

Отчёт о выполнении лабораторной работы №12

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Российский Университет Дружбы Народов

Факультет Физико-Математических и Естественных Наук

Дисциплина: *Операционные системы*

Работу выполняла: Живцова Анна

1032201673

НКН68-01-20

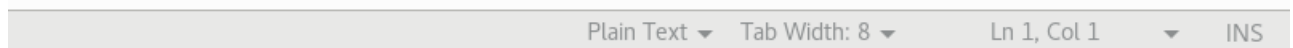
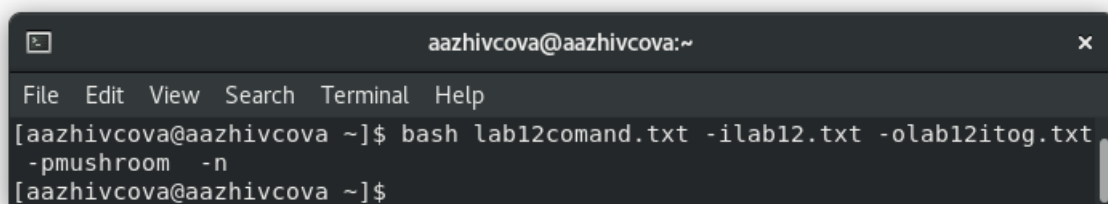
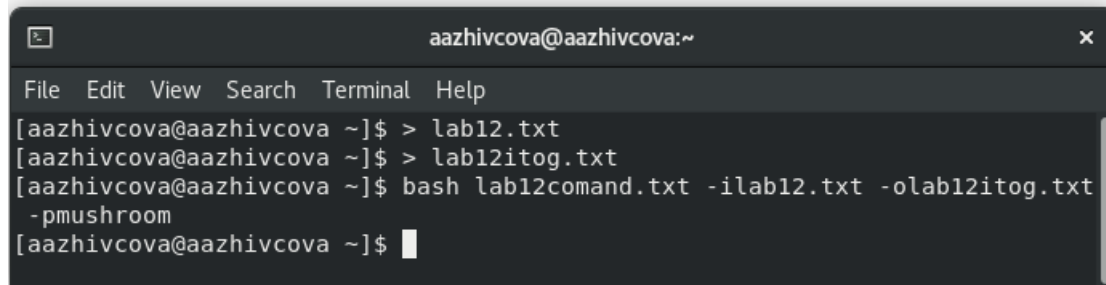
Москва. Дисплейный класс РУДН. 2021г.

Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

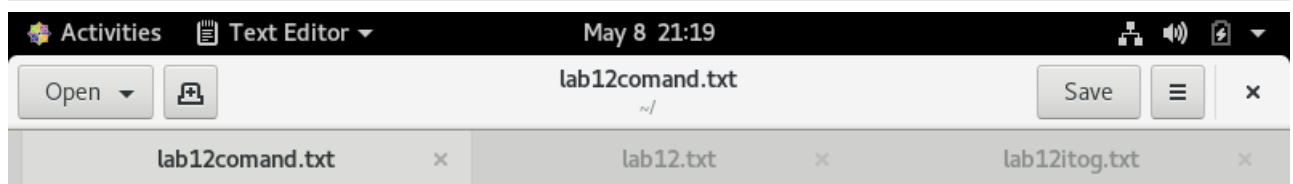
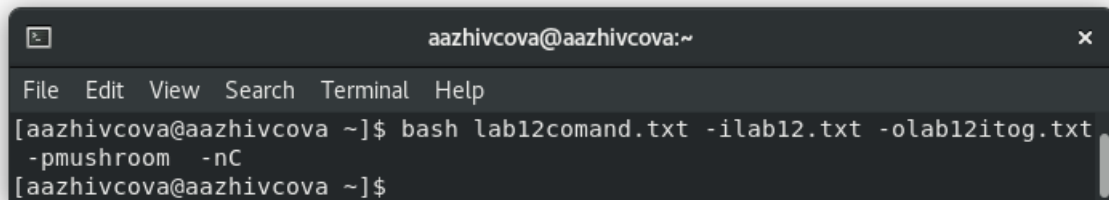
Выполнение работы

