

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 11

дисциплина: Операционные системы

Студент: Губина Ольга Вячеславовна Группа: НПИбд-01-20

Преподаватель: Велиева Татьяна Рефатовна

МОСКВА

2021 г.

Цель работы:

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

Задачи:

1. Получить навыки работы с основными командами языка bash;
2. Научиться писать небольшие командные файлы;
3. Применить их на практике.

Теоретическое введение:

В данной лабораторной работе нам предстоит научиться писать командные файлы и использовать их на практике. Для этого нам необходимо ознакомиться с некоторой теорией.

Командные процессоры (оболочки)

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.

В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

- оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
- C-оболочка (или csh) — надстройка на оболочкой Борна, использующая Сподобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.

Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linuxподобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке bash. В других оболочках большинство команд будет совпадать с описанными ниже.

Переменные в языке программирования bash

Командный процессор *bash* обеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем. Пользователь имеет возможность присвоить переменной значение некоторой строки символов. Например, команда

```
mark=/usr/andy/bin
```

присваивает значение строки символов `/usr/andy/bin` переменной `mark` типа строка символов.

Использование:

```
mv afile ${mark}
```

переместит файл `afile` из текущего каталога в каталог с абсолютным полным именем `/usr/andy/bin`.

Использование значения, присвоенного некоторой переменной, называется *подстановкой*.

Команды read и echo

Команда read позволяет записать значение для переменной с клавиатуры. Она имеет следующий синтаксис:

```
read <variable>
```

Команда echo выводит текст на экран, если имеет вид:

```
echo "Some text"
```

В данном случае она выведет на экран *Some text*.

С помощью данной команды также можно вывести на экран содержимое, например, переменных:

```
echo <variable>
```

С прочей теорией и основами языка bash можно ознакомиться в материалах к *лабораторной работе №17*^[1].

Также в ходе выполнения заданий лабораторной работы я столкнулась в необходимости изучения дополнительных материалов, а именно:

- архивирование файлов в Linux^[2]
- использование массивов в bash^[3]
- различные способы составления списка содержимого каталога без использования команды ls^[4]
- команда find в Linux^[5]
- команда wc в Linux^[6]

Выполнение работы:

Задание 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.

Для этого сначала перейдем в домашний каталог (`cd`), после чего создадим наш командный файл, который будет называться `arch.sh`, командой `touch`. Далее в домашнем каталоге создадим каталог `backup` командой создания каталогов `mkdir`, в нем мы будем создавать резервные копии и архивы с командными файлами (*рисунок 1*).

Чтобы понять структуру архивации файлов и создания архивов воспользуемся справкой `man tar` и изучим команду `tar`, которая позволит нам создать архив (*рисунок 1*). Для создания командных файлов будем использовать текстовый редактор `vi`. Открываем с его помощью будущий командный файл `arch.sh` (`vi arch.sh`) (*рисунок 1*).

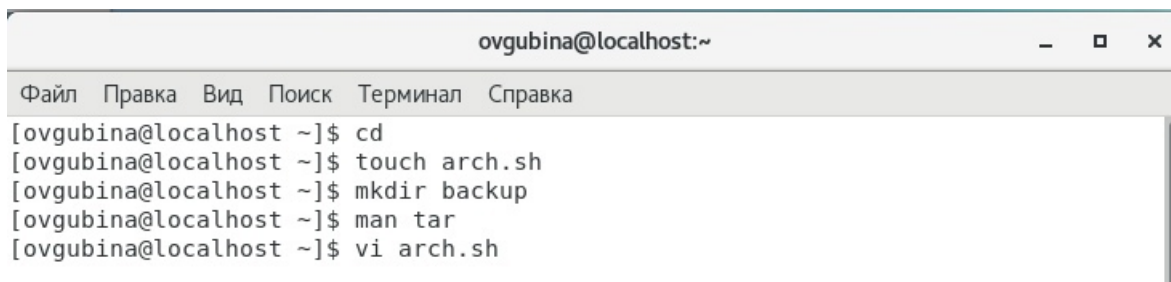
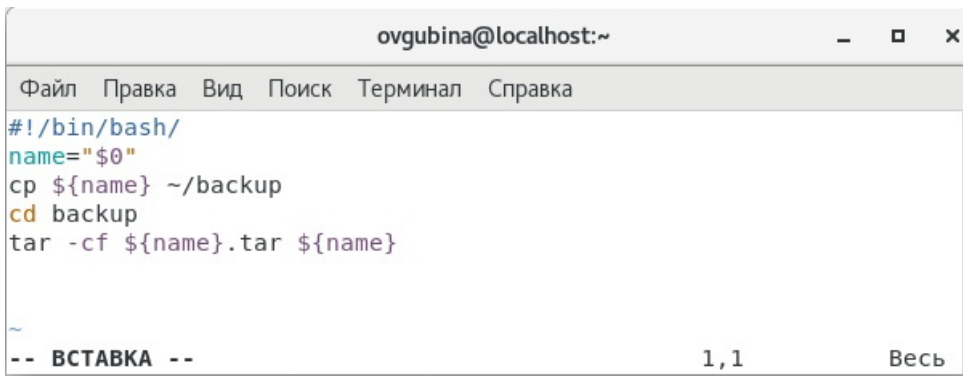


