

# **Лабораторная работа 11**

**Отчет по лабораторной работе 11**

Куркина Евгения Вячеславовна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>17</b>
	<b>Список литературы</b>	<b>18</b>

## Список иллюстраций

3.1	Создание файла, переход в emacs . . . . .	7
3.2	Текст скрипта . . . . .	8
3.3	Сохранение файла . . . . .	8
3.4	Текст первого файла . . . . .	8
3.5	текст второго файла . . . . .	8
3.6	Права доступа . . . . .	9
3.7	Проверка работы скрипта . . . . .	9
3.8	Создание двух файлов . . . . .	10
3.9	Код программы на языке C . . . . .	10
3.10	Сохранение файла . . . . .	10
3.11	Код исполняемого файла . . . . .	11
3.12	права доступа и результат работы скрипта . . . . .	11
3.13	Создание файлов и переход в emacs . . . . .	12
3.14	Код программы . . . . .	12
3.15	Права доступа и проверка работы скрипта . . . . .	12
3.16	Создание файла для последнего скрипта . . . . .	13
3.17	Текст кода . . . . .	13
3.18	Создание каталога и проверка работы скрипта . . . . .	13
3.19	Результат работы . . . . .	14

## Список таблиц

# 1 Цель работы

Здесь приводится формулировка цели лабораторной работы. Формулировки цели для каждой лабораторной работы приведены в методических указаниях.

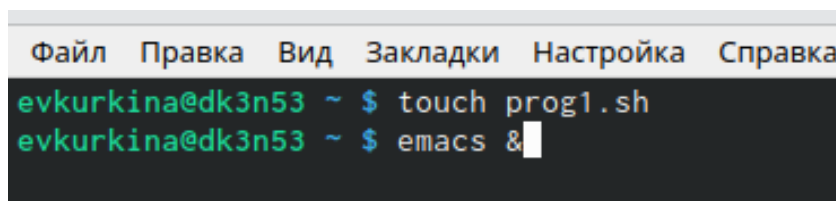
Цель данной лабораторной работы— Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 2 Задание

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

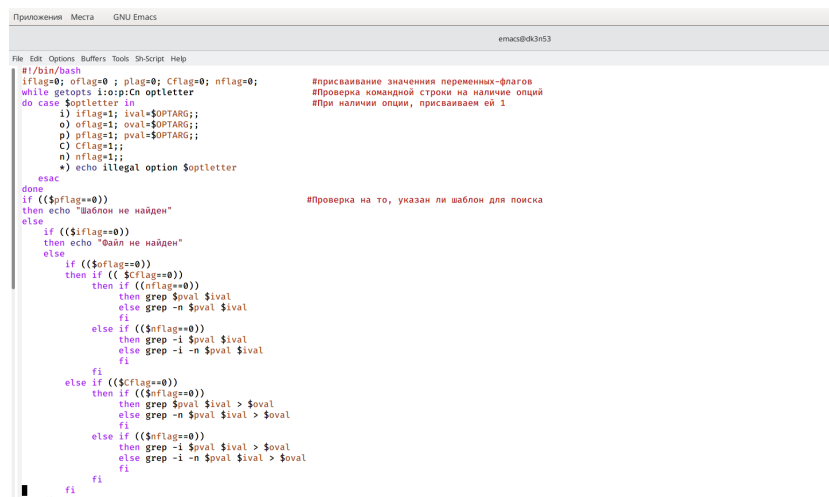
### 3 Выполнение лабораторной работы

1) Создала файл для первого скрипта, перешла в emacs (рис. 3.1). С помощью команды `getopts grep`, написала текст скрипта, который анализирует строку с ключами: `-inputfile`-прочитывает данные из указанного файла `-outputfile`-вывод данных в указанный файл `-p` шаблон `-u` указать шаблон поиска `-C` различить строчные и заглавные буквы `-n` выдать номера строк, а затем поиск указанных строк, которые определяются ключом `-p`. (рис. 3.2). Затем сохранила файл (рис. 3.3). Создала два файла, заполнила их словами (рис. 3.4) (рис. 3.5). Предоставила доступ на исполнение (рис. 3.6), проверила корректную работу скрипта (рис. 3.7).



