

Лабораторная работа №11 по предмету  
Операционные системы

НПМбВ-02-19

Нечаева Виктория Алексеевна

# Содержание

Цель работы	5
Задание	6
Выполнение лабораторной работы	7
Задача 1 . . . . .	7
Задача 2 . . . . .	9
Задача 3 . . . . .	11
Задача 4 . . . . .	13
Выводы	16
Контрольные вопросы	17

## Список таблиц

# Список иллюстраций

1	Рисунок 1 . . . . .	7
2	Рисунок 2 . . . . .	9
3	Рисунок 3 . . . . .	10
4	Рисунок 4 . . . . .	11
5	Рисунок 5 . . . . .	12
6	Рисунок 6 . . . . .	13
7	Рисунок 7 . . . . .	14
8	Рисунок 8 . . . . .	15

## Цель работы

Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

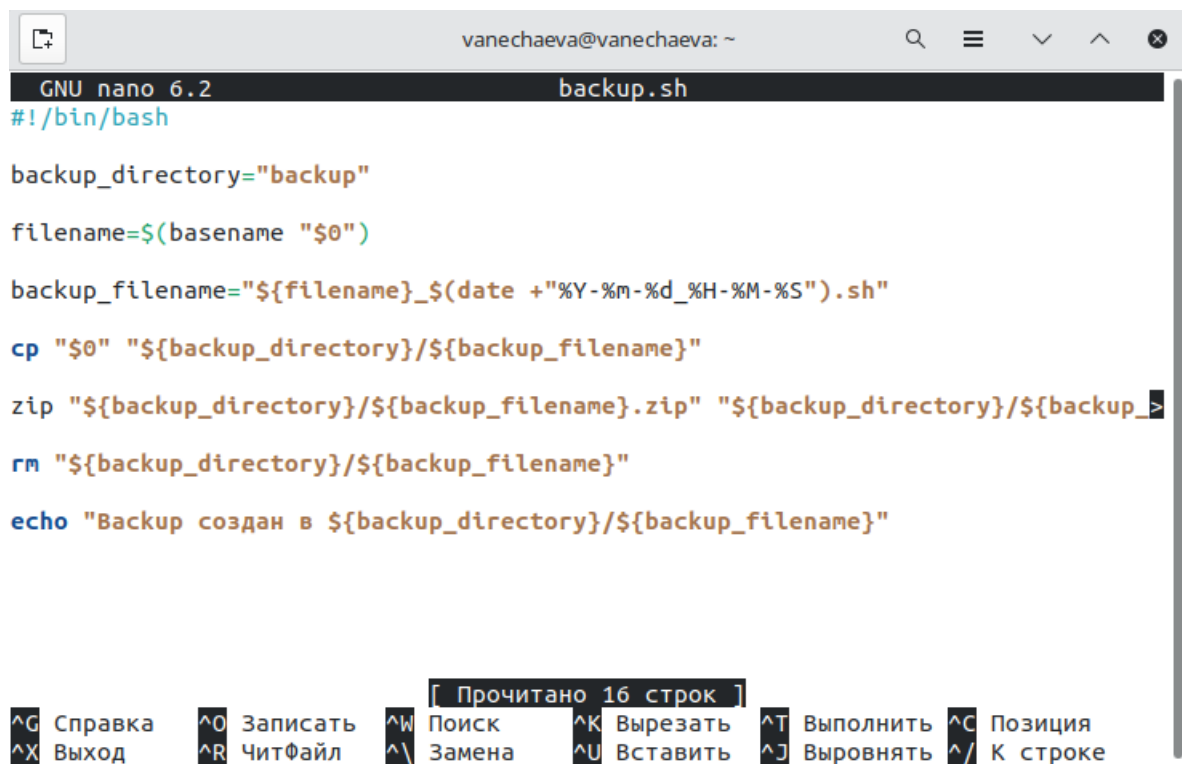
# Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

