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**

Выполнил             В. Д. Ключинский

Проверил                          Н. Ю. Гриценко

Минск 2024

**СОДЕРЖАНИЕ**

[1 Цель работы 2](#_Toc158376840)

[2 Краткие теоретические сведения 4](#_Toc158376841)

[3 Полученные результаты 5](#_Toc158376842)

[Выводы 8](#_Toc158376843)

[Список использованных источников 9](#_Toc158376844)

[Приложение А (обязательное) Листинг исходного кода 10](#_Toc158376845)

**1 ЦЕЛЬ РАБОТЫ**

Целью выполнения лабораторной работы является изучение методов и средств обработки текстовой информации, включая регулярные выражения, и использующих их утилит на примере создание программы-сканера адресов электронной почты из текстового файла.

**2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Регулярные выражения – формальный язык, используемый в компьютерных программах, работающих с текстом, для поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов. Для поиска используется строка-образец, состоящая из символов и метасимволов и задающая правило поиска. Для манипуляций с текстом дополнительно задаётся строка замены, которая также может содержать в себе специальные символы.

Для работы с текстовыми файлами часто используются простейшие утилиты обработки текста, утилита *grep*, интерактивный редактор *vi* и потоковый неинтерактивный редактор *sed*.

Команда *grep* выполняет поиск и выдачу строк, соответствующих заданному регулярному выражению, из заданного файла или поступающих со стандартного ввода. Она также позволяет найти файлы, содержащие заданный текст. Если команда *grep* нашла хотя бы одну подходящую строку, то она завершается с кодом 0 («успех»); если таких строк не найдено, то код завершения равен 1. Если произошла ошибка (файл не найден), то код завершения равен 2. Эти коды можно проверить, например, командой *if* в скрипте, использующем команду *grep* [1].

*AWK* – это скриптовый язык, который полезен при работе в командной строке и широко применяется для обработки текста.  
При использовании *awk* вы можете выбирать данные – один или более отдельных фрагментов текста – на основе заданного критерия. Например, с помощью awk можно выполнять поиск конкретного слова или шаблона во фрагменте текста, а также выбирать определённую строку/столбец в файле.

Команда *sed* – это потоковый редактор текста, работающий по принципу замены. Его можно использовать для поиска, вставки, замены и удаления фрагментов в файле. При этом чаще всего он используется именно для поиска и замены. *SED* позволяет редактировать файлы, не открывая, что существенно ускоряет работу, чем при использовании vi. Помимо этого, *sed* поддерживает регулярные выражения, с помощью которых можно выполнять сложное сопоставление шаблонов [2].

Текстовый редактор *vi* – это один из самых популярных текстовых редакторов в *Linux*. Это редактор на основе командной строки, который предустановлен в большинстве дистрибутивов *Linux*. В любой момент при работе в редакторе *vi* вы находитесь в одном из трёх режимов редактора: командный режим (*command mode*), режим ввода (*insert mode*) и режим последней строки (*last line mode*) [3].

# **3 ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ**

В результате выполнения лабораторной работы был написан скрипт для оболочки *shell,* который выполняет поиск учётных записей электронной почты с заданным расширением из текстового файла с последующей записью в другой файл.

Результаты работы скрипта представлены на рисунке 3.1, рисунке 3.2, рисунке 3.3, рисунке 3.4