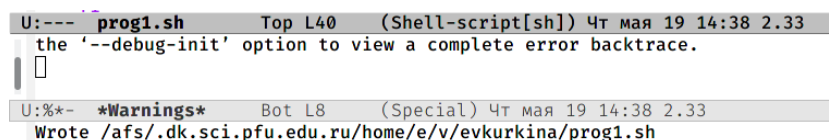
```
Файл  Правка  Вид  Закладки  Настройка  Справка
evkurkina@dk3n53 ~ $ touch prog1.sh
evkurkina@dk3n53 ~ $ emacs &
```

Рис. 3.1: Создание файла, переход в emacs



```
#!/bin/bash
iflag=0; oflag=0; pflag=0; cflag=0; nflag=0; #присваивание значения переменных-флагов
while getopts i:rp:cn optletter #Проверка командной строки на наличие опций
do case $optletter in
  i) iflag=1; ival=$OPTARG;;
  o) oflag=1; oval=$OPTARG;;
  p) pflag=1; pval=$OPTARG;;
  c) cflag=1;;
  n) nflag=1;;
  *) echo illegal option $optletter
esac
done
if (($flag==0)) #Проверка на то, указан ли шаблон для поиска
then echo "Шаблон не найден"
else
  if (($iflag==0))
  then echo "Файл не найден"
  else
    if (($oflag==0))
    then if (($cflag==0))
    then if (($nflag==0))
    then grep $val $val
    else grep -n $val $val
    fi
    else if (($nflag==0))
    then grep -i $val $val
    else grep -i -n $val $val
    fi
    fi
    else if (($cflag==0))
    then if (($nflag==0))
    then grep $val $val > $oval
    else grep -n $val $val > $oval
    fi
    else if (($nflag==0))
    then grep -i $val $val > $oval
    else grep -i -n $val $val > $oval
    fi
    fi
  fi
fi
```

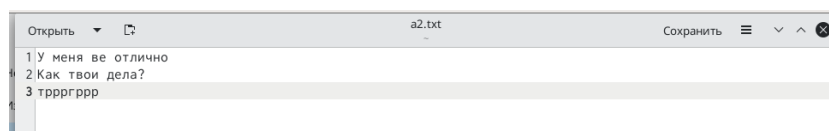
Рис. 3.2: Текст скрипта



```
U:--- prog1.sh Top L40 (Shell-script[sh]) Чт мая 19 14:38 2.33
the '--debug-init' option to view a complete error backtrace.

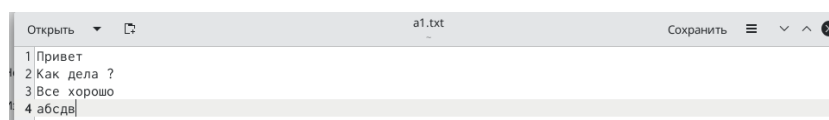
U:%*- *Warnings* Bot L8 (Special) Чт мая 19 14:38 2.33
Wrote /afs/.dk.sci.pfu.edu.ru/home/e/v/evkurkina/prog1.sh
```

Рис. 3.3: Сохранение файла



```
1 У меня ве отлично
2 Как твои дела?
3 trrrgrrrr
```

Рис. 3.4: Текст первого файла



```
1 Привет
2 Как дела ?
3 Все хорошо
4 абсдв
```

