

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. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Задачи:

- 1. Научиться писать небольшие командные файлы с использованием логических управляющих конструкций и циклов;
- 2. 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 №11*^[1].

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

- архивирование файлов в Linux^[2]
- команда `find` в Linux^[3]

- циклы `if`[4]
- команда `xargs`[5]

Выполнение работы:

Задание 1

Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами:

- `-iinputfile` — прочитать данные из указанного файла;
- `-ooutputfile` — вывести данные в указанный файл;
- `-pshablon` — указать шаблон для поиска;
- `-C` — различать большие и малые буквы;
- `-n` — выдавать номера строк.

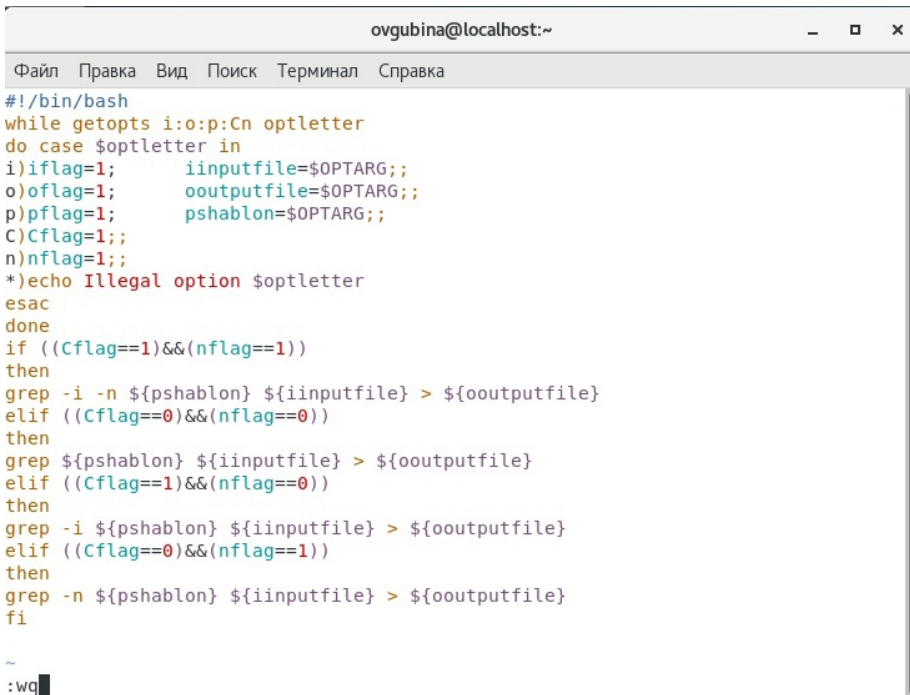
а затем ищет в указанном файле нужные строки, определяемые ключом `-p`

Создадим и откроем командой `vi` командный файл `get.sh`, который будет анализировать командную строку по предложенным в задании ключам (рисунк 1).

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

рисунок 1:

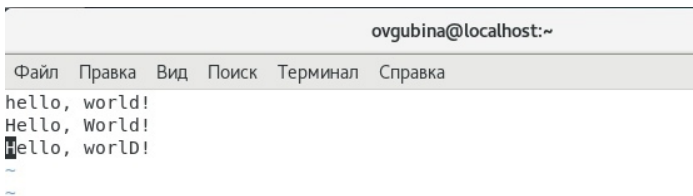
Напишем код для командного файла (рисунк 2). Используем пример использования оператора `getopts` из материалов к лабораторной работе №11, а также циклы `if`, которые будут помогать распознать, какие именно действия нам нужно выполнить в зависимости от упоминания ключей `-C` и `-n`.



```
ovgubina@localhost:~
Файл  Правка  Вид  Поиск  Терминал  Справка
#!/bin/bash
while getopts i:o:p:Cn optletter
do case $optletter in
i)iiflag=1;      iinputfile=$OPTARG;;
o)ooflag=1;      ooutputfile=$OPTARG;;
p)pflag=1;       pshablon=$OPTARG;;
C)Cflag=1;;
n)nflag=1;;
*)echo Illegal option $optletter
esac
done
if ((Cflag==1)&&(nflag==1))
then
grep -i -n ${pshablon} ${iinputfile} > ${ooutputfile}
elif ((Cflag==0)&&(nflag==0))
then
grep ${pshablon} ${iinputfile} > ${ooutputfile}
elif ((Cflag==1)&&(nflag==0))
then
grep -i ${pshablon} ${iinputfile} > ${ooutputfile}
elif ((Cflag==0)&&(nflag==1))
then
grep -n ${pshablon} ${iinputfile} > ${ooutputfile}
fi
~
:wq
```