рисунок 1: создание файла и каталога

Пишем сам командный файл. Для того, чтобы система распознавала его как командный, в первой строке прописываем `#!/bin/bash` (*рисунок 2*).

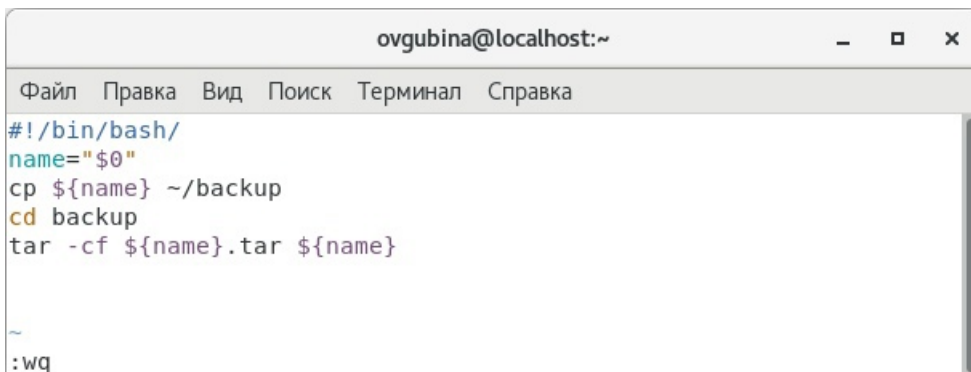
Теперь о структуре кода. Создаем переменную `name`, в которой будет содержаться имя данного командного файла (передаем его указателем `$0`) `name="$0"`. Далее командой `cp` перемещаем копируем файл в недавно созданный каталог `~/backup`. Переходим в данный каталог. Вот теперь воспользуемся командой `tar -cf`, которая позволит нам создать архив, имеющий такое название, какое имеет командный файл, но в формате `.tar` (`${name}.tar`). Ключ `-cf` позволяет нам создать архив и сразу же поместить в него наш командный файл, который мы передаем ссылкой `${name}` (*рисунок 2*).



```
ovgubina@localhost:~  
Файл Правка Вид Поиск Терминал Справка  
#!/bin/bash/  
name="$0"  
cp ${name} ~/backup  
cd backup  
tar -cf ${name}.tar ${name}  
  
~  
-- ВСТАВКА -- 1,1 Весь
```

рисунок 2: создание командного файла

Переходим в командный режим редактора vi нажатием клавиши Esc и, нажав :, переходим в режим последней строки, по средствам которой мы записываем изменения в файл - w - и выходим из редактора - q (рисунок 3).



```
ovgubina@localhost:~  
Файл Правка Вид Поиск Терминал Справка  
#!/bin/bash/  
name="$0"  
cp ${name} ~/backup  
cd backup  
tar -cf ${name}.tar ${name}  
  
~  
:wq
```

рисунок 3: запись изменений и выход

Теперь протестируем созданный файл. Для того, чтобы запустить его как команду необходимо использовать bash (рисунок 4)

```
[ovgubina@localhost ~]$ bash arch.sh  
[ovgubina@localhost ~]$
```

рисунок 4: запуск командного файла

Проверим наш файл. Посмотрим на результат его работы - перейдем в каталог backup, в котором должен был создаться необходимый архив. Видим, что в нем действительно лежит копия исходного файла и создан новый архив (рисунок 5).