# Выполнение лабораторной работы

## Задача 1

Написать скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации необходимо узнать, изучив справку (рис. 1-2)



```
GNU nano 6.2 backup.sh
#!/bin/bash

backup_directory="backup"

filename=$(basename "$0")

backup_filename="${filename}_${date +"%Y-%m-%d_%H-%M-%S"}.sh"

cp "$0" "${backup_directory}/${backup_filename}"

zip "${backup_directory}/${backup_filename}.zip" "${backup_directory}/${backup_>

rm "${backup_directory}/${backup_filename}"

echo "Backup создан в ${backup_directory}/${backup_filename}"
```

[ Прочитано 16 строк ]

<b>^G</b> Справка	<b>^O</b> Записать	<b>^W</b> Поиск	<b>^K</b> Вырезать	<b>^T</b> Выполнить	<b>^C</b> Позиция
<b>^X</b> Выход	<b>^R</b> ЧитФайл	<b>^_\</b> Замена	<b>^U</b> Вставить	<b>^J</b> Выводить	<b>^/</b> К строке

Рис. 1: Рисунок 1

В `backup_directory` задается название каталога куда будет отправляться бэкап скрипта.

В `filename` задается переменная с указателем 0, ссылаясь на название текущего скрипта

`backup_filename` задает название нового бэкапа, в новом названии генерируется дата и время создания

`cp` в данном случае копирует текущий скрипт в `backup`

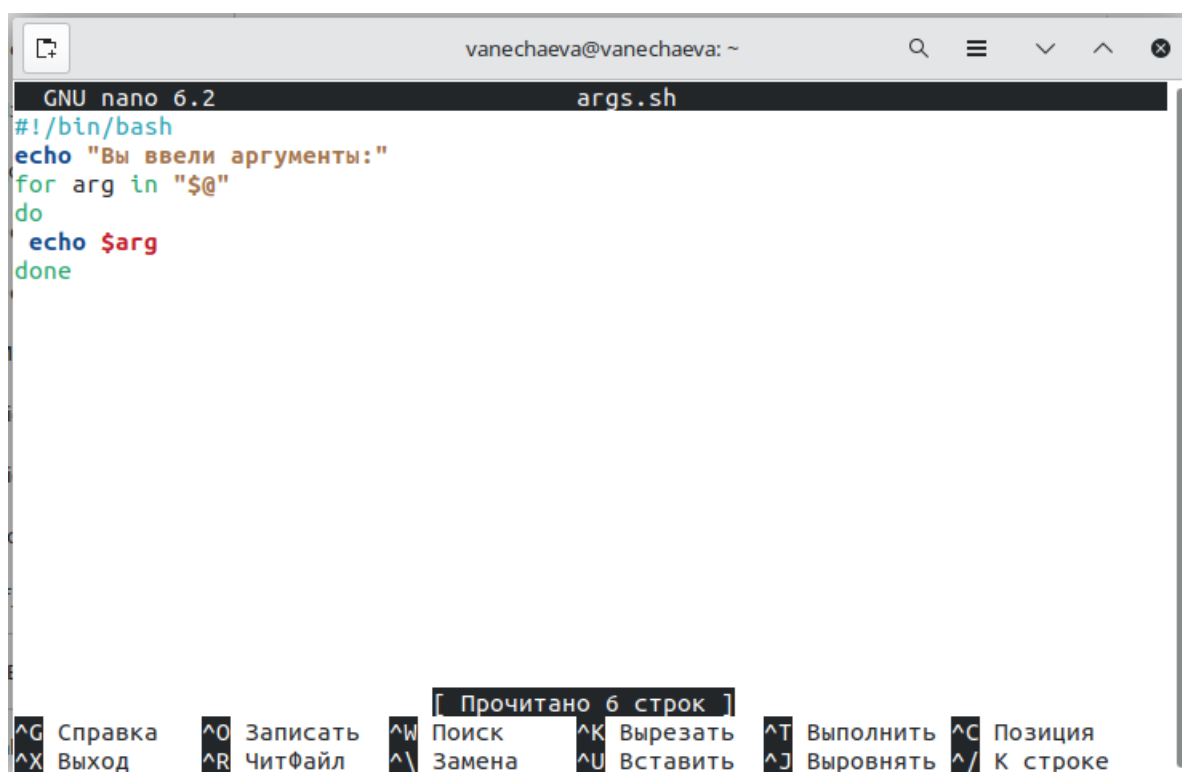
`zip` сжимает созданный в папке `backup` бэкапный файл, так как сжать можно только каталог, файл автоматически кладется в папку `backup`

