

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра технологий программирования

Лабораторная работа №3

По курсу “Системное программирование”

Инструментальные средства обработки данных и программирования

Методические указания по выполнению лабораторной работы

Подготовила: Давидовская М.И.,

Ст. преподаватель кафедры ТП

Минск, 2023 г.

Содержание

1. Методические материалы.....	3
1.1. Перенаправление ввода вывода и фильтрация.....	3
1.2. Использование утилиты awk.....	4
1.2. Поточковый редактор sed.....	7
2. Задания лабораторной работы.....	8
Критерии оценивания.....	8
Содержание отчета по лабораторной работе.....	8
Задание 2.1. Примеры перенаправления потоков ввода-вывода и фильтрации. .9	
Пример 1. Перенаправление стандартного ввода и вывода.....	9
Вопросы:.....	10
Пример 2. Перенаправление стандартного вывода.....	10
Пример 3. Фильтрация по строкам и перенаправление стандартного потока вывода.....	11
Пример 4. Сортировка и перенаправление стандартного потока вывода.....	11
Пример 5. Перенаправление стандартных потоков вывода и вывода ошибок	11
Пример 6. Перенаправление стандартных потоков ввода/вывода и преобразование регистра:.....	12
Пример 7. Использование каналов для создания конвейеров.....	13
Задание 2.2. Обработка данных с помощью awk в примерах.....	13
Примеры к заданию 2.2.....	13
Пример 1.....	13
Пример 2.....	14
Пример 3.....	14
Пример 4.....	15
Пример 5.....	15
Пример 6.....	16
Пример 7.....	16
Пример 8.....	16
Пример 9.....	16
Пример 10.....	16
Пример 11.....	16

Пример 12.....	16
Пример 13.....	16
Пример 14.....	17
Пример 15.....	17
Задание 2.3. Обработка строк файла с применением потокового редактора sed	17
Примеры к заданию 2.3.....	17
Задание 2.4. Для самостоятельной работы.....	19
Варианты.....	19
Вариант 1.....	19
Вариант 2.....	19
Вариант 3.....	20
Вариант 4.....	20
Вариант 5.....	21
Вариант 6.....	22
Вариант 7.....	23
Вариант 8.....	23
Вариант 9.....	24
Вариант 10.....	24
Задание 2.5. Обработка текстовых данных и каналы.....	25

1. Методические материалы

Цели работы:

- Изучить утилиту awk, ее функциональные возможности и синтаксис.
- Изучить потоковый редактор sed и его возможности.

1.1. Перенаправление ввода вывода и фильтрация

Многие утилиты в среде UNIX работают с потоками ввода и вывода. При этом у каждой программы есть стандартный поток ввода (обычно клавиатура) стандартный поток вывода (монитор) и стандартный поток вывода ошибок (тоже монитор). Во многих оболочках имеется возможность изменять стандартные потоки, например, чтобы программа читала данные не с клавиатуры, а из файла, или чтобы программа выводила данные не на экран, а сразу записывала их в файл. Кроме того, можно сделать так, чтобы поток вывода одной программы являлся потоком ввода другой. Эти механизмы называются перенаправлением потоков и значительно увеличивают возможности системы по манипулированию

различными данными.

Рассмотрим этот механизм на конкретных примерах.

1.2. Использование утилиты **awk**

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

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

Утилиту **awk** можно использовать для создания отчетов, поиска заданных текстовых фрагментов, выполнения вычислений на основе вводимых данных, для работы со структурами, напоминающими базы данных, т.е. состоящих из записей, подразделяющихся на поля, разделенные специальными символами.

Схема вызова **awk** выглядит так:

```
$ awk options program file
```

awk воспринимает поступающие к нему данные в виде набора записей. Записи представляют собой наборы полей. Упрощенно, если не учитывать возможности настройки **awk** и говорить о некоем вполне обычном тексте, строки которого разделены символами перевода строки, запись — это строка. Поле — это слово в строке.

Рассмотрим наиболее часто используемые ключи командной строки **awk**:

- **-F fs** — позволяет указать символ-разделитель для полей в записи.
- **-f file** — указывает имя файла, из которого нужно прочесть **awk**-скрипт.
- **-v var=value** — позволяет объявить переменную и задать её значение по умолчанию, которое будет использовать **awk**.
- **-mf N** — задаёт максимальное число полей для обработки в файле данных.
- **-mr N** — задаёт максимальный размер записи в файле данных.
- **-W keyword** — позволяет задать режим совместимости или уровень выдачи предупреждений **awk**.

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

Синтаксис команды **awk** следующий:

\$ 'Шаблон или условие {действие}' имена_файлов

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

Пример 1

```
$ awk '/интерфейс Gnome/ {print}' /home/student/ *
```

Ищутся все файлы в каталоге **/home/student**, в которых встречается последовательность из слов **интерфейс Gnome** и результаты направляются на стандартное устройство вывода — действие **{print}**. В результатах выводятся имена файлов и строки, содержащие шаблон, из домашнего каталога пользователя **student**. Для применения команды заменить слово **student** на имя своего пользователя в операционной системе.

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

```
$ awk '/ SHELLDATA|SHDATA/' /home/student/ *
```

Для применения команды заменить слово **student** на имя своего пользователя в операционной системе.

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

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

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

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

```
$ awk '{print $1, $2, $5}' /home/student/list_students
```

Для применения команды предварительно создайте в домашнем каталоге файл **list_students** и замените имя пользователя **student** в пути к файлу **/home/student/list_students** на имя своего пользователя в операционной системе.