Рис. 3.5: текст второго файла



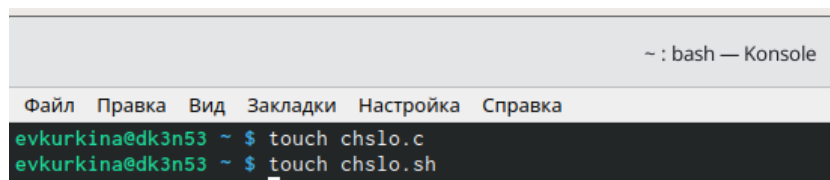
```
evkurkina@dk3n53 ~ $ touch prog1.sh
evkurkina@dk3n53 ~ $ emacs &
[1] 17433
evkurkina@dk3n53 ~ $ touch a1.txt a2.txt
evkurkina@dk3n53 ~ $ chmod +x prog1.sh
evkurkina@dk3n53 ~ $ cat a1.txt
Привет
Как дела ?
Все хорошо
абсдв
evkurkina@dk3n53 ~ $ cat a1.txt
```

Рис. 3.6: Права доступа

```
evkurkina@dk3n53 ~ $ ./prog1.sh -i a1.txt -o a2.txt -p дела -C -n
evkurkina@dk3n53 ~ $ cat a2.txt
2:Как дела ?
evkurkina@dk3n53 ~ $ ./prog1.sh -i a1.txt -o a2.txt -p дела -n
evkurkina@dk3n53 ~ $ cat a2.txt
2:Как дела ?
evkurkina@dk3n53 ~ $ ./prog1.sh -i a1.txt -C -n
./prog1.sh: строка 13: ((: ==0: синтаксическая ошибка: ожидается операнд (неверный маркер «==0»)
^C
evkurkina@dk3n53 ~ $ ./prog1.sh -i a1.txt -o a2.txt -p дела -n
evkurkina@dk3n53 ~ $ cat a2.txt
2:Как дела ?
evkurkina@dk3n53 ~ $ ./prog1.sh -i a1.txt -o a2.txt -p дела -C -n
evkurkina@dk3n53 ~ $ cat a2.txt
2:Как дела ?
evkurkina@dk3n53 ~ $ ./prog1.sh -i a1.txt -C -n
Шаблон не найден
evkurkina@dk3n53 ~ $ ./prog1.sh -i a1.txt -C -n
Шаблон не найден
evkurkina@dk3n53 ~ $ ./prog1.sh -o a2.txt -p дела -C -n
Файл не найден
evkurkina@dk3n53 ~ $
```

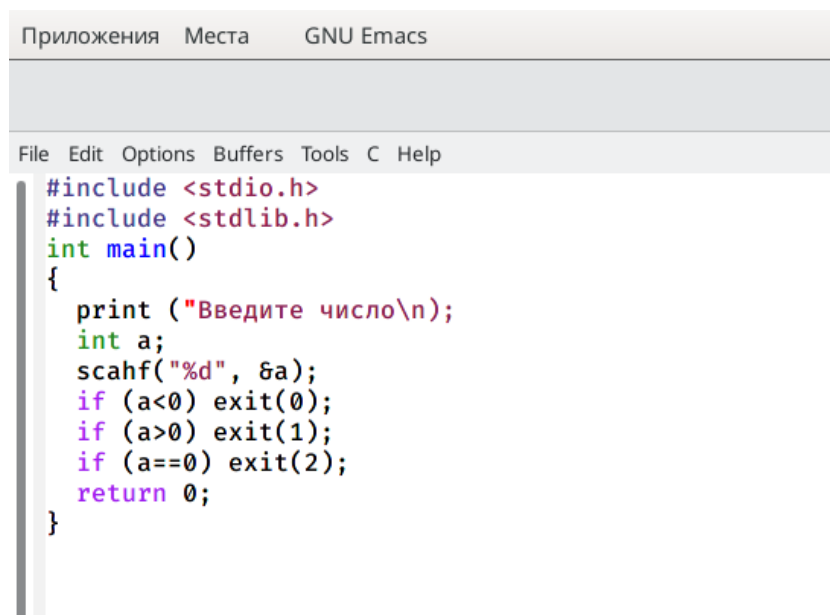
Рис. 3.7: Проверка работы скрипта

2) Создала два новых файла, для второго скрипта (рис. 3.8). На языке С написала программу, которая определяет, введенное число меньше, больше или же равно нулю, затем код завершается с помощью функции `exit(i)` (рис. 3.9), сохранила его (рис. 3.10). После командный файл вызывает данную программу и выдает сообщение о том, какое число было введено пользователем (рис. 3.11). Перешла в консоль, дала права доступа и проверила работу кода (рис. 3.12).



