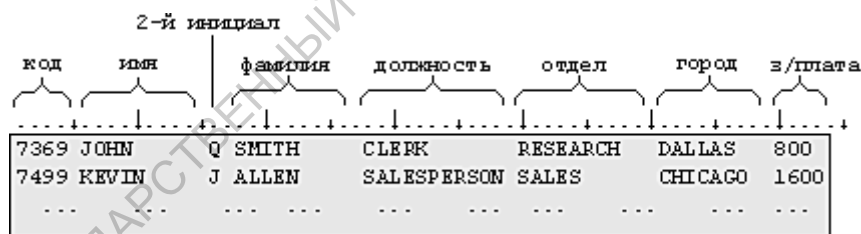


Схема 1. Структура файла query1

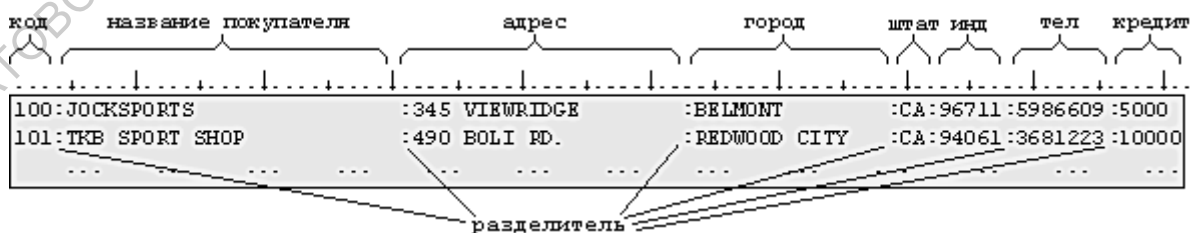


Схема 2. Структура файла query2

код	название товара	макс. цена	мин. цена	дата
100890	!ACE TENNIS NET	!58	!46.4	!01-JAN-89
100860	!ACE TENNIS RACKET I	!35	!28	!01-JUN-90
...

разделитель

Схема 3. Структура файла query3

N заказа	продавец	код покупателя	дата	сумма
612	ALLEN	104	15-JAN-91	5860
605	WARD	106	14-JUL-90	8374
...

Схема 4. Структура файла query4

№ пункта заказа	№ заказа	код товара	реальная цена		общая сумма
			количество	цена	
1	612	100860	30	100	3000
2	612	100861	40.5	20	810
...

Схема 5. Структура файла query5

Используя команду `grep`, выполните поиск в файлах строк, соответствующих определенным критериям. Имена исходных файлов и критерии поиска приводятся в Вашем варианте индивидуального задания. Используя команду `grep`, выполните поиск в файлах строк, соответствующих определенным критериям. Имена исходных файлов и критерии поиска приводятся в Вашем варианте индивидуального задания.

Варианты заданий

Вариант 1

1. В файле `query1` выбрать все строки, в которых имя сотрудника начинается на букву 'R'.
2. В файле `query3` выбрать все строки, в которых в названии есть слово 'TENNIS', а цена установлена в 1990 г.
3. В файле `query4` выбрать все строки, в которых фамилия продавца – 'DUNCAN'.

Вариант 2

1. В файле query4 выбрать все строки, в которых код покупателя – 201.
2. В файле query2 выбрать все строки, в которых индекс начинается с '11'.
3. В файле query3 выбрать все строки, в которых минимальная цена не меньше 10.

Вариант 3

1. В файле query1 выбрать все строки, в которых фамилия сотрудника начинается на букву 'M'.
2. В файле query4 выбрать все строки, в которых сумма не имеет копеек.
3. В файле query3 выбрать все строки, в которых минимальная цена меньше 10, а максимальная цена не меньше 10.

Вариант 4

1. В файле query4 выбрать все строки, в которых фамилия продавца заканчивается буквой 'N'.
2. В файле query1 выбрать все строки, в которых должность – 'MANAGER', а отдел – 'SALES'.
3. В файле query3 выбрать все строки, в которых максимальная цена не меньше 20.

Вариант 5

1. В файле query1 выбрать все строки, в которых код заканчивается цифрами '69'.
2. В файле query2 выбрать все строки, в которых в названии есть 'SPORT'.
3. В файле query4 выбрать все строки, в которых сумма не меньше 1000, но меньше 2000.

Вариант 6

1. В файле query3 выбрать все строки, в которых код заканчивается цифрой '1'.
2. В файле query2 выбрать все строки, в которых в адресе номер дома – '2'.
3. В файле query2 выбрать все строки, в которых кредит не меньше 10000.

Вариант 7

1. В файле query2 выбрать все строки, в которых код заканчивается цифрой '8'.
2. В файле query4 выбрать все строки, в которых N заказа не содержит цифры '4'.
3. В файле query1 выбрать все строки, в которых зарплата меньше 1000.

Вариант 8

1. В файле query1 выбрать все строки, в которых первый инициал – 'K', а второй – 'J'.
2. В файле query3 выбрать все строки, в которых минимальная цена равна 15, а максимальная цена – 20.
3. В файле query1 выбрать все строки, в которых должность – 'MANAGER', а город – 'NEW YORK'.

Вариант 9

1. В файле query4 выбрать все строки, в которых N заказа заканчивается цифрой '4'.
2. В файле query3 выбрать все строки, в которых в названии есть слово 'TENNIS', а цена установлена в 1990 г.
3. В файле query2 выбрать все строки, в которых номер дома не меньше 1000.

Вариант 10

1. В файле query1 выбрать все строки, в которых средний инициал – 'M'.
2. В файле query3 выбрать все строки, в которых цена установлена в январе или феврале любого года.
3. В файле query4 выбрать все строки, в которых сумма не меньше 10000.

Вариант 11

1. В файле query2 выбрать все строки, в которых штат - 'MA'.
2. В файле query3 выбрать все строки, в которых в коде есть два или больше 0 подряд.
3. В файле query1 выбрать все строки, в которых фамилия начинается на букву 'M', а зарплата меньше 1000.

Вариант 12

1. В файле query2 выбрать все строки, в которых город – 'DALLAS'.
2. В файле query4 выбрать все строки, в которых код начинается с цифры '5', а сумма заказа содержит копейки.
3. В файле query1 выбрать все строки, в которых зарплата находится в пределах от 2000 до 2999.

Вариант 13

1. В файле query1 выбрать все строки, в которых город - не 'NEW YORK'.
2. В файле query3 выбрать все строки, в которых цена установлена в 1990 г.
3. В файле query2 выбрать все строки, в которых в названии улицы есть цифры.

Вариант 14

1. В файле query3 выбрать все строки, в которых в названии есть римские цифры.
2. В файле query4 выбрать все строки, в которых дата продажи – 1-е число любого месяца и года.
3. В файле query1 выбрать все строки, в которых должность – не 'SALESPERSON'.

Вариант 15

1. В файле query2 выбрать все строки, в которых номер телефона начинается с '555'.
2. В файле query3 выбрать все строки, в которых в названии есть текст, взятый в кавычки, а в нем - слово 'GUIDE'.
3. В файле query4 выбрать все строки, в которых сумма содержит целое число сотен.

Примеры выполнения заданий

Задание 1

В файле query2 выбрать все строки, в которых в адресе есть улица ("ST").

Решение:

1. Адрес в файле query2 начинается с 31-й позиции (см. структуру файла). Поэтому нужно прежде всего пропустить 30 позиций от начала файла, что можно сделать таким подвыражением: "^.\{30\}" - 30 любых символов от начала файла.