рисунок 2:

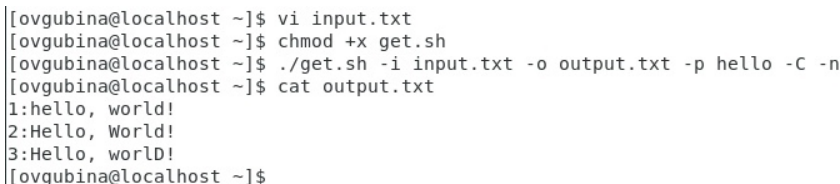
Создали файл `input.txt`, из которого будем читать информацию. В него запишем три одинаковых строки, отличных только прописными и строчными буквами (рисунк 3).



```
ovgubina@localhost:~
Файл  Правка  Вид  Поиск  Терминал  Справка
hello, world!
Hello, World!
Hello, world!
~
~
```

рисунок 3:

Вызываем наш командный файл в качестве команды, в качестве файла для чтения выбираем файл `input.txt`, для записи - `output.txt`, параметр поиска - слово "hello". Сразу проверим работу командного файла, обозначив обе опции `-C` и `-n`. Проосматриваем файл `output.txt` с помощью команды `cat`. Видим, что программа работает исправно (рисунк 4).



```
[ovgubina@localhost ~]$ vi input.txt
[ovgubina@localhost ~]$ chmod +x get.sh
[ovgubina@localhost ~]$ ./get.sh -i input.txt -o output.txt -p hello -C -n
[ovgubina@localhost ~]$ cat output.txt
1:hello, world!
2:Hello, World!
3:Hello, world!
[ovgubina@localhost ~]$
```

рисунок 4:

Задание 2

Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`.

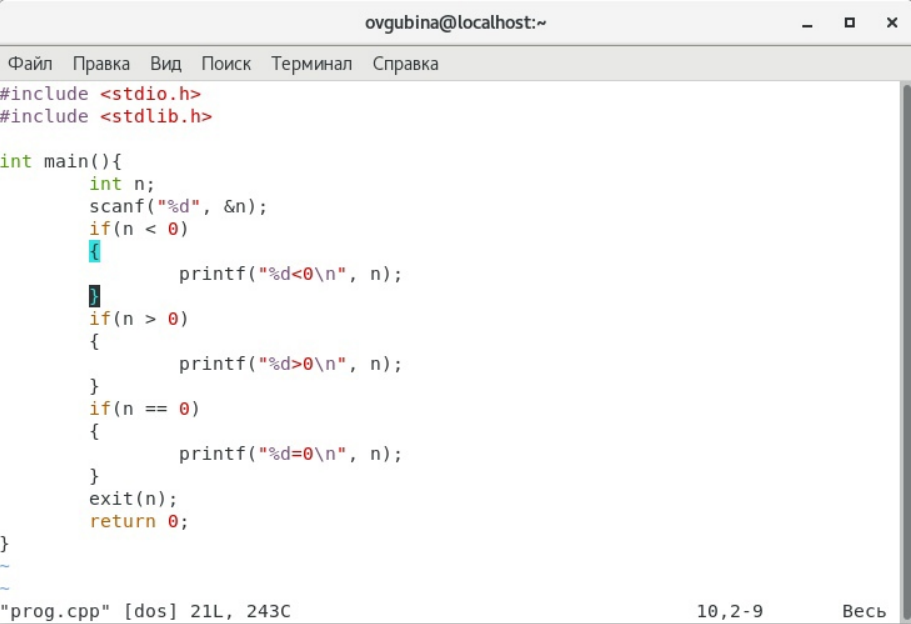
передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды \$?, выдать сообщение о том, какое число было введено.

