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 Лавренова А.С.

Проверил: ассистент кафедры информатики Гриценко Н.Ю.

Минск 2025

СОДЕРЖАНИЕ

Введение3

[1 [Ц](#_agswffz6g6ei)ель работы](#_Toc178067889) 4

[2 Теоритеческие сведения 5](#_Toc178067890)

[3 Описание и пример выполнения программы 8](#_Toc178067891)

Заключение [9](#_Toc178067899)

[Список использованных источников 10](#_Toc178067900)

[Приложение А (обязательное) 11](#_Toc178067901)

# ВВЕДЕНИЕ

Обработка текстовой информации — это важная область компьютерных наук, охватывающая методы и инструменты, используемые для анализа, редактирования и автоматизации работы с текстовыми данными. Сейчас же объем текстовой информации стремительно растет, а значит что навыки обработки данных становятся неотъемлемой частью работы специалистов в различных областях, от программирования до лингвистики и анализа данных.

Данная лабораторная работа будет сосредоточена на использовании регулярных выражений — мощного инструмента, позволяющего эффективно осуществлять поиск и модификацию текстов. Регулярные выражения представляют собой последовательности символов, определяющие шаблоны для поиска. Они находят широкое применение в таких областях, как валидация вводимых данных, парсинг текстов и автоматизация рутинных задач.

В теоретической части работы будут проиллюстрированны основы регулярных выражений, их синтаксис и ключевые функции, позволяющие проводить различные операции с текстом. Также будут рассмотренны типичные сценарии использования регулярных выражений, что поможет лучше понять их возможности и ограничения.

Практическая часть лабораторной работы будет заключаться в написании shell-скрипта, который будет обрабатывать входные текстовые данные в соответствии с заданием. В частности, будет создан «автокорректор», который будет заменять строчные буквы на заглавные в начале предложений. Это задача требует внимательного подхода, поскольку необходимо учитывать различные условия, при которых предложение может начинаться, включая случаи, когда оно начинается с новой строки или после знаков препинания, таких как точки, восклицательные и вопросительные знаки.

Особое внимание будет уделено обработке ошибочных или «неочищенных» входных данных, что является важным аспектом любой разработки.

Таким образом, данная лабораторная работа позволит не только освоить теоретические основы обработки текстовой информации и регулярных выражений, но и получить практические навыки, которые могут быть применены в дальнейшем в различных областях программирования и анализа данных.

1 ЦЕЛЬ РАБОТЫ

Цель данной лабораторной работы заключается в глубоком изучении методов обработки текстовой информации с акцентом на применение регулярных выражений и разработку автоматизированных инструментов для работы с текстовыми данными. В рамках работы будут изучены теоретические основы регулярных выражений, поняты основные концепции и синтаксис, будет освоен навык идентифицирования и применения различных шаблонов для поиска и замены текстовых фрагментов.

Регулярные выражения также будут использованны на пракитке в контексте реальных задач обработки текстовой информации. Это позволит развить навыки, необходимые для решения практических задач в программировании и анализе данных.

Важной частью работы станет разработка эффективного и надежного shell-скрипта, который будет обрабатывать текстовые файлы в соответствии с заданием. Будет создан «автокорректор», который будет заменять строчные буквы на заглавные в начале предложений, учитывая различные грамматические условия и особенности форматирования текста. При этом будут освоенны методы обработки входных данных, которые могут содержать ошибки или не соответствовать ожидаемому формату, что повысит устойчивость и надежность нашего скрипта.

Кроме того, лабораторная работа будет направлена на развитие навыков тестирования и отладки разработанных скриптов, что обеспечит их корректную работу в различных условиях. Будут создаданны тестовые случаи и проведён анализ результатов, что является важным аспектом процесса разработки программного обеспечения.

Таким образом, цель лабораторной работы заключается в интеграции теоретических знаний и практических навыков, что позволит не только освоить основы обработки текстовой информации, но и стать более уверенными и компетентными в использовании регулярных выражений в различных контекстах.

2 ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

Linux — это свободная операционная система (точнее, семейство систем) с открытым исходным кодом. Само название Linux относится к ядру системы, которое является ее ключевым компонентом, управляющим центральным процессором (ЦП), памятью и периферийными устройствами компьютера. [1]

Одним из принципов Unix-систем является широкое использование текстовых данных: конфигурационные файлы, входные и выходные данные программ в \*nix часто организованы в виде обычного текста. Регулярные выражения — это специальным образом записанные строки, используемые для поиска символьных шаблонов в тексте. Чем-то они похожи на групповые символы в оболочке, но их возможности куда шире. Многие утилиты для работы с текстом в Linux и языки программирования включают в себя механизм регулярных выражений.

