

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

ОТЧЁТ  
к лабораторной работе №1  
на тему

**СКРИПТЫ SHELL (УГЛУБЛЕННАЯ ВЕРСИЯ)**

Выполнил: студент гр. 253503  
Тимошевич К.С.

Проверил: ассистент кафедры  
информатики Гриценко Н.Ю.

Минск 2025

## СОДЕРЖАНИЕ

1 Постановка задачи.....	3
2 Описание работы программы.....	4
2.1 Определение режима поиска и обработка аргументов.....	4
2.2 Обход каталогов и поиск файлов.....	4
2.3 Проверка заголовка и вывод содержимого.....	4
3 Ход выполнения программы.....	5
3.1 Примеры выполнения задания.....	5
Вывод.....	6
Список использованных источников.....	7
Приложение А (справочное) Исходный код.....	8

# 1 ПОСТАНОВКА ЗАДАЧИ

Целью данной лабораторной работы является изучение возможностей и конструкций языка программирования *shell* путем разработки и реализации скрипта, выполняющего поиск файлов с обходом дерева каталогов и их дальнейшую обработку. В рамках работы необходимо реализовать два способа поиска файлов.

Первый способ заключается в поиске файлов по имени с использованием регулярного выражения, которое передается в качестве аргумента командной строки.

Второй способ предусматривает поиск по списку имен, представленных в виде ряда аргументов. В обоих случаях поиск должен выполняться рекурсивно, начиная с текущей директории. После нахождения файлов над ними выполняются дополнительные действия: если первая строка файла совпадает с заранее заданным фиксированным заголовком, его содержимое должно быть выведено на экран в виде листинга, где каждая строка нумеруется.

Данный функционал приближен к работе стандартной утилиты *find*, но с расширенными возможностями обработки найденных файлов. В ходе выполнения лабораторной работы требуется продемонстрировать навыки работы с командной оболочкой и допускается использование средств обработки текста, используя встроенные утилиты *Unix*, такие как *find*, *grep*, *sed*, *awk*, *wget* и различные фильтры командной оболочки. Кроме того, в процессе реализации скрипта необходимо учитывать, что размер окна консоли считается фиксированным и не изменяется в ходе его выполнения.

## 2 ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

В данном разделе описаны основные функции скрипта, реализующего поиск файлов с обходом дерева каталогов и их обработку в соответствии с заданными условиями. Описаны методы поиска, проверки содержимого файлов и вывода информации в формате листинга.

### 2.1 Определение режима поиска и обработка аргументов

Скрипт принимает два или более аргументов командной строки [1] Первый аргумент – это заголовок, который должен находиться в первой строке файла, чтобы его содержимое было выведено в формате листинга. Остальные аргументы задают критерии поиска. Если передан один дополнительный аргумент, он интерпретируется как регулярное выражение, используемое для поиска файлов по имени. Если передано два и более дополнительных аргументов, они рассматриваются как список имен файлов, подлежащих поиску. В зависимости от числа аргументов определяется режим работы: поиск по регулярному выражению или поиск по списку имен.

### 2.2 Обход каталогов и поиск файлов

Функция *search\_files* выполняет рекурсивный обход всех подкаталогов, начиная с текущей директории [2]. Для каждого элемента проверяется его существование, после чего определяется его тип. Если элемент является файлом, его имя сравнивается с заданными критериями поиска. В режиме поиска по регулярному выражению используется конструкция `[[ "$filename" =~ $pattern ]]`, проверяющая соответствие имени файла заданному шаблону. [3] В режиме поиска по списку имен выполняется последовательное сравнение имени файла с каждым элементом списка. При совпадении файла с критерием поиска вызывается функция *check\_header* для дальнейшей проверки его содержимого. Если элемент является каталогом, поиск рекурсивно продолжается в его поддиректориях.

