

Лабораторная работа №12 по предмету
Операционные системы

Группа НПМбв-02-19

Нечаева Виктория Алексеевна

Содержание

Цель работы	5
Задание	6
Выполнение лабораторной работы	8
Задача 1	8
Задача 2	10
Задача 3	13
Задача 4	15
Выводы	19
Контрольные вопросы	20

Список таблиц

Список иллюстраций

1	Рисунок 1	9
2	Рисунок 2	10
3	Рисунок 3	11
4	Рисунок 4	12
5	Рисунок 5	13
6	Рисунок 6	14
7	Рисунок 7	15
8	Рисунок 8	16
9	Рисунок 9	16
10	Рисунок 10	17
11	Рисунок 11	18

Цель работы

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

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;
 - `-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 `-r`.
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).

Выполнение лабораторной работы

Задача 1

Сверху задается `getops` с возможными аргументами при запуске программы в терминале. Внизу в блоках `if` проверяется соответствие условиям и `grep` ищет по формату `pattern` подходящие строки. В последнем блоке `if` делается то же самое, но вывод результата происходит в файл `output.txt`


```

GNU nano 6.2
while getopts ":i:o:p:Cn" opt; do
  case ${opt} in
    i ) inputfile=$OPTARG;;
    o ) outputfile=$OPTARG;;
    p ) pattern=$OPTARG;;
    C ) uppercase=1;;
    n ) number=1;;
    \? ) echo "Неверный параметр: -$OPTARG" 1>&2
        exit 1;;
    : ) echo "Параметру -$OPTARG необходим аргумент" 1>&2
        exit 1;;
  esac
done

if [ -n "$inputfile" ]; then
  if [ -n "$uppercase" ]; then
    if [ -n "$number" ]; then
      grep -n "$pattern" "$inputfile"
    else
      grep "$pattern" "$inputfile"
    fi
  else
    if [ -n "$number" ]; then
      grep -ni "$pattern" "$inputfile"
    else
      grep -i "$pattern" "$inputfile"
    fi
  fi
fi

if [ -n "$outputfile" ]; then
  if [ -n "$uppercase" ]; then
    if [ -n "$number" ]; then
      grep -n "$pattern" "$inputfile" > "$outputfile"
    else
      grep "$pattern" "$inputfile" > "$outputfile"
    fi
  else
    if [ -n "$number" ]; then
      grep -ni "$pattern" "$inputfile" > "$outputfile"
    else
      grep -i "$pattern" "$inputfile" > "$outputfile"
    fi
  fi
fi

```

Рис. 1: Рисунок 1

```
vanechaeva@vanechaeva: ~/lab12
one two
two three
two one
two five
two six
vanechaeva@vanechaeva:~/lab12$ nano getopt.sh
vanechaeva@vanechaeva:~/lab12$ ./getopt.sh -i in.txt -o out.txt -p "two" -C -n
1:one two
2:two three
4:two one
5:two five
7:two six
vanechaeva@vanechaeva:~/lab12$ ls
01 getopt.sh  in.txt  out.txt  tar.sh  tmp.sh
vanechaeva@vanechaeva:~/lab12$ cat in.txt
one two
two three
four one
two one
two five
five eight
two six
vanechaeva@vanechaeva:~/lab12$ cat out.txt
1:one two
2:two three
4:two one
5:two five
7:two six
vanechaeva@vanechaeva:~/lab12$
```

Рис. 2: Рисунок 2

Задача 2

Программа запрашивает ввод числа, считывает его, сравнивает согласно условиям и выводит результат. `bash` файл запускает программу на выполнение и `echo &` выводит `pid` который мы задали в программе (берет последнее значение)

The image shows a terminal window with the title bar "vanechaeva@vanechaeva: ~/lab12". Inside the terminal, the GNU nano 6.2 editor is open, editing a file named "start.sh". The editor's content is as follows:

