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:

студент гр. 153504

Сивый А. А.

Проверил:

Гриценко Н. Ю.

Минск 2024

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146631498)

[2 Краткие теоретические сведения 4](#_Toc146631499)

[3 Полученные результаты 5](#_Toc146631500)

[Выводы 7](#_Toc146631501)

[Список использованных источников 8](#_Toc146631502)

[Приложение А (обязательное) Листинг исходного кода 9](#_Toc146631503)

## 1 ЦЕЛЬ РАБОТЫ

Изучить методы и средства обработки текстовой информации, включая регулярные выражения и использующих их утилит. Реализовать скрипт, который в зависимости от параметров переименовывать файлы в соответствии с различными правилами сортировки. Также скрипт имеет возможность убрать ранее добавленные префиксы у всех файлов.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Регулярное выражение – шаблон, пользуясь которым программы вроде sed или awk фильтруют тексты. В шаблонах используются обычные ASCII-символы, представляющие сами себя, и так называемые метасимволы, которые играют особую роль, например, позволяя ссылаться на некие группы символов [1].

Регулярные выражения используют специальные символы, которые позволяют создавать шаблоны для поиска или замены строк определенного формата:

1 Точка – сопоставляется с любым одиночным символом, кроме символа новой строки.

2 Каретка – этот символ указывает на начало строки.

3 Знак доллара – этот символ указывает на конец строки.

4 Звездочка – сопоставляется с нулевым или более повторениями предыдущего символа или группы символов.

5 Плюс – сопоставляется с одним или более повторениями предыдущего символа или группы символов.

6 Вопросительный знак – указывает, что предыдущий символ или группа символов может присутствовать один раз или отсутствовать.

7 Квадратные скобки – используется для задания класса символов.

8 Круглые скобки – этот символ используется для создания группы символов, которые могут быть использованы для определения подвыражений или для применения квантификаторов с несколькими символами сразу.

В Bash регулярные выражения встречаются в различных утилитах, таких как grep, sed, awk, а также используются в операторах сопоставления строк в условных конструкциях. Команды типа grep используются для поиска строк, соответствующий заданному шаблону в файле или выводе другой команды. Sed позволяет выполнять замены и манипуляции с текстом, включая использование регулярных выражений. Оператор сопоставления строк =~ в условных конструкциях позволяет проверять переменные на соответствие регулярному выражению [2].

## 3 ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

В результате выполнения лабораторной работы была написана программа для пред-сортировки файлов. На рисунке 3.1 представлен список файлов в текущей директории.

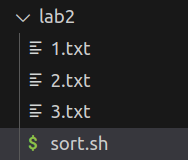


Рисунок 3.1 – Каталог файлов в текущей директории.

Для добавления префиксов нужно запустить скрипт, указав при этом действие (add), после чего необходимо указать критерий для сортировки (дата создания файла, размер или имя) и порядок сортировки (по возрастанию или убыванию). На рисунке 3.2 представлен результат добавления префиксов.

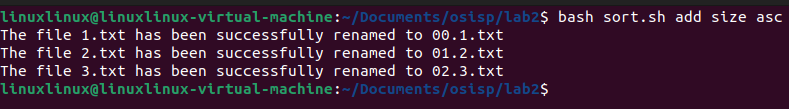


Рисунок 3.2 – Результат добавления префиксов.

В результате работы скрипта ко всем файлам в название добавлен префикс. На рисунке 3.3 представлено содержимое текущей директории.

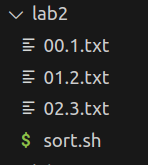


Рисунок 3.3 – Содержимое текущей директории после добавления префикса.

На рисунке 3.3 видно, что в начало к имени каждого файла добавилось порядковое число.

Для удаления префиксов у всех файлов при запуске скрипта нужно передать аргумент remove. На рисунке 3.4 представлен результат удаления.

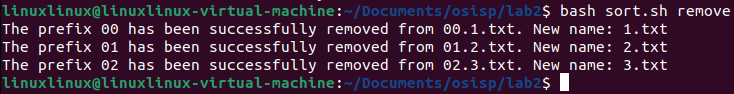


Рисунок 3.4 – Результат удаления префиксов.

Кроме основного функционала данной программы, в скрипте уделено внимание исключительным ситуациям:

1 Осуществляется проверка переданных аргументов, гарантирующая корректное выполнение операций;

2 Префикс имеет динамическую длину. Данное решение позволяет избежать ошибок в случае, если количество файлов, которым необходимо добавить индекс, сильно велико. Для достижения этого в функции add\_prefix\_to\_files расчет длины префикса зависит от количества файлов, что обеспечивает гибкость и универсальность скрипта в различных сценариях использования.