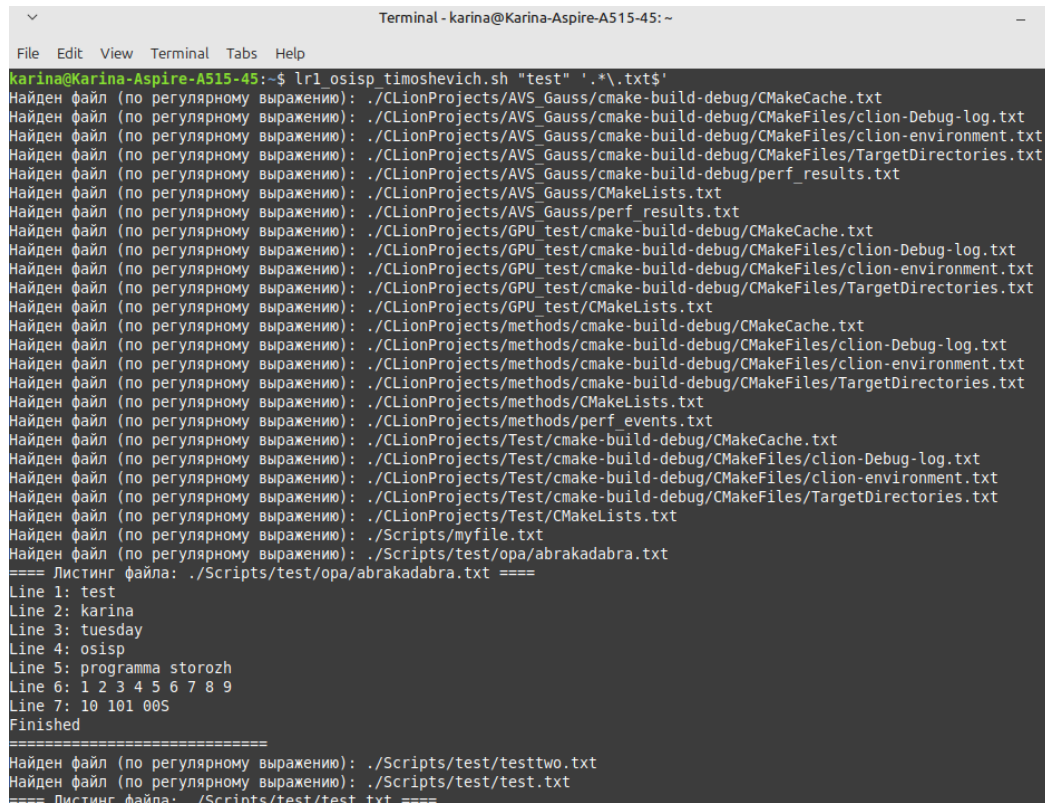
### 2.3 Проверка заголовка и вывод содержимого

Функция *check\_header* проверяет, начинается ли файл с указанного заголовка. Для этого используется команда `head -n 1 "$file"`, извлекающая первую строку. Если заголовок совпадает с переданным аргументом, вызывается функция *print\_with\_numbers*, которая построчно выводит содержимое файла, добавляя к каждой строке её порядковый номер. Для этого применяется цикл `while read line`, который считывает файл построчно, а переменная *countt* увеличивается на каждой итерации, обеспечивая нумерацию строк.