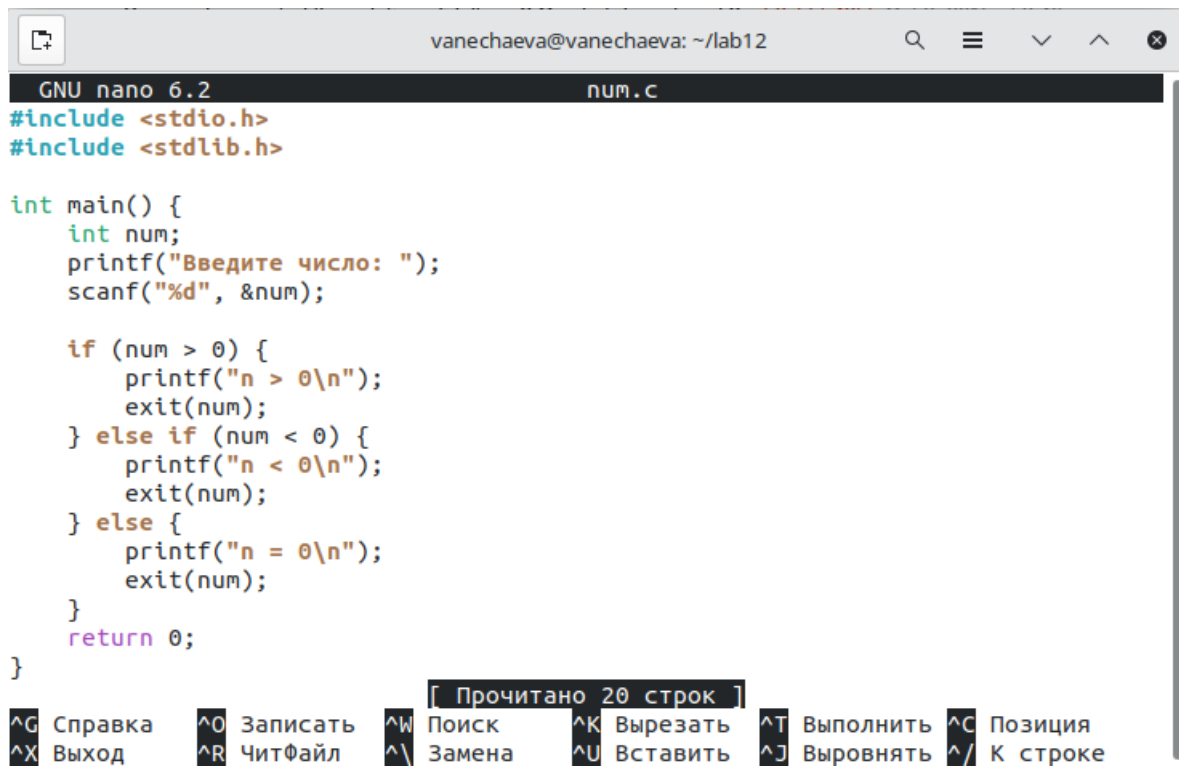
```
GNU nano 6.2 start.sh
#!/bin/bash
./num
echo $?
```

At the bottom of the terminal, a status bar displays the following information:

Прочитано 3 строки

^G Справка	^O Записать	^W Поиск	^K Вырезать	^T Выполнить	^C Позиция
^X Выход	^R ЧитФайл	^_ Замена	^U Вставить	^J Выводить	^/ К строке

