

# **Отчет по лабораторной работе №11**

**Программирование в командном процессоре ОС UNIX. Ветвления и  
циклы**

Кочкарев “sakochkarev” Станислав

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>3</b>
<b>2</b>	<b>Задание</b>	<b>4</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>5</b>
<b>4</b>	<b>Выводы</b>	<b>11</b>
<b>5</b>	<b>Контрольные вопросы</b>	<b>12</b>

# 1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 2 Задание

1. Написать командный файл используя `getopts` и `grep`
2. Написать программу на С и использовать командный файл как интерфейс
3. Написать командный файл создающий и удаляющий файлы
4. Написать командный файл пакующий определенные файлы

## 3 Выполнение лабораторной работы

Первым заданием было написание командного файла, который использует команды `getopts` и `grep` для выполнения поиска с указанными параметрами.

Ниже приведен листинг итогового командного файла (рис. 3.1), а также результат его работы (рис. 3.2).

```

1 #!/bin/zsh
2
3 if [[ $# -lt 1 ]]
4 then
5 echo "No options found!"
6 exit 1
7 fi
8
9 args=()
10
11 while getopts "i:o:p:Cn" opt
12 do
13     case $opt in
14         i) filename=$OPTARG;;
15         o) output=$OPTARG;;
16         p) pattern=$OPTARG;;
17         C) args+=(-i) ;;
18         n) args+=(-n) ;;
19         *) echo "No valid options found";;
20     esac
21 done
22
23 if [[ $output ]]
24 then
25     grep "${args[@]}" "$pattern" "$filename" > "$output";
26 else
27     grep "${args[@]}" "$pattern" "$filename";
28 fi

```

Рис. 3.1: Листинг файла

```

sakochkarev@sakochkarev [14:34:01] [~/work/study/2021-2022/Операционные системы/os-intro/labs/lab11] [master *]
-> % zsh ./1.sh -Cn -i ./report/Makefile -p Wildcard
1:FILES = $(patsubst %.md, %.docx, $(wildcard report.md))
2:FILES += $(patsubst %.md, %.pdf, $(wildcard report.md))

```

Рис. 3.2: Выполнение команды

Следующим заданием было написание программы на языке C, которая читает

ввод пользователя в виде числа и сравнивает это число с нулем. В зависимости от результата, программа завершается с разным кодом. В дополнение к данной программе должен был быть написан командный файл, выступающий интерфейсом для программы. Он должен запускать программу и считывать код выхода. В зависимости от кода выхода программа должна выводить сообщение.

Ниже приведен листинг итогового программного файла (рис. 3.3), командного файла (рис. 3.4), а также результат его работы (рис. 3.5).

A screenshot of a code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is written in C and is line-numbered from 1 to 10. It includes headers for stdio.h and stdlib.h, and a main function that reads an integer from the user and exits with a status code based on whether the input is positive, negative, or zero.

```
1 #include "stdio.h"
2 #include "stdlib.h"
3
4 int main() {
5     int input;
6     scanf("%d", &input);
7     if (input > 0) exit(1);
8     else if (input < 0) exit(2);
9     else exit(0);
10 }
```

Рис. 3.3: Листинг программного файла

```

1 #!/bin/zsh
2
3 ./2c
4 output=$?;
5 if [ $output -eq 0 ]; then
6     echo "Число равно нулю";
7 elif [ $output -eq 2 ]; then
8     echo "Число меньше нуля";
9 elif [ $output -eq 1 ]; then
10    echo "Число больше нуля";
11 fi

```

Рис. 3.4: Листинг командного файла

```

sakochkarev@sakochkarev [15:04:33] [~/work/study/2021-2022/Операционные системы/os-intro/labs/lab11] [master *]
-> % zsh ./2.sh
-1
Число меньше нуля
sakochkarev@sakochkarev [15:04:36] [~/work/study/2021-2022/Операционные системы/os-intro/labs/lab11] [master *]
-> % zsh ./2.sh
0
Число равно нулю
sakochkarev@sakochkarev [15:04:38] [~/work/study/2021-2022/Операционные системы/os-intro/labs/lab11] [master *]
-> % zsh ./2.sh
2
Число больше нуля

```

Рис. 3.5: Выполнение команды

Предпоследним заданием было написание программы, которая создает и удаляет эти созданные файлы.