В результате выполнения на экран выводится следующая таблица, поля которой будут выровнены.

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

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

```
$ awk '{print $0}' /home/student/list_students
```

Для применения команды замените имя пользователя **student** в пути к файлу **/home/student/list_students** на имя своего пользователя в операционной системе.

Результат вывода на экран будет следующий:

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

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

```
$ awk '/5/ {print $0}' /home/student/list_students
```

Бетинов Евгений ЭКТ 2 5
Кошкин Владимир ЭКТ 1 5

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

Для применения команды замените имя пользователя **student** в пути к файлу **/home/student/list_students** на имя своего пользователя в операционной системе.

Пример 6. Выбрать из списка студентов факультета ЭКТ, 1 курса:

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

Команда применяется к файлу **list_students** в текущем каталоге.

Подробнее применение утилиты **awk** рассматривается в материале, опубликованном в курсе «**Системное программирование**» по ссылке <https://edufpmi.bsu.by/mod/resource/view.php?id=28829>.

1.2. Поточковый редактор sed

Команда sed (сокращение от stream editor — «редактор потока») является автоматическим текстовым редактором, который принимает входящий поток (файл или стандартный ввод), изменяет его в соответствии с некоторым выражением и выводит результат в стандартный вывод. Во многих отношениях команда sed подобна команде ed, первичному текстовому редактору Unix. Она обладает множеством операций, инструментами подстановки и возможностями работы с адресацией.

Хотя команда sed является довольно большой и ее детальное рассмотрение выходит за рамки курса, легко понять, как она устроена. В общих чертах, команда sed воспринимает адрес и операцию как один аргумент. Адрес является набором строк, и команда решает, что делать с этими строками.

Очень распространенная задача для команды sed: заменить какое-либо регулярное выражение текстом, например, так:

```
$ sed 's/exp/text/'
```

Так, если требуется заменить первое двоеточие в файле /etc/passwd на символ %, а затем отправить результат в стандартный вывод, следует выполнить следующую команду:

```
$ sed 's:/:/%' /etc/passwd
```

Чтобы заменить все двоеточия в файле /etc/passwd, добавьте спецификатор g в конце операции, как в примере ниже:

```
$ sed 's:/:/g' /etc/passwd
```

Рассмотрим команду, которая работает построчно; она считывает файл /etc/passwd и удаляет строки с третьей по шестую, а затем отправляет результат в стандартный вывод:

```
$ sed 3,6d /etc/passwd
```

В этом примере число 3,6 является адресом (диапазоном строк), а флаг d — операцией (удаление). Если адрес опустить, команда sed будет работать со всеми строками входного потока. Двумя самыми распространенными операциями команды sed являются s (найти и заменить) и d.

В качестве адреса можно также использовать регулярное выражение. Эта команда удаляет любую строку, которая соответствует регулярному выражению exp :

```
$ sed '/exp/d'
```

Подробнее применение потокового редактора sed рассматривается в материале, опубликованном в курсе «**Системное программирование**» по ссылке <https://edufpmi.bsu.by/mod/resource/view.php?id=28830>.

2. Задания лабораторной работы

Критерии оценивания

Оценка 4

Выполнены задания 2.1-2.3. Файлы протоколов команд и меток времени с недочетами. Лабораторная работа сдана с задержкой в 2 недели.

Оценка 5-6

Выполнены задания 2.1.-2.4 (для задания 2.4 задачи 1, 2, 4). Представлен отчет, файлы протоколов и меток времени в git-репозитории. Отчет, файлы протоколов команд и меток времени могут содержать ошибки. Лабораторная работа сдана с задержкой в 1 неделю.

Оценка 7-8

Выполнены задания задания 2.1.-2.5 на хорошем уровне (есть недочеты). Представлен отчет, ответы на вопросы, файлы протоколов и меток времени в git-репозитории. Отчет, файлы протоколов команд и меток времени могут содержать незначительные ошибки. Лабораторная работа сдана с задержкой в 1 неделю.

Оценка 9

Выполнены задания для самостоятельной работы №2.1-2.5 на отличном уровне. Представлен отчет, ответы на вопросы и файлы протоколов и меток времени в git-репозитории. Отчет, файлы протоколов команд и меток времени не содержат ошибок. Лабораторная работа сдана в срок.

Содержание отчета по лабораторной работе

1. Цель работы.
2. Вариант задания.
3. Протоколы выполненных действий по заданиям в виде файлов меток времени и выполненных команд. Протоколы вести только для команд, для **файлов сценариев — нет.**
4. Отчет обязательно должен содержать примеры команд, которые не протоколировались командой script, протоколированные команды — на усмотрение студента, ответы на вопросы и др.
5. Исходные код скриптов опубликовать в каталог /src репозитория, который содержит файл README с описанием сценариев.

Отчет должен быть опубликован в git-репозитории на github. Все результаты лабораторной работы должны быть опубликованы в git-репозитории, ссылка на который доступна в курсе «[Системное программирование](#)».

В файле Readme в корневом каталоге проекта на github должна быть ссылка на отчёт и краткая информация о сценариях в каталоге /src. Отчет опубликовать во внешнем хранилище или в репозитории в каталоге /docs. Если в лабораторной работе необходимо написать программу/ы, то отчёт должен результаты тестов по каждой программе и ответы на контрольные вопросы.

