

# **Лабораторная работа № 12**

**Операционные системы**

Гусева Светлана Алексеевна

# **Содержание**

<b>1 Цель работы</b>	<b>5</b>
<b>2 Задание</b>	<b>6</b>
<b>3 Теоретическое введение</b>	<b>7</b>
<b>4 Выполнение лабораторной работы</b>	<b>9</b>
<b>5 Выводы</b>	<b>16</b>
<b>6 Контрольные вопросы</b>	<b>17</b>
<b>Список литературы</b>	<b>19</b>

## **Список таблиц**

# **Список иллюстраций**

4.1 Результат выполнения задания 1. . . . .	9
4.2 Скрипт задания 1. . . . .	10
4.3 Проверка установки gcc. . . . .	10
4.4 Результат выполнения задания 2. . . . .	11
4.5 Программа на C. . . . .	11
4.6 Командный файл для задания 2. . . . .	12
4.7 Командный файл для задания 3. . . . .	12
4.8 Результат выполнения задания 3. . . . .	13
4.9 Результат выполнения задания 4 (первая часть). . . . .	13
4.10 Скрипт задания 4 (первая часть). . . . .	14
4.11 Результат выполнения задания 4 (вторая часть). . . . .	14
4.12 Скрипт задания 4 (вторая часть). . . . .	15

# **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. Используя команды getopt grep, написать командный файл, который анализирует командную строку с ключами: – -iinputfile – прочитать данные из указанного файла; – -ooutputfile – вывести данные в указанный файл; – -ршаблон – указать шаблон для поиска; – -С – различать большие и малые буквы; – -н – выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом -р.
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.
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).
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).

### 3. Теоретическое введение

Команда getopt осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий:

```
getopt option-string variable [arg ... ]
```

Флаги — это опции командной строки, обычно помеченные знаком минус; Например, для команды ls флагом может являться -F. Иногда флаги имеют аргументы, связанные с ними. Программы интерпретируют флаги, соответствующим образом изменяя своё поведение.

Строка опций option-string — это список 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 getopt может распознать аргумент, то она возвращает истину. Принято включать getopt в цикл while и анализировать введённые данные с помощью оператора case.

Предположим, необходимо распознать командную строку следующего формата:

```
testprog -ifile_in.txt -ofile_out.doc -L -t -r
```

Вот как выглядит использование оператора getopt в этом случае:

```
while getopt o:i:Ltr optletter  
do case $optletter in
```

```
o) oflag=1; oval=$OPTARG;;
i) iflag=1; ival=$OPTARG;;
L) Lflag=1;;
t) tflag=1;;
r) rflag=1;;
*) echo Illegal option $optletter
esac

done
```

Функция getopt включает две специальные переменные среды – OPTARG и OPTIND. Если ожидается дополнительное значение, то OPTARG устанавливается в значение этого аргумента (будет равна file\_in.txt для опции i и file\_out.doc для опции o). OPTIND является числовым индексом на упомянутый аргумент. Функция getopt 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.

## **4. Выполнение лабораторной работы**

1. Используя команды getopt grep, написать командный файл, который анализирует командную строку с ключами: -iinputfile – прочитать данные из указанного файла; -ooutputfile – вывести данные в указанный файл; -rшаблон – 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. Результат работы командного файла (рис. 4.1), скрипт командного файла (рис. 4.2).