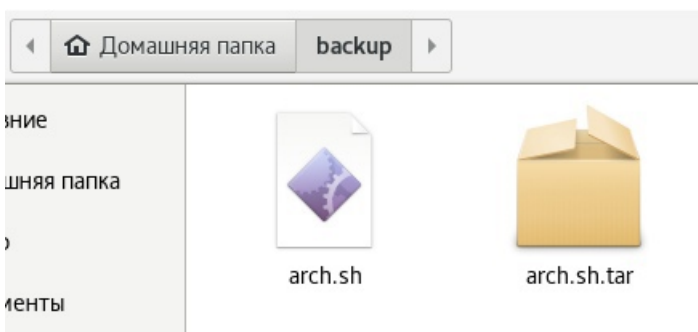


рисунок 5: проверка работы командного файла (1)

Теперь на всякий случай проверим созданный архив, открываем его и видим, что в нем находится все тот же командный файл arch.sh (рисунок 6).

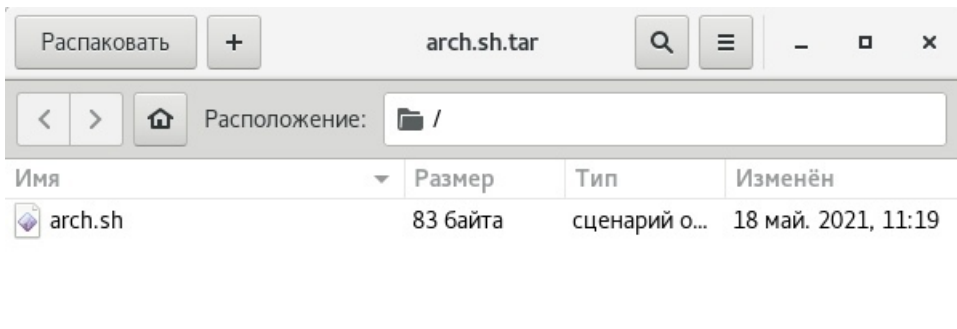


рисунок 6: проверка работы командного файла (2)

Задание 2.

Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов.

Сперва создадим соответствующий командный файл `numbers.sh` и сразу откроем его в редакторе (рисунок 7).

```
[ovgubina@localhost ~]$ vi numbers.sh
```

рисунок 7: создание файла `numbers.sh`

Для того, чтобы вводить неизвестное количество аргументов (даже большее десяти) и обрабатывать их, воспользуемся массивом, который назовем `numbers`. Сначала объявим его: `declare -a numbers`. С помощью команды `echo` выведем на экран сообщение о том, что нужно ввести элементы, притом в качестве разделителя использовать пробелы. Далее командой `read -a numbers` считываем с клавиатуры элементы массива. Выводим строку `Your numbers` и выводим все элементы массива - `echo ${numbers[@]}` (`@` - все элементы массива) (рисунок 8).