2. Улица обозначается в адресе сокращением "ST.", и эта подстрока может стоять в адресе на любом месте, то есть перед ней могут быть и другие символы. Поскольку общая длина адреса - не более 20 символов, перед подстрокой, которую мы ищем, может быть не более 17 любых символов, что определяется подвыражением: ".*\{0,17\}".
3. Наконец, следует указать подстроку, которую мы ищем: "ST.". Поскольку в подстроку входит метасимвол "." (точка), подвыражение для поиска вхождения будет иметь вид: "ST\.".
4. Итоговое регулярное выражение:
`^^.\{31\}.*\{0,17\}ST\."`

5. Протокол выполнения:

```
student@r111wslin01 ~ $ grep "^^.\{31\}.*\{0,17\}ST\." query2
203:REBOUND SPORTS           :2 E. 14TH ST.           :NEW-YORK
:NY:10009:5555989:10000
205:POINT GUARD              :20 THURSTON ST.        :YONKERS
:NY:10956:5554766:3000
211:AT BAT                   :234 BEACHEM ST.        :BROOKLINE
:MA:02146:5557385:8000
212:ALL SPORT                :1000 38TH ST.          :BROOKLYN
:NY:11210:5551739:6000
213:GOOD SPORT              :400 46TH ST.           :SUNNYSIDE
:NY:11104:5553771:5000
214:AL'S PRO SHOP           :45 SPRUCE ST.          :SPRING
:TX:77388:5555172:8000
223:VELO SPORTS             :23 WHITE ST.           :MALDEN
:MA:02148:5554983:5000
228:FITNESS FIRST          :5000 85TH ST.          :JACKSON-
HEIGHTS:NY:11372:5558710:4000
student@r111wslin01 ~ $
```

Задание 2

В файле query1 выбрать все строки, в которых зарплата составляет целое число тысяч.

Решение 1:

1. Согласно структуре файла, столбец зарплаты начинается с позиции 60, поэтому - подвыражение: "^^.\{59\}".
2. Если зарплата составляет целое число тысяч, то в ней содержится одна или несколько цифр, за которыми следует три нуля - подвыражение: "[0-9]\{1,\}000".
3. Однако возможна (теоретически) зарплата, например "10001", поэтому стоит позаботиться о том, чтобы следующие за тремя нулями символы были отличны от значащих цифр. Таких символов может быть сколько угодно, и это условие можно обеспечить подвыражением: "[^0-9]*".
4. Итоговое регулярное выражение:

```
^^.\{59\}[0-9]\{1,\}000[^0-9]*"
```

5. Протокол выполнения:

```
student@r111wslin01 ~ $ grep "^\{59\}[0-9]\{1,\}000[^\{0-9\}]" query1
7569 CHRIS      L ALBERTS    MANAGER      RESEARCH     NEW-YORK     3000
7788 DONALD     T SCOTT      ANALYST      RESEARCH     DALLAS       3000
7799 MATTHEW    G FISHER     ANALYST      RESEARCH     NEW-YORK     3000
7839 FRANCIS   A KING      PRESIDENT    ACCOUNTING   NEW-YORK     5000
7902 JENNIFER  D FORD      ANALYST      RESEARCH     DALLAS       3000
student@r111wslin01 ~ $
```

Решение 2:

1. Поскольку зарплата является последним полем строки файла query1, возможно, можно просто потребовать, чтобы три нуля были последними символами строки и сформулировать регулярное выражение таким образом: "000\$". Однако, такое решение может наткнуться на неочевидное препятствие. Все зависит от того, какими средствами был подготовлен исходный файл query1 (особенно, если он был перенесен из другой системы). Дело в том, что разные программы и редакторы используют разные способы перевода строки, и в конце строки могут оказаться некоторые дополнительные (невидимые "невооруженным глазом" символы. Таким образом, последний 0 в зарплате может еще не быть последним символом строки. Как правило, увидеть эти дополнительные символы можно, выполнив команду cat с опцией -v. В этом случае на выдаче команды cat можно увидеть непечатный символ, показываемый, например, как: "^M".

2. Следующие протокол иллюстрирует этот случай:

```
student@r111wslin01 ~ $ grep "000$" query1
student@r111wslin01 ~ $ cat -v query1
7369 JOHN      Q SMITH      CLERK        RESEARCH     DALLAS       800
7499 KEVIN    J ALLEN      SALESPERSON  SALES        CHICAGO      1600
7505 JEAN     K DOYLE      MANAGER      SALES        NEW-YORK     2850
7506 LYNN     S DENNIS     MANAGER      SALES        DALLAS       2750
7507 LESLIE   D BAKER      MANAGER      OPERATIONS   NEW-YORK     2200
7521 CYNTHIA  D WARD      SALESPERSON  SALES        CHICAGO      1250
7555 DANIEL   T PETERS     SALESPERSON  SALES        NEW-YORK     1250
7557 KAREN    P SHAW      SALESPERSON  SALES        NEW-YORK     1250
7560 SARAH    S DUNCAN     SALESPERSON  SALES        DALLAS       1250
7564 GREGORY   J LANGE     SALESPERSON  SALES        DALLAS       1250
7566 TERRY     M JONES     MANAGER      RESEARCH     DALLAS       2975
7569 CHRIS    L ALBERTS   MANAGER      RESEARCH     NEW-YORK     3000
7600 RAYMOND  Y PORTER    SALESPERSON  SALES        NEW-YORK     1250
7609 RICHARD  M LEWIS     STAFF        OPERATIONS   DALLAS       1800
7654 KENNETH  J MARTIN    SALESPERSON  SALES        CHICAGO      1250
7676 DENISE   D SOMMERS   STAFF        OPERATIONS   CHICAGO      1850
7698 MARION   S BLAKE     MANAGER      SALES        CHICAGO      2850
7782 CAROL    F CLARK     MANAGER      ACCOUNTING   NEW-YORK     2450
7788 DONALD   T SCOTT     ANALYST      RESEARCH     DALLAS       3000
7789 LIVIA     N WEST     SALESPERSON  SALES        DALLAS       1500
7799 MATTHEW  G FISHER    ANALYST      RESEARCH     NEW-YORK     3000
7820 PAUL     S ROSS     SALESPERSON  SALES        BOSTON       1300
7839 FRANCIS  A KING     PRESIDENT    ACCOUNTING   NEW-YORK     5000
7876 DIANE   G ADAMS     CLERK        RESEARCH     DALLAS       1100
```



```

7900 FRED          S JAMES      CLERK        SALES        CHICAGO     950
7902 JENNIFER     D FORD       ANALYST      RESEARCH    DALLAS      3000
7916 GRACE        M ROBERTS    ANALYST      RESEARCH    NEW-YORK    2875
7919 MICHAEL      A DOUGLAS    CLERK        RESEARCH    NEW-YORK    800
7934 BARBARA      M MILLER     CLERK        ACCOUNTING  NEW-YORK    1300
7950 ALICE        B JENSEN     CLERK        SALES       NEW-YORK    750
7954 JAMES        T MURRAY     CLERK        SALES       DALLAS      750
student@r111wslin01 ~ $

```

Решение 3:

1. Уточним логику предыдущего решения, оказавшегося неправильным. За тремя нулями перед концом строки может следовать (а может и не следовать) еще один символ, отличный от значащей цифры:

```
"000[^0-9]\{0,1\}"
```

2. Протокол выполнения:

```

student@r111wslin01 ~ $ grep "000[^0-9]\{0,1\}" query1
7569 CHRIS        L ALBERTS    MANAGER      RESEARCH    NEW-YORK    3000
7788 DONALD       T SCOTT      ANALYST      RESEARCH    DALLAS      3000
7799 MATTHEW      G FISHER     ANALYST      RESEARCH    NEW-YORK    3000
7839 FRANCIS      A KING       PRESIDENT    ACCOUNTING  NEW-YORK    5000
7902 JENNIFER     D FORD       ANALYST      RESEARCH    DALLAS      3000
student@r111wslin01 ~ $

```

Задание 3

В файле query4 выбрать все строки, в которых дата продажи – весна 1990 г.