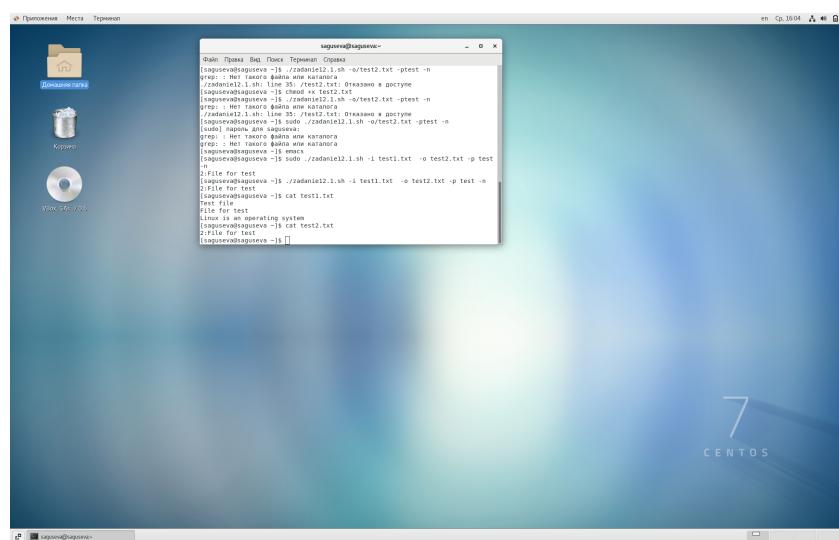


Рис. 4.1: Результат выполнения задания 1.

```
#!/bin/bash -x
#(Ubuntu-Meego-Emacs)
#(zadanie12.1.sh - emacs@zadaniaweb.localdomain)

File Edit Options Buffer Tools Sh-Script Help
File Edit Options Buffer Tools Sh-Script Help
zadanie12.1.sh  All 11  (Shell-scriptish)
Indentation setup for shell-type sh

#!/bin/bash
#(zadanie12.1.sh - emacs@zadaniaweb.localdomain)

while getopts l:opt; do
    case $opt in
        l) fileout=$OPTARG;;
        o) filein=$OPTARG;;
        *) echo "Usage: $0 [-l] [-o] [filein] [fileout]" >> /dev/stderr;
            exit 1;;
    esac
done
if [ -n "$filein" ]
then
    if [ -n ${register} ]
    then
        if [ -n ${number} ]
        then
            grep -n "${pattern}" "$filein"
        else
            grep "${pattern}" "$filein"
        fi
    else
        if [ -n ${number} ]
        then
            grep -n "${pattern}" "$filein"
        else
            grep -i "${pattern}" "$filein"
        fi
    fi
fi
if [ -n "$fileout" ]
then
    if [ -n ${register} ]
    then
        if [ -n ${number} ]
        then
            grep -n "${pattern}" "$filein" > "$fileout"
        else
            grep "${pattern}" "$filein" > "$fileout"
        fi
    else
        if [ -n ${number} ]
        then
            grep -n "${pattern}" "$filein" > "$fileout"
        else
            grep -i "${pattern}" "$filein" > "$fileout"
        fi
    fi
fi
fi
```

Рис. 4.2: Скрипт задания 1.

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. Проверка установки gcc (рис. 4.3), результат выполнения (рис. 4.4), программа на С (рис. 4.5), командный файл (рис. 4.6).

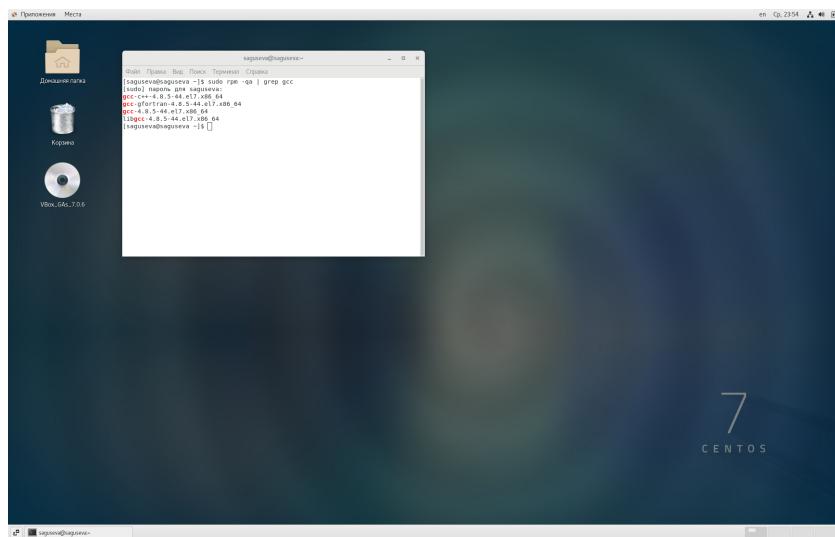


Рис. 4.3: Проверка установки gcc.

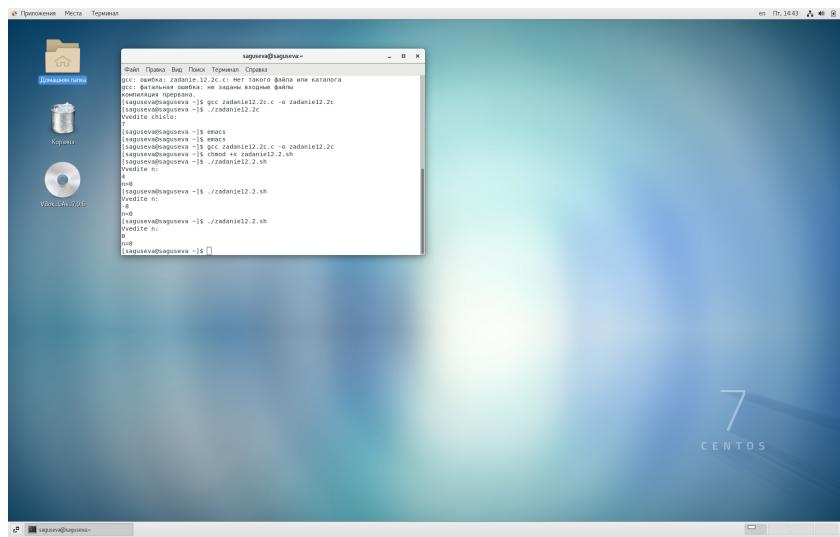


Рис. 4.4: Результат выполнения задания 2.