Создадим и откроем с помощью редактора vi два файла, которые нам пригодятся для этого задания, prog.cpp - файл с программой на языке C, k.sh - командный файл, который вызовет на выполнение файл с кодом и проанализирует введенное в него число (рисунок 5).

```
[ovgubina@localhost ~]$ vi prog.cpp
[ovgubina@localhost ~]$ vi k.sh
```

рисунок 5:

Рассмотрим код файла prog.cpp (рисунок 6). Здесь все стандартно, по канонам программ на C. Вводим число с клавиатуры, сравниваем его значение с нулем и выводим соответствующее сообщение на экран (число больше, меньше нуля или равно ему). Завершаем программу с помощью функции exit(n), передавая информацию в о коде завершения в оболочку

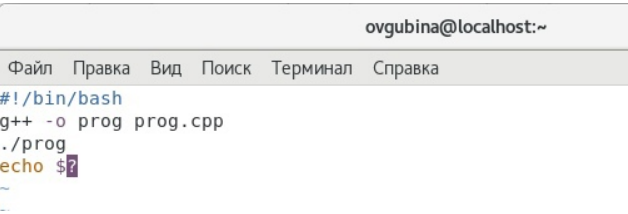


```
ovgubina@localhost:~
Файл  Правка  Вид  Поиск  Терминал  Справка
#include <stdio.h>
#include <stdlib.h>

int main(){
    int n;
    scanf("%d", &n);
    if(n < 0)
    {
        printf("%d<0\n", n);
    }
    if(n > 0)
    {
        printf("%d>0\n", n);
    }
    if(n == 0)
    {
        printf("%d=0\n", n);
    }
    exit(n);
    return 0;
}
~
~
"prog.cpp" [dos] 21L, 243C                               10,2-9      Весь
```

рисунок 6:

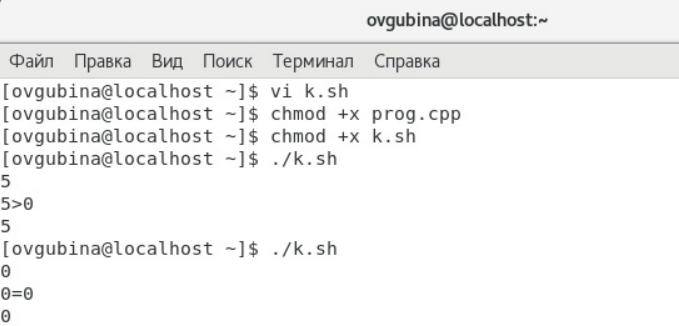
Теперь перейдем к самому командному файлу (рисунок 7). В нем мы сначала компилируем наш код (2 строка), вызываем программу на выполнение (./prog), в конце анализируем и передаем на экран с помощью echo \$?, какое число было введено для сравнения с нулем.



```
ovgubina@localhost:~
Файл  Правка  Вид  Поиск  Терминал  Справка
#!/bin/bash
g++ -o prog prog.cpp
./prog
echo $?
~
~
```

рисунок 7:

Проверим работу файлов (рисунок 8). Вызовем командный файл k.sh. в первом случае введем число 5, видим, что система вывела сообщение о том, что оно больше нуля, и после данной процедуры можно наблюдать результат работы командного файла - вывод числа, которое мы ввели. То же происходит если на ввод подается 0.



```
ovgubina@localhost:~
Файл  Правка  Вид  Поиск  Терминал  Справка
[ovgubina@localhost ~]$ vi k.sh
[ovgubina@localhost ~]$ chmod +x prog.cpp
[ovgubina@localhost ~]$ chmod +x k.sh
[ovgubina@localhost ~]$ ./k.sh
5
5>0
5
[ovgubina@localhost ~]$ ./k.sh
0
0=0
0
```

рисунок 8:

Задание 3

Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (например 1.tmp, 2.tmp, 3.tmp,4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).

Создадим и откроем с помощью редактора vi командный файл files.sh, который будет создавать необходимое число файлов и удалять их. Напишем сам код (рисунок 9). Сперва ввод с клавиатуры числа файлов. Дальше входим в первый цикл for для i от 1 до n, в котором для каждого i создаем файл командой touch, названием которого будет являться цифра, соответствующая значению i. Выводим на экран содержимое текущего каталога, чтобы убедиться, что файлы созданы. Спрашиваем, нужно ли удалить созданные файлы, если ответ да, то снова входим в цикл, только на этот раз не создаем файлы, а удаляем, и вновь проверяем содержимое каталога.

```
ovgubina@localhost:~
Файл Правка Вид Поиск Терминал Справка
#!/bin/bash
n=""
echo "Введите кол-во файлов, которые требуется создать:"
read n
cd
for ((i=1; i<=n; i++))
do
touch $i.txt
done
echo "Файлы каталога:"
ls
answer=""
echo "Вы хотите удалить созданные файлы?(y/n)"
read answer
if (answer=="y")
then
for ((i=1; i<=n; i++))
do
rm $i.txt
done
echo "Файлы каталога:"
ls
fi
```