Пример оформления файла Readme может быть таким:

```
# Overview

Report on LabRabota1.

# Usage

// Заменить <<link>> и <<folder>> на соответствующие ссылки/названия
To check, please, preview report by <<link>> and script files
in <<folder>>.

# Author

Your name and group number.

# Additional Notes

// СКОПИРОВАТЬ И ВСТАВИТЬ ССЫЛКУ НА свой РЕПОЗИТОРИЙ, НАПРИМЕР
https://github.com/maryiad/lab3-task1-gr16-david
```

Каждая лабораторная работа содержит тексты задач и контрольные вопросы, ответы на которые проверяются преподавателем при приёме работы у студента.

Выполнение студентом лабораторной работы и сдача её результатов преподавателю происходит следующим образом:

1. Студент выполняет разработку программ.
2. В ходе разработки студент обязан следовать указаниям к данной задаче (в случае их наличия). Исходные тексты программ следует разрабатывать в соответствии с требованиями к оформлению, приведёнными в приложении.
3. Студент выполняет самостоятельную проверку исходного текста каждой разработанной программы и правильности её работы, а также свои знания по теме лабораторной работы.

Задание 2.1. Примеры перенаправления потоков ввода-вывода и фильтрации

Пример 1. Перенаправление стандартного ввода и вывода

1. Все примеры выполнить в каталоге **task31** репозитория лабораторной работы:

2. Создать файл `file.my` с помощью утилиты `cat` и перенаправления стандартных потоков ввода и вывода:

```
$ cat > file.my <<END
> name
> filename
> listname
> list
> copy
> printname
> university
> university name
> lastname
> last
> lastname
> listname
> firstname
> position name
> job
> job description
> END
```

3. Вывести созданный файл на стандартный вывод с помощью команд `cat` и `nl`:

```
$ cat file.my
$ nl file.my
```

Вопросы:

1. Чем отличается вывод во втором случае от первого?
2. Какая опция команды `cat` позволяет пронумеровать строки?

Пример 2. Перенаправление стандартного вывода

Задание выполняется в каталоге **task31** репозитория лабораторной работы.

Перенаправить один файл в другой (создать копию файла):

```
$ cat file.my > file.my.new
```

команда `cat file.my` должна вывести содержимое файла `file.my` на экран, но в данном случае мы указали, чтобы все данные были помещены в файл `file.my.new` с помощью спецсимвола `>` - перенаправления стандартного потока вывода. Таким образом, мы получили еще один способ копирования файлов.

Пример 3. Фильтрация по строкам и перенаправление стандартного потока вывода

Задание выполняется в каталоге **task31** репозитория лабораторной работы.

1. Выполнить команду:

```
$ grep name file.my > names
```

сохраняет все строки файла `file.my`, содержащие сочетание букв "name" в файле `names`.

При этом предыдущее содержимое файла `names` будет удалено, чтобы избежать этого можно добавить новые данные в конец файла с помощью символов ">>".

2. Выполнить команду:

```
$ grep list file.my >> names
```

новые данные будут добавлены в конец файла `names`.

Для перенаправления потока ввода следует использовать символ "<".

Пример 4. Сортировка и перенаправление стандартного потока вывода

Задание выполняется в каталоге **task31** репозитория лабораторной работы.

Отсортировать строки в файле `file.my` и вывести на экран:

```
$ sort < file.my
```

Перенаправление можно комбинировать. Например, отсортировать и сохранить в файле `file.my.sorted`:

```
$ sort <file.my > file.my.sorted
```

Пример 5. Перенаправление стандартных потоков вывода и вывода ошибок

Задание выполняется в каталоге **task31** репозитория лабораторной работы.

Для перенаправления потока вывода ошибок можно использовать символы "2>".

1. Вывести на стандартный вывод файлы `file.my` и `NameS`. Все ошибки сохранить в файле `errors`:

```
$ cat file.my NameS 2>errors
```

2. Выведите файл `errors` на стандартный вывод. Какая ошибка сохранена в него?

3. Потоки вывода и ошибок можно совместить. Перенаправлять поток ошибок (2) туда же, куда и поток вывода (1), т.е. в файл `output`. Просмотрите содержимое файла `output`:

```
$ cat file.my NameS > output 2>&1
```

4. Что произойдет, если выполнить команду

```
$ cat file.my NameS 2>&1 > someoutput
```

Почему результат отличается от предыдущей команды?

Пример 6. Перенаправление стандартных потоков ввода/вывода и преобразование регистра:

Задание выполняется в каталоге **task31** репозитория лабораторной работы.

1. Если команде `cat` не передается аргумент и стандартный ввод перенаправлен в файл, то весь ввод с клавиатуры до нажатия клавиш `<Ctrl+d>` будет перенаправлен в файл.

```
$ cat > typedin.txt
```

This time, when text is typed at the keyboard,
it is redirected to the file `typedin.txt`.

`<Ctrl+d>`

```
$ ls -l typedin.txt
```

```
$ cat typedin.txt
```

2. Повторите предыдущий шаг, подставив вместо команды `cat` команду `tr`.

```
$ tr 'aeiou' 'AEIOU' > trfile.txt
```

This time, when text is typed at the keyboard,
it is not echoed back to the screen with the translations made.
Instead, it is redirected to the file `trfile.txt`.

`<Ctrl+d>`

```
$ ls -l trfile.txt
```

```
$ cat trfile.txt
```

Какой результат содержит файл `trfile.txt`?