Решение:

1. По структуре файла query4 видно, что дата представляется достаточно легко распознаваемым способом: *год-месяц-число*, таким образом, при поиске даты, удовлетворяющей нашим требованиям можно не привязываться к определенным позициям в строке, а просто искать выражение вида: "[0-9]-*весенний_месяц*-[0-9]".
2. Как распознать *весенний_месяц*? Весенние месяцы - "MAR", "APR", "MAY". Первая буква весеннего месяца должна быть "M" или "A", вторая - "A" или "P", третья - "R" или "Y". Из этих букв можно сложить буквосочетания, обозначающие весенние месяцы, а все другие возможные буквосочетания не являются обозначениями месяцев вообще. Таким образом, шаблон для распознавания весеннего месяца будет: "[MA][AP][RY]".
3. Итоговое регулярное выражение:

```
"[0-9]-[MA][AP][RY]-[0-9]"
```

4. Протокол выполнения:

```

student@r111wslin01 ~ $ grep "[0-9]-[MA][AP][RY]-[0-9]" query4
620 TURNER 100 12-MAR-91 4450
526 WEST   221 04-MAR-90 7700
555 WEST   221 04-MAR-91 8540
528 WEST   224 24-MAR-90 3770

```

```
558 WEST      224 31-MAR-91 1700
503 SHAW      201 25-MAR-89 1876
562 SHAW      203 04-MAY-91 2044.5
536 SHAW      206 21-MAY-90 2135.6
561 ROSS       207 20-APR-91 2558.3
506 DUNCAN    208 27-APR-89 2600.4
530 DUNCAN    208 03-APR-90 3026.5
557 DUNCAN    208 08-MAR-91 2461.8
student@r111wslin01 ~ $
```

Практическая работа 4. Использование перенаправлений и конвейера (программного канала)

Выполнение работы

В составе операционных систем семейства *nix имеется большое число системных утилит, предназначенных для обработки текстов. Утилиты `cat` и `grep`, с которыми Вы уже познакомились, относятся к их числу. Другие утилиты такого рода: `cmp` - сравнение файлов, `cut` - "вырезание" полей из текста и `paste` - сцепление строк файлов, `head` - распечатка начала файла и `tail` - распечатка последних строк файла, `sort` - сортировка, `join` - объединение, `sed` - потоковый текстовый редактор и многие другие.

Каждая из таких утилит выполняет в принципе простую обработку текстового файла, но последовательно применяя одну утилиту за другой, можно скомбинировать их действия таким образом, что итоговое преобразование текста будет достаточно сложным. Обработка текста при помощи последовательных вызовов системных утилит называется в *nix "фильтрацией" текста, а сами утилиты называются фильтрами. Такие названия происходят от метафорического представления прохождения потока данных через набор фильтров, каждый из которых производит отбор каких-то требуемых составляющих, в результате чего мы получаем те данные, которые нам нужны - "отфильтрованные" данные.

В цепочки фильтрации могут включаться и другие команды операционной системы, например, команды файловой системы. Параметры и результаты работы этих команд представляются в виде символьных строк, поэтому они тоже могут быть обработаны текстовыми фильтрами.

По умолчанию большинство команд читает входные данные из потока стандартного ввода (клавиатура) и направляет выходные данные в поток стандартного вывода (экран). Как правило, одним из параметров команды является имя (имена) файла (файлов), который (которые) она обрабатывает. Если такое имя не задано, команда читает входные данные из стандартного ввода. Если в команде может задаваться несколько файлов, то обычно стандартный ввод обозначается среди имен файлов символом '-'.

Имеется, однако, возможность перенаправлять стандартные потоки. Запись вида:

```
команда [аргументы] < файл
```

означает перенаправление стандартного ввода, то есть то, что те данные, которые команда обычно читает с клавиатуры, при этом запуске будут прочитаны ею из файла с именем 'файл'.

Записи вида:

```
команда [аргументы] > файл  
команда [аргументы] >> файл
```

означают перенаправление стандартного вывода, то есть то, что те данные, которые команда обычно выводит на экран, теперь записаны ею в файл с именем файл. Разница между '>' и '>>' состоит в том, что в первом случае файл будет создаваться заново, а во втором, если файл с таким именем уже существует, вывод команды будет добавлен в конец файла.

Запись вида:

```
команда1 [аргументы] | команда2 [аргументы] | ... | командаN [аргументы]
```

определяет конвейер или программный канал. В этом случае стандартный вывод команды¹ будет перенаправлен в стандартный ввод команды². Программный канал является наиболее популярным средством при построении цепочек фильтрации.

В данной практической работе Вам предлагается разработать последовательности команд для решения трех задач обработки текстовых файлов. Основным инструментом для решения этих задач для Вас, по-видимому (но не обязательно), будет редактор `sed` и утилита соединения `join`. Другие средства Вы выберете сами. Однако запретим использовать в этой работе утилиту `awk` – ей будет посвящена отдельная работа.

Задача 1

Создайте текстовый файл с текстом (5-10 строк). Выполнить в соответствии с Вашим вариантом индивидуального задания преобразование этого текстового файла. Результат сохранить в новом файле, исходный файл должен остаться без изменения.

Задача 2

Выполнить в соответствии с Вашим вариантом индивидуального задания выборку данных из файлов `query*`, с которыми вы работали на предыдущей практике.

Задача 3

Таблицы, содержащиеся в этих файлах, образуют "базу данных", концептуальная схема которой показана здесь. Выбрать и вывести на экран в удобном для восприятия формате информацию, определенную в Вашем варианте индивидуального задания.

Варианты заданий

Вариант 1

1. В созданном текстовом файле перенести третью от конца строчку в начало файла.
2. Из информации, содержащейся в файлах query[1-5], определить отделы (название и город), которые получали заказы на общую сумму больше 1000.
3. Определить количество групп пользователей.

Вариант 2

1. В созданном текстовом файле перенести третью от начала строку в конец файла.
2. Из информации, содержащейся в файлах query[1-5], определить города, в которых расположены отделы, выполнявшие заказы в феврале 1991г.
3. Определить количество подкаталогов в /student, к которым нет публичных прав доступа.

Вариант 3

1. В созданном текстовом файле перед каждой строкой вставить текущее время.
2. Из информации, содержащейся в файлах query[1-5], определить фамилии продавцов, которые выполняли заказы на поставку товара 'SP JUNIOR RACKET'.
3. Определить количество подкаталогов в /student, к которым есть публичные права на поиск и чтение в них.

Вариант 4

1. В созданном текстовом файле удалить вторую строку, начинающуюся с буквы 'H'.
2. Из информации, содержащейся в файлах query[1-5], определить штат, в котором был сделан заказ на самую большую общую сумму.
3. Определить количество пользователей из вашей студенческой группы.

Вариант 5

1. В созданном текстовом файле оставить в каждой строке не более двух слов. Слова, выходящие за этот предел поместить в

отдельный файл. В первом файле-результате в тех строках, которые содержат менее двух слов, в конец строки должен быть добавлен символ '='. Во втором файле-результате пустых строк оставаться не должно, а перед непустыми строками должны быть указаны их номера в исходном файле, отделенные от остального текста одним пробелом.

2. Из информации, содержащейся в файлах query[1-5], определить товар, которого было заказано наибольшее количество экземпляров в одном заказе.

3. Определить количество (не подкаталогов и не ссылок) файлов в каталоге /student.

Вариант 6

1. В созданном текстовом файле оставить в каждой строке не более 2-х слов. Остаток перенести в следующую строку. Если вторая строка в паре оказывается пустой - печатать в ней символ '='.

2. Из информации, содержащейся в файлах query[1-5], определить 5 покупателей, которые сделали заказов на наибольшую общую сумму в 1990 г.

3. Определить количество файлов или подкаталогов в корневом каталоге, к которым все имеют полные права доступа.

Вариант 7

1. В созданном текстовом файле первый символ каждой строки заменить на первый символ предыдущей строки. Первая строка остается без изменений.

2. Из информации, содержащейся в файлах query[1-5], определить названия товаров, которые продавались по минимальной цене.

3. Определить количество файлов в каталоге /etc, которые являются символическими ссылками.

Вариант 8

1. В созданном текстовом файле поменять местами два первых и два последних символа каждой строки.

2. Из информации, содержащейся в файлах query[1-5], определить фамилию продавца, который продал товара 'SP JUNIOR RACKET' на максимальную сумму в одном заказе.

3. Определить количество файлов в каталоге /etc, на которые есть более одной жесткой ссылки.

Вариант 9

1. В созданном текстовом файле поменять местами первую и последнюю строки файла.
2. Из информации, содержащейся в файлах query[1-5], определить фамилию продавца, который первым продал товар 'SP JUNIOR RACKET' в 1991 г.
3. Определить количество файлов в каталоге /etc, которые созданы не в этом году.

Вариант 10

1. В созданном текстовом файле после строк, которые заканчиваются точкой или запятой, вставить пустую строку.
2. Из информации, содержащейся в файлах query[1-5], определить названия товаров, которые первыми были выставлены в продажу.
3. Выбрать упорядоченный по алфавиту список подкаталогов в /student, к которым нет публичных прав.

