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 №1

на тему

**СКРИПТЫ SHELL**

Студент А. В. Скворцов

Преподаватель Н. Ю. Гриценко

Минск 2024

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 5](#_Toc146752070)

[Заключение 7](#_Toc146752071)

[Список использованных источников 8](#_Toc146752072)

[Приложение А (обязательное) Листинг кода 9](#_Toc146752073)

1. **ЦЕЛЬ РАБОТЫ**

Изучение элементов и конструкций скриптов *shell*: переменных, параметров, ветвлений, циклов, вычислений, команд *shell* и вызовов внешних программ (*shell, sed, awk, wget*, различные фильтры и т.д.) для решения достаточно сложной задачи, имеющей практическое значение, а также принципов интеграции Unix-программ скриптами *shell*.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Программирование на языке интерпретатора shell приобретает все большую популярность по мере утверждения Linux в качестве удобной в работе и отказоустойчивой операционной системы. Трудно оценить, какое количество пользователей работают с Linux. Эта операционная система распространяется бесплатно, хотя многие компании разрабатывают ее коммерческие варианты. Кроме того, несмотря на сделанные несколько лет назад неутешительные прогнозы специалистов относительно будущего UNIX, данная ОС также не теряет популярности, и число ее приверженцев продолжает расти [1].

Командный язык *shell* (в переводе – раковина, скорлупа) фактически есть язык программирования очень высокого уровня. На этом языке пользователь осуществляет управление компьютером [2].

В Linux автоматизация процессов в значительной степени решается исполняемыми скриптами в *shell* оболочке. Это подразумевает под собой создание файла, содержащего ряд команд, которые исполняются последовательно.

*Bash-скрипт* — это файл, содержащий последовательность команд, которые выполняются программой *bash* построчно. Он позволяет выполнять ряд действий, таких как переход к определенному каталогу, создание папки и запуск процесса с помощью командной строки.

Термины «оболочка» и «*bash*» взаимозаменяемы. Но между ними есть тонкая разница. Термин «оболочка» относится к программе, которая предоставляет интерфейс командной строки для взаимодействия с операционной системой*. Bash (Bourne Again SHell)* является одной из наиболее часто используемых оболочек Unix/Linux и является оболочкой по умолчанию во многих дистрибутивах Linux [3].

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате работы был разработан *bash-скрипт*, который позволяет проходить по дереву каталогов и находить файл по указанному имени, либо по регулярному выражению. Скрипт также вычисляет контрольную сумму файла, выводит файл, название которого укажет пользователь, в виде листинга и предлагает изменить строку с номером, который укажет пользователь.

Чтобы запустить скрипт в него необходимо передать один из параметров *-n* или *-r* для того, чтобы скрипт понял каким образом искать файл: по названию файла, либо по регулярному выражению соответственно. Также программа требует путь на вход, в котором искать файл. После запуска скрипт проходит по дереву каталогов и выводит найденные файлы в консоль и в файл output.txt, также скрипт вычисляет контрольную сумму каждого файла и также выводит ее пользователю. (рисунок 1).

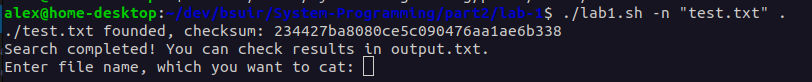


Рисунок 1 – Запуск скрипта

Далее скрипт ожидает пользовательского ввода названия файла, который он хочет отобразить в виде листинга и как только пользователь ввел название файла, консольный вывод очищается, и пользователь видит листинг файла (рисунок 2).

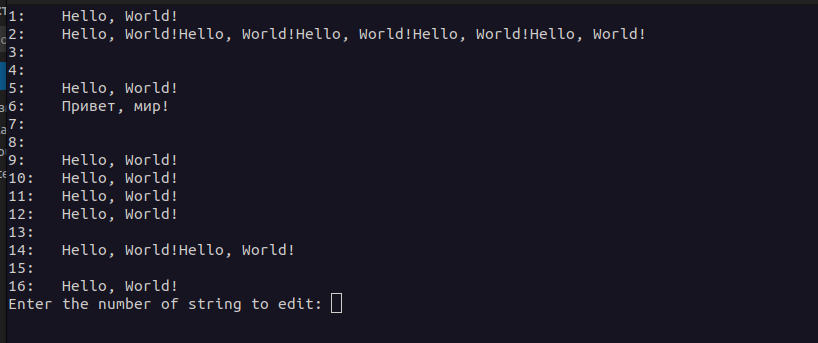


Рисунок 2 – Вывод листинга

После того, как пользователю вывелся файл, пользователь может ввести номер строки, которую он хочет изменить, далее ввести новое содержимое для этой строки и скрипт выполнит замену содержимого в выбранной пользователем строке этого файла (рисунок 3).