3. Команда `cat` принимает в качестве аргумента имя файла или стандартный ввод, перенаправленный из файла. Проверьте это при помощи следующих двух команд:

```
$ cat packages1.txt
```

```
$ cat < packages1.txt
```

4. Однако команда `tr` принимает ввод только из стандартного канала.

```
$ tr 'aeiou' 'AEIOU' < packages1.txt
```

5. В следующем примере стандартный ввод и вывод одновременно перенаправляются

```
$ tr 'aeiou' 'AEIOU' < packages1.txt > packages1.trfile.txt  
$ ls -l packages1.txt packages1.trfile.txt  
$ cat packages1.trfile.txt
```

Пример 7. Использование каналов для создания конвейеров

Задание выполняется в каталоге **task31** репозитория лабораторной работы.

С помощью символа "|" можно организовать перенаправление потока вывода одной программы в поток ввода другой.

1. Содержимое файла file.my направить команде grep и результат работы последней сохранить в файле grep_names.

```
$ cat file.my | grep name > grep_names
```

Такая составная команда называется **конвейер**. В конвейер можно объединять произвольное число команд.

2. Подсчитать количество строк в файле file.my, в которых встречается последовательность символов "name":

```
$ cat file.my | grep name | wc -l
```

3. Вывести в отсортированном виде содержимое файла file.my без дубликатов (команда uniq удаляет дубликаты):

```
$ cat file.my | uniq | sort
```

Стоит отметить, что отдельной программы выполняющей такую функцию нет, но благодаря использованию конвейера такой результат можно получить одной командой.

Задание 2.2. Обработка данных с помощью awk в примерах

Изучите примеры задания 2.2 и выполните их в ОС Ubuntu .

Примеры к заданию 2.2

Пример 1

Предварительно создадим файл для обработки. Для этого выполним следующие действия:

1. Создаем каталог **awk-examples** в каталоге репозитория лабораторной работы.

```
$ mkdir awk-examples
```

2. Перейдите в домашний каталог и выведите сведения о текущем каталоге:

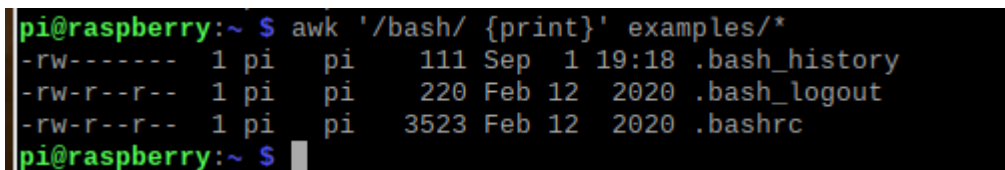
```
$ cd
$ pwd
```

3. Перенаправим вывод команды `ls` расширенном формате в файл, например `log.txt`:

```
$ ls -la > ~/указать путь до каталога awk-examples/log.txt
```

4. И выполним команду для поиска данных в созданном файле или наборе файлов. Будем искать строки, содержащие слово «**bash**», находясь в каталоге репозитория

```
$ awk '/bash/ {print}' awk-examples/*
```



```
pi@raspberrypi:~$ awk '/bash/ {print}' examples/*
-rw----- 1 pi pi 111 Sep 1 19:18 .bash_history
-rw-r--r-- 1 pi pi 220 Feb 12 2020 .bash_logout
-rw-r--r-- 1 pi pi 3523 Feb 12 2020 .bashrc
pi@raspberrypi:~$
```

Ищутся все файлы в каталоге **awk-xamples**, который находится в каталога репозитория лабораторной работы и в которых встречается слово **bash**. Результаты направляются на стандартное устройство вывода — действие **{print}**. В результатах выводятся строки, содержащие шаблон.

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

```
$ awk '/config/' awk-examples/*
```

В качестве результата получим записи со словом `config`.

Пример 2

Найти по указанному пути файлы исходных текстов на языке C, включающие слова, например, `QUIT` или `SETDATE`:

```
$ awk '/QUIT|SETDATE/' /usr/include/protocols/*
```

Пример 3

1. Создайте в каталоге **awk-examples** текстовый файл **myfile** следующего содержания:

```
This is a test.

This is the second test.

This is the third test.

This is the fourth test.
```

2. Просмотрите файл:

```
$ cat myfile
```

```
maryia@ThinkPad-E470:~/examples$ cat myfile
This is a test.
This is the second test.
This is the third test.
This is the fourth test.
```

Одна из основных функций **awk** заключается в возможности манипулировать данными в текстовых файлах. Делается это путём автоматического назначения переменной каждому элементу в строке. По умолчанию **awk** назначает следующие переменные каждому полю данных, обнаруженному им в записи:

- **\$0** — представляет всю строку текста (запись).
- **\$1** — первое поле.
- **\$2** — второе поле.
- **\$n** — n-ное поле.

2. Выполните вывод первого элемента каждой строки:

```
$ awk '{print $1}' myfile
```

3. Выполните вывод третьего элемента каждой строки:

```
$ awk '{print $3}' myfile
```

Пример 4

1. **awk** позволяет изменять регистр символов. В примере ниже необходимо ввести команду и две строки текста, одна из которых будет преобразована в строку в верхнем регистре:

```
$ awk '/foo/ { print toupper($0); }'
```

```
This line contains bar.
```

```
This line contains foo.
```

```
THIS LINE CONTAINS FOO.
```

2. Для завершения ввода нажмите комбинацию клавиш **Ctrl+C**.