`rm` удаляет `backup` файл, так как по заданию нужен именно архив

`echo` выводит в консоль инфо о созданном бэкапе







```
GNU nano 6.2 args.sh
#!/bin/bash
echo "Вы ввели аргументы:"
for arg in "$@"
do
    echo $arg
done
```

Прочитано 6 строк ]

<b>^G</b> Справка	<b>^O</b> Записать	<b>^W</b> Поиск	<b>^K</b> Вырезать	<b>^T</b> Выполнить	<b>^C</b> Позиция
<b>^X</b> Выход	<b>^R</b> ЧитФайл	<b>^_</b> Замена	<b>^U</b> Вставить	<b>^J</b> Выводить	<b>^/</b> К строке

Рис. 3: Рисунок 3

В коде скрипта только цикл по всем аргументам, указанным при вызове скрипта в командной строке. Выводит более 10 аргументов.

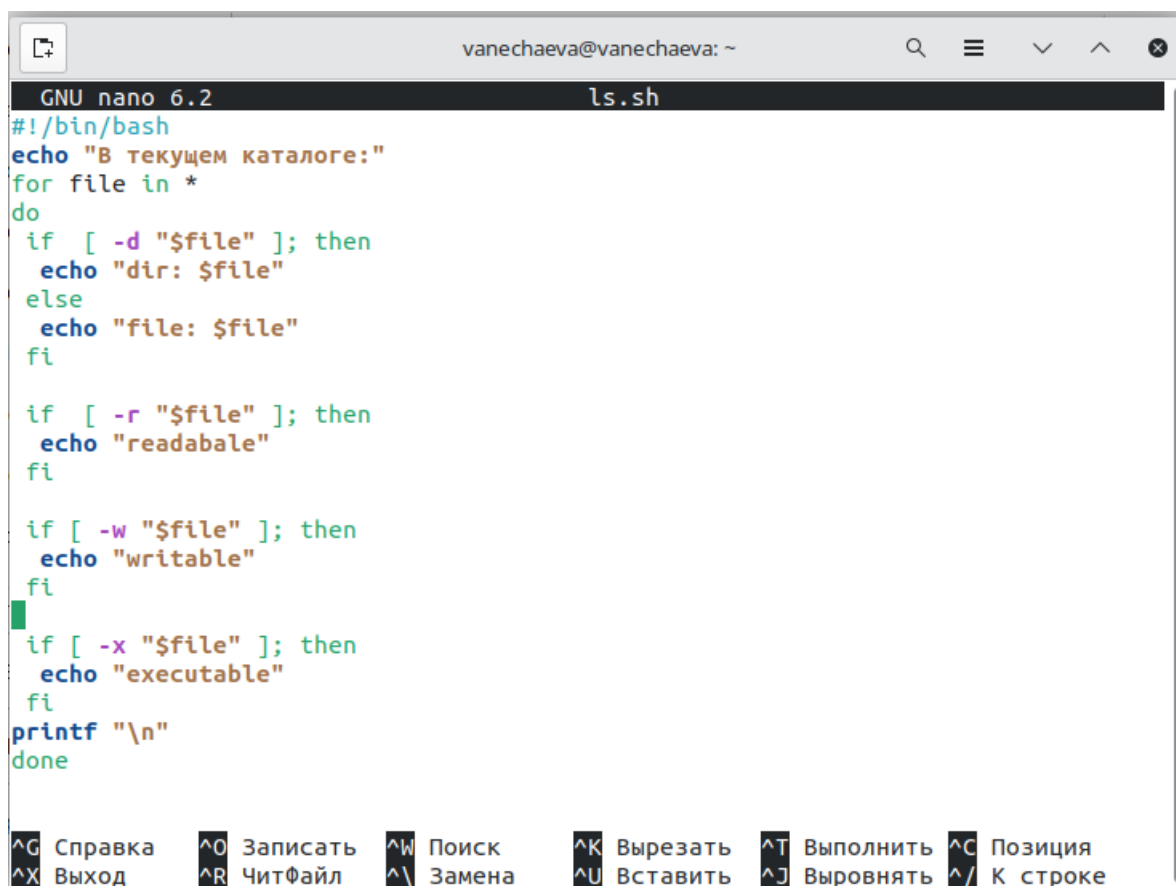
A terminal window titled 'vanechaeva@vanechaeva: ~' with standard window controls. The terminal shows a script being executed with various arguments. On the left margin, line numbers 2 through 13 are visible. The script output lists the arguments: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, -1, f, -d, 3737, sdsd, and @. The prompt returns to the shell.

```
vanechaeva@vanechaeva: ~$ ./args.sh 1 2 3 4 5 6 7 8 9 10 11 12 13 -1 f -d 3737 sdsd @
Вы ввели аргументы:
1
2
3
4
5
6
7
8
9
10
11
12
13
-1
f
-d
3737
sdsd
@
vanechaeva@vanechaeva: ~$
```

