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

ОТЧЕТ

к лабораторной работе №2

на тему

**ОБРАБОТКА ТЕКСТОВОЙ ИНФОРМАЦИИ. РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ**

Студент В. П. Бычко

Преподаватель Н. Ю. Гриценко

Минск 2024

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc159347008)

[2 Краткие теоретические сведения 4](#_Toc159347009)

[3 Результаты выполнения лабораторной работы 7](#_Toc159347010)

[Выводы 8](#_Toc159347011)

[Список использованных источников 9](#_Toc159347012)

[Приложение А (обязательное) Листинг кода 10](#_Toc159347013)

## 1 ЦЕЛЬ РАБОТЫ

Целью выполнения данной лабораторной работы является изучение методов и средств обработки текстовой информации, включая регулярные выражения, и использующих их утилит. Разработать и реализовать программу, осуществляющую замену строчных букв на заглавные в начале предложений, т.е. в начале документа и после точки, не находящейся внутри.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Исторически первый стандартный текстовый редактор в Unix-системах – однострочный командный редактор ed (EDitor). Важная особенность: постоянное отображение редактируемого текста не предполагается, все действия над текстом (в т.ч. отображение) выполняются по командам. Расширенная версия ed – ex (EXtended editor). Тоже однострочный, но существенно расширен набор команд для сложного поиска и модификации в тексте. Это множестве команд поддерживается и в редакторе vi.[1]

В современных системах ed, как правило, присутствует, а вместо «настоящего» ex команда ex запускает редактор vi (vim) в ex-mode. Так или иначе, ed (и ex) сейчас почти не применяются, но они послужили основой для ряда современных утилит (точнее, их семейств): визуального редактора vi, потокового редактора sed и утилиты поиска в тексте или потоке grep.

Редактор ed нельзя считать в полной мере фильтром: его поток ввода – это команды и их данные. Однако sed и grep – типичные фильтры, имеющие собственный командный язык.

Редакторы ed, vi и sed загружают текстовый документ в буфер и выполняют над ним операции согласно командам, поступающим из потока ввода (stdin) или файла. Текст в течение всего сеанса работы остается в буфере, запись в файл только по команде, вывод – тоже (для ed и sed). В зависимости от конкретной утилиты и способа ее выполнения, источником документа может служить как файл, так и входной поток (stdin).

«Потоковый редактор» sed (Stream EDitor) – неинтерактивное средство обработки текстов. Можно рассматривать как сложный универсальный фильтр для разнообразной обработки текстовых данных, имеющий собственный специализированный командный язык с широким использованием регулярных выражений. Основной сценарий использования – применение к входному потоку текстовых данных одной или нескольких команд, но может работать и с входным файлом или множеством файлов (аналогично большинству программ-фильтров). Результаты всегда передаются в поток и при необходимости могут быть перенаправлены.[2]

Утилита (точнее, семейство утилит) grep – программа-фильтр для поиска по образцу строк в файлах и/или потоках. Одна из наиболее часто используемых программ. По умолчанию grep выводит строки, в которых обнаружено соответствие с заданным образцом поиска. Образец рассматривается как регулярное выражение. Таким образом, grep выполняет более узкий по сравнению с sed набор функций и поэтому не нуждается в отдельном командном языке, но одновременно он имеет дополнительные возможности именно поиска (фильтрации).[3]

Регулярное выражение (regular expression или regexp) – последовательность символов и метасимволов (специальных символов), которую может интерпретировать (разобрать) конечный автомат. И наоборот, конечный автомат может быть определен через возможность обработки им регулярных выражений. В более узком смысле, регулярное выражение – способ описать множество текстовых строк через соответствие данному регулярному выражению. Аналогично, регулярное выражение может быть использовано для порождения множества соответствующих ему текстовых строк.

awk – специализированный «язык сканирования и обработки шаблонов» (pattern scanning and processing language) и семейство интерпретаторов этого языка. Другие определения: «язык обработки структурированных текстов», «язык построчного разбора и обработки входного потока» и т.п.

awk предполагает, что входной файл или поток организован как последовательность записей (строк), каждая из которых представляет собой последовательность полей. Разбиение потока на записи и поля – автоматически на основании разделителей записей и полей. Таким образом, входные данные в интерпретации awk могут рассматриваться как таблица с переменным от строки к строке количеством столбцов, а основными единицами структурирования данных будут поток, файл, запись и поле. Поток – все данные, проходящие через awk. Поток может быть безымянным, если он попадает на вход программы путем перенаправления, или состоять из одного или нескольких файлов, если они задаются явным образом.

Команда END открывает блок кода, отрабатывающий после того, как будут обработаны все входные данные. Команде END соответствует команда BEGIN, которая открывает блок кода, отрабатывающий перед началом обработки входных данных.[4]

В awk присутствуют предопределенные переменные. Например, символ или строка, которая является разделителем записей входного потока хранится в переменной RS, аналогично в FS хранится символ разделения полей. Для обращения к данным используются конструкции: «$0» для обращения ко всей текущей записи входного потока без разделения на поля и «$n» для обращения к отдельному полю текущей записи, где n – индекс поля, который начинается с единицы.

Поддерживаемый набор операций и их синтаксис практически совпадает с языком C, включая инкремент, декремент (префиксная и постфиксная форма различаются) и совмещение арифметической операции с присваиванием. Для чисел дополнительно могут быть определены операции «^» или «\*\*» – возведение в степень (зависит от версии awk). Для строк дополнительно определена операция «~» и «!~» – проверка на соответствие регулярному выражению (в дополнение к обычному точному сравнению). Для объединения (катенации) строк достаточно записать их без разделителя, специальная операция не предусмотрена. Битовые операции не предусмотрены.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы была реализована программа для автокоррекции предложений. На рисунке 3.1 представлен результат запуска программы.

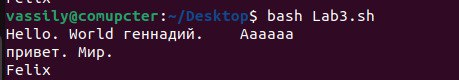


Рисунок 3.1 – Результат запуска программы

На рисунке 3.2 представлен корректируемый текст.

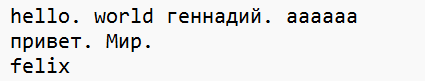


Рисунок 3.2 – Корректируемый текст

# ВЫВОДЫ

В ходе выполнения данной лабораторной работы была разработана и реализована программа, которая выполняет автокоррекцию текста, а именно – производит замену строчных букв на заглавные в начале предложений, т.е. в начале документа и после точки, не находящейся внутри.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] [Электронный ресурс]. – Режим доступа: https://parallel.uran.ru/book/export/html/493 – Дата доступа 06.02.2024

[2] [Электронный ресурс]. – Режим доступа: https://losst.pro/komanda-sed-linux – Дата доступа: 06.02.2024

[3] [Электронный ресурс]. – Режим доступа: https://highload.today/grep-regex/ – Дата доступа: 06.02.2024

[4] [Электронный ресурс]. – Режим доступа: hhttps://www.opennet.ru/docs/RUS/bash\_scripting\_guide/x14802.html – Дата доступа: 07.02.2024

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

**lab2.sh**

#!/&in/bash

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

s/^(.)/\U\1/;

s/\.\n([а-яa-z])/\.\n\U\1/g;

s/([.?!])\s+([а-яa-z])/\1 \U\2/g' Text.txt