Пример 5

1. Создать в каталоге **awk-xamples** текстовый файл **list_students**, отсортированный по алфавиту, каждая строка которого включает: фамилию, имя, факультет, курс, рейтинговая оценка, и состоящий из следующих строк-записей:

```
Асташко Иван ПИ 1 6
```

```
Бузун Евгений КБ 1 9
```

```
Кравченя Леонид ПМ 1 4
```

```
Кошкин Владимир ИН 1 7
```

```
Липин Федор ПИ 2 4
```

Пенов Николай ПМ 1 5

Яшин Петр КБ 4 8

2. Вывести список, состоящий из фамилий, имен и рейтинговых оценок:

```
$ awk '{print $1,$2,$5}' list_students
```

Пример 6

Выбрать из исходного файла *list_students* студентов с рейтинговой оценкой 8 и распечатать все значения полей для выбранных записей:

```
$ awk '/8/ {print $0}' list_students
```

В случае, если оценка встречается в других полях, например, курс 4 и оценка 4, то условия выборки недостаточны.

Пример 7

Выбрать из списка студентов, рейтинговая оценка которых 4:

```
$ awk '($5==4)' list_students
```

Пример 8

Выбрать из списка студентов специальности КБ, 1 курса:

```
$ awk '($3=="КБ")&&($4==1)' list_students
```

Пример 9

Выбрать из файла со списком студентов все имена с длиной >5

```
$ awk '(length($2)>5) {print}' list_students
```

Пример 10

Вывести данные исходного файла в виде таблицы и пронумеровать строки:

```
$ awk '{print NR, $0}' list_students
```

Пример 11

Вывести все имена студентов в верхнем регистре:

```
$ awk '{print toupper ($2)}' list_students
```

Пример 12

Посчитать суммарный балл оценок студентов:

```
awk '{sum += $5} END {print("SUM=",sum)}' list_students
```

Пример 13

Присутствует ли фамилия Леонов в списке файла *list_students*?

```
$ awk '($1 ~ /Кравченя/)' list_students
```


Пример 14

Проверить, есть ли запись студента Липина в списке файла *list_students*, причем фамилия может быть написана с прописной или строчной буквы:

```
$ awk '($1 ~ /[Лл]ипин/)' list_students
```

Вывести все записи кроме студента Липин:

```
$ awk '($1 ~ /[Лл]ипин/)' list_students
```

Пример 15.

1. Создайте в каталоге **awk-xamples** текстовый файл *colours.csv* вида:

```
name,color,amount
apple,red,4
banana,yellow,6
strawberry,red,3
grape,purple,10
apple,green,8
plum,purple,2
kiwi,brown,4
potato,brown,9
pineapple,yellow,5
```

2. Выберите все записи из файла *colours.csv*, количество которых больше 5, выполнив команду:

```
$ awk -F", " '$3>6 {print $1, $2}' colours.csv > output.txt
```

В команде выше явно был указан разделитель данных, т.е. “,”. В результате выполнения команды будет создан файл, содержащий записи согласно условию.

Задание 2.3. Обработка строк файла с применением потокового редактора sed

Изучите примеры задания 2.3 и выполните их в ОС Ubuntu.

Примеры к заданию 2.3

Создайте в каталоге репозитория лабораторной работы каталог **sed-examples** и все команды выполните в нем в качестве текущего:

1. Сохраните данные ниже в файл с именем books:

Book one.

The second book.

The third.

This is book four.

Five.

This is the sixth.

This is book seven.

Eighth and last.

2. Чтобы вывести все строки файла и продублировать строки согласно шаблону, т. е. содержащие слово book выполните команду

```
$ sed '/book/ p' books
```

3. Чтобы выбрать определенные строки, например содержащие слово book, выполните команду

```
$ sed -n '/book/ p' books
```

4. Чтобы вывести часть файла, например, строки с 2 по 5, выполните команду

```
$ sed -n '2,5 p' books
```

5. Для выполнения более сложных и длинных инструкций можем использовать файл программы для sed. Выполним команду из примера 4, указав параметры в файле records.

Содержимое файла records:

```
2,5 p
```

Пример команды из примера 4 с выполнением инструкций из файла:

```
$ sed -n -f records books
```

6. В данном примере выбираем строку 3 и используем инструкцию Добавить, чтобы добавить разделитель строк и текст «My favorite book.» к третьей строке:

Содержимое файла appends

```
3 a\
```

```
My favorite book.
```

Пример команды с добавлением строки:

```
$ sed -f appends books
```

7. В данном примере требуется вставить разделитель строк и текст «SKARBONKA.» перед строками, в которых содержится слово «This»:

Содержимое файла insert

```
/This/ i\
```

```
SKARBONKA.
```

Пример команды с со вставкой строк:

```
$ sed -f insert books
```

8. В следующем примере выполняется замена слова book на nove l

```
$ sed -n 's/book/novel/ p' books
```

Задание 2.4. Для самостоятельной работы

Перед выполнением задачи 3 из задания 2.4 изучите теорию <https://awk.js.org/help.html#programs-user-fn> и разберите практический пример <https://awk.js.org/help.html#practice-user-fn>.

Варианты

Вариант 1.

Задача 1. Перенаправьте вывод команды `ls`, просматривающей содержимое домашнего каталога в файл `file_list.txt`. В выводе `ls` отображать все файлы, включая скрытые, `inode` (индексный дескриптор) файла и размер в килобайтах, с сортировкой по размеру.