Рис. 3: Рисунок 3



```
GNU nano 6.2 num.c
#include <stdio.h>
#include <stdlib.h>

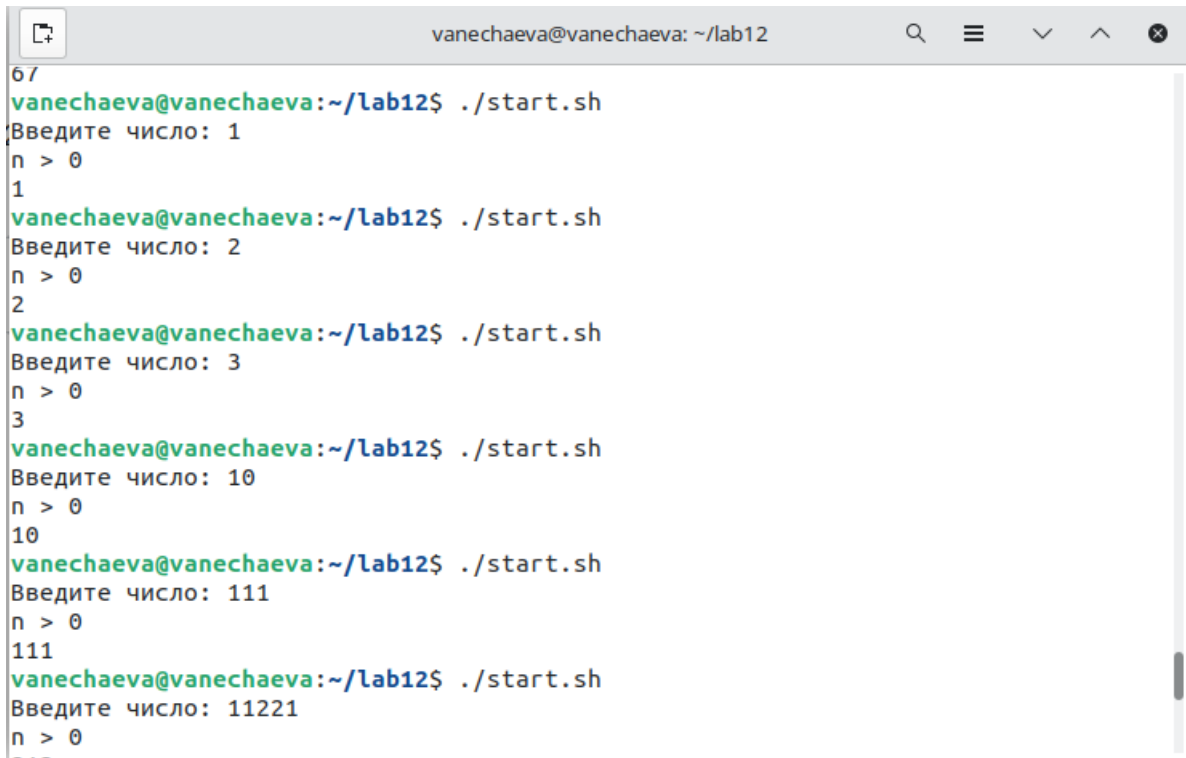
int main() {
    int num;
    printf("Введите число: ");
    scanf("%d", &num);

    if (num > 0) {
        printf("n > 0\n");
        exit(num);
    } else if (num < 0) {
        printf("n < 0\n");
        exit(num);
    } else {
        printf("n = 0\n");
        exit(num);
    }
    return 0;
}
```

[Прочитано 20 строк]

^G Справка	^O Записать	^W Поиск	^K Вырезать	^T Выполнить	^C Позиция
^X Выход	^R ЧитФайл	^\ Замена	^U Вставить	^J Выводить	^/_ К строке