В Linux есть два движка регулярных выражений:

* ядро, поддерживающее стандарт POSIX Basic Regular Expression (BRE);
* ядро, поддерживающее стандарт POSIX Extended Regular Expression (ERE). [2]

Большинство утилит Linux соответствуют как минимум стандарту POSIX BRE, но некоторые утилиты (в том числе) понимают только часть стандарта BRE. Одной из причин такого ограничения является желание сделать такие утилиты как можно быстрее в текстовом редакторе.sed

Для работы в основном используются следующие символы:

* " ext" - слова начинающиеся с text;
* "text/" - слова, заканчивающиеся на text;
* "^" - начало строки;
* "$" - конец строки;
* "a-z" - диапазон от a до z;
* "[^t]" - не буква t;
* "[" - воспринять символ [ буквально;
* "." - любой символ;
* "a|z" - а или z.

Регулярные выражения в основном используются со следующими командами:

* grep - утилита поиска по выражению;
* egrep - расширенный grep;
* fgrep - быстрый grep;
* rgrep - рекурсивный grep;
* sed - потоковый текстовый редактор.

А особенно с [утилитой grep](https://wiki.merionet.ru/servernye-resheniya/39/rukovodstvo-po-komande-grep-v-linux/). Данная утилита используется для сортировки результатов чего либо, передавая ей результаты по конвейеру. Эта утилита осуществляет поиск и передачу на стандартный вывод результат его. Её можно запускать с различными ключами, но можно использовать ее другие варианты, которые представлены выше.

И есть еще потоковый текстовый редактор. Это не полноценный текстовый редактор, он просто получает информацию построчно и обрабатывает. После чего выводит на стандартный вывод. Он не изменяет текстовый вывод или текстовый поток, он просто редактирует перед тем как вывести его для нас на экран.

Функционал регулярных выражений в grep разделяется на два уровня: базовый и расширенный. Для активации расширенного набора требуется добавить параметр -E.

Регулярное выражение следует помещать в одинарные кавычки, чтобы предотвратить интерпретацию и расширение выражения оболочкой (например, Bash) перед запуском процесса grep. [3]

Использование косых кавычек может вызвать выполнение текста между ними как подпроцесса bash. Если содержимое окажется допустимой командой, то вместо регулярного выражения будет текст, который вернет команда, что приведет к некорректному выводу.

Также возможно использование двойных кавычек, что позволит передавать в регулярное выражение переменные среды, которые будут применены при вызове команды grep.

В этой главе были рассмотрены теоретические аспекты работы с регулярными выражениями. Регулярные выражения, как мощный инструмент для работы с текстовыми данными, играют важную роль в автоматизации обработки информации. Мы обсудили, как они позволяют осуществлять поиск и замену текстовых фрагментов, а также какие утилиты, такие как grep, sed и их расширенные версии, поддерживают этот функционал.

Также была освещена важность правильного использования синтаксиса регулярных выражений и необходимость обрамлять их в кавычки, чтобы избежать нежелательной интерпретации оболочки. Понимание основ регулярных выражений и их применение в реальных задачах становится неотъемлемой частью работы специалистов в области программирования и анализа данных.

Таким образом, освоение теоретических основ регулярных выражений и их практических применений позволит значительно упростить и автоматизировать работу с текстовой информацией, что является важным навыком для будущих специалистов в области информационных технологий.

**3 ОПИСАНИЕ И ПРИМЕР ВЫПОЛНЕНИЯ ПРОГРАММЫ**

В данном разделе мы подробно рассмотрим программу «Автокорректор», написанную на языке Bash. Эта программа предназначена для автоматической замены строчных букв на заглавные в начале предложений. Она полезна для обработки текстов, где необходимо привести первую букву каждого предложения к заглавному регистру. Это особенно актуально для создания более читабельных и правильно оформленных текстов.

Программа выполняет следующие действия:

1. Программа проверяет, существует ли входной файл input.txt. Если файл отсутствует, программа выводит соответствующее сообщение об ошибке и завершает выполнение.
2. Если файл существует, программа проверяет, пуст ли он. В случае, если файл пустой, выводится предупреждение, и программа завершает выполнение.
3. Программа использует утилиту sed для обработки текста. Она заменяет первую букву в файле на заглавную и заглавные буквы после знаков препинания (точка, восклицательный и вопросительный знаки). Это позволяет привести текст к стандартному виду.

Для демонстрации работы программы создадим текстовый файл input.txt с произвольным содержимым. Например, заполним текстом продемонстрированном на рисунке 3.1.

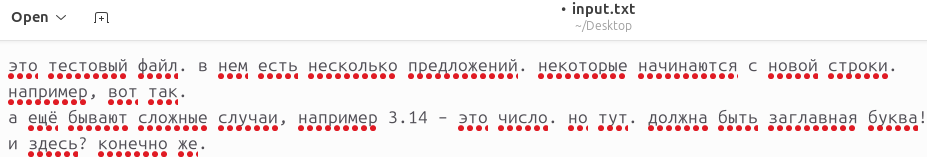


Рисунок 3.1 – Файл input.txt

Далее при успешном запуске программы( файл input.txt существует и не пустой), в терминал будет выведена информация, которая изображена на рисунке 3.2.