## 3 ХОД ВЫПОЛНЕНИЯ ПРОГРАММЫ

### 3.1 Примеры выполнения задания

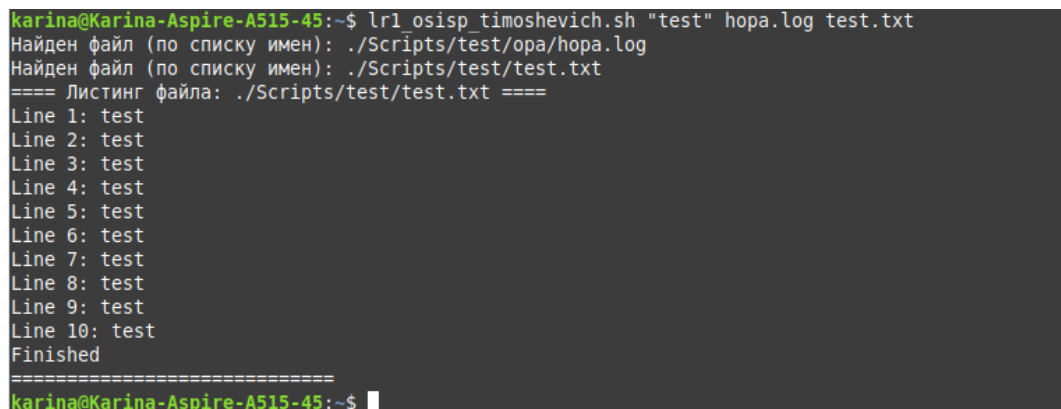
На рисунке 3.1 показан вывод результатов, полученных по итогу выполнения скрипта с передачей ему регулярного выражения.



```
Terminal - karina@Karina-Aspire-A515-45: ~
File Edit View Terminal Tabs Help
karina@Karina-Aspire-A515-45:~$ ./lrl osisp_timoshevich.sh "test" '.*\.txt$'
Найден файл (по регулярному выражению): ./CLionProjects/AVS_Gauss/cmake-build-debug/CMakeCache.txt
Найден файл (по регулярному выражению): ./CLionProjects/AVS_Gauss/cmake-build-debug/CMakeFiles/clion-Debug-log.txt
Найден файл (по регулярному выражению): ./CLionProjects/AVS_Gauss/cmake-build-debug/CMakeFiles/clion-environment.txt
Найден файл (по регулярному выражению): ./CLionProjects/AVS_Gauss/cmake-build-debug/CMakeFiles/TargetDirectories.txt
Найден файл (по регулярному выражению): ./CLionProjects/AVS_Gauss/cmake-build-debug/perf_results.txt
Найден файл (по регулярному выражению): ./CLionProjects/AVS_Gauss/CMakeLists.txt
Найден файл (по регулярному выражению): ./CLionProjects/AVS_Gauss/perf_results.txt
Найден файл (по регулярному выражению): ./CLionProjects/GPU_test/cmake-build-debug/CMakeCache.txt
Найден файл (по регулярному выражению): ./CLionProjects/GPU_test/cmake-build-debug/CMakeFiles/clion-Debug-log.txt
Найден файл (по регулярному выражению): ./CLionProjects/GPU_test/cmake-build-debug/CMakeFiles/clion-environment.txt
Найден файл (по регулярному выражению): ./CLionProjects/GPU_test/cmake-build-debug/CMakeFiles/TargetDirectories.txt
Найден файл (по регулярному выражению): ./CLionProjects/GPU_test/CMakeLists.txt
Найден файл (по регулярному выражению): ./CLionProjects/methods/cmake-build-debug/CMakeCache.txt
Найден файл (по регулярному выражению): ./CLionProjects/methods/cmake-build-debug/CMakeFiles/clion-Debug-log.txt
Найден файл (по регулярному выражению): ./CLionProjects/methods/cmake-build-debug/CMakeFiles/clion-environment.txt
Найден файл (по регулярному выражению): ./CLionProjects/methods/cmake-build-debug/CMakeFiles/TargetDirectories.txt
Найден файл (по регулярному выражению): ./CLionProjects/methods/CMakeLists.txt
Найден файл (по регулярному выражению): ./CLionProjects/methods/perf_events.txt
Найден файл (по регулярному выражению): ./CLionProjects/Test/cmake-build-debug/CMakeCache.txt
Найден файл (по регулярному выражению): ./CLionProjects/Test/cmake-build-debug/CMakeFiles/clion-Debug-log.txt
Найден файл (по регулярному выражению): ./CLionProjects/Test/cmake-build-debug/CMakeFiles/clion-environment.txt
Найден файл (по регулярному выражению): ./CLionProjects/Test/cmake-build-debug/CMakeFiles/TargetDirectories.txt
Найден файл (по регулярному выражению): ./CLionProjects/Test/CMakeLists.txt
Найден файл (по регулярному выражению): ./Scripts/myfile.txt
Найден файл (по регулярному выражению): ./Scripts/test/opa/abracadabra.txt
==== Листинг файла: ./Scripts/test/opa/abracadabra.txt ====
Line 1: test
Line 2: karina
Line 3: tuesday
Line 4: osisp
Line 5: программа storozh
Line 6: 1 2 3 4 5 6 7 8 9
Line 7: 10 101 005
Finished
=====
Найден файл (по регулярному выражению): ./Scripts/test/testtwo.txt
Найден файл (по регулярному выражению): ./Scripts/test/test.txt
==== Листинг файла: ./Scripts/test/test.txt ====
```

Рисунок 3.1 - Работа скрипта при поиске по регулярному выражению

На рисунке 3.2 продемонстрирован результат работы скрипта при использовании списка искомых файлов.



```
karina@Karina-Aspire-A515-45:~$ ./lrl osisp_timoshevich.sh "test" hopa.log test.txt
Найден файл (по списку имен): ./Scripts/test/opa/hopa.log
Найден файл (по списку имен): ./Scripts/test/test.txt
==== Листинг файла: ./Scripts/test/test.txt ====
Line 1: test
Line 2: test
Line 3: test
Line 4: test
Line 5: test
Line 6: test
Line 7: test
Line 8: test
Line 9: test
Line 10: test
Finished
=====
karina@Karina-Aspire-A515-45:~$
```

Рисунок 3.2 - Работа скрипта при передаче списка

## ВЫВОД

В ходе выполнения лабораторной работы был разработан и реализован скрипт на языке *shell*, который выполняет поиск файлов с заданными параметрами и их дальнейшую обработку. Скрипт решает задачу поиска файлов в дереве каталогов, а также обработки их содержимого, выводя информацию в удобном формате. Важной частью работы является использование двух режимов поиска: через регулярное выражение и через список имен файлов, что позволяет значительно расширить возможности скрипта в зависимости от требований пользователя.