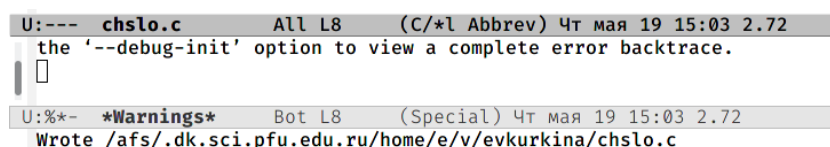
```
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
evkurkina@dk3n53 ~ $ touch chslo.c
evkurkina@dk3n53 ~ $ touch chslo.sh
```

Рис. 3.8: Создана два файла



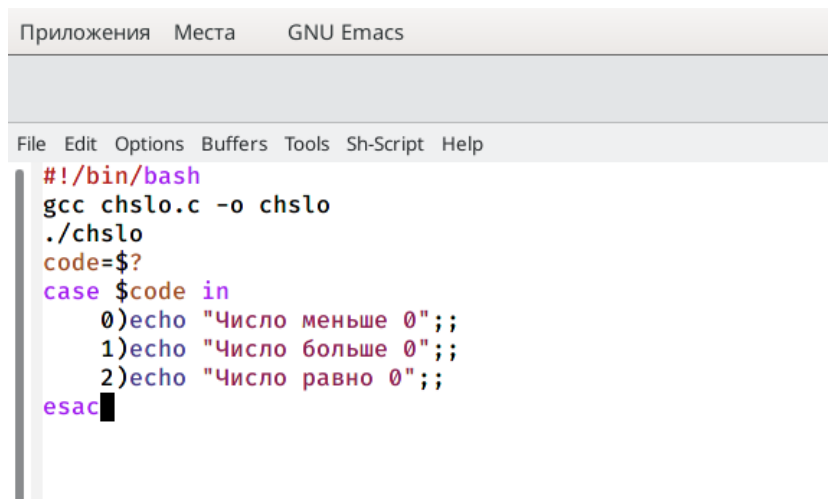
```
Приложения  Места  GNU Emacs
File Edit Options Buffers Tools C Help
#include <stdio.h>
#include <stdlib.h>
int main()
{
    print ("Введите число\n");
    int a;
    scanf ("%d", &a);
    if (a<0) exit(0);
    if (a>0) exit(1);
    if (a==0) exit(2);
    return 0;
}
```

Рис. 3.9: Код программы на языке C



```
U:--- chslo.c All L8 (C/*l Abbrev) Чт мая 19 15:03 2.72
the '--debug-init' option to view a complete error backtrace.
U:%*- *Warnings* Bot L8 (Special) Чт мая 19 15:03 2.72
Wrote /afs/.dk.sci.pfu.edu.ru/home/e/v/evkurkina/chslo.c
```