Рисунок 3.2 – Успешное выполнение

После выполнения программы содержимое файла input.txt будет изменено (рисунок 3.3).

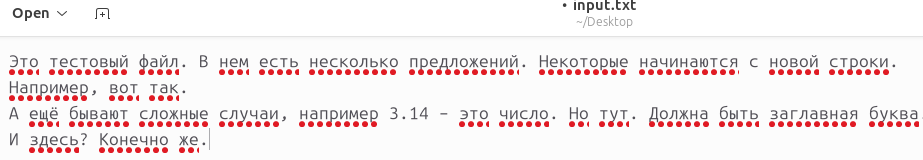


Рисунок 3.3 – Файл input.txt после отработки программы

Программа также реализует базовую обработку ошибок, что позволяет сделать её более устойчивой. Рассмотрим случаи, когда могут возникнуть ошибки, и как они обрабатываются.

Если файл отсутствует, программа выведет сообщение об ошибке, изображённое на рисунке 3.4.



Рисунок 3.4 – Файл не найден

Если файл существует, но он пустой, программа выдаст предупреждение, изображённое на рисунке 3.5.



Рисунок 3.5 – Файл пуст

Программа «Автокорректор» демонстрирует простоту и эффективность использования Bash для обработки текстов. Она позволяет автоматизировать процесс исправления текста, существенно экономя время и усилия пользователя.

Благодаря встроенной обработке ошибок, программа становится более надежной и удобной в использовании. Возможность работы с текстовыми файлами и преобразования их содержимого открывает широкие возможности для автоматизации рутинных задач, связанных с текстом.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы была разработана программа «Автокорректор», предназначенная для автоматической обработки текстов на языке Bash. Основная цель работы заключалась в изучении основ работы с текстовыми файлами, обработки данных и применения регулярных выражений для решения практических задач.

Разработка программы позволила ознакомиться с основными принципами работы с оболочкой Unix/Linux и инструментами, такими как sed, которые используются для текстовой обработки. Был реализован функционал, позволяющий автоматически заменять строчные буквы на заглавные в начале предложений, что является важным аспектом для создания читаемого и грамотно оформленного текста. Эта задача имеет практическое применение в различных областях, включая обработку документов, генерацию отчетов и автоматизацию рутинных задач.

Кроме того, в процессе работы было уделено внимание обработке ошибок, что повышает надежность программы. Были реализованны проверки на существование файла и его пустоту, что позволяет избежать неожиданных сбоев во время выполнения. Такие меры по обеспечению устойчивости программы являются важной частью разработки программного обеспечения и демонстрируют необходимость внимательного отношения к деталям.

Важным аспектом работы стало изучение процесса тестирования и отладки программы. Было проведено тестирование на различных примерах текстов, анализируя полученные результаты и внося необходимые корректировки в код. Это позволило не только убедиться в корректности работы программы, но и улучшить навыки программирования и анализа.

В будущем данная программа может быть расширена новыми функциональными возможностями. Например, можно добавить поддержку дополнительных языков, улучшить алгоритмы обработки текста или интегрировать программу с графическим интерфейсом пользователя. Также интересным направлением для дальнейшего развития является использование машинного обучения для автоматического исправления ошибок и улучшения качества текста.

Таким образом, лабораторная работа не только достигла поставленных целей, но и предоставила возможность углубить знания в области программирования, обработки текстов и разработки программного обеспечения. Даже простые инструменты, такие как Bash, могут быть мощными средствами для решения задач автоматизации и обработки данных. Опыт, полученный в ходе этой работы, будет полезен в будущих проектах и научных исследованиях, связанных с программированием и обработкой текстовой информации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Linux и UNIX: программирование в shell. Руководство разработчика. Дэвид Тейнсли – [Электронный ресурс]. – Режим доступа: https://coollib. com/b/275969/read?ysclid=m6ujdy2lag999562018.

[2] Командная строка в Linux. Уильям Шоттс – [Электронный ресурс]. – Режим доступа: https://itsecforu.ru/wp-content/uploads/2018/01/Komandnaya \_stroka\_Linux.pdf/.

[3] Регулярные выражения Bash: полный гайд – [Электронный ресурс]. – Режим доступа: https://timeweb.cloud/tutorials/linux/regulyarnye-vyrazheniya-bash-gajd?ysclid=m82zxlerds129168253.

ПРИЛОЖЕНИЕ А

(обязательное)

Исходный код

#!/bin/bash

input\_file="input.txt"

# Проверяем, существует ли файл

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

echo "Ошибка: Файл '$input\_file' не найден." >&2

exit 1

fi

# Проверяем, пуст ли файл

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

echo "Предупреждение: Файл '$input\_file' пуст." >&2

exit 1

fi

# Обрабатываем текст с учетом различных случаев

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

s/^[[:space:]]\*([[:lower:]])/\U\1/; # Заглавная первая буква в файле

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

mv "$input\_file.tmp" "$input\_file"

echo "Обработка завершена. Исправленный текст сохранён в '$input\_file'"