Вариант 11

1. В созданном текстовом файле перенести последнее слово в каждой строке в новую строку. Для строк, состоящих из одного слова - не делать ничего.
2. Из информации, содержащейся в файлах query[1-5], определить названия товаров, которые заказывались вместе с товаром 'SP JUNIOR RACKET'.
3. Выбрать упорядоченный по алфавиту список подкаталогов в /student, к которым есть публичные права на поиск и чтение в них.

Вариант 12

1. В созданном текстовом файле перенести первое слово каждой строки в начало следующей строки.
2. Из информации, содержащейся в файлах query[1-5], определить названия товаров, которые когда-либо заказывал покупатель 'JOCKSPORTS'.
3. Выбрать упорядоченный по алфавиту список пользователей из вашей студенческой группы.

Вариант 13

1. В созданном текстовом файле во всех четных строках перенести первое слово строки в конец строки. Строки, содержащие только одно слово, не изменяются.

2. Из информации, содержащейся в файлах query[1-5], определить фамилии продавцов, которые когда-либо продавали товары по их максимальной цене.

3. Выбрать упорядоченный по алфавиту список файлов (не подкаталогов, не ссылок) в каталоге /student.

Вариант 14

1. В созданном текстовом файле во всех нечетных строках перенести последнее слово строки в начало строки. Строки, содержащие только одно слово, не изменяются.

2. Из информации, содержащейся в файлах query[1-5], определить названия товаров, на которые не было заказов в 1990 г.

3. Выбрать упорядоченный по алфавиту список файлов в каталоге /etc, на которые есть более одной жесткой ссылки.

Вариант 15

1. В созданном текстовом файле поменять местами четные строки с нечетными.

2. Из информации, содержащейся в файлах query[1-5], определить названия товаров, которые не были в продаже в 1990 г.

3. Выбрать упорядоченный по алфавиту список файлов в каталоге /etc, которые созданы не в этом году.

Примеры выполнения заданий

Задание 1

Работа всех вариантов демонстрируется на обработке файла с именем Hum-Dum.txt, исходное содержимое которого показано в следующем протоколе:

```
student@r111wslin01 ~ $ cat Hum-Dum
Humpty-Dumpty
Set on the wall.
Humpty-Dumpty
Had a greate fall.
And all the king's horses,
And all the king's man.
Can not Humpty,
Can not Dumpty,
Humpty-Dumpty,
Dumpty-Humpty,
Set on this wall
Again.
student@r111wslin01 ~ $
```

Ниже приводятся некоторые общие соображения по решению задач 1-го задания.

В большинстве случаев мы можем легко сформулировать операторы потокового редактора, необходимые для выполнения заданного преобразования, за исключением одного компонента - адреса строки, к которой это преобразование должно быть применено. В некоторых случаях нам также заранее неизвестен и текст, который необходимо вставить в целевую строку. Поэтому типовая схема решения следующая:

1. Определяются номера строк, к которым должны быть применены преобразования, а при необходимости - также и текст, который должен вставляться в целевой файл. Для определения применяются статические (заранее известные и заложенные в текст команд) команды потокового редактора. Результаты этого шага сохраняются в файле.

2. Путем редактирования результатов 1-го шага динамически формируются команды `sed` для выполнения преобразования, содержащие адреса и тексты, определенные на 1-м шаге. Эти команды сохраняются в файле.

3. Выполняется редактирование исходного текста командами, сохраненными на 2-м шаге.

Возможны два подхода к получению номеров строк, подлежащих преобразованию:

1. выполнить нумерование всех строк исходного файла (это можно сделать командой `pr` или командой `cat` с опцией `-n`), а затем из пронумерованной последовательности строк выбрать строку, предназначенную для обработки с ее номером; такой прием применен в решениях для вариантов 1 и 2;

2. выбрать номер строки, предназначенной для обработки, при помощи команды `sed "="`; такой прием применен для варианта 3.

Поскольку Unix имеет большое число утилит для обработки текстов, причем функций многих утилит перекрываются, каждая из предложенных задач может быть решена множеством различных способов.

Задание 1, вариант 1

Каждое второе слово каждой строки вывести в отдельную следующую строку. Если в строке только одно слово, ничего не делать.

Решение:

```
pr -n ' ' -T Hum-  
Dum.txt |
```

```
sed -n 's/^ */p' |
```

```
cut -f1,3 -d' ' |
```

```
sed -n ' /^[^0-9]/p'  
>temp01
```

Вывести файл без заголовка и лишних строк, но с номерами строк.

Удалить головные пробелы.

Вырезать номер и 2-е слово.

Удалить строки, содержащие только номер, результат сохранить в 1-м временном файле.


```

cut -f1 -d' ' temp01
|
sed -n 's/$/a\\p'
>temp02

cut -f2 -d' ' temp01
>temp03

paste -d'\n' temp02
temp03 >temp01

sed -f temp01 Hum-
Dum.txt >result

rm -f temp*

```

Выбрать из 1-го временного файла номера строк.

Добавить к ним 'a\' и сохранить во 2-м временном файле.

Выбрать из 1-го временного файла слова и сохранить в 3-м временном файле.

Сцепить построчно 2-й и 3-й временные файлы, вставив между сцепляемыми строками перевод строки. Результат сохранить в 1-м временном файле. Результат - набор команд sed на вставку.

Выполнить команды из 1-го временного файла, сохранить результат.

Удалить временные файлы.

Протокол выполнения:

```

student@r111wslin01 ~ $ pr -n' ' -T Hum-Dum.txt |
> sed -n 's/^ *//p' |
> cut -f1,3 -d' ' |
> sed -n ' /^[^0-9]/p' >temp01
student@r111wslin01 ~ $ cut -f1 -d' ' temp01 |
> sed -n 's/$/a\\p' >temp02
student@r111wslin01 ~ $ cut -f2 -d' ' temp01 >temp03
student@r111wslin01 ~ $ paste -d'\n' temp02 temp03 >temp01
student@r111wslin01 ~ $ sed -f temp01 Hum-Dum.txt >result
student@r111wslin01 ~ $ rm -f temp*
student@r111wslin01 ~ $ cat result
Humpty-Dumpty
Set on the wall.
on
Humpty-Dumpty
Had a greate fall.
a
And all the king's horses,
all
And all the king's man.
all
Can not Humpty,
not
Can not Dumpty,
not
Humpty-Dumpty,
Dumpty-Humpty,
Set on this wall
on
Again.
student@r111wslin01 ~ $

```

Обратите внимание на то, что в вышеприведенном протоколе некоторые приглашения системы выглядят как "`"`", а некоторые – как "`>`". Система печатает приглашение "`>`" (вторичное приглашение), если предыдущая команда заканчивается знаком конвейера "`|`" и, следовательно, обязательно ожидается ввод следующей команды. В противном случае печатается первичное

приглашение – """. Символы первичного и вторичного приглашения определяются переменными окружения PS1 и PS2 соответственно.

Задание 1, вариант 2

Первый символ каждой строки заменить на первый символ следующей строки. Последняя строка остается без изменений.

Решение:

```
pr -n ' ' -T Hum-Dum.txt |
cut -c7 |
sed -n '2,$p' >temp01
```

Вывод с нумерацией строк.

Выделение 1-го символа.

Удаление 1-й строки, символ из нее никуда не подставляется. 1-е символы строк сохраняются во временном файле.

```
pr -n ' ' -T Hum-Dum.txt |
sed -n 's/^[ ]*//p' |
cut -f1 -d' ' |
paste -d' ' - temp01 |
```

Вывод с нумерацией строк.

Удаление головных пробелов.

Вывод только столбца номеров.

Сцепление номера строки (входной поток) с 1-м символом следующей строки (временный файл).