1. Используя команды `getopts` `grep`, написала 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; `-п` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-р`. (см рисунки ниже [командный файл в действии 1](#) [командный файл в действии 2](#) [командный файл в действии 3](#) [командный файл исходный текст](#))

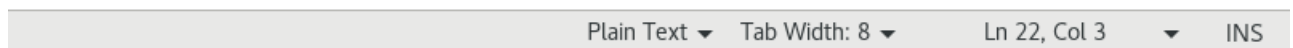


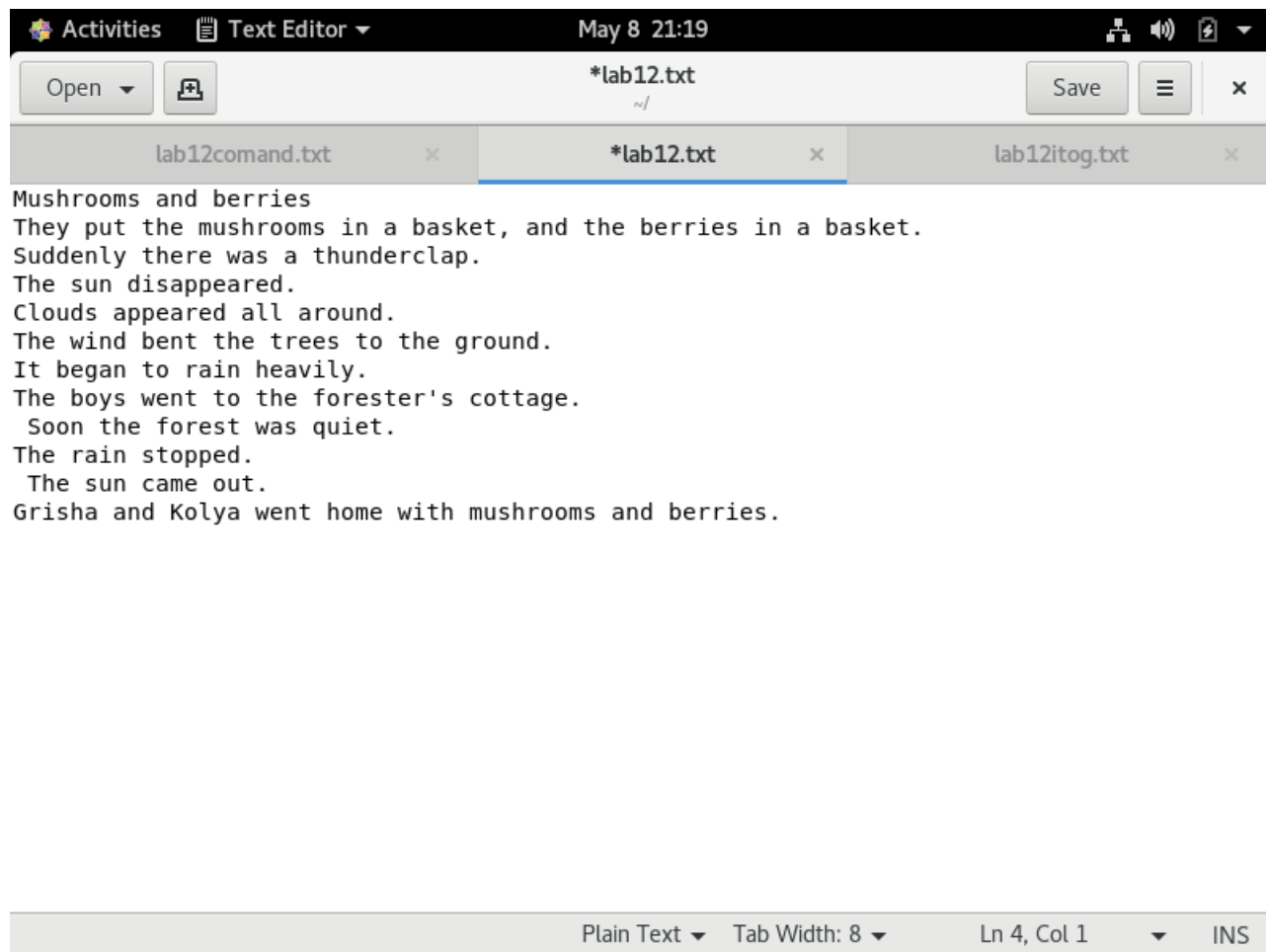


2:They put the mushrooms in a basket, and the berries in a basket.
12:Grisha and Kolya went home with mushrooms and berries.