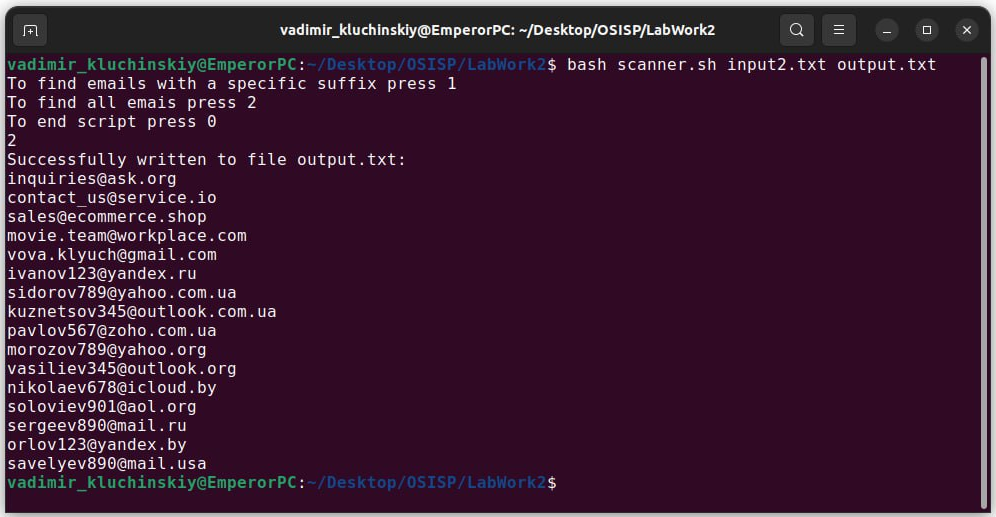


Рисунок 3.1 – Результат работы скрипта при выборе учётных записей с любыми расширениями

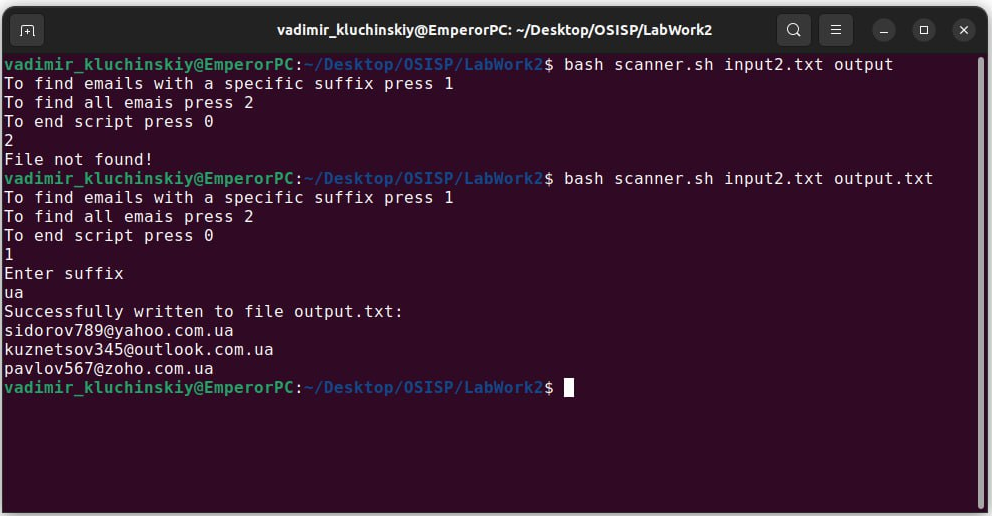


Рисунок 3.2 – Результат работы скрипта при выборе учётных записей с расширением «ua»