Рис. 4: Рисунок 4

### Задача 3

Написать командный файл — аналог команды `ls` (без использования самой этой команды и команды `dir`). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.



```
GNU nano 6.2 ls.sh
#!/bin/bash
echo "В текущем каталоге:"
for file in *
do
if [ -d "$file" ]; then
echo "dir: $file"
else
echo "file: $file"
fi

if [ -r "$file" ]; then
echo "readabale"
fi

if [ -w "$file" ]; then
echo "writable"
fi

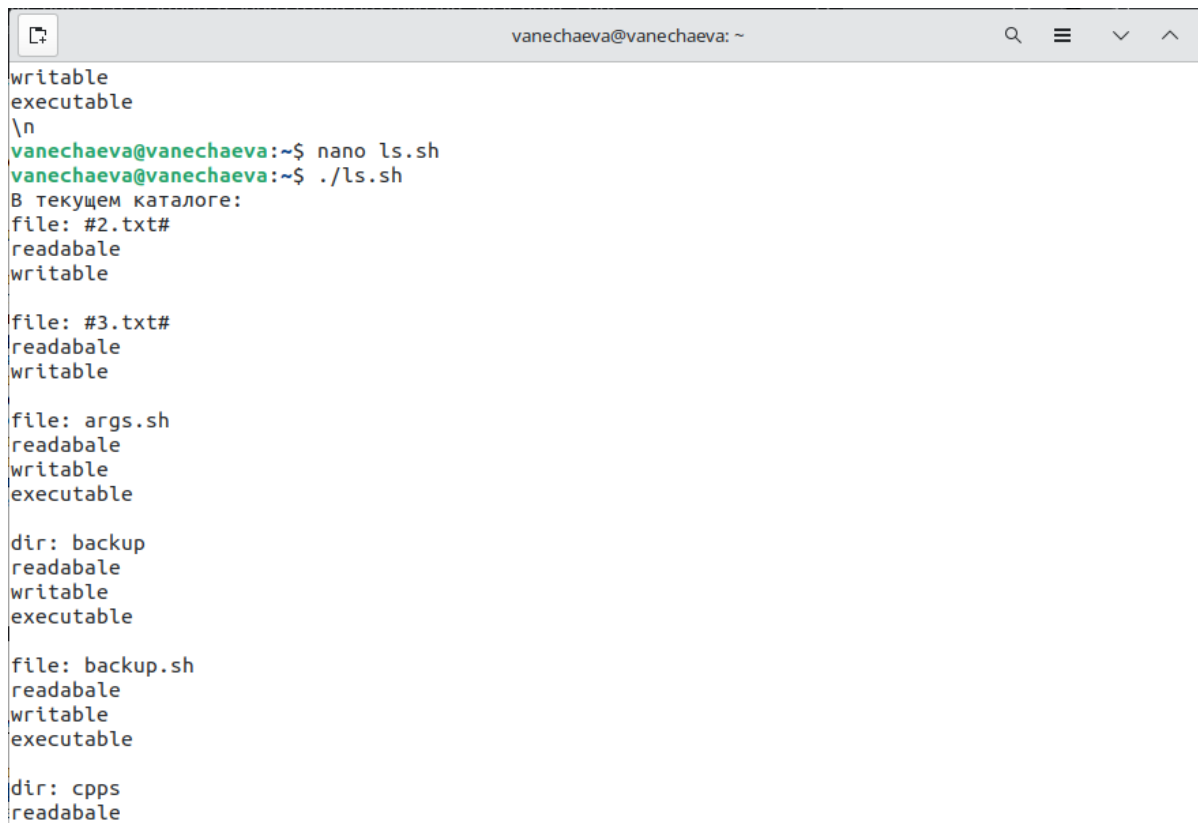
if [ -x "$file" ]; then
echo "executable"
fi
printf "\n"
done
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция  
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^\_ К строке

Рис. 5: Рисунок 5

Цикл проходит по всем файлам директории и проверяет на несколько критериев: Если тип это файл (-f) или каталог (-d), оно выводится в консоли. Таким образом выводятся все файлы и каталоги. Остается вывести инфо о доступах.

В трех следующих проверках условий проверяются права доступа -r (readable), -w (writable), -x (executable), информация об этом также выводится в консоли.