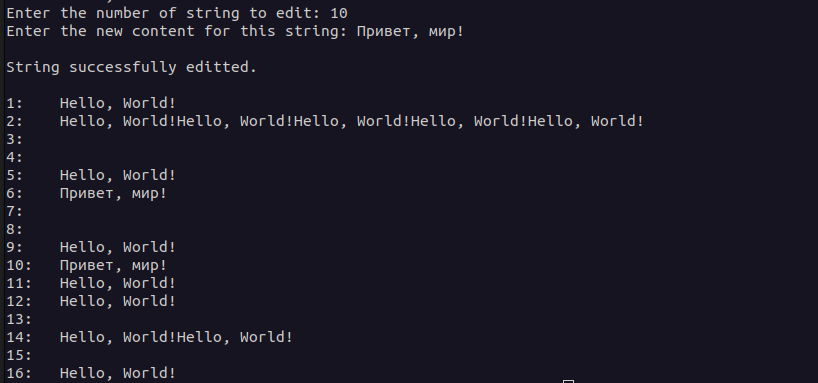


Рисунок 3 – Изменение строки файла

После изменения строки файла, в консоль выводится этот файл заново и, если открыть измененный файл в любом другом приложении, можно увидеть, что изменения применились к файлу и его содержимое изменилось. Также скрипт проверяет все вводимые пользователем данные: не позволяет открыть для изменения файл, который не существует, проверяет валидность номера строки, а также наличие необходимых для запуска скрипта параметров имени файла или регулярного выражения.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы был разработан *bash-скрипт*, позволяющий взаимодействовать с деревом каталогов и файловой системой в целом, в котором используются основные конструкции скриптов shell, такие как переменные, ветвления, циклы. Также были изучены внешние программы, такие как *awk*, позволяющей работать с текстовыми данными и *sed*, позволяющей вносить изменения в файлы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Тейнсли Д. Linux и UNIX: программирование в shell. Руководство разработчика: Пер.. с англ. – СПб: «БХВ-Петерберг», 2001. – 464 с.
2. ПРОГРАММИРОВАНИЕ НА shell (UNIX) [Электронный ресурс]. – Режим доступа: https://www.opennet.ru/docs/shell/shell.html. – Дата доступа: 05.02.2024.
3. Работа с командной строкой Linux & Shell [Электронный ресурс]. – Режим доступа: https://medium.com/@kkonstantinov/bash-скрипты-4c6be424edae. – Дата доступа: 06.02.2024.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – lab1.sh

#!/bin/bash

usage() {

echo "Usage: $0 [-r <regex>|-n <filename>] <dir\_path>"

exit 1

}

if [ $# -lt 2 ]; then

usage

fi

pattern=""

filename=""

path=""

while getopts ":r:n:" opt; do

case $opt in

r)

pattern="$OPTARG"

;;

n)

filename="$OPTARG"

;;

\?)

echo "Invalid option: -$OPTARG" >&2

usage

;;

esac

done

shift $((OPTIND - 1))

path="$1"

if [ -z "$pattern" ] && [ -z "$filename" ]; then

echo "Please specify either a regex pattern (-r) or a filename (-n)."

usage

fi

if [ ! -d "$path" ]; then

echo "No such file or directory."

exit 1

fi

if [ -e "output.txt" ]; then

> output.txt

fi

calculate\_checksum() {

local file="$1"

checksum=$(md5sum "$1" | awk '{print $1}')

echo "$checksum"

}

traverse\_directory() {

local dir="$1"

for file in "$dir"/\*; do

if [ -f "$file" ]; then

if [ -n "$pattern" ]; then

if [[ $(basename "$file") =~ $pattern ]]; then

checksum=$(calculate\_checksum "$file")

echo "$file founded, checksum: $checksum"

echo "$(basename "$file, checksum: $checksum")" >> output.txt

fi

elif [ -n "$filename" ]; then

if [ "$(basename "$file")" = "$filename" ]; then

checksum=$(calculate\_checksum "$file")

echo "$file founded, checksum: $checksum"

echo "$(basename "$file, checksum: $checksum")" >> output.txt

fi

fi

elif [ -d "$file" ]; then

traverse\_directory "$file"

fi

done

}

cat\_file() {

local filename="$1"

lineNumber=1

while IFS= read -r line; do

printf "%-5s %s\n" "$lineNumber:" "$line"

lineNumber=$((lineNumber+1))

done < "$filename"

}

traverse\_directory "$path"

echo "Search completed! You can check results in output.txt."

read -p "Enter file name, which you want to cat: " filename

if [ -f "$filename" ]; then

clear

cat\_file "$filename"

read -p "Enter the number of string to edit: " editLineNumber

read -p "Enter the new content for this string: " newLine

if [ "$editLineNumber" -ge 1 ] 2>/dev/null; then

sed -i "${editLineNumber}s/.\*/${newLine}/" "$filename"

echo -e "\nString successfully editted.\n"

cat\_file "$filename"

else

echo "Incorrect string number."

fi

else

echo "$filename: no such file or directory."

fi

exit 0