

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

Студент: гр.153502

Миненков Я. А.

Проверил: Гриценко Н. Ю.

Минск 2024

СОДЕРЖАНИЕ

1 Формулировка задачи	3
2 Теоретические сведения	4
3 Описание функций программы.....	5
Список использованных источников	7
Приложение А (обязательное) Исходный код программы	8

1 ФОРМУЛИРОВКА ЗАДАЧИ

Целью выполнения лабораторной работы является изучение элементов и конструкций скриптов *shell*: переменных, параметров, ветвлений, циклов, вычислений, команд *shell* и вызовов внешних программ для решения достаточно сложной задачи, имеющей практическое значение, а также принципов интеграции *Unix*-программ скриптами *shell*.

В качестве задачи требуется написать скрипт для оболочки *shell*, который обеспечит получение заданным образом организованной выходной информации.

2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Bash представляет собой просто макропроцессор, выполняющий команды. Термин «макропроцессор» означает функциональность, при которой текст и символы расширяются для создания более крупных выражений.

Оболочка Unix является одновременно интерпретатором команд и языком программирования. В качестве интерпретатора команд оболочка предоставляет пользовательский интерфейс для богатого набора утилит GNU. Возможности языка программирования позволяют комбинировать эти утилиты. Файлы, содержащие команды, могут быть созданы и сами станут командами. Эти новые команды имеют тот же статус, что и системные команды в таких каталогах, как */bin*, что позволяет пользователям или группам создавать собственные среды для автоматизации своих общих задач [1].

Z shell, *zsh* — одна из современных командных оболочек *UNIX*, использующаяся непосредственно как интерактивная оболочка, либо как скриптовый интерпретатор. *Zsh* является расширенным аналогом, а также имеет обратную совместимость с *bourne shell*, имея большое количество улучшений [2].

В *Bash* нет типов данных. В *Bash* переменная способна хранить числовые значения, отдельные символы или строки символов. Для доступа к существующей переменной используется *\$*. Также возможно обращение к элементу массива, в качестве индекса могут выступать константы и переменные, а также специальное значение «@» — обращение ко всему списку значений.

Практически любая программа требует выполнения различных наборов команд в зависимости от условий. Интерпретатор *Bash* оснащен специальным оператором *if/then*. Он способен принимать выражение и преобразовывать его результат в логическое значение «правда» или «ложь». Если результатом проверки является «правда», оператор *if* осуществляет выполнение содержащихся в нем команд [3].

При написании сценариев командной строки может возникнуть ситуация, когда нужно организовать и перенаправление сообщений об ошибках, и перенаправление стандартного вывода. Для того, чтобы этого добиться, нужно использовать команды перенаправления для соответствующих дескрипторов с указанием файлов, куда должны попадать ошибки и стандартный вывод. Для этого используются операторы *>* и *>>* [4].

3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ

Программа обеспечивает поиск файлов с обходом дерева каталогов следующим образом:

- поиск файлов по имени, с заданным регулярным выражением. Регулярное выражение передается как аргумент командной строки. Поиск начинается с введенной директории;

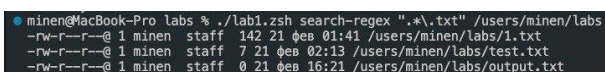
- поиск файлов из заданного списка.

- подсчет байтов каждого файла директории и вывод общей суммы всех файлов.

- подсчет контрольной суммы файлов.

- вывод листинга файла, с заданной первой строкой

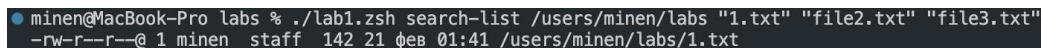
На рисунке 1 произведён поиск файлов по заданному регулярному выражению.



```
minen@MacBook-Pro labs % ./lab1.zsh search-regex ".*\.txt" /users/minen/labs
-rw-r--r--@ 1 minen  staff  142 21 фев 01:41 /users/minen/labs/1.txt
-rw-r--r--@ 1 minen  staff   7 21 фев 02:13 /users/minen/labs/test.txt
-rw-r--r--@ 1 minen  staff   0 21 фев 16:21 /users/minen/labs/output.txt
```

Рисунок 1 – Пример поиска файла по регулярному выражению

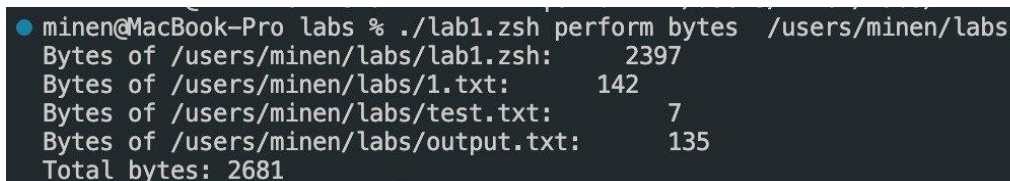
На рисунке 2 произведен поиск файлов по заданному списку файлов из директории `/users/minen/labs`.



```
minen@MacBook-Pro labs % ./lab1.zsh search-list /users/minen/labs "1.txt" "file2.txt" "file3.txt"
-rw-r--r--@ 1 minen  staff  142 21 фев 01:41 /users/minen/labs/1.txt
```