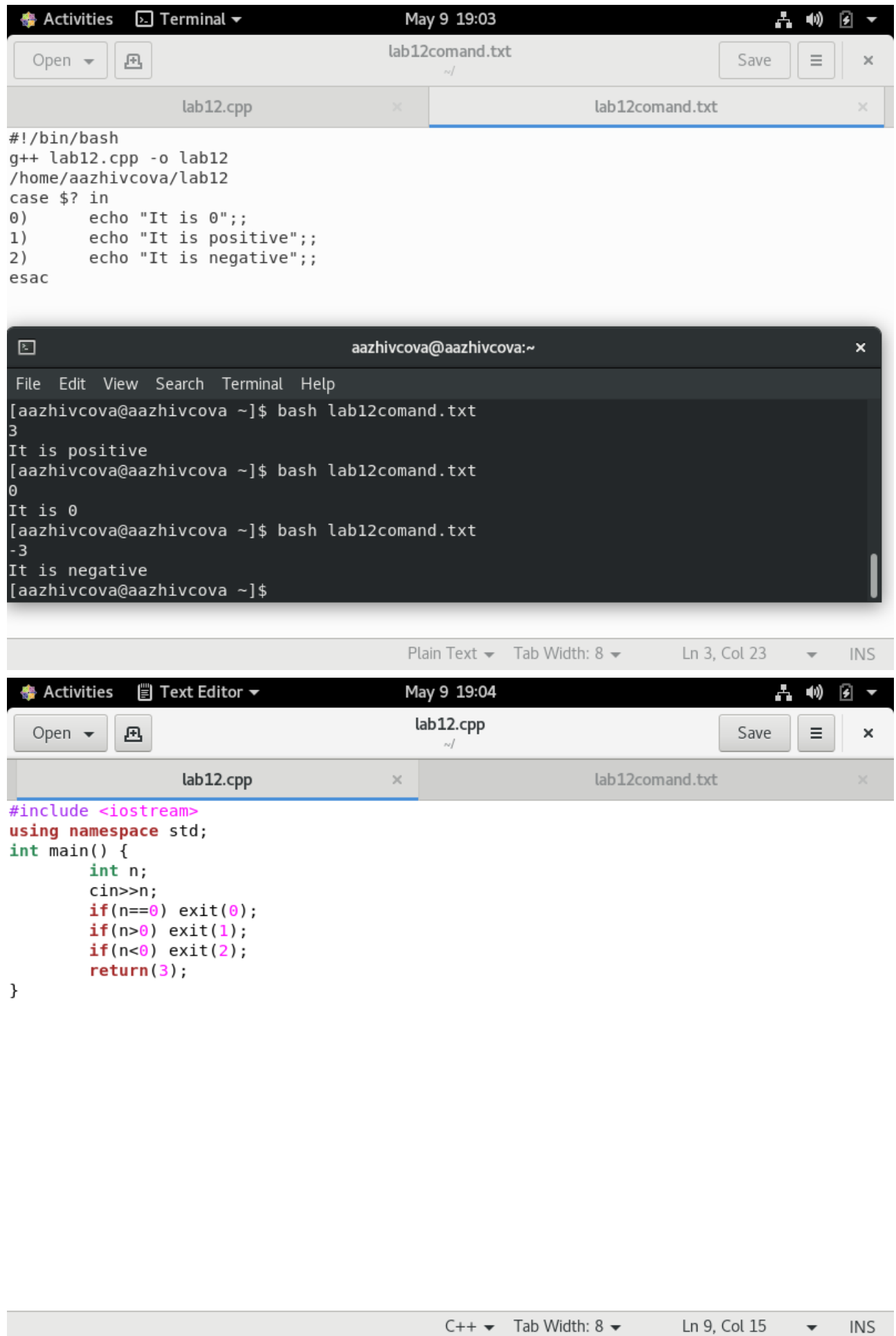


```
while getopts i:o:p:Cn var
do case $var in
i)      iflag=1;          ival=$OPTARG;;
o)      oflag=1;          oval=$OPTARG;;
p)      pflag=1;          pval=$OPTARG;;
C)      Cflag=1;;
n)      nflag=1;;
*)      echo illegal option $val
esac
done
if (( nflag==1 && Cflag==1 ))
then grep -n "$pval" $ival > $oval
fi
if (( nflag==1 && Cflag==0 ))
then grep -ni "$pval" $ival > $oval
fi
if (( nflag==0 && Cflag==1 ))
then grep "$pval" $ival > $oval
fi
if (( nflag==0 && Cflag==0 ))
then grep -i "$pval" $ival > $oval
fi
```





