Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЁТ

к лабораторной работе №2

на тему

**ОБРАБОТКА ТЕКСТОВОЙ ИНФОРМАЦИИ.**

**РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ.**

Выполнил: студент гр.253504 Казакевич Г.С.

Проверил: ассистент кафедры информатики Гриценко Н.Ю.

Минск 2025

**СОДЕРЖАНИЕ**

[1 Формулировка задачи 3](#_Toc183307053)

[2 Краткие теоритические сведения 4](#_Toc183307054)

[3 Описание функций программы 5](#_Toc183307055)

[Заключение 7](#_Toc183307056)

[Список использованных источников 8](#_Toc183307057)

[Приложение А (обязательное) исходный код программы 9](#_Toc183307058)

# **1 ФОРМУЛИРОВКА ЗАДАЧИ**

В данной лабораторной работе необходимо изучить и освоить инструменты обработки текстовой информации в *Unix*-подобных системах с использованием средств командной строки, таких как *sed*, *awk*, а также командного интерпретатора *bash*. Основное внимание уделяется разработке скриптов для автоматической трансформации входных текстовых данных в соответствии с заданными условиями. При этом важно получить практические навыки работы с фильтрами потоков, регулярными выражениями, а также основами построения устойчивых к ошибкам скриптов.

Целью работы является не только реализация функциональности для конкретной задачи, но и формирование общего понимания принципов обработки текстов в *Linux*-среде, включая взаимодействие между различными утилитами и построение цепочек команд. В частности, требуется реализовать «автокорректор» — скрипт, автоматически заменяющий строчные буквы на заглавные в начале предложений. Коррекция должна производиться в начале документа, после точки, а также после знаков «!» и «?», при этом необходимо избегать изменений внутри чисел и учитывать переносы предложений на новую строку.

Дополнительно необходимо предусмотреть обработку ошибочных или неочищенных входных данных, чтобы скрипт мог корректно реагировать на нестандартные или повреждённые входные файлы. Выполнение данной лабораторной работы позволит закрепить знания по использованию текстовых утилит *Unix*, научиться применять регулярные выражения на практике и углубить понимание *shell*-программирования в контексте автоматизации обработки данных.

# **2 КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ**

В *Unix*-подобных операционных системах широкое применение находят инструменты командной строки для обработки текстовой информации. Среди них особенно выделяются утилиты *sed*, *awk*, а также интерпретатор команд *bash*, позволяющие создавать мощные и гибкие скрипты для автоматизации обработки файлов и потоков данных. Такие скрипты могут выполнять трансформации текста, фильтрацию данных, поиск по шаблонам, замену строк и другие операции без необходимости ручного редактирования [1].

*Sed* (*stream* *editor*) используется для обработки текста с помощью регулярных выражений, выполняя подстановки, удаление, вставку и изменение строк в потоках данных. Он особенно эффективен для автоматических замен и редактирования больших текстовых файлов [2]. Например, с помощью команды *s*/*old*/*new*/*g* можно заменить все вхождения одного слова на другое.

*Awk* — это язык обработки текстовых данных, ориентированный на построчную обработку файлов. Он позволяет работать с полями, разделёнными пробелами или другими символами, применять условия и выполнять действия над текстом. *Awk* широко используется для форматирования вывода, подсчётов и извлечения информации из таблиц и логов [3].

Командный интерпретатор *bash* обеспечивает обёртку над этими инструментами, позволяя объединять их в скриптах, реализующих сложную логику. С помощью условий (*if*, *case*), циклов (*for*, *while*) и встроенных команд *bash* можно управлять потоком выполнения, проверять корректность входных данных и формировать отчёты .

Обработка текста требует грамотного использования регулярных выражений, которые позволяют задавать шаблоны поиска. В данной работе это особенно важно, поскольку задача заключается в автоматической замене строчных букв на заглавные в начале предложений, включая случаи, когда предложения начинаются с новой строки. При этом необходимо избегать замены внутри чисел и учитывать различные знаки окончания предложений (., !, ?).

Скрипт должен быть устойчивым к ошибкам — это означает проверку на пустые строки, отсутствие необходимых файлов или неверный формат входных данных. Для этого применяются конструкции обработки ошибок и условия, позволяющие скрипту корректно реагировать на нестандартные ситуации.

Таким образом, использование *sed*, *awk* и *bash* в рамках данной лабораторной работы позволяет не только решить конкретную задачу автоматического исправления текста, но и получить ценные практические навыки работы с инструментами командной строки, необходимыми для эффективной обработки данных в системном администрировании, разработке и анализе логов.

# **3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ**

В данной программе автокорректора реализована автоматизированная обработка текстового файла с целью исправления ошибок, связанных с использованием регистра

Если входной файл присутствует, скрипт задействует утилиту *sed* с опцией -*E*, включающей расширенные регулярные выражения. На начальном этапе используется конструкция :*a*;*N*;$!*ba*;, объединяющая содержимое файла в единый текстовый блок. Это необходимо для корректной обработки переносов строк и случаев, когда знак окончания предложения и начальная буква следующего предложения находятся в разных строках.