рисунок 9:

Проверим работу командного файла (рисунок 10). в качестве числа на ввод подадим число 2. По выводам команды видим, что два соответствующих файла сначала были созданы, а потом удалены, поскольку мы согласились на их удаление, введя согласие - y .

```
[ovgubina@localhost ~]$ vi files.sh
[ovgubina@localhost ~]$ chmod +x files.sh
[ovgubina@localhost ~]$ ./files.sh
Введите кол-во файлов, которые требуется создать:
2
Файлы каталога:
1.txt      feathers    k.sh      new_directory  prog.cpp    Документы
2.txt      files.sh    k.sh~     #new_file.txt# prog.cpp~    Загрузки
abc1       file.txt    #lab07.sh# numbers.sh     quick.cpp    Изображения
arch.sh    find.sh     lab07.sh  output.txt     ski.plases   Музыка
australia  get.sh      lab07.sh~ pandoc.1       text.txt     Общедоступные
backup     get.sh~     ls.sh     prog           work         Рабочий стол
bin        input.txt   my_os     prog.c~        prog.c~      Видео
           input.txt   my_os     prog.c~        prog.c~      Шаблоны
Вы хотите удалить созданные файлы?(y/n)
y
Файлы каталога:
abc1       file.txt    #lab07.sh# numbers.sh     quick.cpp    Изображения
arch.sh    find.sh     lab07.sh  output.txt     ski.plases   Музыка
australia  get.sh      lab07.sh~ pandoc.1       text.txt     Общедоступные
backup     get.sh~     ls.sh     prog           work         Рабочий стол
bin        input.txt   my_os     prog.c~        prog.c~      Видео
           input.txt   my_os     prog.c~        prog.c~      Шаблоны
feathers    k.sh      new_directory  prog.cpp    Документы
files.sh    k.sh~     #new_file.txt# prog.cpp~    Загрузки
[ovgubina@localhost ~]$ █
```

рисунок 10:

Задание 4

Написать командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду `find`).

Напишем командный файл `tar.sh`, который будет архивировать все файлы директории, отредактированные не позднее чем 7 дней назад (рисунок 11). Для этого сначала будем вводить директорию, в которой будем работать, и имя будущего массива с клавиатуры, для этого в коде используем команду `read`. Когда обе переменные обозначены, воспользуемся командой `find`, которая поможет нам найти подходящие нам файлы. В ней мы использовали следующие опции:

- `.` - поиск осуществляется в текущем каталоге
- `-mtime -7` - файлы, редактированные не позднее чем 7 дней назад
- `-type f` - поиск именно файлов, без каталогов (папок)
- `-print0` - позволяет выводить полный путь к файлу на стандартном выходе, за которым следует нулевой символ

Далее используем конвейер и создаем архив с заданным с клавиатуры именем при помощи команды `tar` (рисунок 11). `xarg` - флаг `-0` `xargs` эффективно использует пространство в именах файлов, поэтому мы его используем для того, чтобы закинуть все найденные файлы в архив. Ключи `-cvzf` позволяют:

- `-c` - создать архив в `linux`
- `-v` - показать подробную информацию о процессе работы
- `-z` - сжать архив с помощью `Gzip`
- `-f` - обозначить файлы для записи архива

После выводим на экран содержимое заданной директории командой `ls`, чтобы удостовериться, что архив был создан.

```
ovgubina@localhost:~  
Файл Правка Вид Поиск Терминал Справка  
#!/bin/bash  
directory=""  
arch_name=""  
echo "Введите директорию:"  
read directory  
echo "Введите имя архива, который нужно создать:"  
read arch_name  
cd $directory  
find . -mtime -7 -type f -print0 | xargs -0 tar -cvzf ${arch_name}.tar  
echo "Файлы директории:"  
ls  
cd  
~
```

рисунок 11:

Выведем наш файл на проверку (рисунок 12). В качестве директории для работы обозначим директорию `new_directory`, в которую предварительно были помещены: 2 старых файла с прошлых лабораторных работ (`abc1`, `feathers`) и файл отредактированный в день выполнения работы - `text.txt`. В качестве имени архива задаем число выполнения лр. Видим, что сначала система выводит на экран файл, который удовлетворяет нашим условиям, и это действительно соответствует действительности. После он показывает нам содержимое заданной директории. Видим, что архив был успешно создан.

```
[ovgubina@localhost ~]$ vi tar.sh  
[ovgubina@localhost ~]$ ./tar.sh  
Введите директорию:  
new_directory  
Введите имя архива, который нужно создать:  
25_05  
./text.txt  
Файлы директории:  
25_05.tar abc1 feathers text.txt
```

рисунок 12:

Теперь проверим директорию на наличие архива. Видим, что он на месте (рисунок 13).