2. Написала 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 вызывает эту программу и, проанализировав с помощью команды `$?`, выдает сообщение о том, какое число было введено. (см рисунки ниже [командный файл в действии программа](#))



The image shows a Linux desktop environment with two windows open. The top window is a terminal titled "Terminal" with a timestamp of "May 9 19:03". It shows the execution of a script named "lab12comand.txt". The script uses a case statement to check if a number is positive, zero, or negative. The bottom window is a text editor titled "Text Editor" with a timestamp of "May 9 19:04". It shows the source code of a C++ program named "lab12.cpp".

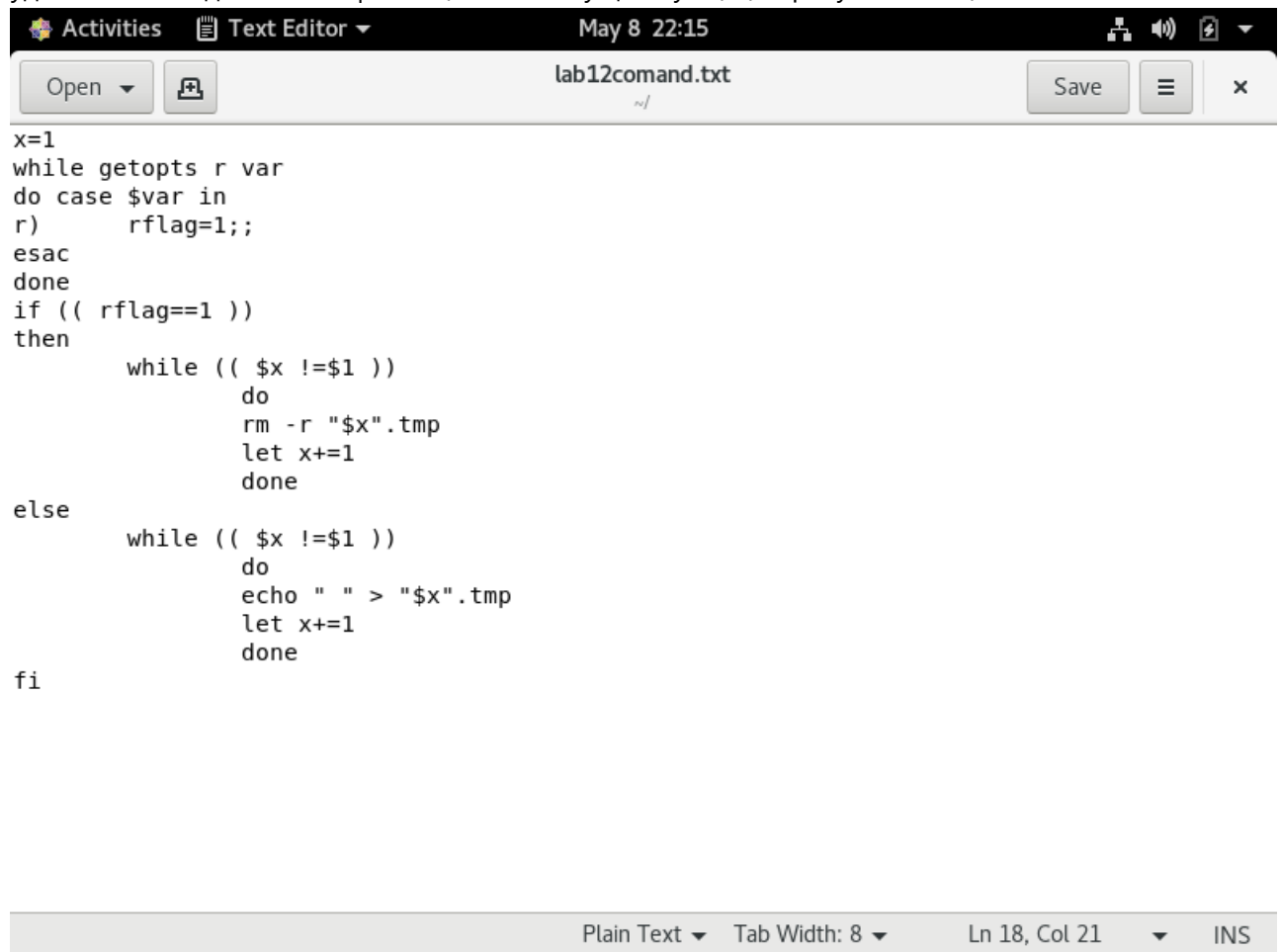
```
#!/bin/bash
g++ lab12.cpp -o lab12
/home/aazhivcova/lab12
case $? in
0)      echo "It is 0";;
1)      echo "It is positive";;
2)      echo "It is negative";;
esac
```

```
aazhivcova@aazhivcova:~
File Edit View Search Terminal Help
[aazhivcova@aazhivcova ~]$ bash lab12comand.txt
3
It is positive
[aazhivcova@aazhivcova ~]$ bash lab12comand.txt
0
It is 0
[aazhivcova@aazhivcova ~]$ bash lab12comand.txt
-3
It is negative
[aazhivcova@aazhivcova ~]$
```