```
vanechaeva@vanechaeva: ~  
writable  
executable  
\\n  
vanechaeva@vanechaeva:~$ nano ls.sh  
vanechaeva@vanechaeva:~$ ./ls.sh  
В текущем каталоге:  
file: #2.txt#  
readabale  
writable  
  
file: #3.txt#  
readabale  
writable  
  
file: args.sh  
readabale  
writable  
executable  
  
dir: backup  
readabale  
writable  
executable  
  
file: backup.sh  
readabale  
writable  
executable  
  
dir: cpps  
readabale
```

Рис. 6: Рисунок 6

## Задача 4

Написать командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки.



```
GNU nano 6.2 filecount.sh
#!/bin/bash
directory="."
if [[ "$1" != .* && -d "$1" ]]; then
    directory="$1"
    shift
fi

if [ "$1" = "" ]; then
    echo "Пожалуйста, введите аргумент с расширением."
    exit
fi

extension="$1"

count=$(find "$directory" -type f -name "*$extension" | wc -l)

echo "Количество файлов расширения $extension в указанном каталоге: $count"
```

[ Прочитано 17 строк ]

<b>^G</b> Справка	<b>^O</b> Записать	<b>^W</b> Поиск	<b>^K</b> Вырезать	<b>^T</b> Выполнить	<b>^C</b> Позиция
<b>^X</b> Выход	<b>^R</b> ЧитФайл	<b>^I</b> Замена	<b>^U</b> Вставить	<b>^J</b> Выводить	<b>^_</b> К строке