Получается, например: '3 H'.

```
sed '$d' |
```

Удаление последней строки - в нее ничего не подставляется.

```
sed -n 's/ /s\\/^\\.\\.//p' |
```

Замена пробела между номером и символом на служебные символы. Получается: '3s/^./H'.

```
sed -n 's/$/\\//p' >temp01
```

Добавление служебных символов в конец строки. Получается: '3s/^./H' - команда sed на изменение 1-го символа строки. (Обратите внимание: эта команда не заканчивается флагом 'p'.) Команды сохраняются во временном файле.

```
sed -f temp01 Hum-Dum.txt
>result
```

Временный файл используется как командный скрипт sed. (Обратите внимание: если бы в командах скрипта не было бы флага 'p', а в данной команде не было бы опции -n, не вывелась бы последняя строка, для которой нет подстановки в скрипте). Результат сохраняется.

```
rm -f temp*
```

Удаление временного файла.

Протокол выполнения:

```
student@r111wslin01 ~ $ pr -n ' ' -T Hum-Dum.txt |
> cut -c7 |
> sed -n '2,$p' >temp01
student@r111wslin01 ~ $ pr -n ' ' -T Hum-Dum.txt |
> sed -n 's/^[ ]*//p' |
> cut -f1 -d' ' |
> paste -d' ' - temp01 |
> sed '$d' |
> sed -n 's/ /s\\/^\\.\\.//p' |
> sed -n 's/$/\\//p' >temp01
```

```

student@r111wslin01 ~ $ sed -f temp01 Hum-Dum.txt >result
student@r111wslin01 ~ $ rm -f temp*
student@r111wslin01 ~ $ cat result
Sumpty-Dumpty
Het on the wall.
Humpty-Dumpty
Aad a greate fall.
And all the king's horses,
Cnd all the king's man.
Can not Humpty,
Han not Dumpty,
Dumpty-Dumpty,
Sumpty-Humpty,
Aet on this wall
Again.
student@r111wslin01 ~ $

```

Задание 1, вариант 3

В предпоследней строке, которая заканчивается точкой, поменять местами первое слово с последним.

Решение:

```
sed -n '/\.$/= ' Hum-
Dum.txt |
```

```
tail -2 |
```

```
head -n1 > temp01
```

```
sed -n 's/$/s\ /
\.*\ / /p/p' temp01
>temp02
```

```
sed -n -f temp02 Hum-
Dum.txt >temp03
```

```
sed -n 's/$/s\ \ [ \ ^
\ ] * \ / /gp/p' temp01
>temp02
```

```
sed -n -f temp02 Hum-
Dum.txt >temp04
```

```
paste -d' ' temp01
temp03 |
```

```
sed -n 's/ /s\ \ [ \ ^
\ ] * $ \ / /p' |
```

```
sed -n 's/$/ \ / /p'
>temp03
```

```
paste -d' ' temp01
temp04 |
```

```
sed -n 's/ /s\ \ [ \ ^
\ ] * \ / /p' |
```

Вывод номеров всех строк, которые заканчиваются точкой.

Вывод двух последних из этих номеров.

Вывод первого из двух последних номеров. Сохранение в temp01.

Формирование команды sed на вывод 1-го слова строки с номером, сохраненным в temp01. Команда сохраняется в temp02.

Выполнение команды из temp02, запись 1-го слова в temp03.

Формирование команды sed на вывод последнего слова строки с номером, сохраненным в temp01. Команда сохраняется в temp02.

Выполнение команды из temp02, запись последнего слова в temp04.

Сцепление через пробел номера строки и последнего слова строки.

Формирование команды sed на замену в строке с номером, выбранным из temp01, 1-го слова на текст, выбранный из temp03.

Завершение формирование команды и сохранение ее в temp03.

Сцепление через пробел номера строки и 1-го слова строки.

Формирование команды sed на замену в строке с номером, выбранным из temp01, последнего слова на текст, выбранный из temp04.

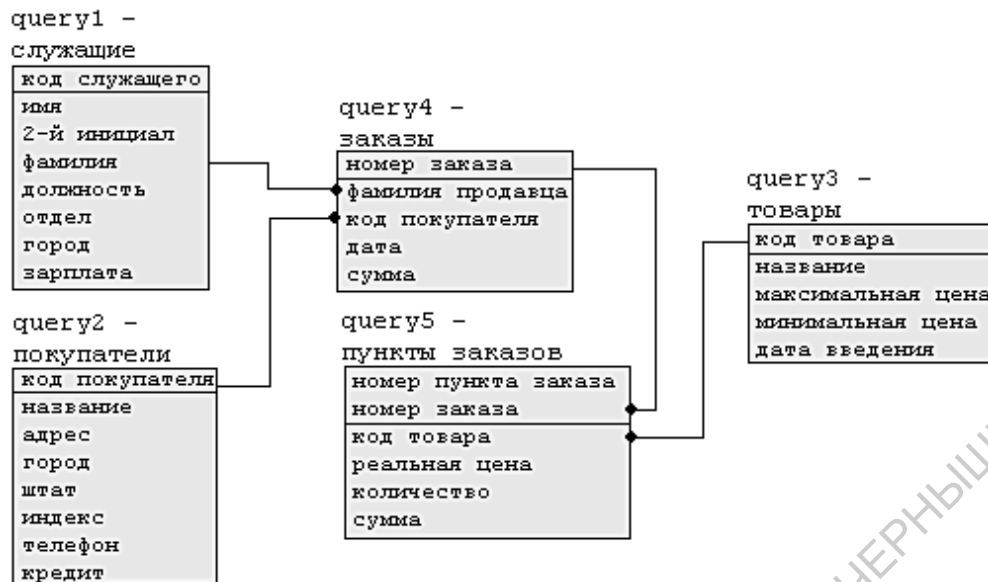
<pre>sed -n 's/\$/ \\/p' >temp04 sed -f temp03 Hum- Dum.txt sed -f temp04 >result rm -f temp*</pre>	<p>Завершение формирование команды и сохранение ее в temp04.</p> <p>Выполнение для исходного файла команды из temp03.</p> <p>Выполнение для того, что получилось, команды из temp04.</p> <p>Удаление временных файлов.</p>
--	--

Протокол выполнения:

```
student@r111wslin01 ~ $ sed -n '/\.$/= ' Hum-Dum.txt |
> tail -2 |
> head -n1 > temp01
student@r111wslin01 ~ $ sed -n 's/$/s\ \. \*\ \\/p/p' temp01 >temp02
student@r111wslin01 ~ $ sed -n -f temp02 Hum-Dum.txt >temp03
student@r111wslin01 ~ $ sed -n 's/$/s\ \[\^ \]\* \\/gp/p' temp01 >temp02
student@r111wslin01 ~ $ sed -n -f temp02 Hum-Dum.txt >temp04
student@r111wslin01 ~ $ paste -d' ' temp01 temp03 |
> sed -n 's/ /s\ \ [\^ \]\* $\ / /p' |
> sed -n 's/ $\ \\/p' >temp03
student@r111wslin01 ~ $ paste -d' ' temp01 temp04 |
> sed -n 's/ /s\ \[\^ \]\* \\/p' |
> sed -n 's/ $\ \\/p' >temp04
student@r111wslin01 ~ $ sed -f temp03 Hum-Dum.txt |
> sed -f temp04 >result
> student@r111wslin01 ~ $ rm -f temp*
student@r111wslin01 ~ $ cat result
Humpty-Dumpty
Set on the wall.
Humpty-Dumpty
Had a greate fall.
And all the king's horses,
man. all the king's And
Can not Humpty,
Can not Dumpty,
Humpty-Dumpty,
Dumpty-Humpty,
Set on this wall
Again.
student@r111wslin01 ~ $
```

Задание 2

Работа всех вариантов этого задания происходит на наборе файлов query[1-5]. В содержимом этих файлов легко видится регулярность структуры. Это содержимое представляют собой таблицы (реляционные таблицы) и составляют как бы базу данных, схема которой показана ниже:



Варианты задания 2 представляют собой задания на выборку данных из "таблиц" этой "базы данных". Утилиты *nix предоставляют в наше распоряжение следующие средства, которые в той или иной мере могут служить аналогом реляционных операций:

Операция *проекции* может быть осуществлена вырезанием определенных полей строки - командой cut.

Операция *ограничения* может быть осуществлена какой-либо из утилит, осуществляющих поиск строки по шаблону - grep или sed.

