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 К. А. Тимофеев

Проверил Н. Ю. Гриценко

Минск 2024

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения данной лабораторной работы является изучение методов и средств обработки текстовой информации, включая регулярные выражения, и использующих их утилит. Кроме того, необходимо разработать скрипт для предварительной сортировки файлов по таким параметрам, как размер файла, дата последнего изменения, длине имени и алфавитный порядок. Должна быть возможность сортировки одновременно по нескольким параметрам. Должна быть возможность сортировки по возрастанию и по убыванию.

2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Обработка текстовой информации в среде оболочки часто выполняется с использованием различных утили и команд, таких как «*grep*», «*getopts*», «*read*».

Утилита «*grep*», используется для поиска строк в текстовых файлах, которые соответствуют определенному шаблону или регулярному выражению. Данная утилита эффективна для быстрого поиска строк по паттернам, но не имеет возможности выполнения сложных манипуляция с данными.

Регулярные выражения – мощный инструмент для работы с текстовой информацией. Они используются для поиска, сопоставления и манипулирования строками текста с использованием определенных шаблонов. Регулярные выражения широко используются в программировании, обработке текстов, поиске информации в текстовых файлах и многочисленных других областях.

Утилита «*getopts*» позволяет разбить аргументы на набор параметров с значениями или без них, предоставляя классический интерфейс использования команды через консоль.[1]

Утилита «*read*» считывает данные из потока ввода. При помощи оператора «<» в качестве потока можно использовать файл для построчного считывания.[2]

Для выполнения данной лабораторной работы были использованы следующие сведения и концепции:

1 Циклы в скриптах *bash*: циклы позволяют выполнять множество однотипных операций пока выполняется некоторое условие.

2 Функции в скриптах *bash*: функции в скриптах *bash* позволяют удобно вызывать повторяющиеся блоки кода.

3 Массивы в скриптах *bash*: массивы в скриптах *bash* позволяют хранить упорядоченные наборы данных в одной переменной и обращаться к данным по числовому индексу.

4 Регулярные выражения: регулярные выражения позволили эффективно обрабатывать строковые данные. Благодаря им из строк извлекались определенные подстроки, строки проверялись на соответствие условиям и т. д.

5 Команда «*mv*»: команда позволила переименовывать файлы.

3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы был разработан небольшой *bash*-скрипт, который позволяет предварительно сортировать файлы по нескольким параметрам в любом порядке путем добавления перед именем файла числового индекса. Вид каталога до работы скрипта представлен на рисунке 3.1.

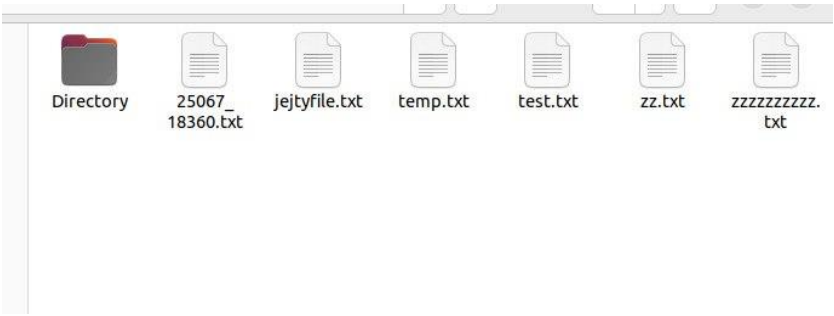

Рисунок 3.1 – Каталог до работы *bash*-скрипта

Скрипт имеет возможность сортировки в случайном порядке. Результат данной сортировки представлен на рисунке 3.2.

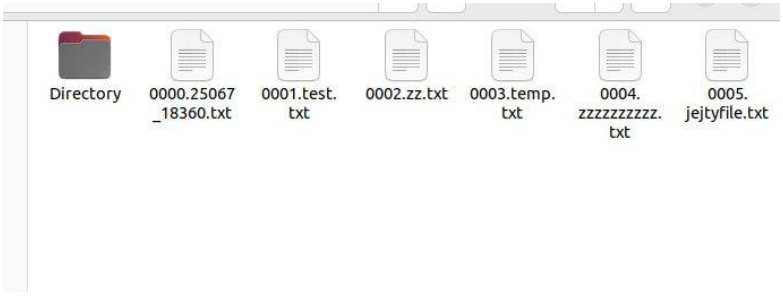

Рисунок 3.2 – Результат работы скрипта со случайной сортировкой

При сортировке скрипт игнорирует каталоги. Также при помощи флага «*-h*» скрипт может вывести справочную информацию о себе. Результат вызова с флагом «*-h*» показан на рисунке 3.3.