Рис. 7: Рисунок 7

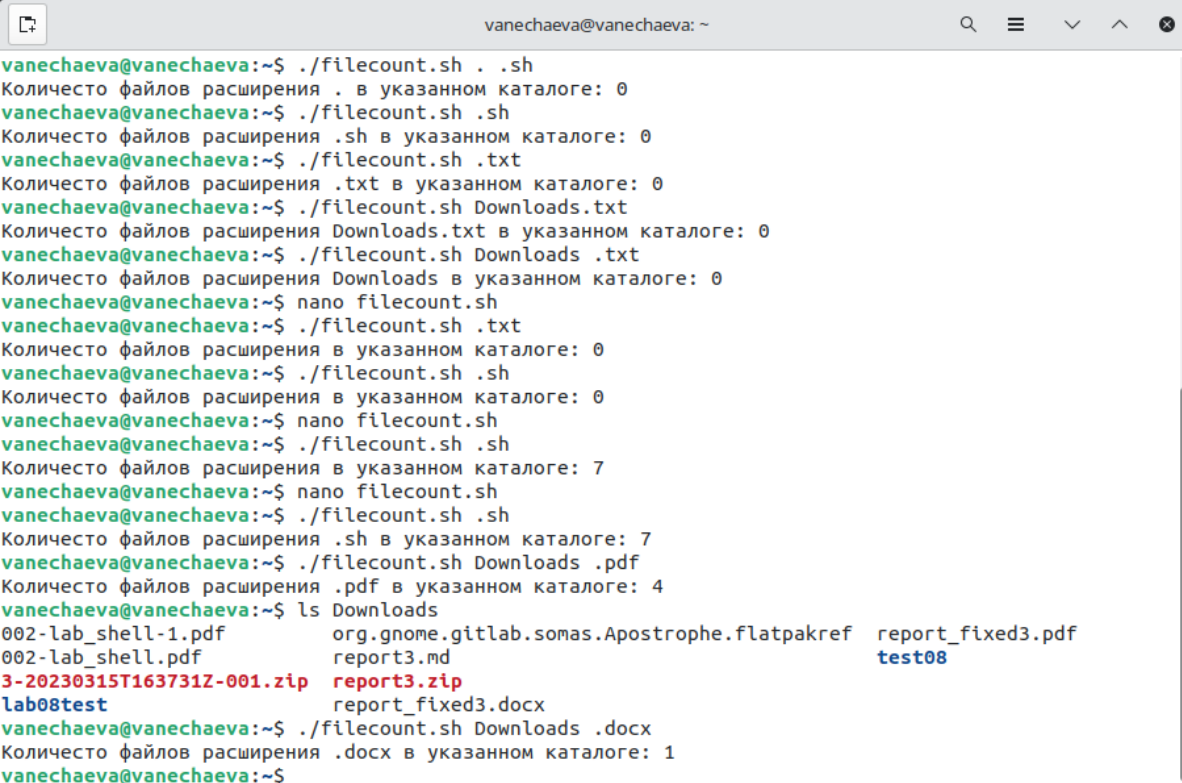
Директория по умолчанию устанавливается текущая - “.”

В первом условии if проверяется, что первый аргумент это НЕ расширение и что первый аргумент это каталог. Если условия подходят, то переменной directory вместо значения по умолчанию присваивается каталог из первого аргумента. shift сдвигает аргумент влево, чтобы скрипт мог прочитать расширение, указанное пользователем.

Во второй проверке if проверяется условие для случая, если мы решили не указывать директорию в аргументе и указали только нужное расширение. Проверяется, что первый аргумент пустой, и в таком случае выводится сообщение о необходимости указать расширение и скрипт завершает выполнение с помощью exit.

Далее extension присваивается значение первого введенного нами аргумента. count ищет с помощью find объекты типа f (файл) с названием по маске extension вве-

денного нами, конвеер передает результат поиска функции wc которая с опцией -l считает количество строк и в последней строке через echo мы его выводим в консоли.