Рис. 4: Рисунок 4

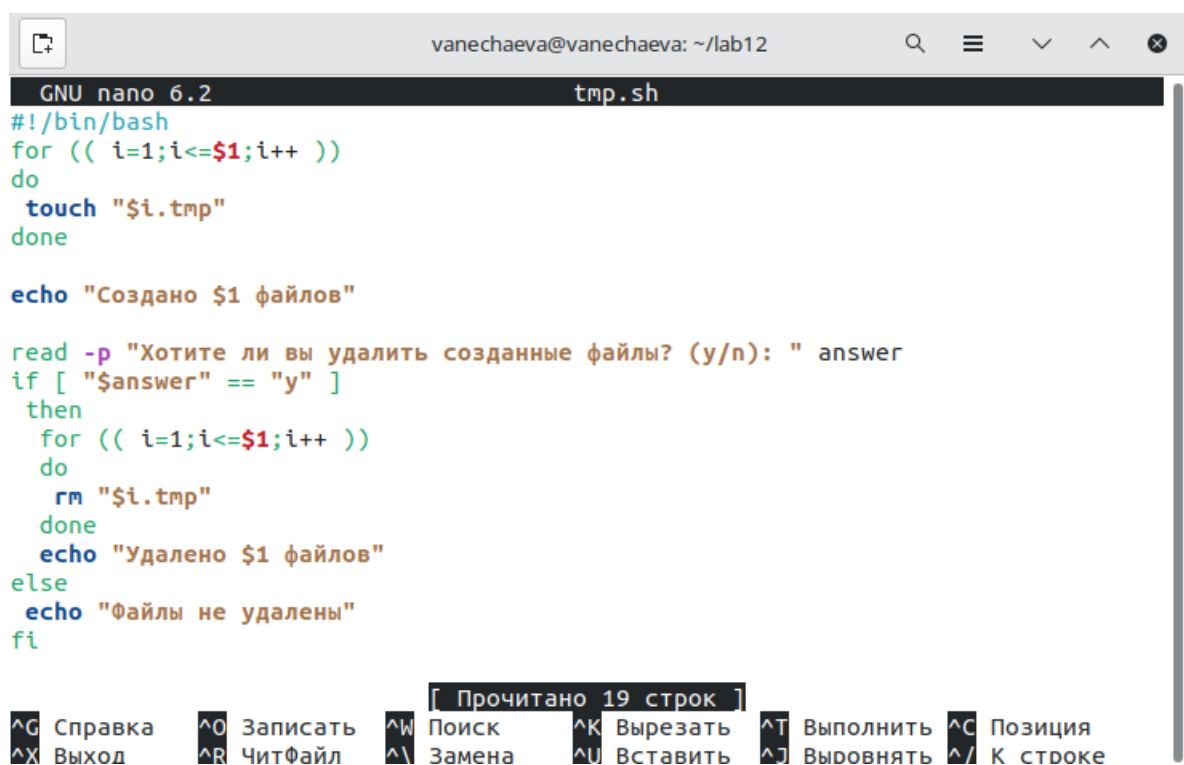


```
67
vanechaeva@vanechaeva:~/lab12$ ./start.sh
Введите число: 1
n > 0
1
vanechaeva@vanechaeva:~/lab12$ ./start.sh
Введите число: 2
n > 0
2
vanechaeva@vanechaeva:~/lab12$ ./start.sh
Введите число: 3
n > 0
3
vanechaeva@vanechaeva:~/lab12$ ./start.sh
Введите число: 10
n > 0
10
vanechaeva@vanechaeva:~/lab12$ ./start.sh
Введите число: 111
n > 0
111
vanechaeva@vanechaeva:~/lab12$ ./start.sh
Введите число: 11221
n > 0
...
```