Рис. 3.10: Сохранение файла

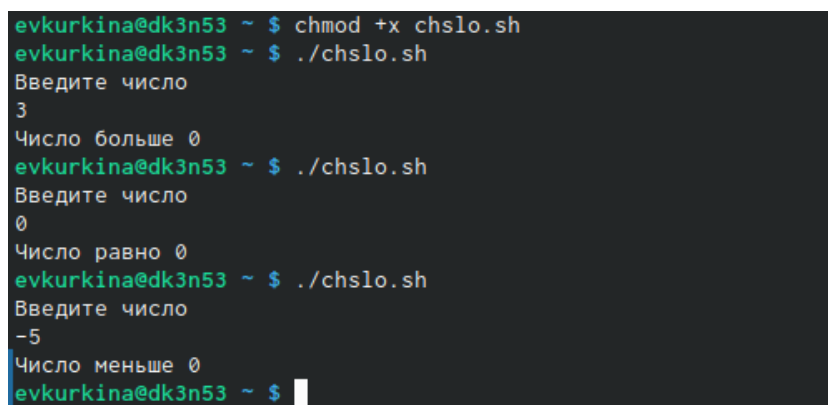


```
Приложения  Места  GNU Emacs

File Edit Options Buffers Tools Sh-Script Help

#!/bin/bash
gcc chslo.c -o chslo
./chslo
code=$?
case $code in
  0)echo "Число меньше 0" ;;
  1)echo "Число больше 0" ;;
  2)echo "Число равно 0" ;;
esac
```

Рис. 3.11: Код исполняемого файла



```
evkurkina@dk3n53 ~ $ chmod +x chslo.sh
evkurkina@dk3n53 ~ $ ./chslo.sh
Введите число
3
Число больше 0
evkurkina@dk3n53 ~ $ ./chslo.sh
Введите число
0
Число равно 0
evkurkina@dk3n53 ~ $ ./chslo.sh
Введите число
-5
Число меньше 0
evkurkina@dk3n53 ~ $
```

Рис. 3.12: права доступа и результат работы скрипта

- 3) Создала новый файл для 3его скрипта (рис. 3.13). Написала код исполняемого файла, который создает указанное пользователем число файлов, которые пронумерованы от 1 до N. Число файлов передается аргументом в командной строке, также этот скрипт может удалить все созданные им файлы(рис. 3.14). Далее, я предоставила доступ к исполнению, создала три файла, а затем удалила их , с помощью кода.(рис. 3.15)

```

~ : bash — Kc
Файл  Правка  Вид  Закладки  Настройка  Справка
evkurkina@dk3n53 ~ $ touch files.sh
evkurkina@dk3n53 ~ $ emacs &
[1] 25396
evkurkina@dk3n53 ~ $ █

```

Рис. 3.13: Создание файлов и переход в emacs

```

emacs@dk3n53
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
opt=$1;
format=$2;
number=$3;
function Files()
{
    for (( i=1; i<=$number; i++ )) do
        file=$(echo $format | tr '#' "$i")
        if [ $opt == "-r" ]
        then
            rm -f $file
        elif [ $opt == "-c" ]
        then
            touch $file
        fi
    done
}
Files█