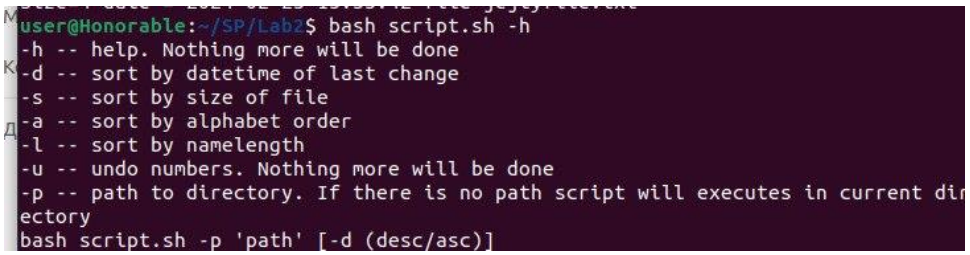

Рисунок 3.3 – Вывод справочной информации

В скрипте предусмотрена обработка неправильных входных данных. Пример работы скрипта в таком режиме представлен на рисунке 3.4.

Рисунок 3.4 – Вывод информации об ошибке в аргументе

При указании нескольких параметров для сортировки скрипт будет группировать результаты. При повторном запуске в каталоге предыдущие префиксы будут удалены.

ВЫВОДЫ

В ходе лабораторной работы были изучены методы и средства обработки текстовой информации, в скриптах *bash*, включая регулярные выражения, и использующих их утилит. Кроме того, в ходе лабораторной работы был разработан скрипт, использующий регулярные выражения для предварительной сортировки файлов по нескольким параметрам.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] How to Use Bash Getopts With Examples [Электронный ресурс]. – Режим доступа: https://kodekloud.com/blog/bash-getopts/.

[2] read — записываем ввод пользователя в переменную [Электронный ресурс]. – Режим доступа: https://linux.cttit.ru/read/.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг исходного кода