Рис. 5: Рисунок 5

Задача 3

В аргументе задается количество файлов, которое мы ходим создать. Цикл проходит от 1 до $i = 1$ и создает такое количество файлов. Далее программа спрашивает пользователя, хочет ли он удалить созданные файлы, если пользователь отвечает “у”, то только что созданные файлы удаляются в цикле по одному, если “н” (или любую другую букву), то файлы не удаляются.



```
GNU nano 6.2 tmp.sh
#!/bin/bash
for (( i=1;i<=$1;i++ ))
do
    touch "$i.tmp"
done

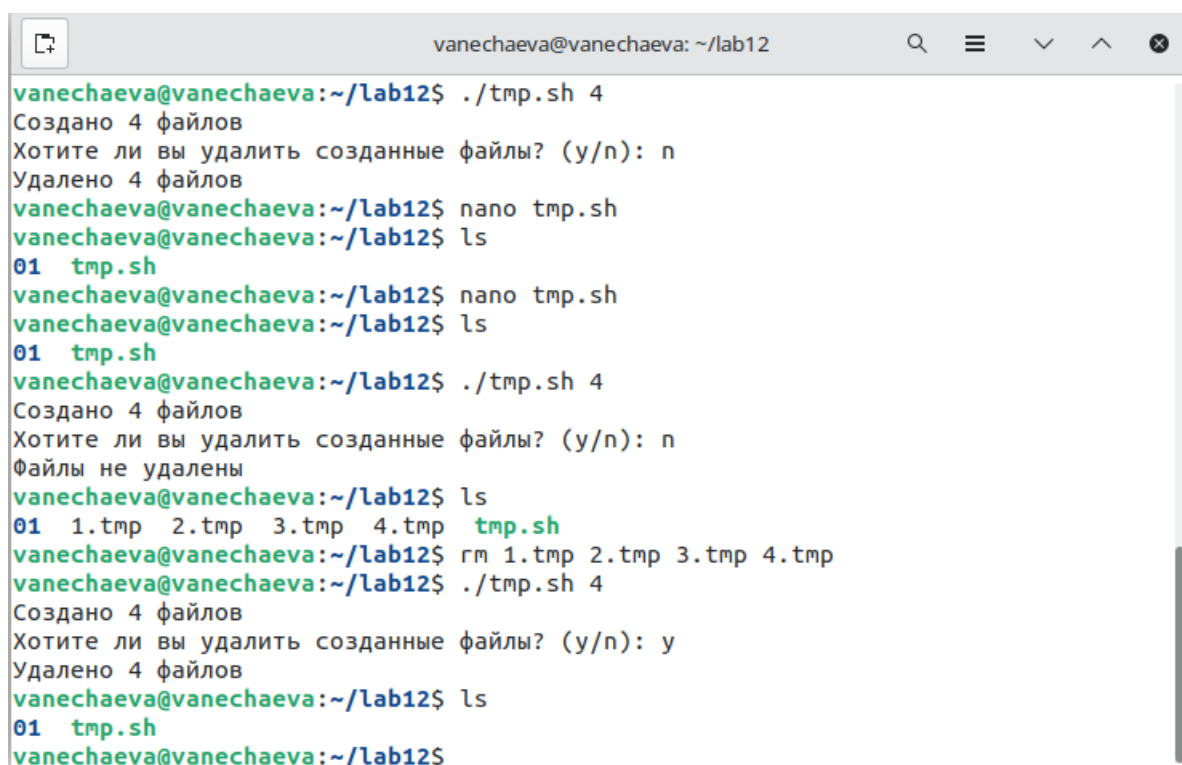
echo "Создано $1 файлов"

read -p "Хотите ли вы удалить созданные файлы? (y/n): " answer
if [ "$answer" == "y" ]
then
    for (( i=1;i<=$1;i++ ))
    do
        rm "$i.tmp"
    done
    echo "Удалено $1 файлов"
else
    echo "Файлы не удалены"
fi
```

[Прочитано 19 строк]

^G Справка	^O Записать	^W Поиск	^K Вырезать	^T Выполнить	^C Позиция
^X Выход	^R ЧитФайл	^I Замена	^U Вставить	^J Выровнять	^/_ К строке