```

Рис. 3.14: Код программы

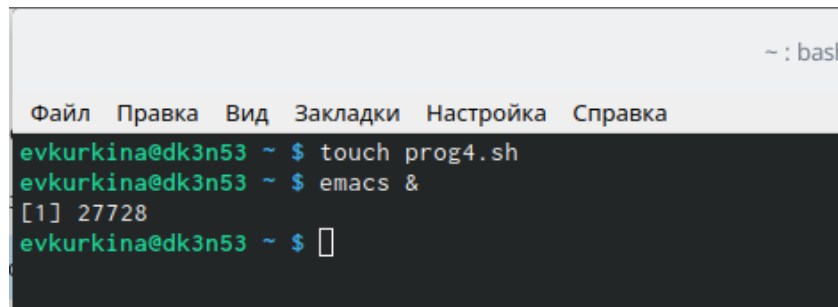
```

evkurkina@dk3n53 ~ $ chmod +x files.sh
evkurkina@dk3n53 ~ $ ./files.sh -c abc#.txt 3
evkurkina@dk3n53 ~ $ ls
a1.txt      blog2.sh~   example.txt~  '#lab07.sh#'  prog2.sh      text.txt
a2.txt      chslo       file254       lab07.sh~     prog2.sh~     tmp
abc1        chslo.c~   file254.txt  'Matrisa(54)'  progals.sh    vozrast
abc1.txt    chslo.c~   file2.doc     may           program1      work
abc2.txt    chslo.sh~  file.doc      monthly      program12     Видео
abc3.txt    chslo.sh~  file.pdf      my_os        Programploshad  Документы
Architecture_PC  conf.txt   files.sh      os-intro     programsum     Загрузки
australia   evkurkina.github.io  files.sh~    play         progl.s       Изображения
backup      example2.txt~  file.txt     poil         program2      Музыка
backup.sh   example2.txt~  format.sh    presentation.md  public        Общеизвестные
backup.sh~  example3.txt~  format.sh~   presentation.pdf  public_html   препрограмма
bin         example4.txt~  GNUstep      prog1.sh      reports
blog        example.txt   lab01        prog1.sh~     ski.plases
blog2.sh    example.txt   lab01        prog1.sh~     temp
evkurkina@dk3n53 ~ $ ./files.sh -r abc#.txt 3
evkurkina@dk3n53 ~ $ ls
a1.txt      chslo.c~   feathers      '#lab07.sh#'  prog1.sh~     ski.plases
a2.txt      chslo.c~   file254       lab07.sh~     prog2.sh      temp
abc1        chslo.sh~  file254.txt  'Matrisa(54)'  prog2.sh~     text.txt
abc1.txt    chslo.sh~  file2.doc     may           progals.sh    tmp
Architecture_PC  conf.txt   file.doc      monthly      program1      vozrast
australia     evkurkina.github.io  file.pdf      my_os        program12     work
backup         example2.txt~  files.sh      os-intro     Programploshad  Видео
backup.sh      example2.txt~  file.txt     poil         programsum     Документы
backup.sh~     example3.txt~  format.sh    play         progl.s       Изображения
bin            example4.txt~  format.sh~   presentation.md  public        Музыка
blog           example.txt   GNUstep      presentation.pdf  public_html   Общеизвестные
blog2.sh       example.txt   lab01        prog1.sh      reports       препрограмма
chslo          example.txt~  lab01        prog1.sh~     temp
evkurkina@dk3n53 ~ $ █

```

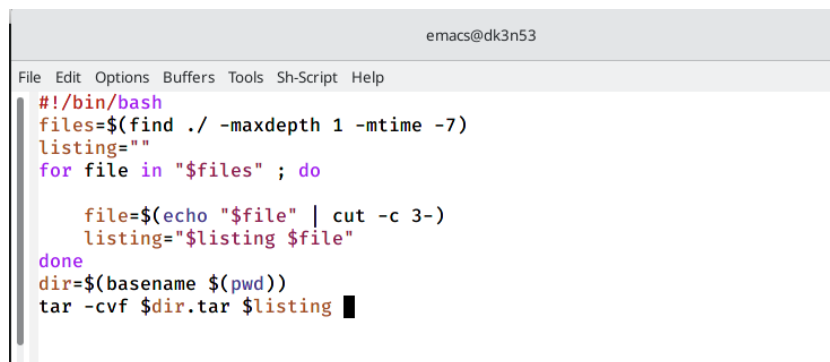
Рис. 3.15: Права доступа и проверка работы скрипта

- 4) Создала файл для последнего 4го скрипта(рис. 3.16). Написала текст исполняемого файла, который с помощью команды tar собирает в архив все файлы, которые находятся в этой директории.(рис. 3.17). Далее предоставила права доступа на выполнение, создала отдельный каталог, куда поместила файлы, а после проверила работу программы (рис. 3.18) (рис. 3.19).