Операция *естественного соединения* может быть осуществлена утилитой join. Следует однако помнить, что для применения утилиты join таблицы должны быть отсортированы по тому столбцу, по которому происходит соединение, это можно сделать при помощи утилиты sort.

Дубликаты в файлах-таблицах могут быть устранены при помощи утилиты uniq или утилиты sort с опцией -u. Следует иметь в виду, что в первом случае дубликатами считаются совпадения полных строк, а во втором - только тех полей, по которым выполняется сортировка.

Рассматриваемые нами утилиты не предоставляют тех возможностей, которые предоставляют агрегатные функции SQL, однако функции MAX() и MIN() можно промоделировать, выполнив сортировку таблицы и выбрав затем первую (утилита head) или последнюю (утилита tail) строку.

В пояснениях к нашим примерам выполнения мы часто используем терминологию реляционных операций.

Большинство утилит, работающих с полями форматированного текста, по умолчанию предполагают символом-разделителем полей символ табуляции. Однако работать с символом табуляции неудобно, поскольку он по умолчанию явно не отображается. Поэтому разумно назначать разделителем какой-либо отображаемый символ. Обратите внимание на то, что в наших файлах-таблицах используются различные разделители полей. Для выполнения операции соединения необходимо установить общий разделитель для обеих соединяемых таблиц.

Мы всегда выполняли проекцию (отбор необходимых столбцов) прежде, чем соединение. Возможно выполнять проекцию и после соединения. Во втором случае может даже быть сэкономлено несколько команд, но упреждающий отбор только необходимых столбцов уменьшает объем промежуточных результатов, чем существенно облегчает отладку.

Задание 2 вариант 1

Определить фамилии продавцов, которые выполняли заказы, состоящие из более чем 5 пунктов.

Решение:

```
sed 's/ \{1,\}/\:/g'
../metod/query5 |
cut -f1,2 -d':' |
sort +0 -1 -t':' -n |

sed '/^[1-4]:/d' >temp01
sort +1 -2 -t':' >temp01

sed 's/ \{1,\}/\:/g'
../metod/query4 |
cut -f1,2 -d':' |
sort +0 -1 -t':' |
join -j1 1 -j2 2 -t':' -
temp01 |
cut -f2 -d':' |
sort +0 -1 -t':' -u
>result

rm -f temp*
```

Установка в файле query5 разделителя ":".

Проекция по номеру пункта и номеру заказа.

Числовая (опция -n) - сортировка по номеру пункта.

Удаление тех строк, в которых номер пункта 1-4.

Сортировка по номеру заказа. Результат сохраняется в temp01.

Установка в файле query4 разделителя ":".

Проекция по номеру заказа и фамилии продавца.

Сортировка по номеру заказа.

Соединение с сохраненным в temp01.

Проекция по фамилии продавца.

Сортировка по фамилии продавца с устранением дубликатов.

Удаление временного файла.

Протокол выполнения:

```
student@r111wslin01 ~ $ sed 's/ \{1,\}/\:/g' ../metod/query5 |
> cut -f1,2 -d':' |
> sort +0 -1 -t':' -n |
> sed '/^[1-4]:/d' |
> sort +1 -2 -t':' >temp01
student@r111wslin01 ~ $ sed 's/ \{1,\}/\:/g' ../metod/query4 |
> cut -f1,2 -d':' |
> sort +0 -1 -t':' |
> join -j1 1 -j2 2 -t':' - temp01 |
> cut -f2 -d':' |
> sort +0 -1 -t':' -u >result
student@r111wslin01 ~ $ rm -f temp*
student@r111wslin01 ~ $ cat result
DUNCAN
ROSS
SHAW
TURNER
WARD
student@r111wslin01 ~ $
```

Задание 2, вариант 2

Определить покупателей, которые ничего не покупали в феврале 1990 г.

Решение:

<code>cut -f1,2 -d':'</code>	Проекция по коду и имени покупателя.
<code>../method/query2 </code>	
<code>sort +0 -1 -t':'</code>	Сортировка по коду, сохранение.
<code>>temp01</code>	
<code>sed 's/ \{1,\}/\:/g'</code>	Установка разделителя ':'.
<code>../method/query4 </code>	
<code>cut -f3,4 -d':'</code>	Выделение кода покупателя и даты.
<code>sed -n '/-FEB-91/p' </code>	Ограничение по дате.
<code>sort +0 -1 -t':'</code>	Сортировка по коду покупателя, ликвидация дубликатов кодов.
<code>-u </code>	
<code>join -j1 1 -j2 1 -t':'</code>	Соединение по коду покупателя (правое внешнее).
<code>-a2 - temp01 </code>	
<code>sed -n</code>	
<code>'/^[^:]*:[^:]*\$/'p </code>	Выборка тех строк, в которых только один раз встречается разделитель полей, то есть, нет даты.
<code>cut -f2 -d':'</code>	Проекция по имени покупателя.
<code> </code>	
<code>sort +0 -1 >result</code>	Сортировка по имени.
<code>rm -f temp*</code>	Удаление временных файлов.

Протокол выполнения:

```
student@r111wslin01 ~ $ cut -f1,2 -d':' ../method/query2 |
> sort +0 -1 -t':' >temp01
student@r111wslin01 ~ $ sed 's/ \{1,\}/\:/g' ../method/query4 |
> cut -f3,4 -d':' |
> sed -n '/-FEB-91/p' |
> sort +0 -1 -t':' -u |
> join -j1 1 -j2 1 -t':' -a2 - temp01 |
> sed -n '/^[^:]*:[^:]*$/'p |
> cut -f2 -d':' |
> sort +0 -1 >result
student@r111wslin01 ~ $ rm -f temp*
student@r111wslin01 ~ $ cat result
AL AND BOB'S SPORTS
AL'S PRO SHOP
ALL SPORT
AT BAT
BOB'S FAMILY SPORTS
BOB'S SWIM, CYCLE, AND RUN
CENTURY SHOP
EVERY MOUNTAIN
FAST BREAK
GOOD SPORT
HIT, THROW, AND RUN
HOOPS
JOCKSPORTS
JOE'S BIKE SHOP
JUST BIKES
JUST TENNIS
POINT GUARD
REBOUND SPORTS
SHAPE UP
```

STADIUM SPORTS
 THE ALL AMERICAN
 THE COLISEUM
 THE OUTFIELD
 THE POWER FORWARD
 THE TOUR
 TKB SPORT SHOP
 VELO SPORTS
 WHEELS AND DEALS
 WOMENS SPORTS
 student@r111wslin01 ~ \$

Задание 2, вариант 3

Определить имена и фамилии всех служащих фирмы, которые работают в одном городе с президентом.

В принципе, эту задачу можно решить, и не прибегая к реляционным операциям, но мы решим ее именно таким образом, потому что среди не рассматриваемых нами имеются варианты, которые требуют именно такого подхода.

Хотя все выбираемые данные находятся в одной таблице, нам необходимо будет применить здесь автосоединение - соединение таблицы с самой собой.

Решение:

<pre>sed 's/ \{1,\}/\:/g' ./metod/query1 >temp01</pre>	<p>Установка разделителя ':' в файле query1 (просто для удобства). Сохранение в temp01.</p>
<pre>sed -n '/:PRESIDENT:/p' temp01 >temp02</pre>	<p>Выделение строки президента. Сохранение в temp02.</p>
<pre>sed '/:PRESIDENT:/d' temp01 </pre>	<p>Удаление строки президента.</p>
<pre>cut -f4,6,7 -d':' temp01 >temp03</pre>	<p>Выделение полей фамилии, названия отдела, города. Сохранение в temp03.</p>
<pre>cut -f6 -d':' temp02 >temp01</pre>	<p>Выделение названия отдела в строке президента. Сохранение в temp01.</p>
<pre>sort +1 -2 -t':' temp03 </pre>	<p>Сортировка остальных служащих по полю названия отдела.</p>
<pre>join -j1 2 -j2 1 -t':' - temp01 </pre>	<p>Соединение с названием отдела президента, сохраненным в temp01.</p>
<pre>cut -f2 -d':' sort +0 -1 -t':' >temp04</pre>	<p>Выделение из результата фамилий служащих. Сортировка фамилий и сохранение их в temp04.</p>
<pre>cut -f7 -d':' temp02 >temp01</pre>	<p>Выделение города в строке президента. Сохранение в temp01.</p>
<pre>sort +2 -3 -t':' temp03 </pre>	<p>Сортировка остальных служащих по полю города.</p>
<pre>join -j1 3 -j2 1 -t':' - temp01 </pre>	<p>Соединение с городом президента, сохраненным в temp01.</p>
<pre>cut -f2 -d':' sort +0 -1 -t':' </pre>	<p>Выделение из результата фамилий служащих. Сортировка фамилий.</p>


```
join -j1 1 -j2 1 - temp04 Соединение тех, у кого совпадает отдел, с теми, у
>result кого совпадает город. Результат сохраняется.
rm -f temp* Удаление временных файлов.
```