Рис. 6: Рисунок 6

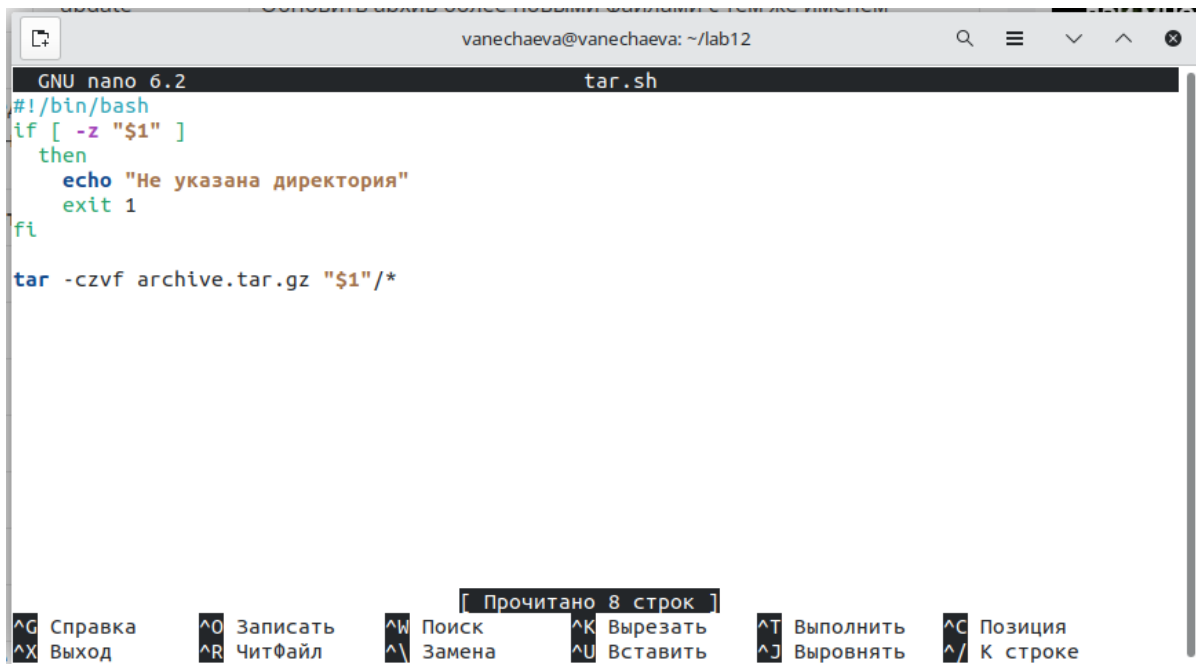
A terminal window titled 'vanechaeva@vanechaeva: ~/lab12' with standard window controls. The terminal shows a script being executed with the command './tmp.sh 4'. The script creates 4 files and asks if they should be deleted, with 'n' as the answer. Then, the user runs 'nano tmp.sh' and 'ls', showing '01 tmp.sh'. The script is run again, but this time files are not deleted. The user then runs 'ls' showing '01 1.tmp 2.tmp 3.tmp 4.tmp tmp.sh', followed by 'rm 1.tmp 2.tmp 3.tmp 4.tmp'. Finally, the script is run a third time, and the user answers 'y' to delete the files. The terminal ends with 'ls' showing '01 tmp.sh' and the prompt 'vanechaeva@vanechaeva:~/lab12\$'.

```
vanechaeva@vanechaeva:~/lab12$ ./tmp.sh 4
Создано 4 файлов
Хотите ли вы удалить созданные файлы? (y/n): n
Удалено 4 файлов
vanechaeva@vanechaeva:~/lab12$ nano tmp.sh
vanechaeva@vanechaeva:~/lab12$ ls
01 tmp.sh
vanechaeva@vanechaeva:~/lab12$ nano tmp.sh
vanechaeva@vanechaeva:~/lab12$ ls
01 tmp.sh
vanechaeva@vanechaeva:~/lab12$ ./tmp.sh 4
Создано 4 файлов
Хотите ли вы удалить созданные файлы? (y/n): n
Файлы не удалены
vanechaeva@vanechaeva:~/lab12$ ls
01 1.tmp 2.tmp 3.tmp 4.tmp tmp.sh
vanechaeva@vanechaeva:~/lab12$ rm 1.tmp 2.tmp 3.tmp 4.tmp
vanechaeva@vanechaeva:~/lab12$ ./tmp.sh 4
Создано 4 файлов
Хотите ли вы удалить созданные файлы? (y/n): y
Удалено 4 файлов
vanechaeva@vanechaeva:~/lab12$ ls
01 tmp.sh
vanechaeva@vanechaeva:~/lab12$
```

Рис. 7: Рисунок 7

Задача 4

Программа проверяет указан ли в строке вызова аргумент. Если нет, то программа завершается, если да, то создается .tar.gz