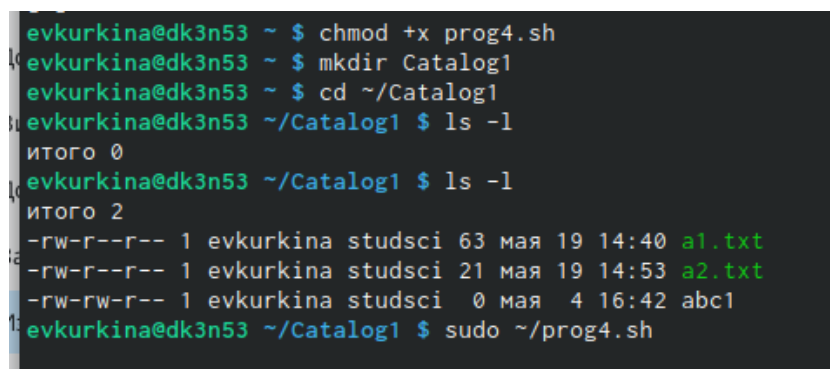
```
~ : bash
Файл Правка Вид Закладки Настройка Справка
evkurkina@dk3n53 ~ $ touch prog4.sh
evkurkina@dk3n53 ~ $ emacs &
[1] 27728
evkurkina@dk3n53 ~ $
```

Рис. 3.16: Создание файла для последнего скрипта



```
emacs@dk3n53
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
files=$(find ./ -maxdepth 1 -mtime -7)
listing=""
for file in "$files" ; do
    file=$(echo "$file" | cut -c 3-)
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
```

Рис. 3.17: Текст кода



```
evkurkina@dk3n53 ~ $ chmod +x prog4.sh
evkurkina@dk3n53 ~ $ mkdir Catalog1
evkurkina@dk3n53 ~ $ cd ~/Catalog1
evkurkina@dk3n53 ~/Catalog1 $ ls -l
итого 0
evkurkina@dk3n53 ~/Catalog1 $ ls -l
итого 2
-rw-r--r-- 1 evkurkina studsci 63 мая 19 14:40 a1.txt
-rw-r--r-- 1 evkurkina studsci 21 мая 19 14:53 a2.txt
-rw-rw-r-- 1 evkurkina studsci 0 мая 4 16:42 abc1
evkurkina@dk3n53 ~/Catalog1 $ sudo ~/prog4.sh
```

Рис. 3.18: Создание каталога и проверка работы скрипта

```
evkurkina@edk3n53 ~/Catalog1 $ ~/prog4.sh
a1.txt
a2.txt
evkurkina@edk3n53 ~/Catalog1 $ tar -tf Catalog1.tar
a1.txt
a2.txt
evkurkina@edk3n53 ~/Catalog1 $
```

Рис. 3.19: Результат работы

#### 5) Ответы на контрольные вопросы:

1). Команда `getopts` осуществляет синтаксический анализ командной строки, выделяя флаги, используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg...]` Флаги – это опции командной строки, обычно помеченные знаком минус; Например, для команды `ls` флагом может являться `-F`. Строка опций `option-string` – это описок возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, то она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введённые данные с помощью оператора `case`. Функция `getopts` включает две специальные переменные среды `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента. Функция `getopts` также понимает переменные типа массив, следовательно, можно использовать её в функции не только для синтаксического анализа аргументов функций, но и для анализа введённых пользователем данных.

2). При перечислении имён файлов текущего каталога можно использовать следующие символы:

- \* – соответствует произвольной, в том числе и пустой строке;
- ? – соответствует любому одинарному символу;

[c1-c2] – соответствует любому символу, лексикографически находящемуся между символами c1 и c2;  
z]\*–соответствует произвольному имени файла в текущем каталоге, начинающемуся с л

3). Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости от результатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути, являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.

4). Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.

5). Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования `bash`: это команда `true`, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда `false`, которая всегда возвращает код завершения, неравный нулю (т.е. ложь). Примеры бесконечных циклов: `while true do echo hello andy done until false do echo hello mike done`.

6). Строка `if test -f mans/i.s, mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).

7). Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла `while` служебного слова `while` на `until` условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.



## 4 Выводы

Во время выполнения данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## **Список литературы**