3 В скрипте реализован механизм игнорирования исполняемого файла, чтобы избежать случайного воздействия на сам сценарий в процессе его выполнения. Это достигается путем проверки имени файла на равенство имени исполняемого скрипта, что обеспечивает безопасность и надежность работы скрипта.

4 Проверка существования файла перед его обработкой позволяет избежать потенциальных ошибок в случае, если файл не существует в указанном месте.

5 Реализована проверка наличия префикса в имени файла перед его удалением.

## ВЫВОДЫ

В ходе выполнения лабораторной работы были изучены методы и средства обработки текстовой информации, а также применены регулярные выражения для написания скрипта, который выполняет переименование файлов в соответствии с заданным правилами сортировки и убирает ранее добавленные префиксы. Полученные знания о регулярных выражениях и их применение в реальном коде позволят эффективно работать с текстовой информацией и автоматизировать процессы обработки файлов.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Брайан Керниган, Роб Пайк. UNIX — универсальная среда программирования. – М.: Изд. Дом Вильямс, 1992. – 230 с.

[2] Shell [Электронный ресурс]. – Режим доступа: https://www.shellscript.sh/ – Дата доступа 02.02.2024

[3] Среда Shell [Электронный ресурс]. – Режим доступа: https://tyapk.ru/blog/post/how-to-run-shell-files – Дата доступа 02.02.2024

**ПРИЛОЖЕНИЕ А**

## (обязательное)

## Листинг исходного кода

Листинг 1 – Программный код разработанного приложения

#!/bin/bash

add\_prefix\_to\_file() {

local filename=$1

local prefix=$2

if [ -e "$filename" ]; then

new\_filename="${prefix}.${filename}"

mv "$filename" "$new\_filename"

echo "The file $filename has been successfully renamed to $new\_filename"

else

echo "File $filename is not found"

fi

}

remove\_prefix\_from\_file() {

local filename=$1

prefix="${filename%%.\*}"

if [[ "$prefix" =~ ^[0-9]+$ ]]; then

if [ -e "$filename" ]; then

if [[ "$filename" == \*.\*.\* ]]; then

new\_filename="${filename#\*.}"

mv "$filename" "$new\_filename"

echo "The prefix $prefix has been successfully removed from $filename. New name: $new\_filename"

else

echo "Error: The file $filename does not contain a prefix"

fi

else

echo "Find $filename не is not found"

fi

fi

}

get\_sorted\_files() {

local sort\_option=$1

local sort\_order=$2

local files=()

case $sort\_option in

"date")

files=($(ls -lt --time=atime | awk '{print $9}'))

;;

"size")

files=($(ls -lS | awk '{print $9}'))

;;

"name")

files=($(ls))

;;

\*)

echo "Incorrect sort option. Use 'date', 'size' or 'name'"

exit 1

;;

esac

if [ "$sort\_order" = "desc" ]; then

reversed\_files=()

for ((i=${#files[@]}-1; i>=0; i--)); do

reversed\_files+=("${files[$i]}")

done

files=("${reversed\_files[@]}")

fi

echo "${files[@]}"

}

add\_prefix\_to\_files() {

local files=("${@}")

local file\_count=${#files[@]}

local prefix\_length=$(( ${#file\_count} + 1 ))

local current\_prefix=0

local formatted\_prefix=""

for file in "${files[@]}"; do

if [ "$file" != "${BASH\_SOURCE[0]}" ]; then

formatted\_prefix=$(printf "%0${prefix\_length}d" "$current\_prefix")

add\_prefix\_to\_file "$file" "$formatted\_prefix"

((current\_prefix++))

fi

done

}

remove\_prefix\_from\_files() {

local files=("${@}")

for file in "${files[@]}"; do

remove\_prefix\_from\_file "$file"

done

}

if [ "$#" -eq 0 ]; then

echo "Error: Use '$0 remove' or '$0 add <sort\_option> <sort\_order>'"

exit 1

fi

case "$1" in

"remove")

if [ "$#" -ne 1 ]; then

echo "Error: Use '$0 remove'"

exit 1

fi

remove\_prefix\_from\_files $(ls)

;;

"add")

if [ "$#" -ne 3 ]; then

echo "Error: Use 'add <sort\_option> <sort\_order>'"

exit 1

fi

sort\_option="$2"

sort\_order="$3"

files=($(get\_sorted\_files "$sort\_option" "$sort\_order"))

add\_prefix\_to\_files "${files[@]}"

;;

\*)

echo "Error: Use '$0 remove' or '$0 add <sort\_option> <sort\_order>'"

exit 1

;;

esac