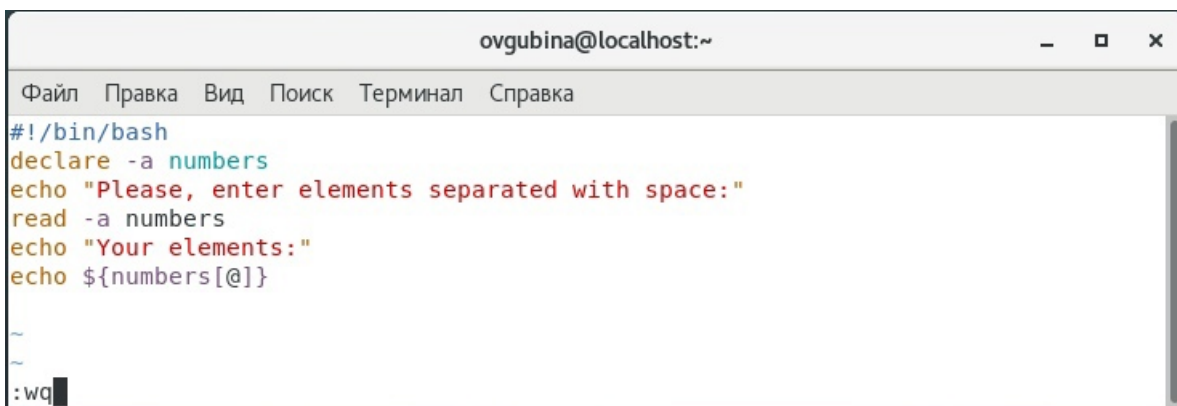


рисунок 8: набор текста файла, сохранение и выход

Проверим работу нашего файла. Видим, что он работает исправно и действительно выводит те числа, которые мы ввели (рисунок 9).

```
[ovgubina@localhost ~]$ vi numbers.sh
[ovgubina@localhost ~]$ bash numbers.sh
Please, enter elements separated with space:
1 2 3 4 5 6 7 8 9 10 3567 0872 12308
Your elements:
1 2 3 4 5 6 7 8 9 10 3567 0872 12308
[ovgubina@localhost ~]$
```

рисунок 9: результат работы командного файла

Задание 3.

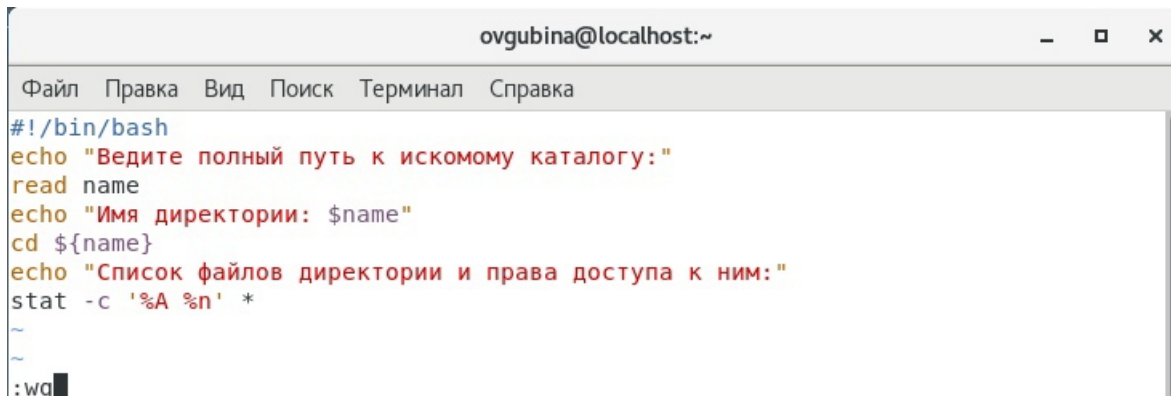
Написать командный файл — аналог команды `ls` (без использования самой этой команды и команды `dir`). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.

Сперва создадим соответствующий командный файл `ls.sh` и сразу откроем его в редакторе (рисунок 10).

```
[ovgubina@localhost ~]$ vi ls.sh
```