Протокол выполнения:

```
student@r111wslin01 ~ $ sed 's/ \{1,\}/\:/g' ../metod/query1 >temp01
student@r111wslin01 ~ $ sed -n '/:PRESIDENT:/p' temp01 >temp02
student@r111wslin01 ~ $ sed '/:PRESIDENT:/d' temp01 |
> cut -f4,6,7 -d':' temp01 >temp03
student@r111wslin01 ~ $ cut -f6 -d':' temp02 >temp01
student@r111wslin01 ~ $ sort +1 -2 -t':' temp03 |
> join -j1 2 -j2 1 -t':' - temp01 |
> cut -f2 -d':' |
> sort +0 -1 -t':' >temp04
student@r111wslin01 ~ $ cut -f7 -d':' temp02 >temp01
student@r111wslin01 ~ $ sort +2 -3 -t':' temp03 |
> join -j1 3 -j2 1 -t':' - temp01 |
> cut -f2 -d':' |
> sort +0 -1 -t':' |
> join -j1 1 -j2 1 - temp04 >result
student@r111wslin01 ~ $ rm -f temp*
student@r111wslin01 ~ $ cat result
CLARK
KING
MILLER
student@r111wslin01 ~ $
```

Задание 3

Источником входных данных для всех вариантов этого задания может быть команда печати содержимого каталогов `ls` или команда поиска файлов `find`. Выходной поток первой команды направляется в конвейер, где он может последовательно обрабатываться командами обработки текстов: `grep`, `sed`, `cut`, `sort` и т.п. Если в задании требуется подсчитать число элементов, в последнем звене конвейера может быть применена команда `wc`.

При выполнении задания следует иметь в виду, что пользовательские группы в системе совпадают с кодами студенческих групп (например: "ap109", "ap070b" и т.д.), все коды начинаются с букв "ap"; а имена пользователей формируются как: имя_группыnn, где nn - порядковый номер в группе.

Задание 3 вариант 1

Определить общее количество студенческих групп.

Решение:

```
ls -ld ../ * |
```

Команды выполняются из домашнего каталога пользователя - `/home/имя_пользователя`, а информацию о созданных для групп каталогах можно получить из каталога `/home/`, который может адресоваться из текущего каталога как: `../`. Выводим информацию о содержимом этого каталога. Опция `-l` указывается, чтобы получить полную информацию, включая группу, опция `-d` предотвращает обход подкаталогов. Печать команды `ls` перенаправляется в поток.

<pre>grep "^d.\{24\}ap" </pre>	<p>Признак подкаталога - буква "-d" в первой позиции выдачи команды ls, а имена групп начинаются с 25-й позиции выдачи. Команде grep задается шаблон, который определяет признак каталога в 1-й позиции и имя группы, начинающееся с букв "ap".</p>
<pre>sed -n 's/[]\{2,\}/ /gp' </pre>	<p>Поскольку дальше потребуется выделять поля, избавимся от множественных пробелов с помощью команды sed.</p>
<pre>cut -f4 -d ' ' </pre>	<p>Выделяется 4-е поле, содержащее имя группы.</p>
<pre>sort </pre>	<p>Результат сортируется, это понадобится для следующей команды. Поскольку сейчас в тексте остался только один столбец, никаких опций для сортировки мы не указываем.</p>
<pre>uniq </pre>	<p>Одна и та же группа повторяется для многих каталогов, поэтому следует избавиться от повторяющихся строк.</p>
<pre>wc -l</pre>	<p>Команда wc подсчитывает число оставшихся строк, результат выводится на печать.</p>

Протокол выполнения:

```
student@r111wslin01 ~ $ ls -lad ../ * | grep "^d.\{24\}ap" | sed -n 's/[ ]\{2,\}/ /gp' | cut -f4 -d ' ' |
> sort +0 -1 | uniq | wc -l
3
student@r111wslin01 ~ $
```

Задание 3 вариант 2

Определить файлы в каталоге /etc, которые являются символическими ссылками. Вывести имена файлов и имена тех файлов, на которые они ссылаются, упорядочив список по первому имени.

Решение:

<pre>ls -ld /etc/* </pre>	<p>Выводится информация о содержимом интересующего нас каталога. Печать команды ls перенаправляется в поток.</p>
<pre>grep "^[l]" </pre>	<p>Выделяются те строки, которые имеют в 1-й позиции букву "l" - признак мягкой ссылки.</p>
<pre>sort +8 -9 </pre>	<p>Выполняется сортировка по 9-му столбцу - имени файла</p>
<pre>sed -n 's/[]\{2,\}/ /gp' </pre>	<p>Множественные пробелы заменяются одним пробелом.</p>
<pre>cut -f9-100 -d ' '</pre>	<p>Выделяются поля, начиная с 9-го - и до конца.</p>

Протокол выполнения:

```
student@r111wslin01 ~ $ ls -ld /etc/* | grep "^[l]" | sort +8 -9 | sed -n 's/[ ]\{2,\}/ /gp' | cut -f9-100 -d ' '
/etc/grub.conf -> ../boot/grub/grub.conf
/etc/init.d -> rc.d/init.d
/etc/rc -> rc.d/rc
/etc/rc0.d -> rc.d/rc0.d
/etc/rc1.d -> rc.d/rc1.d
/etc/rc2.d -> rc.d/rc2.d
/etc/rc3.d -> rc.d/rc3.d
```

```
/etc/rc4.d -> rc.d/rc4.d
/etc/rc5.d -> rc.d/rc5.d
/etc/rc6.d -> rc.d/rc6.d
/etc/rc.local -> rc.d/rc.local
/etc/rc.sysinit -> rc.d/rc.sysinit
/etc/rmt -> ../sbin/rmt
student@r111wslin01 ~ $
```

Практическая работа 5. Текстовый процессор awk

Выполнение работы

Выполняя предыдущую работу, Вы, возможно, не раз посетовали на отсутствие в элементарных фильтрах `*nix` вычислительных и логических возможностей. Этот недостаток компенсируется в утилите `awk`, которая, являясь одной из утилит, работающих с регулярными выражениями, в то же время предоставляет программисту алгоритмические и вычислительные возможности, базирующиеся на синтаксисе языка `C`. Как вы увидите ниже, многие из тех предыдущих задач, для решения которых нам приходилось создавать длинные конвейерные цепочки команд, могут быть решены одним обращением к утилите `awk`.

`awk` может применяться также и как фильтр и, сочетаясь в цепочке фильтрации с другими утилитами, неограниченно расширять возможности командного управления.

Хотя синтаксис внутреннего языка `awk` базируется на синтаксисе языка `C`, следует помнить, что язык `awk` - язык обработки текстов, в этом языке существует единственный тип данных - строка символов, а при выполнении вычислительных операций происходят "прозрачные" преобразования строковых операндов в числовые и числовых результатов - в строковый тип. В данной практической работе Вам предлагается разработать программы `awk` для решения задач обработки текстовых файлов. Вы можете комбинировать вызов `awk` с вызовами других утилит, но в большинстве случаев в этом нет необходимости.

В задаче 1 выполните при помощи `awk` ту же обработку текстового файла, которую Вы выполняли в Задаче 1 работы №4.

В задачах 2,3 выполните при помощи `awk` обработку файлов `query[1-5]`, в соответствии с Вашим вариантом индивидуального задания.

Варианты заданий

Вариант 1

1. В созданном для практической работы №4 текстовом файле перенести третью от конца строчку в начало файла.
2. В файле query1 определить среднюю зарплату продавцов ("SALESPERSON").
3. В файле query2 определить трех покупателей, которым предоставлены наибольшие кредиты.