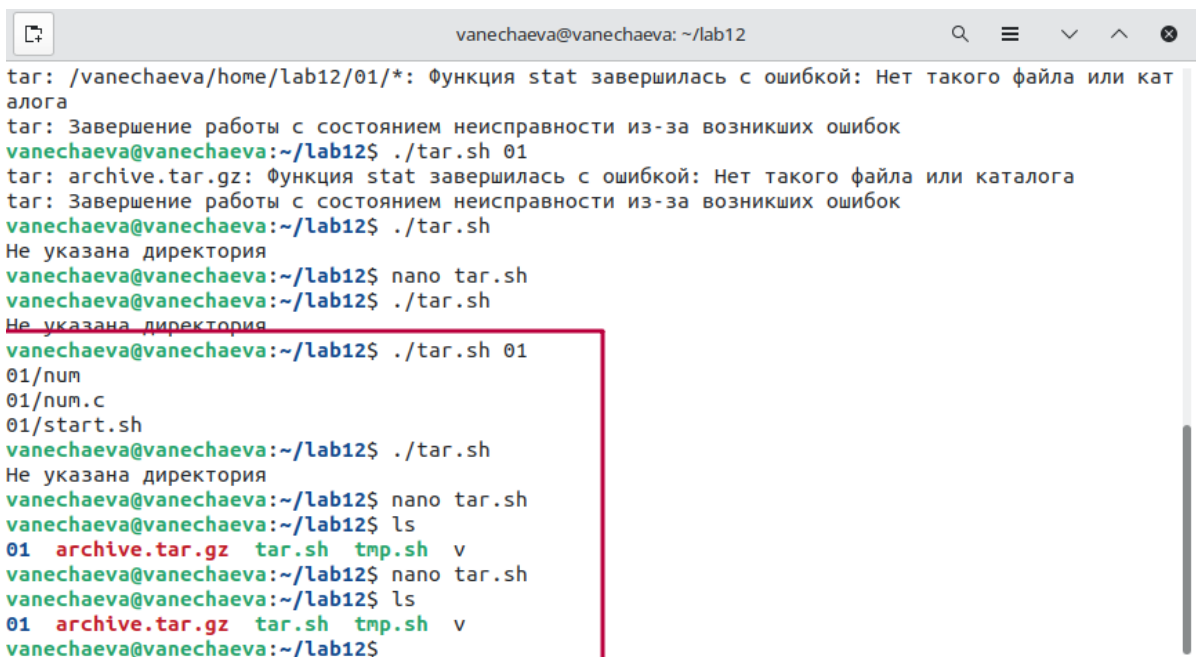
```
GNU nano 6.2 tar.sh
#!/bin/bash
if [ -z "$1" ]
then
    echo "Не указана директория"
    exit 1
fi

tar -czvf archive.tar.gz "$1"/*
```

Прочитано 8 строк

^G Справка	^O Записать	^W Поиск	^K Вырезать	^T Выполнить	^C Позиция
^X Выход	^R ЧитФайл	^M Замена	^U Вставить	^J Выводить	^/_ К строке

Рис. 8: Рисунок 8



```
tar: /vanechaeva/home/lab12/01/*: Функция stat завершилась с ошибкой: Нет такого файла или каталога
tar: Завершение работы с состоянием неисправности из-за возникших ошибок
vanechaeva@vanechaeva:~/lab12$ ./tar.sh 01
tar: archive.tar.gz: Функция stat завершилась с ошибкой: Нет такого файла или каталога
tar: Завершение работы с состоянием неисправности из-за возникших ошибок
vanechaeva@vanechaeva:~/lab12$ ./tar.sh
Не указана директория
vanechaeva@vanechaeva:~/lab12$ nano tar.sh
vanechaeva@vanechaeva:~/lab12$ ./tar.sh
Не указана директория
vanechaeva@vanechaeva:~/lab12$ ./tar.sh 01
01/num
01/num.c
01/start.sh
vanechaeva@vanechaeva:~/lab12$ ./tar.sh
Не указана директория
vanechaeva@vanechaeva:~/lab12$ nano tar.sh
vanechaeva@vanechaeva:~/lab12$ ls
01 archive.tar.gz tar.sh tmp.sh v
vanechaeva@vanechaeva:~/lab12$ nano tar.sh
vanechaeva@vanechaeva:~/lab12$ ls
01 archive.tar.gz tar.sh tmp.sh v
vanechaeva@vanechaeva:~/lab12$
```