Задача 2. Напишите AWK-программу для вывода четных строк списка автомобилей из файла `cars.txt` (<https://edufpmi.bsu.by/mod/resource/view.php?id=28836>), в котором буквами в верхнем регистре будет обозначен только производитель. Модель и последующие поля каждой строки должны появляться в том виде, в котором они сохранены в файле `cars`.

Задача 3. Создайте скрипт `awk`, содержащий функцию для создания массива 3x4, функции определения максимального и минимального элемента массива. Функции вызываются из функции `main()`.

Задача 4. Создайте файл из 10 строк, в каждой строке которого текст содержит от 5 до 12 слов и разделен запятыми. Всего в каждой строке должно быть две запятых. Напишите `sed`-команду, которая выводит на стандартный вывод данные из файла с заменой второй запятой на «|».

Вариант 2.

Задание 1. Создайте в каталоге текущего задания новый каталог `docs` с помощью команды `mkdir` с правами доступа 751 двумя способами: 1) только с помощью команды `mkdir` и выводом о выполнении команды 2) с помощью `hex`, `chmod` и `mkdir`. Вывод об успешном выполнении команды сохранять для каждого способа.

Задача 2. Напишите AWK-программу, которая использует файл `cars.txt` (<https://edufpmi.bsu.by/mod/resource/view.php?id=28836>) выводит данные о всех автомобилях, цена которых меньше \$6000, с названием модели в верхнем регистре и отправляет эти данные на стандартный вывод, а также подсчитывает суммарное значение стоимости для выведенных и выводит последней строкой.

Задача 3. Создайте скрипт `awk`, обрабатывающий файл со списком студентов группы и их примерные оценки по экзаменам последней сессии (не менее пяти оценок для каждого), например *Иванов Иван 7 6 8 5 9*. Разработайте функции для

расчета суммы баллов каждого студента, среднюю оценку для каждого студента и выводящие список в следующем формате с учетом разделителей и отступов:

Иванов Иван:

сумма баллов: 35

средний балл: 7

Задача 4. Создайте текстовый файл из 20 строк в каждой строке не менее 5 слов, содержащий не менее 12 слов cat в разном регистре. Напишите sed-программу по имени `ins`, которая копирует файл на стандартный вывод, заменяя все встречающиеся слова cat в нижнем регистре на dog и предваряя каждую измененную строку строкой предупреждения «Следующая строка была изменена:».

Вариант 3.

Задача 1. Создайте псевдоним `lr`, который вызывает команду `ls` со следующими возможностями: псевдоним перечисляет файлы в длинном формате. Перенаправьте вывод команды `lr` для домашнего каталога с сортировкой по типу файла и фильтрацией по дате больше >10 числа независимо от месяца в файл `file_list.txt`.

Задача 2. Напишите AWK-программу, которая использует файл `cars.txt` (<https://edufpmi.bsu.by/mod/resource/view.php?id=28836>) выводит данные о всех автомобилях, цена которых превышает \$5000, и отправляет эти данные на стандартный вывод, а также подсчитывает среднее значение стоимости для выведенных и выводит последней строкой.

Задача 3. Создайте скрипт awk, обрабатывающий файл со списком студентов группы и их примерные оценки по экзаменам последней сессии (не менее пяти оценок для каждого), например *Иванов Иван 7 6 8 5 9*. Разработайте функции для расчета средней оценки для каждого студента, сортировки в прямом порядке оценок и выводящие список в следующем формате с учетом разделителей и отступов:

Иванов Иван:

средний балл: 7

сортировка прямая: 5 6 7 8 9

Задача 4. Создайте файл, содержащий сведения об авторе и названии книги, фамилии автора содержатся в разном регистре, например *Иванов, иВанов, ИВанов, иваНоВ* и т.д. Напишите сценарий sed, копирующую на стандартный вывод строки файла для одного из авторов с заменой вывода его имени и фамилии независимо от регистрации на значения в с первой заглавной.

Вариант 4.

Задача 1. Используя команду `who`, определите пользователей (авторизуйтесь не

менее чем под тремя разными учетными записями), работающих в системе и отобразите данные о дате и времени загрузки системы, текущий уровень, на котором выполняются процессы пользователя, и сведения о пользователе. Подсчитайте количество пользователей, сортировку по имени в прямом порядке выведите на стандартный вывод, а в обратном — в файл `users.txt`.

Задача 2. Используйте AWK для определения, сколько строк в файле `/usr/share/dict/words` (проверить на Ubuntu) содержат подстроку `abul`, а также выводите их нумеруя. Определите самую длинную строку. Если данный файл отсутствует выполнять задание для любого текстового файла, который можно найти в подкаталогах каталога `/usr/share`.

Задача 3. Создайте скрипт `awk`, обрабатывающий файл со списком студентов группы и их примерные оценки по экзаменам последней сессии (не менее пяти оценок для каждого), например *Иванов Иван 7 6 8 5 9*. Разработайте функции для расчета суммы баллов для каждого студента, сортировки в обратном порядке оценок и выводящие список в следующем формате с учетом разделителей и отступов:

Иванов Иван:

сумма баллов: 35

сортировка обратная: 9 8 7 6 5

Задача 4. Напишите `sed`-программу по имени `div`, которая копирует файл из 50 строк на стандартный вывод, причем копирует строки 5-10 в файл по имени `first` и копирует последние 12 строк в файл по имени `last`.

Вариант 5.