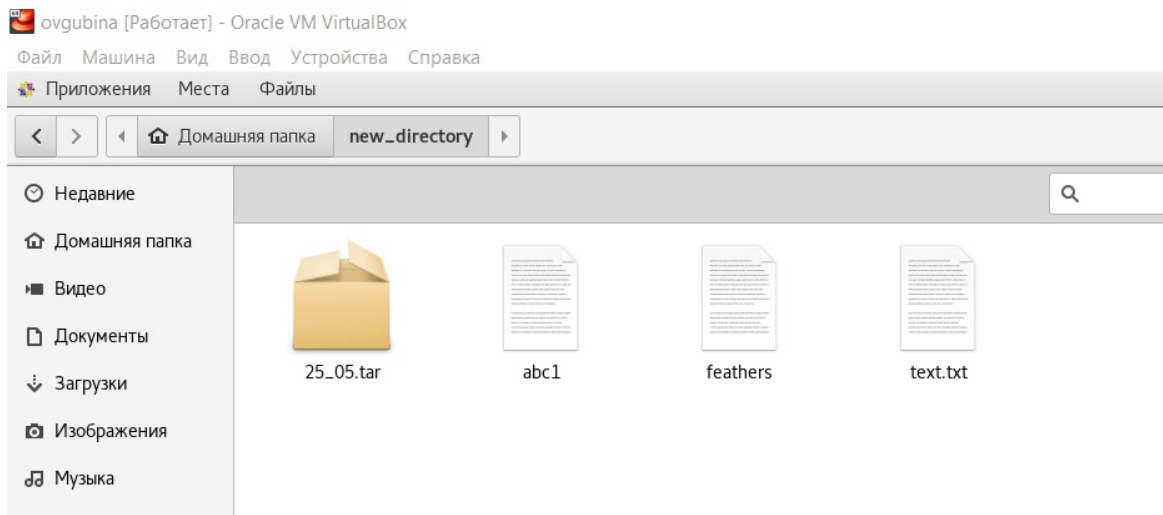


рисунок 13:

Откроем сам архив. Видим, что в нем действительно содержится файл `text.txt`, удовлетворяющий нашим условиям (рисунок 14).

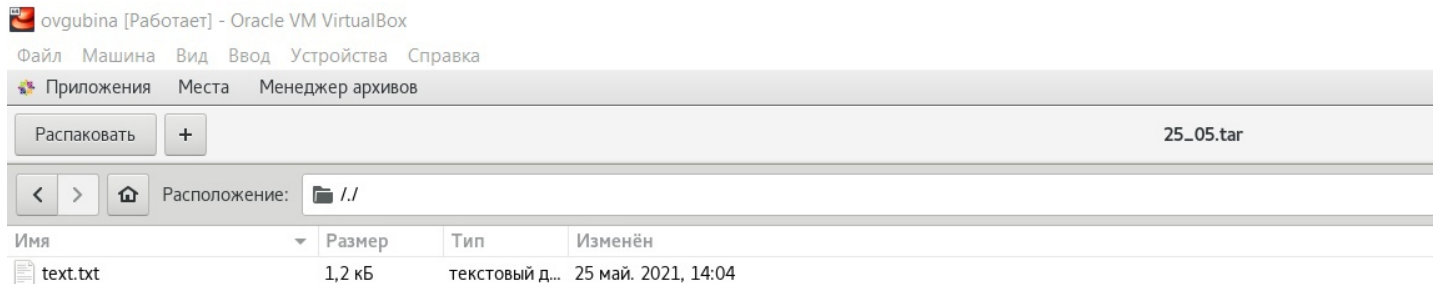


рисунок 14:

Вывод:

Изучила основы программирования в оболочке ОС UNIX/Linux. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Библиография:

[1] [Лабораторная работа №11](#)

[2] [Архивирование файлов в Linux](#)

[3] [Команда find в Linux](#)

[4] [Циклы if](#)

[5] [Команда xargs](#)