Рис. 9: Рисунок 9

Аналогично, только еще если условие поиска по файлам, которые были изменены менее недели назад.

```
vanechaeva@vanechaeva:~$ ls
Desktop  Downloads  lab12  Pictures  snap  Templates  work
Documents  fifo      Music  Public   tar.sh  Videos

vanechaeva@vanechaeva:~$ ./tar.sh Downloads
Downloads/002-lab_shell-1.pdf
Downloads/002-lab_shell.pdf
Downloads/3-20230315T163731Z-001.zip
Downloads/4_1.sh
Downloads/4.sh
Downloads/lab08test/
Downloads/lab08test/002-lab_shell.pdf
Downloads/ls.sh
Downloads/org.gnome.gitlab.somas.Apostrophe.flatpakref
Downloads/report3.md
Downloads/report3.zip
Downloads/report_fixed3.docx
Downloads/report_fixed3.pdf
Downloads/tar.sh
Downloads/test08/
Downloads/test08/txt.txt
Downloads/002-lab_shell.pdf
Downloads/ls.sh
Downloads/report_fixed3.pdf
Downloads/002-lab_shell-1.pdf
Downloads/report3.zip
Downloads/report_fixed3.docx
Downloads/tar.sh
Downloads/test08/txt.txt
Downloads/lab08test/002-lab_shell.pdf
Downloads/4_1.sh
Downloads/4.sh
Downloads/report3.md
```

Рис. 10: Рисунок 10



```
GNU nano 6.2 tar.sh
#!/bin/bash
if [ -z "$1" ]
then
    echo "Не указана директория"
    exit 1
fi

find "$1" -type f -mtime -7 | xargs tar -czvf archive.tar.gz "$1"/*
```

Рис. 11: Рисунок 11

Выводы

В ходе данной лабораторной работы мною были написаны более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Контрольные вопросы

1. Каково предназначение команды `getopts`?

Команда `getopts` в UNIX-подобных операционных системах используется для разбора аргументов командной строки. Она позволяет программистам написать скрипты, которые могут обрабатывать опции и параметры командной строки, переданные пользователем.

2. Какое отношение метасимволы имеют к генерации имён файлов?

Метасимволы в UNIX-подобных операционных системах используются для генерации имен файлов и путей к ним. Например, символ звездочки (*) соответствует любой последовательности символов в имени файла, а символ вопросительного знака (?) соответствует одному любому символу. Также метасимволы используются для выполнения поиска и замены текста в файле при использовании утилит, таких как `sed` и `awk`.

3. Какие операторы управления действиями вы знаете?

Операторы управления действиями в UNIX-подобных операционных системах включают в себя:

`if/else`: позволяет выполнить определенное действие, если условие истинно, и другое действие, если условие ложно. `for`: используется для выполнения набора команд или действий для каждого элемента в списке или диапазоне значений. `while`: выполняет определенные действия, пока условие истинно. `case`: позволяет выполнить одно действие из нескольких в зависимости от значения

переменной. `break`: используется для выхода из текущего цикла. `continue`: прерывает текущую итерацию цикла и переходит к следующей.

4. Какие операторы используются для прерывания цикла?

Для прерывания цикла в UNIX-подобных операционных системах используются следующие операторы:

`break`: используется для немедленного выхода из цикла. `continue`: используется для пропуска текущей итерации цикла и перехода к следующей.

5. Для чего нужны команды `false` и `true`?

Команды `false` и `true` являются базовыми утилитами в UNIX-подобных операционных системах, которые возвращают код возврата 1 и 0 соответственно. Обычно эти команды используются в комбинации с другими командами и операторами для контроля над процессами, запускаемыми в скриптах.

6. Что означает строка `if test -f mans/i.$s`, встречающаяся в командном файле?

Данная строка используется для проверки наличия файла в определенном месте в файловой системе. Она проверяет, существует ли файл с именем `mans/i.$s`, где `$s`, `$i` - переменные, заданные ранее в скрипте.

7. Объясните различия между конструкциями `while` и `until`

Конструкция `while` используется для выполнения блока кода, пока условие истинно. В то время как конструкция `until` используется для выполнения блока кода до тех пор, пока условие не станет истинным. Другими словами, `while` выполняет блок кода, пока условие не станет ложным, а `until` выполняет блок кода, пока условие не станет истинным.