Задача 1. Задача 1. Создайте псевдоним `lr`, который вызывает команду `ls` со следующими возможностями: псевдоним классифицирует файлы, присоединяя индикатор типа файла к именам файлов. Перенаправьте вывод команды `lr` для домашнего каталога в файл `file_list.txt`, подсчитав количество файлов и записав это значение последней строкой в файл `file_list.txt`.

Задача 2. Напишите `awk`-программу, которая читает данные из файла <https://www.rfc-editor.org/rfc/rfc-ref.txt>, выводит строки с фразой «*Network Protocol*» подсчитывает их общее количество и количество строк в файле `rfc-ref.txt`.

Задача 3. Ознакомьтесь с документацией из руководства `man` по функции `strftime()`. Изучите примеры команд с применением функций `systemtime()`, `strftime()`:

```
$ echo | awk '{print systemtime();}'
```

```
$ echo | awk '{print strftime();}'
```

```
$ echo | awk '{print strftime("%d-%m-%y %H-%M-%S",systemtime());}'
```

Создайте скрипт `awk`, состоящий функции получения текущего времени и преобразования в `unixtime` с помощью `systemtime()`, функции, форматирующей с помощью `strftime()` полученное значение и выводящей следующие значения для полученной даты:

time["second"] — seconds (0 - 59)
time["minute"] — minutes (0 - 59)
time["hour"] — hours (0 - 23)
time["althour"] — hours (0 - 12)
time["monthday"] — day of month (1 - 31)
time["month"] — month of year (1 - 12)
time["monthname"] — name of the month
time["shortmonth"] — short name of the month

Созданные функции вызывать в main().

Задача 4. Сохраните вывод команды `ls -la` для каталога `/etc` в файл. Напишите скрипт `sed`, выполняющую замену «hosts» на «HOSTS», «group» на «GRPOUP», «groups» на «GRPOUPS» и выводящую на экран все строки за исключением строк, начинающихся с символа «d».

Вариант 6.

Задача 1. Перенаправьте вывод списка файлов в длинном формате (расширенный формат) с сортировкой по размеру в файл `list.txt` (если файл существует, то новые строки в него должны быть добавлены) и вывести на стандартный вывод строки 4 первые строки из файла `list.txt`.

Задача 2. Напишите `awk`-программу, которая читает данные из файла <https://www.rfc-editor.org/rfc/rfc100.txt> и выводит первое слово каждой строки, если первым символом строки является буква, буквами в нижнем регистре, а последнее слово каждой строки буквами, переведенными в верхний регистр.

Задача 3. Ознакомьтесь с документацией из руководства `man` по функции `strftime()`. Изучите примеры команд с применением функций `systime()`, `strftime()`:

```
$ echo | awk '{print systime();}'  
$ echo | awk '{print strftime();}'  
$ echo | awk '{print strftime("%d-%m-%y %H-%M-%S",systime());}'
```

Создайте скрипт `awk`, состоящий функции получения текущего времени и преобразования в `unixtime` с помощью `systime()`, функции, форматирующей с помощью `strftime()` полученное значение и выводящей следующие значения для полученной даты:

time["year"] — year modulo 100 (0 - 99)
time["fullyear"] — full year
time["weekday"] — day of week (Sunday = 0)
time["altweekday"] — day of week (Monday = 0)
time["dayname"] — name of weekday
time["shortdayname"] — short name of weekday
time["yearday"] — day of year (0 - 365)
time["time zone"] — abbreviation of time zone name

time["ampm"] — AM or PM designation

time["weeknum"] — week number, Sunday first day

time["altweeknum"] — week number, Monday first day

Задача 4. Напишите sed-команду, копирующую файл cars.txt (<https://edufpmi.bsu.by/mod/resource/view.php?id=28836>) на стандартный вывод, удаляя при этом все строки, содержащие «chevy», и подсчитывающие количество удаленных строк и среднюю стоимость автомобилей «chevy».

Вариант 7.

Задача 1. Создайте псевдоним lr, который вызывает команду ls со следующими возможностями: псевдоним перечисляет файлы в порядке времени их модификации. Перенаправьте вывод команды lr в текущем каталоге в файл file_list.txt, а первые 7 строк вывода в файл seven-rows.txt.

Задача 2. Напишите AWK-программу для вывода списка автомобилей из файла cars.txt (<https://edufpmi.bsu.by/mod/resource/view.php?id=28836>), которая выводит самый старый и самый новый автомобиль ford с выводом модели заглавными буквами, сумму и среднюю стоимость всех автомобилей ford.

Задача 3. Написать сценарий AWK, подсчитывающий количество уникальных слов в текстовом файле, выводящий общее количество слов, уникальное слово и его количество в файле. Подсчет и вывод реализовать в виде отдельных функций, которые из вызвать из main().

Задача 4. Напишите sed-сценарий, копирующий файл на стандартный вывод, удаляя при этом все пустые строки (например, строки без символов) и строки, начинающиеся с символа (, или буквы в верхнем регистре, сохраняя результат в отдельный файл.

Вариант 8.

Задача 1. Записать команду с перенаправлением, получающую сведения о файлах из домашнего каталога в длинном формате, вырезающую только записи обычных файлов, передающую полученный список за исключением первых трех строк команде sort, которая перенаправляет результат команде wc, подсчитывающей количество строк.

Задача 2. Напишите awk-программу, которая читает данные из файла <https://www.rfc-editor.org/rfc/rfc1139.txt> и выводит второе слово каждой строки с буквами, переведенными в верхний регистр, подсчитывает количество измененных слов и количество уникальных слов и выводит результаты подсчета.