Рисунок 2 – Пример поиска файлов по заданному списку

На рисунке 3 отображен вывод количества байтов каждого файла и общей суммы всех файлов.



```
minen@MacBook-Pro labs % ./lab1.zsh perform bytes /users/minen/labs
Bytes of /users/minen/labs/lab1.zsh:      2397
Bytes of /users/minen/labs/1.txt:          142
Bytes of /users/minen/labs/test.txt:         7
Bytes of /users/minen/labs/output.txt:     135
Total bytes: 2681
```

Рисунок 3 – Пример подсчета байтов

На рисунке 4 произведен вывод листинга файла с заголовком «*qwerty*».

```
minen@MacBook-Pro labs % ./lab1.zsh perform list /users/minen/labs "qwerty"
Listing file: /users/minen/labs/test.txt
1  qwerty
2  sdfsfds
3  sdfsfds
4  dsf
5  s
6  fs
7  fd
```

Рисунок 4 – Пример вывода листинга с заданным заголовком

На рисунке 5 произведен подсчет контрольной суммы заданных файлов.

```
minen@MacBook-Pro labs % ./lab1.zsh perform checksum /users/minen/labs
Checksum of /users/minen/labs/lab1.zsh: 2563659592
Checksum of /users/minen/labs/1.txt: 3947026241
Checksum of /users/minen/labs/test.txt: 3133825526
Checksum of /users/minen/labs/output.txt: 3863500113
Total checksum: 13508011472
```

Рисунок 5 – Пример подсчета контрольной суммы

Результат выполнения команды записывается в файл.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Bash Reference Manual [Электронный ресурс]. – Режим доступа: https://www.gnu.org/software/bash/manual/bash.html#What-is-Bash_003f. – Дата доступа: 21.02.2024.

[2] Языковые конструкции и внутренние переменные Bash [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Zsh>. – Дата доступа: 21.02.2024.

[3] Zsh [Электронный ресурс]. – Режим доступа: <https://www.freecodecamp.org/news/bash-scripting-tutorial-linux-shell-script-and-command-line-for-beginners/>. – Дата доступа: 21.02.2024.

[4] Использование циклов и выражений в сценариях Bash [Электронный ресурс]. – Режим доступа: <https://freehost.com.ua/faq/articles/ispolzovanie-tsiklov-i-virazhenij-v-stsenarijah-bash>. – Дата доступа: 21.02.2024.

ПРИЛОЖЕНИЕ А

(обязательное)

Исходный код программы

```
#!/usr/bin/env zsh

cat /dev/null > output.txt

show_help() {
    echo "Usage: ./lab1.sh <action> <arguments> [directory]"
    echo "Actions:"
    echo "  search-regex <pattern> [directory] - Search files by name
using a regular expression pattern"
    echo "  search-list <directory> <name1> <name2> ... - Search files by
name using a list of names"
    echo "  perform <action> [directory] <search_string> - Perform actions
on found files"
    echo "  Available actions: list, checksum, bytes"
    echo "  help - Show this help message"
}

if [[ "$1" == "help" ]]; then
    show_help
    exit 0
fi

search_files_by_regex() {
    local pattern="$1"
    local directory="${2:-.}"
    find "$directory" -type f -regex "$pattern" -exec ls -l {} \;
}

search_files_by_list() {
    local names=("$@")
    local directory="${names[1]}"
    shift
    find "$directory" -type f \( -name "${names[2]}" $(printf -- "-o -name
%s " "${names[@]:3}")\) -exec ls -l {} \;
}

perform_actions() {
    local action="$1"
    shift
    local directory="${1:-.}"
    local search_string="$2"
    case "$action" in
        "list")
            while IFS= read -r file; do
                local first_line=$(head -n 1 "$file")
                if [[ "$first_line" == "$search_string" ]]; then
                    echo "Listing file: $file" | tee -a output.txt
                    cat -n "$file" | tee -a output.txt
                fi
            done
        ;;
    esac
}
```



```

        fi
    done <<< "$(find "$directory" -type f)"
    ;;
checksum")
    local total_checksum=0
    while IFS= read -r file; do
        local checksum=$(cksum "$file" | awk '{print $1}')
        echo "Checksum of $file: $checksum" | tee -a output.txt
        ((total_checksum += checksum))
    done <<< "$(find "$directory" -type f)"
    echo "Total checksum: $total_checksum" | tee -a output.txt
    ;;
bytes")
    local total_bytes=0
    while IFS= read -r file; do
        local bytes=$(wc -c < "$file")
        echo "Bytes of $file: $bytes" | tee -a output.txt
        ((total_bytes += bytes))
    done <<< "$(find "$directory" -type f)"
    echo "Total bytes: $total_bytes" | tee -a output.txt
    ;;
*)
    echo "Invalid action: $action" | tee -a output.txt
    ;;
esac
}

action="$1"
shift

case "$action" in
    "search-regex")
        search_files_by_regex "$@"
        ;;
    "search-list")
        search_files_by_list "$@"
        ;;
    "perform")
        perform_actions "$@"
        ;;
    *)
        echo "Invalid action: $action" | tee -a output.txt
        ;;
esac

```