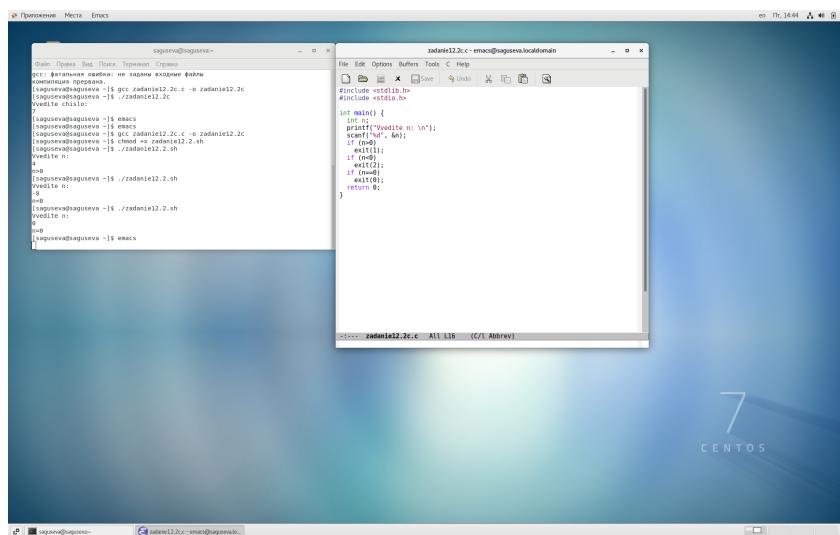


Рис. 4.5: Программа на С.

```

sagurova@zagonov:~$ gcc -o zadanie12_2c zadanie12_2c.c
sagurova@zagonov:~$ ./zadanie12_2c
1
2
3

```

Рис. 4.6: Командный файл для задания 2.

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). Командный файл (рис. 4.7), результат выполнения задания (рис. 4.8).

```

sagurova@zagonov:~$ ./zadanie12_3.sh
1
2
3
4
5
6
7
8
9
10
rm: cannot remove '10.tmp': No such file or directory
rm: cannot remove '9.tmp': No such file or directory
rm: cannot remove '8.tmp': No such file or directory
rm: cannot remove '7.tmp': No such file or directory
rm: cannot remove '6.tmp': No such file or directory
rm: cannot remove '5.tmp': No such file or directory
rm: cannot remove '4.tmp': No such file or directory
rm: cannot remove '3.tmp': No such file or directory
rm: cannot remove '2.tmp': No such file or directory
rm: cannot remove '1.tmp': No such file or directory

```

Рис. 4.7: Командный файл для задания 3.

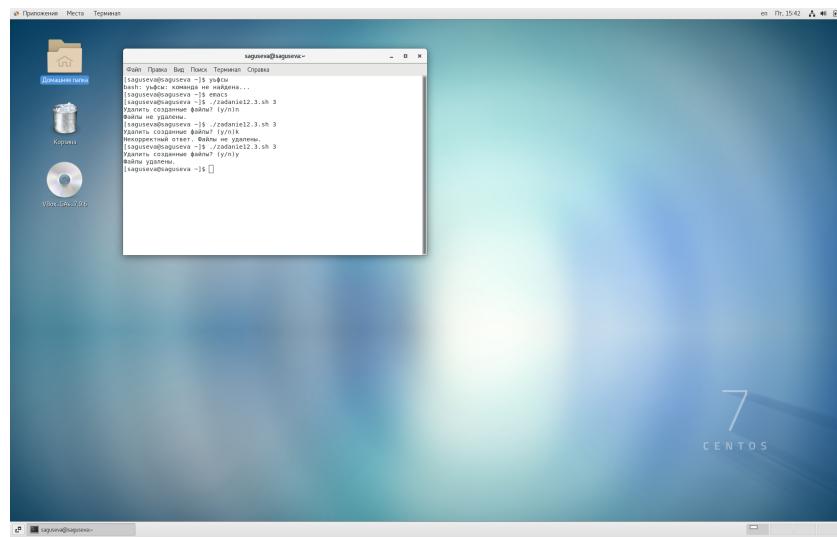


Рис. 4.8: Результат выполнения задания 3.

4. Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Результат выполнения задания (рис. 4.9), скрипт (рис. 4.10).

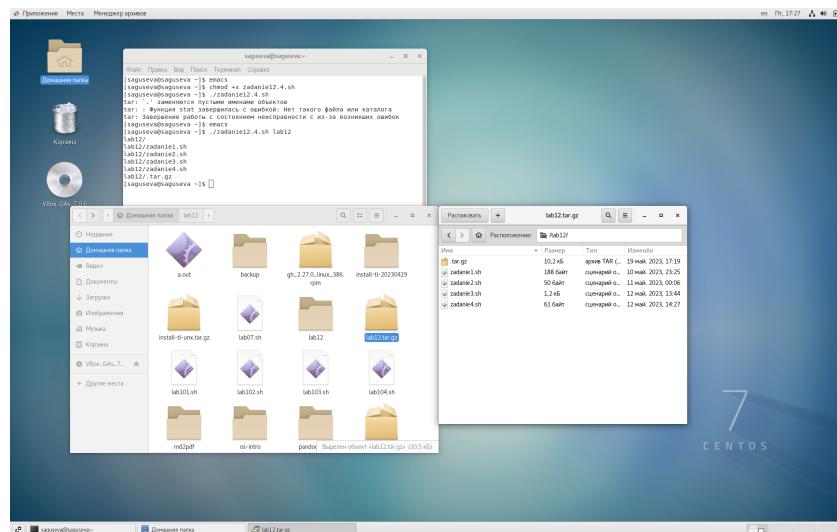


Рис. 4.9: Результат выполнения задания 4 (первая часть).

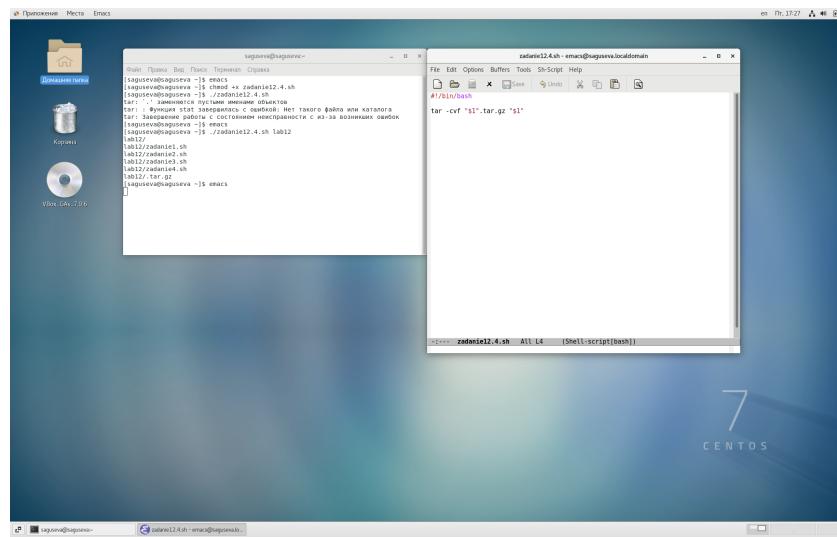


Рис. 4.10: Скрипт задания 4 (первая часть).

Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find). Результат выполнения задания (рис. 4.11), скрипт (рис. 4.12).

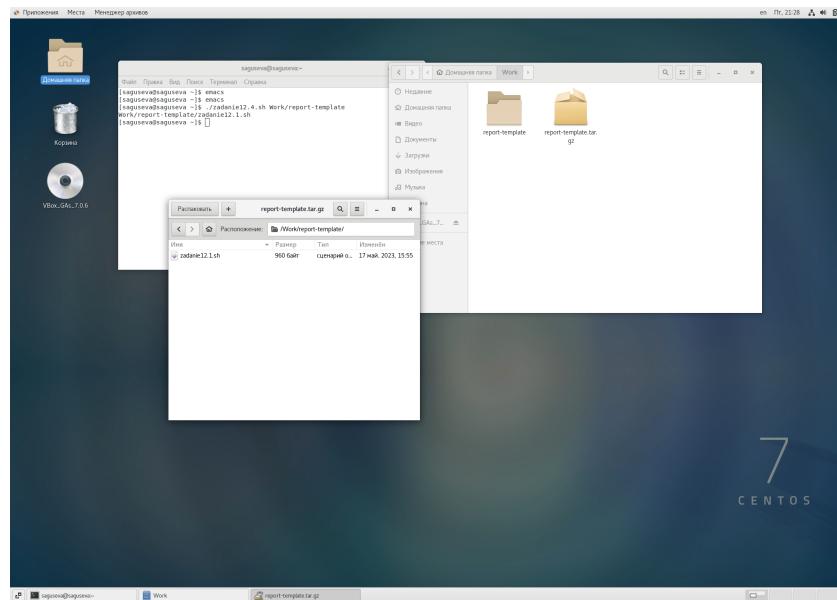


Рис. 4.11: Результат выполнения задания 4 (вторая часть).