Первая команда замены *s*/^[[:*space*:]]\*([[:*lower*:]])/\*U*\1/ находит в начале документа возможные пробелы, за которыми следует строчная буква, и преобразует её в заглавную. Такой подход обеспечивает правильное оформление начала текста.

Вторая команда *s*/([.!?])([[:*space*:]\*n*]+)([[:*lower*:]])/\1\2\*u*\3/*g* предназначена для обработки предложений, завершающихся точкой, восклицательным или вопросительным знаком. Она находит такие знаки препинания, за которыми следуют пробельные символы или перенос строки и строчная буква, и преобразует эту букву в заглавную.

На рисунке 3.1 представлено исходное состояние текстового файла до обработки автокорректором.

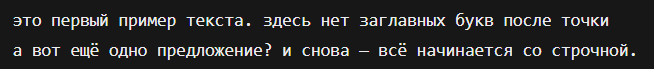


Рисунок 3.1 – Исходный текстовый файл (до обработки)

После выполнения скрипта автокорректора текст преобразуется таким образом, что первая буква документа и первая буква после каждого знака окончания становятся заглавными. Результат работы программы отображается в файле *output*.*txt*, как показано на рисунке 3.2.

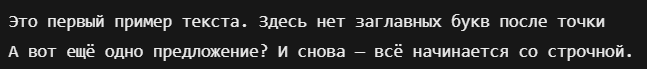


Рисунок 3.2 – Текстовый файл после обработки автокорректором

Таким образом, разработанный автокорректор демонстрирует эффективность использования регулярных выражений и возможностей *Unix*-утилит для автоматизации обработки текстовой информации, обеспечивая стабильную работу даже при ошибочном или неочищенном вводе.

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения лабораторной работы была успешно реализована программа автокорректора, разработанная с использованием *Bash*-скриптов и утилиты *sed*. Основной задачей скрипта стала автоматическая обработка текстового файла с целью корректировки регистра букв в начале предложений. Программа эффективно выполняет анализ содержимого входного файла, проверяет его наличие, а в случае отсутствия — корректно завершает выполнение с соответствующим уведомлением, что повышает устойчивость и надёжность работы скрипта при наличии некорректных или неочищенных данных.

Ключевая часть логики построена на использовании расширенных регулярных выражений. Скрипт объединяет содержимое всего файла в один блок, что позволяет корректно обрабатывать предложения, начинающиеся с новой строки. Затем производится замена строчных букв в начале текста и после знаков окончания предложений (таких как точка, восклицательный и вопросительный знаки) на заглавные. Благодаря этому подходу достигается единообразное оформление текста вне зависимости от его структуры и форматирования.

Дополнительно, реализация автокорректора показала высокую эффективность в обработке многострочных данных, содержащих случайные пробелы, переносы строк и знаки препинания. Скрипт способен обрабатывать тексты различной длины и сложности без необходимости ручного вмешательства, что делает его универсальным инструментом для автоматизации подобных задач.

Лабораторная работа также способствовала углублению знаний в области *shell*-программирования, регулярных выражений и работы с текстовыми потоками в *Unix*/*Linux*-среде. Полученные навыки можно применить при создании более сложных утилит для форматирования, фильтрации и анализа текстовых данных, что особенно актуально для системных администраторов, разработчиков и специалистов по обработке информации.

Таким образом, все цели лабораторной работы были достигнуты. Разработанная программа представляет собой надёжное, удобное и расширяемое решение для автоматического исправления регистра в текстовых документах, демонстрируя на практике эффективность *Unix*-инструментов и возможности автоматизации повседневных задач.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] *Introduction* *to* *Linux* *Shell* *and* *Shell* *Scripting* [Электронный ресурс]. – Режим доступа: *https*://*www*.*geeksforgeeks*.*org*/*introduction*-*linux*-*shell*-*shell*-*scripting*/

[2] *Sed* *Command* *in* *Linux*/*Unix* *with* *examples* [Электронный ресурс]. – Режим доступа: *https*://*www*.*geeksforgeeks*.*org*/*sed*-*command*-*in*-*linux*-*unix*-*with*-*examples*/

[3] *AWK* *command* *in* *Unix*/*Linux* *with* *examples* [Электронный ресурс]. – Режим доступа: *https*://*www*.*geeksforgeeks*.*org*/*awk*-*command*-*unixlinux*-*examples*/

# **ПРИЛОЖЕНИЕ А (обязательное) Исходный код программного продукта**

#!/bin/bash

input\_file="input.txt"

output\_file="output.txt"

if [ ! -f "$input\_file" ]; then

echo "Error: File '$input\_file' not found." >&2

exit 1

fi

sed -E ':a;N;$!ba;

s/^[[:space:]]\*([[:lower:]])/\U\1/;

s/([.!?])([[:space:]\n]+)([[:lower:]])/\1\2\u\3/g' "$input\_file" > "$output\_file"