```
vanechaeva@vanechaeva:~$ ./filecount.sh . .sh
Количество файлов расширения . в указанном каталоге: 0
vanechaeva@vanechaeva:~$ ./filecount.sh .sh
Количество файлов расширения .sh в указанном каталоге: 0
vanechaeva@vanechaeva:~$ ./filecount.sh .txt
Количество файлов расширения .txt в указанном каталоге: 0
vanechaeva@vanechaeva:~$ ./filecount.sh Downloads.txt
Количество файлов расширения Downloads.txt в указанном каталоге: 0
vanechaeva@vanechaeva:~$ ./filecount.sh Downloads .txt
Количество файлов расширения Downloads в указанном каталоге: 0
vanechaeva@vanechaeva:~$ nano filecount.sh
vanechaeva@vanechaeva:~$ ./filecount.sh .txt
Количество файлов расширения в указанном каталоге: 0
vanechaeva@vanechaeva:~$ ./filecount.sh .sh
Количество файлов расширения в указанном каталоге: 0
vanechaeva@vanechaeva:~$ nano filecount.sh
vanechaeva@vanechaeva:~$ ./filecount.sh .sh
Количество файлов расширения в указанном каталоге: 7
vanechaeva@vanechaeva:~$ nano filecount.sh
vanechaeva@vanechaeva:~$ ./filecount.sh .sh
Количество файлов расширения .sh в указанном каталоге: 7
vanechaeva@vanechaeva:~$ ./filecount.sh Downloads .pdf
Количество файлов расширения .pdf в указанном каталоге: 4
vanechaeva@vanechaeva:~$ ls Downloads
002-lab_shell-1.pdf      org.gnome.gitlab.somas.Apostrophe.flatpakref  report_fixed3.pdf
002-lab_shell.pdf       report3.md                                     test08
3-20230315T163731Z-001.zip  report3.zip
lab08test                report_fixed3.docx
vanechaeva@vanechaeva:~$ ./filecount.sh Downloads .docx
Количество файлов расширения .docx в указанном каталоге: 1
vanechaeva@vanechaeva:~$
```

Рис. 8: Рисунок 8

## Выводы

В ходе данной лабораторной работы мною были изучены основы программирования в оболочке ОС UNIX/Linux.



# Контрольные вопросы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются?

Командная оболочка — программа, через которую пользователь или администратор управляет операционной системой и установленными программами, используя командную строку. У этого понятия множество синонимов — shell, командный интерпретатор и командный процессор. Пример `bash`, `zsh`.

2. Что такое POSIX?

POSIX - это стандарт, описывающий интерфейс между операционной системой и прикладной программой

3. Как определяются переменные и массивы в языке программирования `bash`?

Переменная инициализруется с помощью значка доллара.

Инициализация, или объявление массивов происходит либо явно, с использованием команды `declare`, либо при указании массива прямо в коде. Обратите внимание на то, что если не использовать команду `declare` – то массив по-умолчанию будет считаться индексированным.

Примеры объявления массивов:

```
$ declare -a array
```

```
$ array=( [1]=one [2]=two )
```

```
$ echo ${array[@]}
```

```
one two
```

4. Какие арифметические операции можно применять в языке программирования bash?

```
let a=5+4
echo $a # 9
let "a = 5 + 4"
echo $a # 9
let a++
echo $a # 10
let "a = 4 * 5"
echo $a # 20
let "a = $1 + 30"
echo $a
```

5. Что означает операция (( ))?

В двойных скобках выполняется целочисленная арифметика.

6. Какие стандартные имена переменных Вам известны?

arg

7. Что такое метасимволы?

Метасимволом называется особая комбинация символов, которые в реальной строке могут совпадать с разными символами. Кроме того, некоторые метасимволы совпадают с границами между символами (например с началом строки).

8. Как экранировать метасимволы?

С помощью бэкслепа

9. Как создавать и запускать командные файлы?

Создать файл filename.sh, дать права на запуск `chmod +x filename.sh`, запустить скрипт `./filename.sh`

10. Каким образом можно выяснить, является файл каталогом или обычным файлом?

в консоли через `ls`

в шелл скрипте `[ -d "/path/dir/" ] && echo "Directory /path/dir/ exists"`

11. Как передаются параметры в командные файлы?

При запуске скрипта указываются через пробел

12. Назовите специальные переменные языка `bash` и их назначение.

`$*` – все аргументы; `$@` – все аргументы; `$#` – количество аргументов; `$0` – имя скрипта;