рисунок 10: создание файла ls.sh

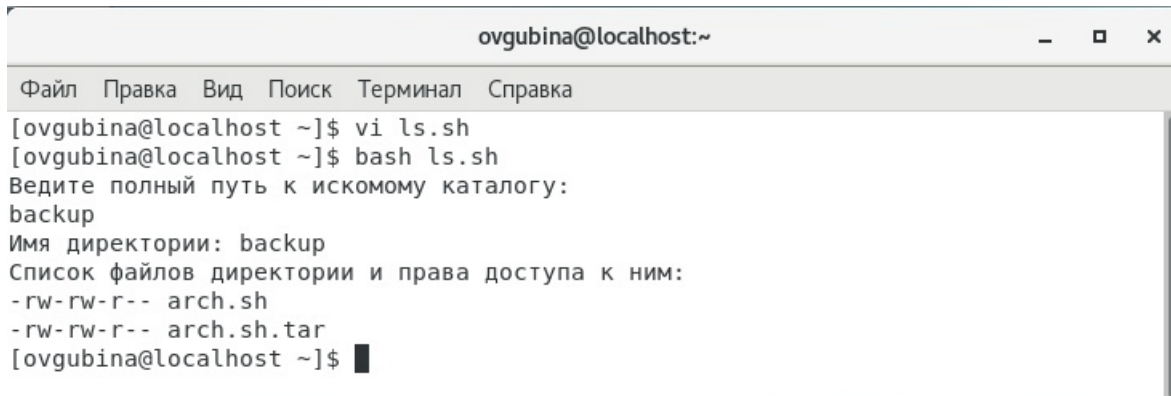
Пишем текст командного файла. Сначала выведем сообщение о вводе имени каталога, который мы хотим рассмотреть, - `echo`. Команда `read` позволит нам считать введенную с клавиатуры директорию в переменную `name`. Выводим имя директории и переходим в заданный каталог: `cd ${name}`. Выведем строку-сообщение о выводе файлов каталога и прав доступа к ним командой `echo`. Выведем содержимое текущего каталога командой `stat: stat -c '%A %n' *`. Где `-c` является ключом, который выведет наши файлы построчно, `%A` - вывод прав доступа в формате, читаемом для человека, а не машины, `%n` - названия файлов, `*` - указывает на текущий каталог (рисунок 11).



```
ovgubina@localhost:~  
Файл Правка Вид Поиск Терминал Справка  
#!/bin/bash  
echo "Ведите полный путь к искомому каталогу:"  
read name  
echo "Имя директории: $name"  
cd ${name}  
echo "Список файлов директории и права доступа к ним:"  
stat -c '%A %n' *  
~  
~  
:wq
```

рисунок 11: набор текста файла, сохранение и выход

Посмотри на результаты работы нашего командного файла. Рассмотрим директорию, созданную в задании 1 (рисунок 12), и домашний катлог (рисунок 13). Видим, что файл работает исправно, все, что требовалось, выполнили.



```
ovgubina@localhost:~  
Файл Правка Вид Поиск Терминал Справка  
[ovgubina@localhost ~]$ vi ls.sh  
[ovgubina@localhost ~]$ bash ls.sh  
Ведите полный путь к искомому каталогу:  
backup  
Имя директории: backup  
Список файлов директории и права доступа к ним:  
-rw-rw-r-- arch.sh  
-rw-rw-r-- arch.sh.tar  
[ovgubina@localhost ~]$
```

рисунок 12: результат работы командного файла

```
[ovgubina@localhost ~]$ bash ls.sh
Ведите полный путь к искомому каталогу:
/home/ovgubina
Имя директории: /home/ovgubina
Список файлов директории и права доступа к ним:
-rw-rw-r-- abc1
-rw-rw-r-- arch.sh
drwxr--r-- australia
drwxrwxr-x backup
drwxr-xr-x bin
-rw-rw-r-- feathers
-rw-rw-r-- file.txt
-rw-rw-r-- #lab07.sh#
-rw-rw-r-- lab07.sh
-rw-rw-r-- lab07.sh~
-rw-rw-r-- ls.sh
-r-xr--r-- my_os
drwxrwxr-x new_directory
-rw-rw-r-- #new_file.txt#
-rw-rw-r-- numbers.sh
-rw-rw-r-- quick.cpp
drwxrwxr-x ski.plases
-rw-rw-r-- text.txt
drwxr-xr-x work
drwxr-xr-x Видео
drwxr-xr-x Документы
drwxr-xr-x Загрузки
drwxr-xr-x Изображения
drwxr-xr-x Музыка
drwxr-xr-x Общедоступные
drwxr-xr-x Рабочий стол
drwxr-xr-x Шаблоны
[ovgubina@localhost ~]$ █
```

рисунок 13: результат работы командного файла

Задание 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.