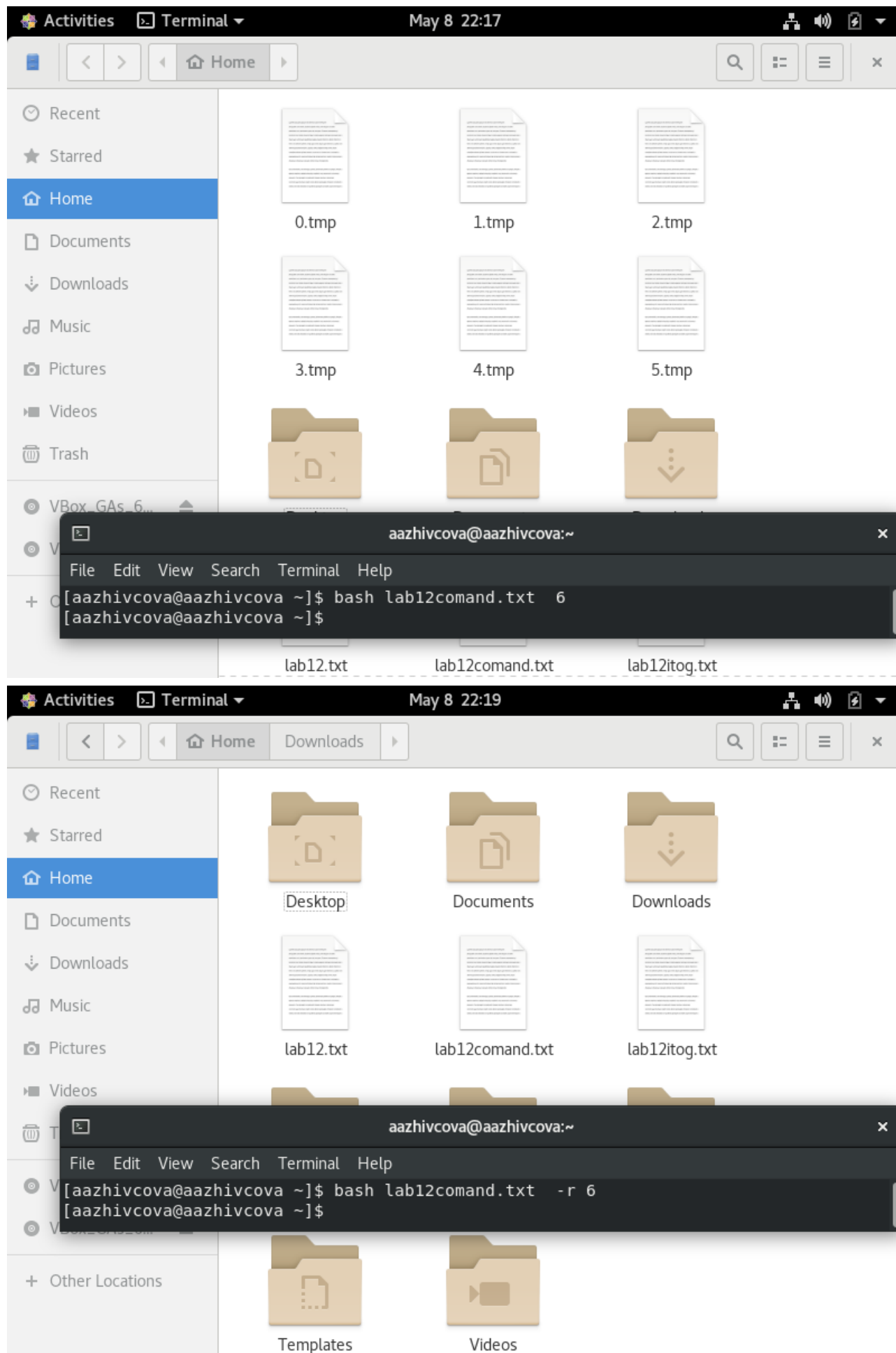
```
#include <iostream>
using namespace std;
int main() {
    int n;
    cin>>n;
    if(n==0) exit(0);
    if(n>0) exit(1);
    if(n<0) exit(2);
    return(3);
}
```