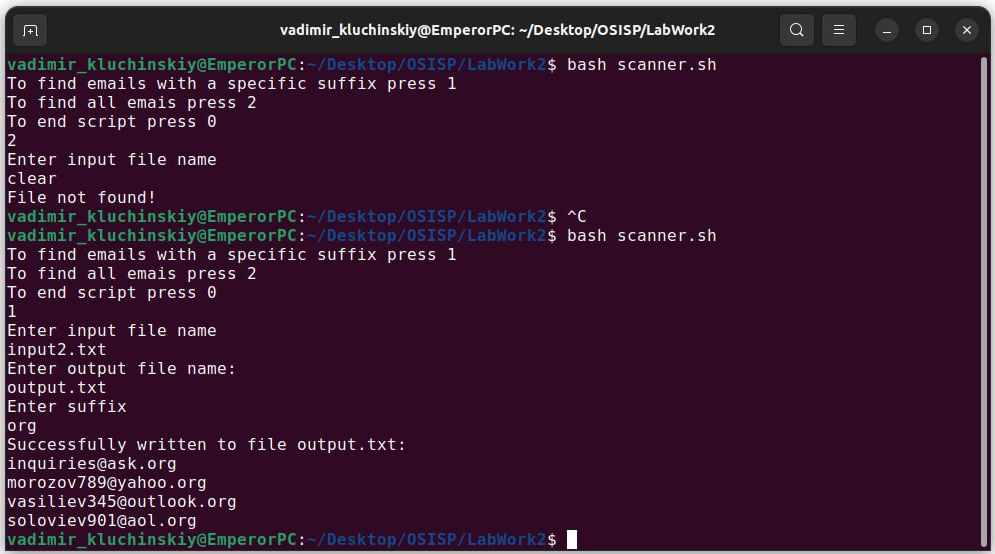


Рисунок 3.3 – Результат работы скрипта при вводе имен файлов не через параметры

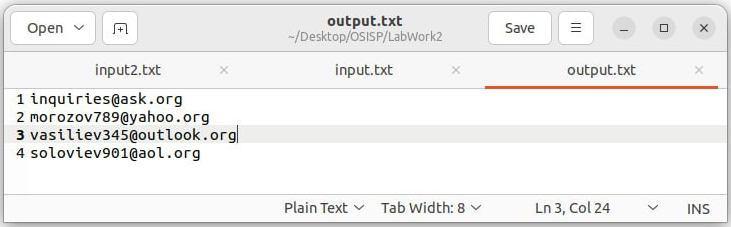


Рисунок 3.4 – Результат работы скрипта в виде записанной информации в выходной файл

Таким образом, созданный скрипт реализует поставленную задачу: находит в текстовом файле все учётные записи с заданным расширением и записывает их в выходной файл.

# **ВЫВОДЫ**

В ходе выполнения данной лабораторный работы была создана программа для оболочки *shell*, которая находит в текстовом файле все учётные записи с заданным расширением и записывает их в выходной файл. Была изучена программа *grep* для поиска подстрок с использованием расширенных регулярных выражений.

Для редактирования кода был выбран редактор текста «*VSCode*».

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Тейнсли, Д. Linux и другие UNIX-подобные операционные системы: программирование в Shеll. / Д. Тейнсли. – СПб.: БВХ-Петербург, 2006. – 1056 с: ил.

[2] Щупаков, Ю. Руководство по командам и shell-программированию, операционная система Linux. / Ю. Щупаков. – СПб: Питер, 2008. – 592 с.

[3] BashTutorials [Электронный ресурс]. – Режим доступа: https://prog /bash-tutorials/. – Дата доступа: 01.02.2024.

# **ПРИЛОЖЕНИЕ А**

# **(обязательное)**

# **Листинг исходного кода**

Листинг 1 – Программный код разработанного приложения:

#!bin/bash/

base\_regex="([a-zA-Z0-9\_\-\.\+]+)@([a-zA-Z0-9\_\-\.]+)\."

standart\_suffix="([a-zA-Z]{2,5})"

echo "To find emails with a specific suffix press 1"

echo "To find all emais press 2"

echo "To end script press 0"

read variant

if [ $variant -eq 0 ]

then

exit 0

fi

if [ -z "$1" ]; then

echo "Enter input file name"

read input\_file

else

input\_file=$1

fi

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

echo "File not found!"

exit 1

fi

if [ -z "$1" ]; then

echo "Enter output file name:"

read output\_file

else

output\_file=$2

fi

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

echo "File not found!"

exit 1

fi

if [ $variant -eq 1 ]

then

echo "Enter suffix"

read suffix

if [[ $suffix =~ [.,] ]]

then

echo "Error! Suffix cant contain {.,}"

exit 1

else

grep -E -o "$base\_regex$suffix" $input\_file > $output\_file

fi

else

grep -E -o "$base\_regex$standart\_suffix" $input\_file > $output\_file

fi

echo "Successfully written to file $output\_file:"

cat $output\_file