Задача 3. Написать сценарий AWK, подсчитывающий количество частоту встречаемости слов в текстовом файле, выводящий всего строк, всего слов в файле, слово и его количество в файле. Подсчет и вывод реализовать в виде отдельных функций, которые из вызвать из main().

Задача 4. Напишите sed-программу по имени stringscopy, которая выводит файл на стандартный вывод, копирует первые 6 строк в файл по имени firstf, а остальную часть файла в файл по имени lastf, добавив в каждый из файлов его

имя.

Вариант 9.

Задача 1. Записать команду с перенаправлением, получающую сведения о файлах из каталога /etc в длинном формате, вырезающую только записи обычных файлов, сортирующей список и выводящей список на стандартный вывод с нумерацией строк.

Задача 2. Напишите AWK-программу, которая использует файл cars.txt (<https://edufpmi.bsu.by/mod/resource/view.php?id=28836>) выводит данные о всех автомобилях, цена которых в интервале \$3000-\$7000, с названием модели в верхнем регистре и отправляет эти данные на стандартный вывод, а также подсчитывает суммарное значение стоимости для выведенных и выводит последней строкой.

Задача 3. Создайте скрипт awk, которая удаляет из файла history строки с одинаковыми командами (аргументы команды не учитывать), сохраняет изменения в новый файл history-copу, подсчитывает и выводит количество удаленных строк. Реализовать каждую из операций в виде функции, которую вызвать из main().

Задача 4. Напишите sed-программу по имени animals, которая копирует файл на стандартный вывод, заменяя все встречающиеся слова bird на animal и предваряя каждую измененную строку строкой предупреждения «Следующая строка была изменена:» .

Вариант 10.

Задача 1. Записать команду с перенаправлением, выполняющую поиск файлов в каталоге /etc по имени, например host, передает результаты поиска команде ls с перенаправлением ошибок на /dev/null.

Задача 2. Напишите AWK-программу, которая использует файл cars.txt (<https://edufpmi.bsu.by/mod/resource/view.php?id=28836>) выводит данные о всех автомобилях, цена которых меньше \$3000, с названием модели в нижнем регистре и отправляет эти данные на стандартный вывод, а также подсчитывает среднее значение стоимости для выведенных и выводит последней строкой.

Задача 3. Напишите AWK-сценарий сортировки записей файла системных паролей /etc/passwd по определённому. Номер поля, по которому выполняется сортировка передать в качестве аргумента сценария. Результаты сортировки сохранить в отдельный файл для поля, содержащего имя для входа, и поля, содержащего идентификатор пользователя.

Задача 4. Напишите sed-команду, копирующую файл на стандартный вывод, удаляя при этом все строки начинающиеся со строчной буквы и строки, начинающиеся с символа % или символов пунктуации (, : .).

Задание 2.5. Обработка текстовых данных и каналы

Выполните задания с применением утилит `grep`, `find`, `exec`, `sort` и т. д. в зависимости от задания:

1. Создайте одной командой с помощью регулярного выражения 5 файлов `yourlastnameN.txt`, где `yourlastname` — Ваша фамилия, `N` — номер файла. Сохраните в каждый из файлов не менее 5 студентов группы по дисциплине Системное программирование. Не менее в 3 файлах должна быть Ваша фамилия. Используя `ls`, `find`, `exec` и `grep`, выведете только строки из всех файлов, содержащие Вашу фамилию.
2. Дополните предыдущую команду командами `tee` и `wc` и сохраните найденные строки в файл `yourlastname.txt` и выведите количество строк и символов на стандартный вывод.
3. Используя команды `ls` и `sort` выведете все файлы из домашнего каталога с сортировкой по размеру.
4. Создайте файл `short_list_spec.txt` со списком специальностей факультета, содержащий поля *трехбуквенное название специальности* (например ПИН — прикладная информатика), *год обучения*, и *количество студентов*, разделенные пробелом. Используя команды `cat`, `sort` и `head`, выводите первые 11 строк, отсортированные в обратном порядке по 2 столбцу.
5. Дополните команду из предыдущего задания командой `sed` с заменой названия краткого названия каждой специальности на полное название и сохранением результата в `full_list_spec.txt`.
5. Создайте файл `zadacha55.txt`, например со следующим текстом:

```
linux is my life
linux has changed my life
linux is best and everthing to me..:)
```

Перенаправьте данные со стандартного вывода команды `cat`, на ввод команды `tr`, преобразовав все строчные буквы в заглавные, и результат вывода перенаправьте в файл, например `output55.txt`.

6. Перенаправьте файл `zadacha55.txt` на стандартный ввод команды `tr`. Команда `tr` должна выполнить замену строчных букв на прописные и перенаправить вывод в файл `output56.txt`.

7. Используя команды `echo` и `tr`, из фразы «the linux staff» удалите букву «t».

```
wc
```

8. Создайте файл `linux_os.txt`, содержащий следующие строки:

```
Red Hat
```

CentOS

Fedora

Debian

Scientific Linux

OpenSuse

Ubuntu

Xubuntu

Linux Mint

Pearl Linux

Slackware

Mandriva

Используя команду `wc`, посчитайте и выведите следующие данные:

- количество строк в файле, количество слов и количество байт;
- только количество строк в файле;
- только количество слов в файле;
- длину самой длинной строки;
- количество символов в файле;