3. Написала 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 умеет удалять все созданные им файлы (если они существуют). (см рисунки ниже)

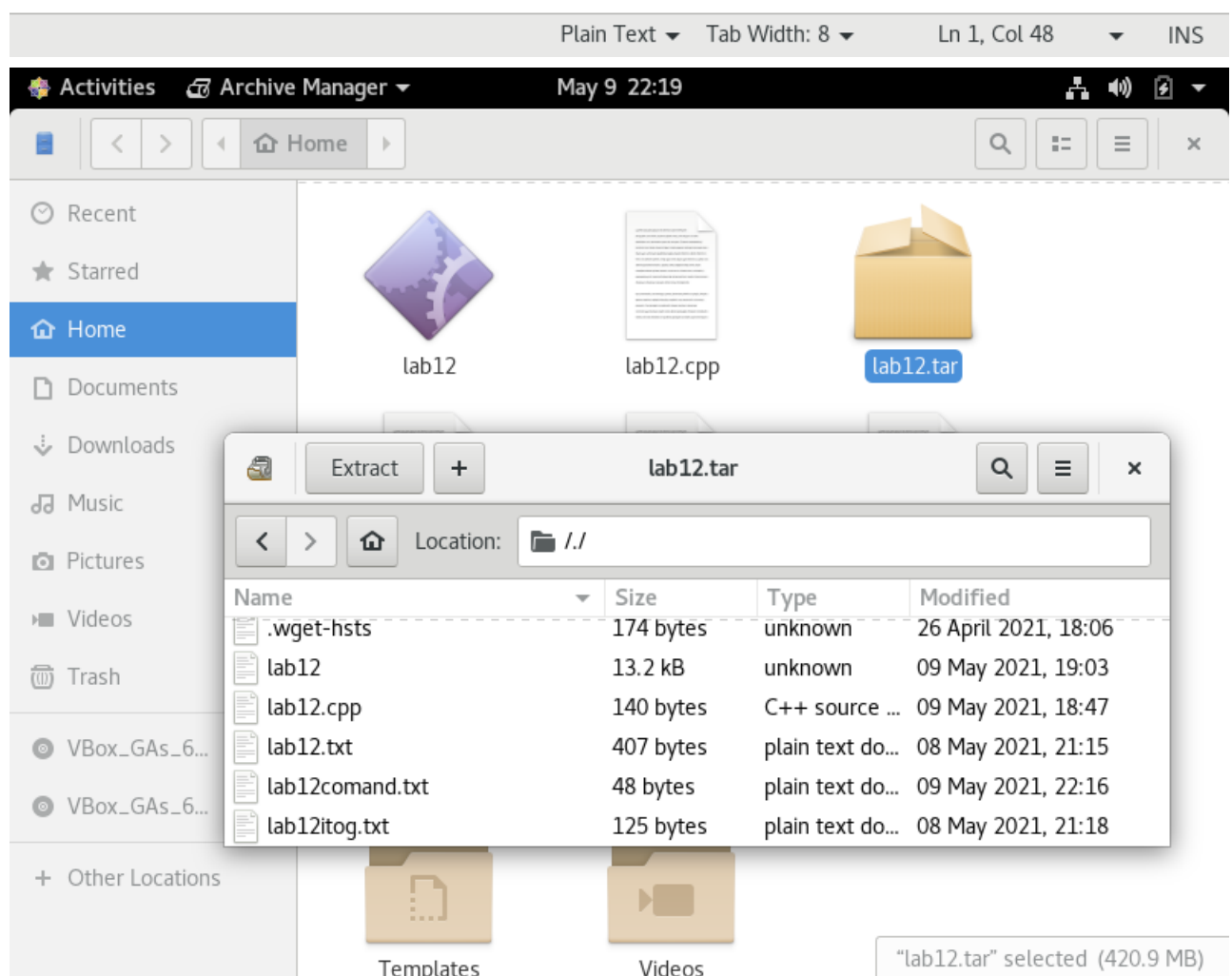
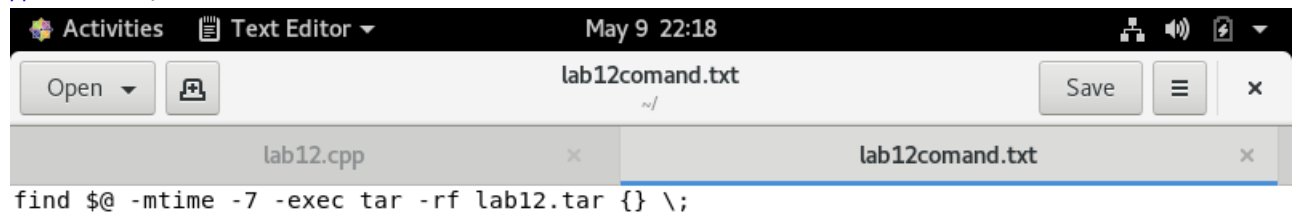


```
x=1
while getopts r var
do case $var in
r)      rflag=1;;
esac
done
if (( rflag==1 ))
then
    while (( $x !=$1 ))
    do
        rm -r "$x".tmp
        let x+=1
    done
else
    while (( $x !=$1 ))
    do
        echo " " > "$x".tmp
        let x+=1
    done
fi
```