Ниже приведен листинг итогового командного файла (рис. 3.6), а также результат его работы (рис. 3.7).



```
1 #!/bin/zsh
2
3 subcommand=$1
4 case $subcommand in
5 create) if [ $# -ne 2 ]; then echo "No number provided"; fi;
6         for ((i=1;i≤$2;i++)); do touch "$i.tmp"; done;;
7 #delete) ls | grep -P "[0-9]+\tmp$" | xargs -d"\n" rm;;
8 delete) rm -i [0-9]*\tmp;;
9 *) echo "No valid command specified";;
```

Рис. 3.6: Листинг командного файла

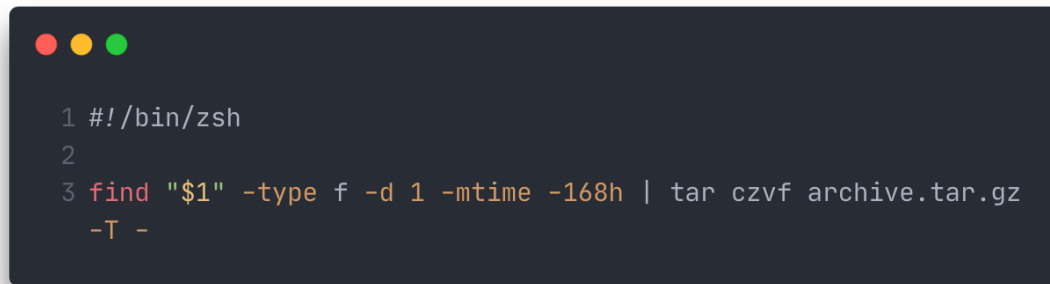
```
sakochkarev@sakochkarev [15:27:53] [~/work/study/2021-2022/Операционные системы/os-intro/labs/lab11] [master *]
-> % ls
total 48
drwxr-xr-x 13 squidass staff 416B May 20 15:27 ./
drwxr-xr-x 21 squidass staff 672B May 18 17:45 ../
-rw-r--r--@ 1 squidass staff 6.0K May 19 21:54 .DS_Store
-rw-r--r-- 1 squidass staff 412B May 20 14:42 1.sh
-rw-r--r-- 1 squidass staff 0B May 20 15:27 1.tmp
-rw-r--r-- 1 squidass staff 242B May 20 15:03 2.sh
-rw-r--r-- 1 squidass staff 0B May 20 15:27 2.tmp
lrwxr-xr-x 1 squidass staff 49B May 20 14:58 2c@ -> /Users/squidass/Downloads/M4/cmake-build-debug/M4
-rw-r--r-- 1 squidass staff 286B May 20 15:27 3.sh
-rw-r--r-- 1 squidass staff 0B May 20 15:27 3.tmp
drwxr-xr-x 4 squidass staff 128B May 18 19:47 presentation/
drwxr-xr-x 8 squidass staff 256B May 20 14:07 report/
-rw-r--r-- 1 squidass staff 112B May 20 14:42 test.txt
sakochkarev@sakochkarev [15:27:54] [~/work/study/2021-2022/Операционные системы/os-intro/labs/lab11] [master *]
-> % zsh ./3.sh delete
remove 1.tmp?
remove 2.tmp?
remove 3.tmp?
```

Рис. 3.7: Выполнение команды

И последним заданием было написание командного файла, который должен паковать все файлы в указанной директории, время изменения которых не позднее недели.