Сперва создадим соответствующий командный файл `find.sh` и сразу откроем его в редакторе (рисунок 14).

```
drwxr-xr-x Шаблоны
[ovgubina@localhost ~]$ vi find.sh
```

рисунок 14: создание файла `find.sh`

Напишем сам командный файл. Введем обозначения двух переменных: `dirt`, в которую мы запишем рассматриваемую директорию, и `format`, в которую запишем искомый формат файла. Им сопутствуют два вывода `echo`, сообщающих пользователю о том, что именно необходимо ввести в данный момент. `${dirt}` - переходим в требуемую директорию. Ищем (команда `find`) в ней ("." - текущая директория) файлы по именам (`-name`), в которых встречается нам введенный формат. Конвейером считываем нереализованный вывод и командой `wc -l` считаем его строки, т.е. - файлы, найденные в данной директории и соответствующие требованиям. (рисунок 15)

```
ovgubina@localhost:~
Файл Правка Вид Поиск Терминал Справка
#!/bin/bash
dirt=""
echo "Введите директорию:"
read dirt
format=""
echo "Введите требуемый формат:"
read format
cd ${dirt}
echo "Файлов с таким форматом в данной директории: "
find . -name "*.${format}" | wc -l

:wq
```

рисунок 15: ввод текста файла

Посмотрим на результат работы написанного файла. Введем с клавиатуры путь к домашней директории, будем искать в ней файлы формата txt, видим, что найдем 41 файл (*рисунок 16*). Проверить такое количество файлов проблематично, поэтому осуществим проверку, используя директорию australia. Откроем ее и запустим файл с теми же требованиями. В выводе найден один файл, соответствующий требованиям, смотрим на содержимое каталога, видим, что это действительно так (*рисунок 17*).

```
[ovgubina@localhost ~]$ vi find.sh
[ovgubina@localhost ~]$ bash find.sh
Введите директорию:
/home/ovgubina
Введите требуемый формат:
txt
Файлов с таким форматом в данной директории:
41
[ovgubina@localhost ~]$ █
```

рисунок 16: результат работы

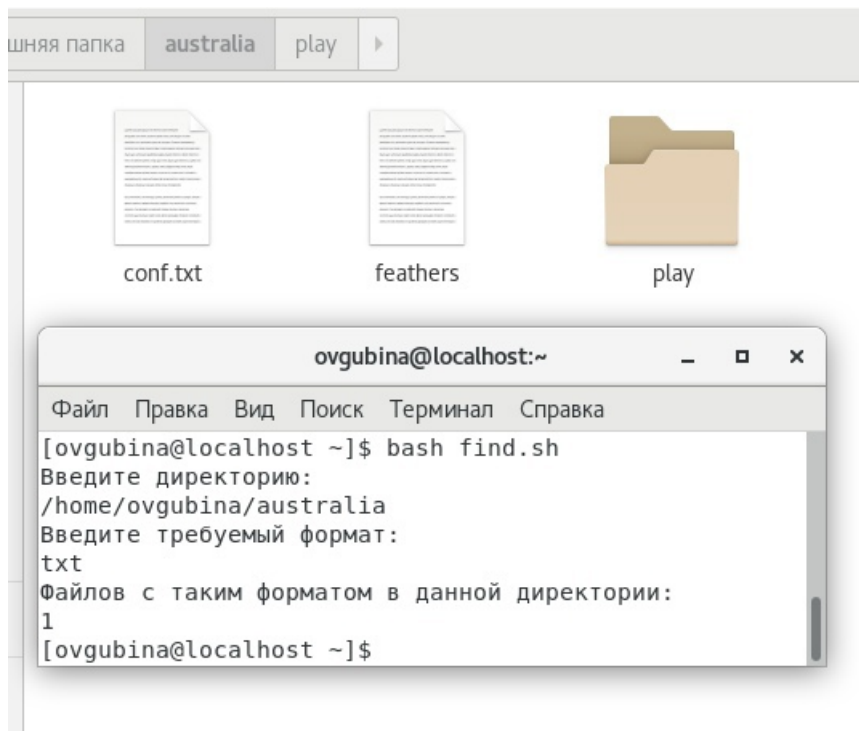


рисунок 17: результат работы

Вывод:

Изучила основы программирования в оболочке ОС UNIX/Linux. Изучила основы языка bash, научилась писать небольшие командные файлы.

Библиография:

- [1] [Лабораторная работа №11](#)
- [2] [Архивирование файлов в Linux](#)
- [3] [Использование массивов в bash](#)
- [4] [Различные способы составления списка содержимого каталога без использования команды ls](#)
- [5] [Команда find в Linux](#)
- [6] [Команда wc в Linux](#)