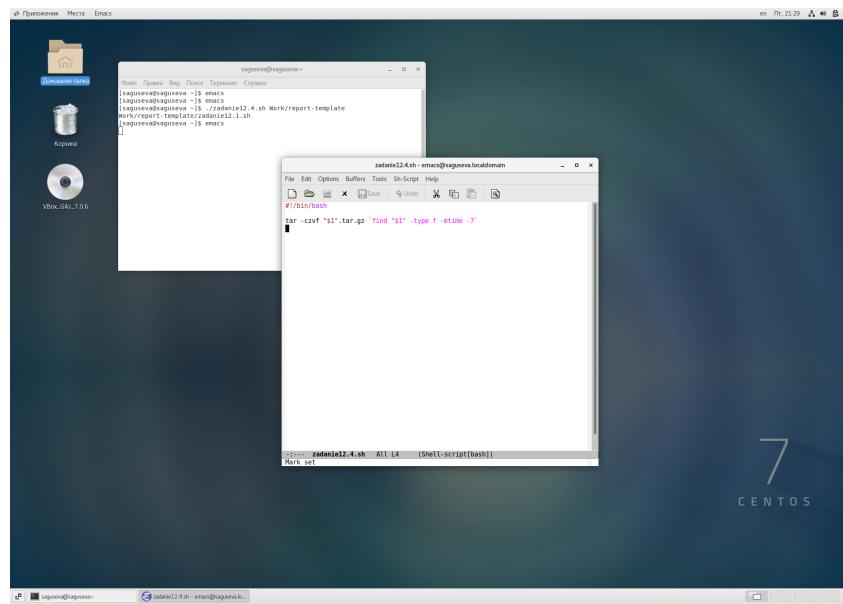


Рис. 4.12: Скрипт задания 4 (вторая часть).

## **5. Выводы**

В ходе выполнения лабораторной работы мной были изучены основы программирования в оболочке ОС UNIX. Я научилась писать более сложные командные файлы с использованием логических управляемых конструкций и циклов.

## 6. Контрольные вопросы

1. Каково предназначение команды getopt?

Команда getopt является встроенной командой командной оболочки bash, предназначеннной для разбора параметров сценариев. Она обрабатывает исключительно однобуквенные параметры как с аргументами, так и без них и этого вполне достаточно для передачи сценариюм любых входных данных.

2. Какое отношение метасимволы имеют к генерации имён файлов?

После всех подстановок в каждом слове команды ищутся символы \*, ?, и [. Если находится хотя бы один из них, то это слово рассматривается как шаблон имен файлов и заменяется именами файлов, удовлетворяющих данному шаблону (в алфавитном порядке). Если ни одно имя файла не удовлетворяет шаблону, то он остается неизменным. Значения указанных символов:

- любая строка, включая и пустую

? один любой символ

[...] любой из указанных между ними символов.

Пара символов, разделенных знаком -, означает любой символ, который находится между ними, включая и их самих. Если первым символом после [” идет “!”, то указанные символы не должны входить в имя файла

3. Какие операторы управления действиями вы знаете?

Оператор условия if/else; операторы цикла for, while, until; оператор выбора case; оператор выхода из цикла break; оператор перехода к следующей итерации цикла continue.

4. Какие операторы используются для прерывания цикла?

Break, continue.

5. Для чего нужны команды false и true?

False и true возвращают соответственно 0 и 1. Используются для проверки условий. 6. Что означает строка if test -f mans/i.\$s, встреченная в командном файле?

Проверка существования файла с именем mans/i.\$s, \$s и \$i - переменные, заданные в командном файле.

7. Объясните различия между конструкциями while и until.

При использовании while код выполняется, пока условие не станет ложным, при использовании until - пока условие не станет истинным.

## **Список литературы**

::: <https://linux-faq.ru/page/komanda-getopts>  
<https://www.linuxlib.ru/shell/gl1.htm?ysclid=lhtkde2kyu465851927>  
:::