4. Написала командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории. запаковываются только те файлы, которые были изменены менее недели

тому назад(использовала команду find). (см рисунки ниже [командный файл](#) [командный файл в действии 1](#))



Контрольные вопросы

1. Весьма необходимой при программировании является команда `getopts`, которая осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg...]`. Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные 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 `getopts` в этом случае:

```
while getopts 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 `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента (будет равно `file_in.txt` для опции `i` и `file_out.doc` для опции `o`). `OPTIND` является числовым индексом на упомянутый аргумент. Функция `getopts` также понимает переменные типа массив, следовательно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.
2. При перечислении имен файлов текущего каталога можно использовать следующие символы:
 - — соответствует произвольной, в том числе и пустой строке;
 - ? — соответствует любому одному символу;
 - [c1-c1] — соответствует любому символу, лексикографически находящемуся между символами c1 и c2.
 - echo * — выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`;
 - ls *.c — выведет все файлы с последними двумя символами, равными `.c`.
 - echo prog.? — выдаст все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.`
 - [a-z]* — соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.
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 Вам 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 дела являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда.

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 перестает быть правильным. Пример бесконечного цикла `while`, с прерыванием в момент, когда файл перестает существовать:

```
while true do if [! -f $file] then break fi sleep 10 done
```

5. 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
6. Введенная строка означает условие существования файла `man$s/$i.$s`
7. Если речь идет о 2-х параллельных действиях, то это `while`. когда мы показываем, что сначала делается 1-е действие. потом оно заканчивается при наступлении 2-го действия, применяем `until`.

Библиография

<https://infopedia.su/24x10498.html>

<https://sgwww.livejournal.com/8836.html>

<https://ipc.susu.ru/44852.html>

Вывод

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