Во-первых, был реализован механизм обработки аргументов командной строки, который позволяет задавать как регулярные выражения для поиска, так и перечень имен файлов. Это позволяет пользователю выбирать наиболее удобный и подходящий способ поиска файлов, что повышает универсальность скрипта.

Во-вторых, скрипт использует рекурсивный обход всех подкаталогов, начиная с текущей директории. Это позволяет искать файлы в любой глубине дерева каталогов, обеспечивая полный охват файловой системы. Реализация рекурсивного обхода была выполнена с использованием стандартных конструкций языка *shell*, что обеспечило простоту и наглядность кода. Для каждого найденного файла скрипт проверяет, соответствует ли его имя одному из заданных критериев (регулярному выражению или имени из списка), а затем, если файл найден, проверяется его содержимое.

В-третьих, в процессе выполнения скрипт проверяет первую строку файла на соответствие указанному заголовку. Это гарантирует, что вывод содержимого файла будет производиться только в том случае, если файл соответствует установленным критериям. Для удобства вывода содержимого файл построчно выводится с нумерацией строк. Это делает результат работы скрипта легко воспринимаемым и позволяет оперативно ориентироваться в содержимом каждого файла.

Скрипт проверяет наличие прав на чтение каждого файла перед попыткой вывести его содержимое. В случае отсутствия прав на чтение выводится сообщение об ошибке, что позволяет избежать необработанных исключений и делает скрипт более надежным. Также предусмотрена проверка на правильность передачи аргументов командной строки, что гарантирует правильную работу программы при любых входных данных.

В результате выполнения работы был создан скрипт, который выполняет задачи поиска и обработки файлов в системе, обеспечивая гибкость при выборе критерия поиска и удобный формат вывода результатов.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Bash-скрипты: параметры и ключи командной строки [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/companies/ruvds/articles/326328/>. – Дата доступа: 02.02.2024.
- [2] Bash-скрипты: функции и разработка библиотек [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/companies/ruvds/articles/327248/>. – Дата доступа: 02.02.2024.
- [3] Bash-скрипты, часть 9: регулярные выражения [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/companies/ruvds/articles/327896/>. – Дата доступа: 02.02.2024.

## ПРИЛОЖЕНИЕ А

### (справочное)

### Исходный код

```
#!/bin/bash

if [[ $# -lt 2 ]]; then
    echo "Использование: $0 'заголовок' 'регулярное_выражение' | имя1 имя2 ..."
    exit 1
fi

header="$1"
shift

if [[ $# -eq 1 ]]; then
    pattern="$1"
    search_mode="regex"
else
    file_list=("$@")
    search_mode="list"
fi

print_with_numbers() {
    local file="$1"
    local count=1

    echo "==== Листинг файла: $file ====="
    countt=1
    cat $file | while read line
    do
        echo "Line $countt: $line"
        countt=$(( $countt + 1 ))
    done
    echo "Finished"
    echo "===== "
}

search_files() {
    local dir="$1"

    for item in "$dir"/*; do
        if [[ -e "$item" ]]; then
            if [[ -f "$item" ]]; then
                filename=$(basename "$item")

                if [[ "$search_mode" == "regex" && "$filename" =~ $pattern ]];
then
                    echo "Найден файл (по регулярному выражению): $item"
                    check_header "$item"
                elif [[ "$search_mode" == "list" ]]; then
                    for name in "${file_list[@]}; do
                        if [[ "$filename" == "$name" ]]; then
                            echo "Найден файл (по списку имен): $item"
                            check_header "$item"
                            break
                        fi
                    done
                fi
            fi
        fi

        if [[ -d "$item" ]]; then
```



```

        search_files "$item"
    fi
done
}

check_header() {
    local file="$1"

    if [[ -r "$file" ]]; then
        first_line=$(head -n 1 "$file")
        if [[ "$first_line" == "$header" ]]; then
            print_with_numbers "$file"
        fi
    else
        echo "Ошибка: Нет прав на чтение файла $file"
    fi
}

search_files "."

```