Листинг 1 – Программный код разработанного приложения

```
#!/bin/bash

function getsize(){
	# local tmpfile="$(($RANDOM+0))_$(($RANDOM+0)).txt"
	local res=$(du "$1")

	if [[ "$res" =~ ([0-9]+) ]];
	then
	return ${BASH_REMATCH[1]}
	else
		echo "unable to parse string $res"
		exit 1
	fi

}
function sizeComparator(){
	local testA=0
	local testB=0

	getsize $1
	local size1=$?
	getsize $2
	local size2=$?
	echo "size1 = $size1 ; size2 = $size2"

	if ((size1 == size2))
	then
		# echo "equal length"
		return 1
	fi

	(( size1 < size2 )) || testA=1
	[ "$3" = "desc" ] || testB=1

	if [ "$testA" -ne "$testB" ]
	then
		return 1
	else
		return 0
	fi
}
function getdate(){
	local res=$(stat "$1")
	if [[ "$res" =~ ([0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}) ]];
	then
		retval=${BASH_REMATCH[1]}
	else
		echo "can't parse date of $1"
	fi
}
function dateComparator(){
	local testA=0
	local testB=0

	getdate $1
	local date1=$retval
	getdate $2
	local date2=$retval

	if [ "$date1" = "$date2" ];
	then
		return 1
	fi

	[[ "$date1" < "$date2" ]] || testA=1
	[ "$3" = "desc" ] || testB=1
	if [ "$testA" -ne "$testB" ]
	then
```

```
                return 1
        else
                return 0
        fi
}
function alphabetComparator(){
        local testA=0
        local testB=0
        [[ "$1" < "$2" ]] || testA=1
        [ "$3" = "desc" ] || testB=1

        if  [ "$testA" -ne "$testB" ]
        then
                return 1
        else
                return 0
        fi
}
function namelengthComparator(){
        local testA=0
        local testB=0
        echo "length of $1 = ${#1} length of $2 = ${#2}"
        if ((${#1} == ${#2}))
        then
                # echo "equal length"
                return 1
        fi

        (( ${#1} < ${#2} )) || testA=1
        [ "$3" = "desc" ] || testB=1

        if [ "$testA" -ne "$testB" ]
        then
                return 1
        else
                return 0
        fi
}
function random(){
        local arraysize=${#filesarr[@]}
        local randcount
        local i
        ((randcount = arraysize * 2 ))
        echo "count = $randcount"
        for ((i=0; i < randcount; ++i))
        do
                local i1
                ((i1=$RANDOM % arraysize))
                local i2
                ((i2=$RANDOM % arraysize))
                # echo "i1= $i1 i2=$i2"
                local tmp=${filesarr[i1]}
                filesarr[$i1]=${filesarr[i2]}
                filesarr[$i2]=$tmp
        done
}
function mysort(){
        local comparator=$1
        local i
        local j
        for ((i=0; i < ${#filesarr[@]}; ++i))
        do
                for ((j=0; j < ${#filesarr[@]} - i - 1; ++j ))
                do
                        # echo "i = $i j= $j"
                        if  $comparator ${filesarr[j]} ${filesarr[j+1]} $2
                        then
                                # echo "swap"
                                local tmp=${filesarr[j]}
                                filesarr[$j]=${filesarr[j+1]}
                                filesarr[((j+1))]=$tmp
                        fi
                done
        done
}
function undoNumbers(){
        for ((i = 0; i < ${#filesarr[@]}; ++i))
        do
```

```
                    local filename=${filesarr[i]}
                    if [[ $filename =~ ^[0-9]{4}\.(.*) ]];
                    then
                              # echo "filename without number ${BASH_REMATCH[1]}"
                              mv "$filename" "${BASH_REMATCH[1]}"
                    fi
          done
}
function getFiles(){
          local tmpfile="$(($RANDOM+0))_$(($RANDOM+0)).txt"
          $(ls -p >> $tmpfile)
          local tmpfile2="$(($RANDOM+0))_$(($RANDOM+0)).txt"
          grep ".*[^/]$" $tmpfile >> $tmpfile2
          rm $tmpfile
          local i=0
          while read line;do
                    if [ $line = $tmpfile ] || [ $line = $tmpfile2 ] || [ $line = $scriptname ]
                    then
                              continue
                    fi
                    filesarr[$i]=$line
                    echo "DEBUG i = $i line = $line filesarr[i] = ${filesarr[i]}"
                    ((i=i+1))
                    # echo ${filesarr[*]}
          done < "$tmpfile2"
          rm "$tmpfile2"
}
function showHelp(){
          echo "-h -- help. Nothing more will be done"
          echo "-d -- sort by datetime of last change"
          echo "-s -- sort by size of file"
          echo "-a -- sort by alphabet order"
          echo "-l -- sort by namelength"
          echo "-u -- undo numbers. Nothing more will be done"
          echo "-p -- path to directory. If there is no path script will executes in current
directory"
          echo "bash script.sh -p 'path' [-d (desc/asc)]"
}
scriptname="$0"
declare -a filesarr
declare -a sortstack
i=0
while getopts ':hrua:d:l:s:p:' opt; do
          case "$opt" in
          a)
                    if [ $OPTARG = "desc" ] || [ $OPTARG = "asc" ]
                    then
                              sortstack[$i]="mysort alphabetComparator $OPTARG"
                              ((i=i+1))
                    else
                              exit 1
                    fi
          ;;
          d)
                    if [ $OPTARG = "desc" ] || [ $OPTARG = "asc" ]
                    then
                              sortstack[$i]="mysort dateComparator $OPTARG"
                              ((i=i+1))
                    else
                              echo "error in arg of $opt . It can't be $OPTARG"
                              exit 1
                    fi
          ;;
          l)
                    if [ $OPTARG = "desc" ] || [ $OPTARG = "asc" ]
                    then
                              sortstack[$i]="mysort namelengthComparator $OPTARG"
                              ((i=i+1))
                    else
                              echo "error in arg of $opt . It can't be $OPTARG"
                              exit 1
                    fi
          ;;
          s)
                    if [ $OPTARG = "desc" ] || [ $OPTARG = "asc" ]
                    then
                              sortstack[$i]="mysort sizeComparator $OPTARG"
                              ((i=i+1))
```

```
                    else
                              echo "error in arg of $opt . It can't be $OPTARG"
                              exit 1
                    fi
          ;;
          r)
                    # echo "in random"
                    sortstack[$i]="random"
                    ((i=i+1))
          ;;
          h)
                    showHelp
                    exit 0
          ;;
          u)
                    # echo " in undoNumbers case"
                    getFiles
                    undoNumbers
                    exit 0
          ;;
          p)
                    cd "$OPTARG"
                    if (($? != 0));
                    then
                              echo "Error in path $OPTARG"
                              exit 1
                    fi
          ;;
          :)
                    echo -e "Option requires an argument desc/asc"
                    exit 1
          ;;
          ?)
                    echo -e "There is no such option. Use -h to get help"
                    exit 1
          ;;
  esac
  echo "option"
done
getFiles
undoNumbers
for file in ${filesarr[@]}
do
          echo "file $file"
done
for ((i=i-1; i >= 0; --i))
do
          eval "${sortstack[i]}"
done
for file in ${filesarr[@]}
do
          getdate "$file"
          getsize "$file"
          echo "size=$? date = $retval file $file "
done
for ((i=0; i < ${#filesarr[@]}; ++i))
do
          declare number
          file=${filesarr[i]}
          if ((i <= 9));
          then
                    number=000$i
          elif ((i <= 99))
          then
                    number=00$i
          elif ((i <= 999))
          then
                    number=0$i
          else
                    number=$i
          fi
          mv $file "$number.$file"
done
```