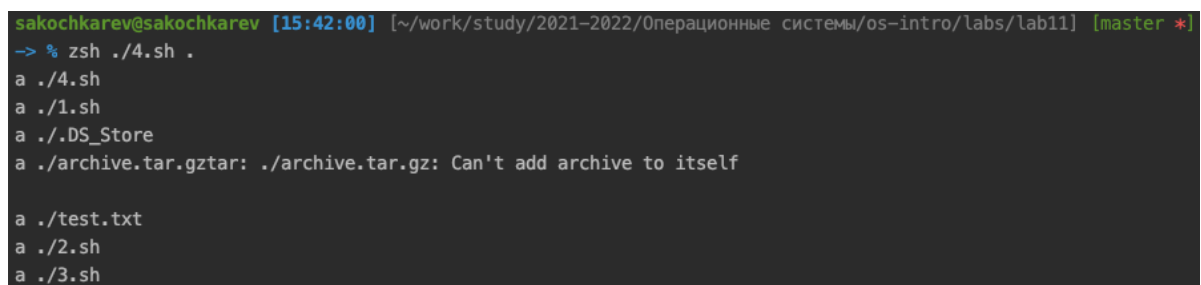
Ниже приведен листинг итогового командного файла (рис. 3.8), а также резуль-

тат его работы (рис. 3.9).

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains a numbered list of commands:

```
1 #!/bin/zsh
2
3 find "$1" -type f -d 1 -mtime -168h | tar czvf archive.tar.gz
-T -
```

Рис. 3.8: Листинг командного файла

A terminal window showing the execution of a script. The prompt is 'sakochkarev@sakochkarev [15:42:00] [~/work/study/2021-2022/Операционные системы/os-intro/labs/lab11] [master \*]'. The user runs 'zsh ./4.sh .' and then enters several commands: './4.sh', './1.sh', './.DS\_Store', './archive.tar.gz', './test.txt', './2.sh', and './3.sh'. The output shows the script running and an error message: 'a ./archive.tar.gztar: ./archive.tar.gz: Can't add archive to itself'.

```
sakochkarev@sakochkarev [15:42:00] [~/work/study/2021-2022/Операционные системы/os-intro/labs/lab11] [master *]
-> % zsh ./4.sh .
a ./4.sh
a ./1.sh
a ./DS_Store
a ./archive.tar.gztar: ./archive.tar.gz: Can't add archive to itself
a ./test.txt
a ./2.sh
a ./3.sh
```

Рис. 3.9: Выполнение команды

## 4 Выводы

По выполнении лабораторной работы мы изучили основы программирования в оболочке ОС UNIX, а также научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 5 Контрольные вопросы

1. Команда `getopts` предназначена для парсинга аргументов командной строки. Она используется для обработки всех входящих аргументов и позволяет устанавливать и выполнять действия на основе переданных аргументов.
2. Метасимволы позволяют задать шаблон имен файлов, например `*.txt` для всех файлов с расширением `.txt` или более сложный `[0-9]*\ .tmp` для всех файлов формата `<число>.tmp`.
3.
  - `for`
  - `if`
  - `case`
  - `while`
4.
  - `break` – для полного прерывания цикла
  - `continue` – для пропуска оставшихся команд внутри итерации
5. Команды `true` и `false` сами по себе просто возвращают значение `true` или `false` соответственно. Значения же нужны для булеановых переменных и проверок. Например установить значение переменной в цикле как `true`, чтобы после завершения цикла можно было понять, что цикл сделал что-то конкретное.
6. Данная строка проверяет, что существует некий файл(ы), находящийся в директории `man<переменная s>/` (например `man123/`) с названием `<переменная i>.<переменная s>` (например `test.123`). Пример полного

пути к файлу для проверки – `mantxt/test.txt`. Если файл существует и является обычным файлом, то выполняется `then`.

7. Можно сказать, что `until` является противоположностью `while`: если `while` выполняется пока его условие действительно, то `until` выполняется пока условие не верно, однако в любом случае выполняется хотя бы один раз.