Вариант 2

1. В созданном для практической работы №4 текстовом файле перенести третью от начала строку в конец файла.
2. В файле query2 определить средний кредит для каждого штата.
3. В файле query3 определить 5 самых дорогих товаров (по максимальной цене).

Вариант 3

1. В созданном для практической работы №4 текстовом файле перед каждой строкой вставить текущее время.
2. В файле query4 определить среднюю сумму заказа для 1990 г.
3. В файле query4 определить количество продаж для каждого продавца.

Вариант 4

1. В созданном для практической работы №4 текстовом файле удалить вторую строку, начинающуюся с буквы 'H'.
2. В файле query1 определить количество сотрудников на каждой должности.
3. В файле query1 определить отделы, в которых нет менеджеров ("MANAGER").

Вариант 5

1. В созданном для практической работы №4 текстовом файле оставить в каждой строке не более двух слов. Слова, выходящие за этот предел поместить в отдельный файл. В первом файле-результате в тех строках, которые содержат менее двух слов, в конец строки должен быть добавлен символ '='. Во втором файле-результате пустых строк оставаться не должно, а перед непустыми строками должны быть указаны их номера в исходном файле, отделенные от остального текста одним пробелом.

2. В файле query1 определить общее число отделов в фирме.

3. В файле query3 считая, что первое слово в названии товара - название фирмы, определить число товаров каждой фирмы.

Вариант 6

1. В созданном для практической работы №4 текстовом файле оставить в каждой строке не более 2-х слов. Остаток перенести в следующую строку. Если вторая строка в паре оказывается пустой - печатать в ней символ '='.

2. В файле query2 определить покупателей, у которых в адресе указана улица или т.п. ("ST.", "RD.", etc.), и их количество.

3. В файле query3 определить товары, максимальная цена которых больше 20, и средний процент по ним.

Вариант 7

1. В созданном для практической работы №4 текстовом файле первый символ каждой строки заменить на первый символ предыдущей строки. Первая строка остается без изменений.

2. В файле query3 определить товары, в названии которых фигурирует "BALL", и их общее количество.

3. В файле query2 определить города, в которых есть более одного покупателя.

Вариант 8

1. В созданном для практической работы №4 текстовом файле поменять местами два первых и два последних символа каждой строки.
2. В файле query1 определить города, в которых есть отделы "SALES".
3. В файле query4 определить три самых больших заказа.

Вариант 9

1. В созданном для практической работы №4 текстовом файле поменять местами первую и последнюю строки файла.
2. В файле query1 определить максимальную зарплату менеджеров.
3. В файле query2 определить для каждого штата покупателя с максимальным кредитом.

Вариант 10

1. В созданном для практической работы №4 текстовом файле после строк, которые заканчиваются точкой или запятой, вставить пустую строку.
2. В файле query1 определить трех сотрудников, получающих самую высокую зарплату.
3. В файле query3 считая, что первое слово в названии товара - название фирмы, определить фирмы, все товары которых выставлены в продажу в один день.

Вариант 11

1. В созданном для практической работы №4 текстовом файле перенести последнее слово в каждой строке в новую строку. Для строк, состоящих из одного слова - не делать ничего.
2. В файле query1 определить сумму зарплаты, которая встречается чаще других.
3. В файле query3 считая, что первое слово в названии товара - название фирмы, определить фирмы, все товары которых выставлены в продажу в один день.

Вариант 12

1. В созданном для практической работы №4 текстовом файле перенести первое слово каждой строки в начало следующей строки.
2. В файле query1 определить разность между максимальной и минимальной зарплатой.
3. В файле query2 определить для каждого штата число покупателей, в названии которых есть "SPORT".

Вариант 13

1. В созданном для практической работы №4 текстовом файле во всех четных строках перенести первое слово строки в конец строки. Строки, содержащие только одно слово, не изменяются.
2. В файле query2 определить число покупателей для каждого штата.
3. В файле query3 определить товар/товары, который появился в продаже последним.

Вариант 14

1. В созданном для практической работы №4 текстовом файле во всех нечетных строках перенести последнее слово строки в начало строки. Строки, содержащие только одно слово, не изменяются.
2. В файле query2 определить разность между максимальной и минимальной суммой кредита.
3. В файле query3 определить количество товаров, выставленных в каждом месяце (с учетом года).

Вариант 15

1. В созданном для практической работы №4 текстовом файле поменять местами четные строки с нечетными.
2. В файле query4 определить разность между максимальной и минимальной суммой заказа.
3. В файле query2 определить первые две цифры почтового индекса для каждого штата.

Примеры выполнения заданий

С примерами использования утилиты awk Вы можете ознакомиться в мануале по данной утилите.

Список литературы

1. Таненбаум Э. С., Херберт Б. Современные операционные системы. 4-е изд. – Издательский дом «Питер», 2015.
2. Таненбаум Э. С., Таненбаум Э. С. Компьютерные сети:[пер. с англ.]. – Издательский дом «Питер», 2012.
3. Таненбаум Э. С. Архитектура компьютера:[пер. с англ.]. – Издательский дом «Питер», 2011.
4. Cannon J. Command Line Kung Fu: Bash Scripting Tricks, Linux Shell Programming Tips, and Bash One-liners. – CreateSpace Independent Publishing Platform, 2014.
5. Cooper M. Advanced Bash Scripting Guide. – Рипол Классик, 2014.
6. Бессонов Л. В., Брагина И. Г. Операционные системы. Компьютерные сети: Пособие для студентов, обучающихся по дополнительной специальности «Компьютерная графика и веб-дизайн» // Саратов: Изд-во «Научная книга», 2009. – 44 с.
7. Брагина И.Г. О применение программных средств в обучении математике/И.Г. Брагина, А.В. Букушева//Наука, образование, общество: актуальные вопросы и перспективы развития: Сборник научных трудов по материалам Международной научно-практической конференции 30 мая 2015 г.: в 3 частях. Часть II. М.: «АР-Консалт», 2015. С. 109-110.
8. Брагина И.Г., Сергеева Н.В., Бессонов Л.В. Основы теории вероятностей: Учебное пособие – Саратов, 2014.
9. Бессонов Л. В., Брагина И. Г. Информационные технологии в профессиональной деятельности преподавателя: Учеб. пособие. – Саратов: Изд-во «Научная книга», 2014. – 28 с.
10. Дмитриев П.О. Практикум по веб-программированию. Часть 1. Теоретическое введение в язык PHP: Учебное пособие для студентов, обучающихся по направлению подготовки бакалавриата 09.03.03 «Прикладная информатика» — Саратов: ООО Издательский Центр «Наука», 2016. — 40 с.
11. Бессонов Л.В. Практикум по веб-программированию. Упражнения и практические задания: Учебно-методическое пособие для студентов, обучающихся по направлению подготовки бакалавриата 09.03.03 «Прикладная информатика» — Саратов: ООО Издательский Центр «Наука», 2016. — 40 с.
12. Бессонов Л.В. Операционные системы. Опорный конспект лекций: Учебное пособие для студентов, обучающихся по направлению подготовки бакалавриата 09.03.03 «Прикладная информатика», 38.03.05 «Бизнес-информатика» — Саратов: ООО Издательский Центр «Наука», 2016. — 44 с.

Учебное издание

Донник А. М.

Операционные системы

Практикум

*Учебное пособие для студентов, обучающихся
по направлениям подготовки бакалавриата
09.03.03 «Прикладная информатика»,
38.03.05 «Бизнес-информатика»,
02.03.01 «Математика и компьютерные науки»*

Подписано в печать 01.12.2016. Формат 60x84 1/16. Бумага офсетная.
Гарнитура Times New Roman. Печать RISO. Объем 2,5 печ. л.
Тираж 100 экз. Заказ № 206.

ООО Издательский Центр «Наука»
410012, г. Саратов, ул. Пугачевская, 117, оф. 50

Отпечатано с готового оригинал-макета
Центр полиграфических и копировальных услуг
Предприниматель Серман Ю.Б. Свидетельство № 3117
410012, Саратов, ул. Московская, д.152, офис 311, тел. 26-18-19

САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО