

Отчет по лабораторной №13

Камалиева Лия Дамировна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Контрольные вопросы	15
5	Выводы	18
	Список литературы	19

Список иллюстраций

4.1	рис.1.1	8
4.2	рис.1.2	9
4.3	рис.1.3	10
4.4	рис.1.4	10
4.5	рис.1.5	11
4.6	рис.1.6	12
4.7	рис.1.7	13
4.8	рис.1.8	13
4.9	рис.1.9	14
4.10	рис.1.10	14

Список таблиц

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

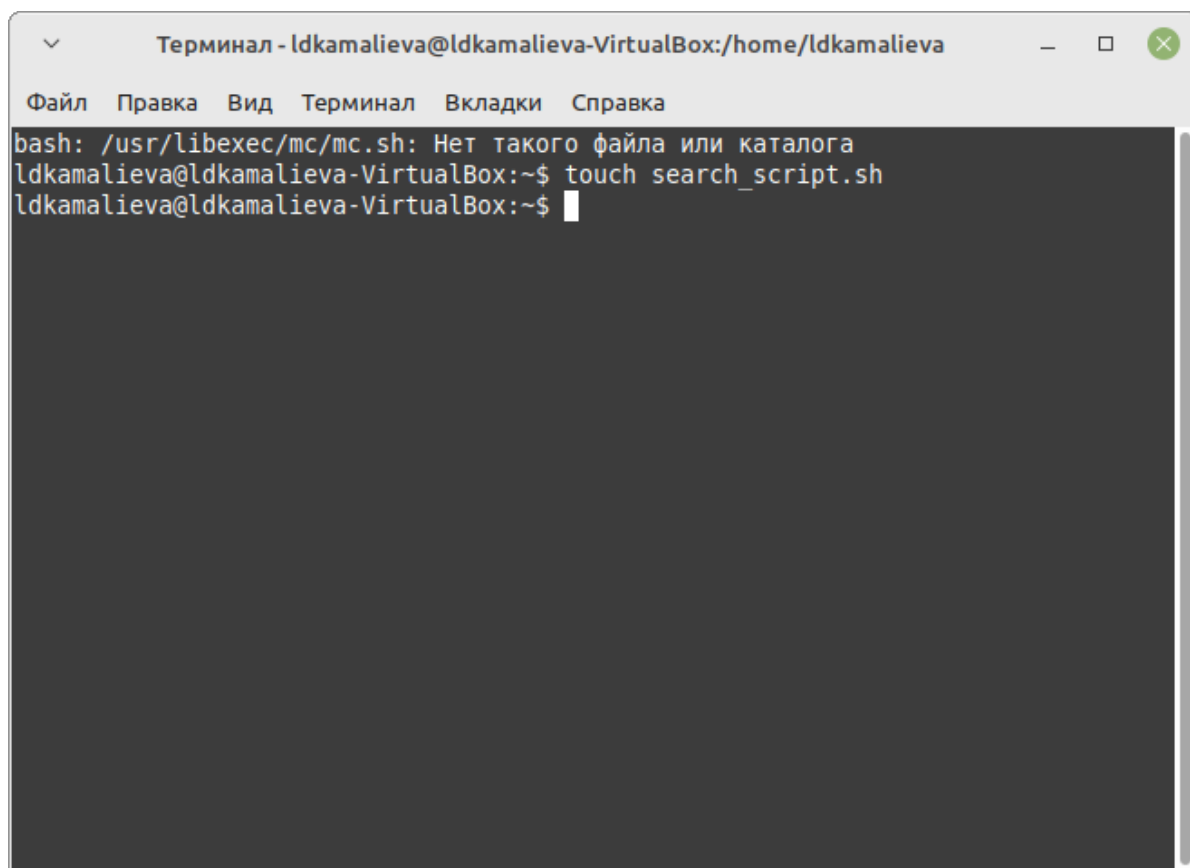
1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-iinputfile` — прочитать данные из указанного файла; `-ooutputfile` — вывести данные в указанный файл; `-rшаблон` — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-p`.
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. Команд- ный 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 ∞ (например `1.tmp`, `2.tmp`, `3.tmp`, `4.tmp` и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же ко- мандный файл должен уметь удалять все созданные им файлы (если они существуют).
4. Написать командный файл, который с помощью команды `tag` 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

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 C-подобный 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)

4 Выполнение лабораторной работы

Шаг 1. Создаю файл `research_script.sh`



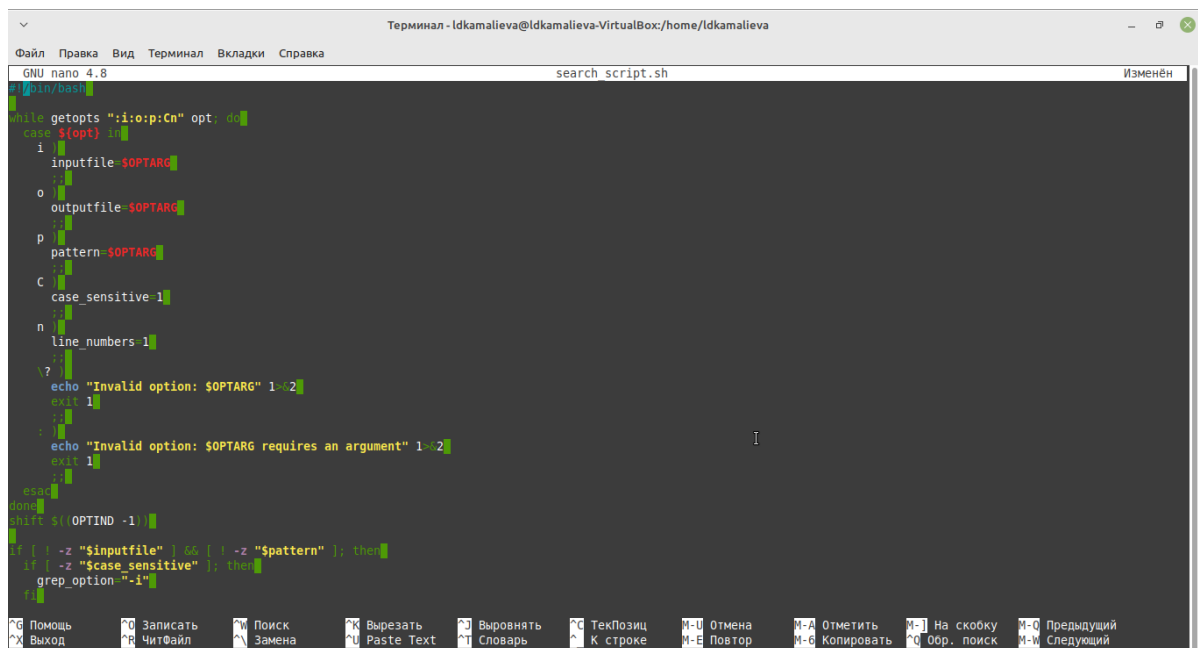
The image shows a terminal window titled "Терминал - ldkamaliev@ldkamalieva-VirtualBox:/home/ldkamalieva". The window has a menu bar with "Файл", "Правка", "Вид", "Терминал", "Вкладки", and "Справка". The terminal output shows a message from the shell: "bash: /usr/libexec/mc/mc.sh: Нет такого файла или каталога". Below this, the user enters the command "touch search_script.sh" and the prompt returns to "ldkamalieva@ldkamalieva-VirtualBox:~\$".

```
Терминал - ldkamaliev@ldkamalieva-VirtualBox:/home/ldkamalieva
Файл  Правка  Вид  Терминал  Вкладки  Справка
bash: /usr/libexec/mc/mc.sh: Нет такого файла или каталога
ldkamalieva@ldkamalieva-VirtualBox:~$ touch search_script.sh
ldkamalieva@ldkamalieva-VirtualBox:~$
```

Рис. 4.1: рис.1.1

Шаг 2. пишу скрипт, по заданию 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; — -C — различать большие и малые буквы; — -n — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом -p.



```
GNU nano 4.8 search_script.sh
#!/bin/bash
while getopts ":i:o:p:Cn" opt; do
  case $opt in
    i)
      inputfile=$OPTARG
      ;;
    o)
      outputfile=$OPTARG
      ;;
    p)
      pattern=$OPTARG
      ;;
    C)
      case_sensitive=1
      ;;
    n)
      line_numbers=1
      ;;
    ?)
      echo "Invalid option: $OPTARG" 1>&2
      exit 1
      ;;
    *)
      echo "Invalid option: $OPTARG requires an argument" 1>&2
      exit 1
      ;;
  esac
done
shift $(OPTIND -1)
if [ ! -z "$inputfile" ] && [ ! -z "$pattern" ]; then
  if [ -z "$case_sensitive" ]; then
    grep_option="-i"
  fi
```

Рис. 4.2: рис.1.2

Шаг 3. сохраняю файл

```
Терминал - ldkamalieva@ldkamalieva-VirtualBox: /home/ldkamalieva
GNU nano 4.8 search_script.sh
#!/bin/bash
while getopts ":i:o:p:C:n" opt; do
  case $opt in
    i)
      inputfile=$OPTARG
      ;;
    o)
      outfile=$OPTARG
      ;;
    p)
      pattern=$OPTARG
      ;;
    C)
      case_sensitive=1
      ;;
    n)
      line_numbers=1
      ;;
    ?)
      echo "Invalid option: $OPTARG" 1>&2
      exit 1
      ;;
    :)
      echo "Invalid option: $OPTARG requires an argument" 1>&2
      exit 1
      ;;
  esac
done
shift $(($OPTIND - 1))
if [ ! -z "$inputfile" ] && [ ! -z "$pattern" ]; then
  if [ -z "$case_sensitive" ]; then
    grep -oi "$pattern" "$inputfile"
  else
    grep -o "$pattern" "$inputfile"
  fi
fi
```

Рис. 4.3: рис.1.3

Шаг 4. используя команду `chmod +x research_script.sh`

```
Терминал - ldkamalieva@ldkamalieva-VirtualBox: /home/ldkamalieva
bash: /usr/libexec/ncurses/ncurses: Нет такого файла или каталога
ldkamalieva@ldkamalieva-VirtualBox:~$ touch search_script.sh
ldkamalieva@ldkamalieva-VirtualBox:~$ nano search_script.sh
ldkamalieva@ldkamalieva-VirtualBox:~$ chmod +x search_script.sh
ldkamalieva@ldkamalieva-VirtualBox:~$
```

Рис. 4.4: рис.1.4

Шаг 5. открываю программу gedit

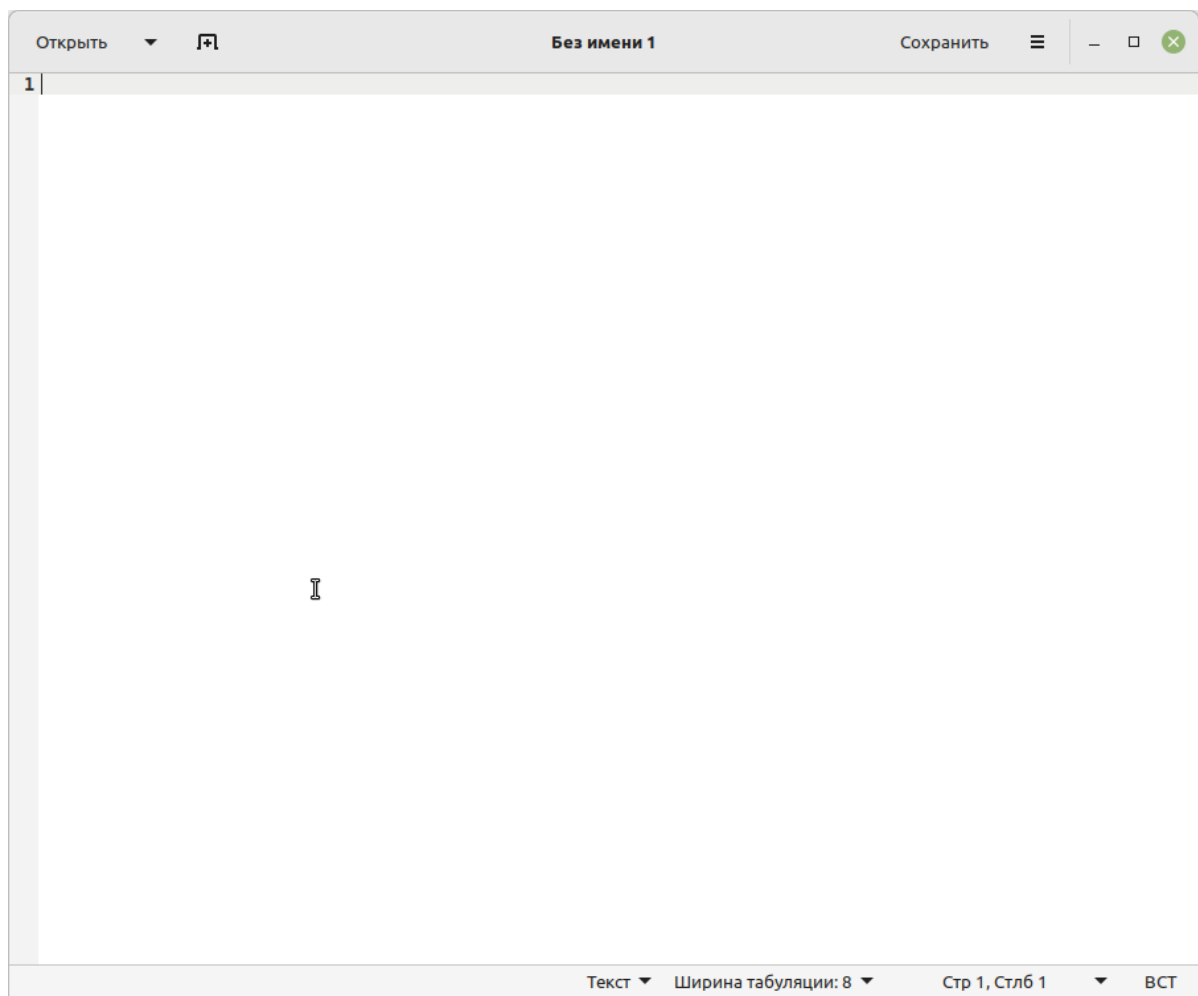
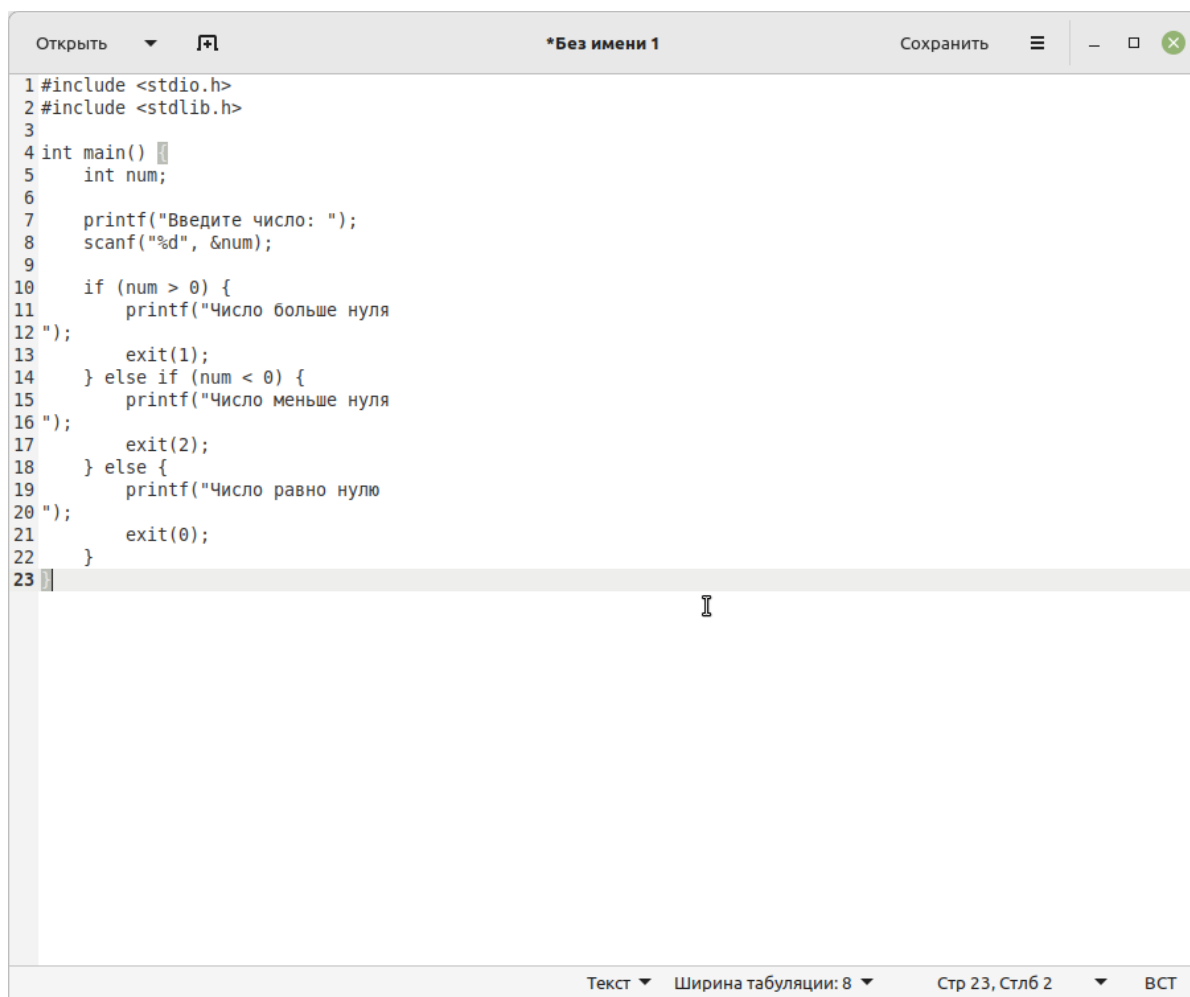


Рис. 4.5: рис.1.5

Шаг 6. прописываю скрипт



The screenshot shows a code editor window with a title bar containing "Открыть", a file icon, "*Без имени 1", "Сохранить", and window control buttons. The editor contains the following C code:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int num;
6
7     printf("Введите число: ");
8     scanf("%d", &num);
9
10    if (num > 0) {
11        printf("Число больше нуля
12 ");
13        exit(1);
14    } else if (num < 0) {
15        printf("Число меньше нуля
16 ");
17        exit(2);
18    } else {
19        printf("Число равно нулю
20 ");
21        exit(0);
22    }
23 }
```

The status bar at the bottom indicates "Текст", "Ширина табуляции: 8", "Стр 23, Стлб 2", and "ВСТ".

Рис. 4.6: рис.1.6

Шаг 7. прописываю еще один скрип для запуска

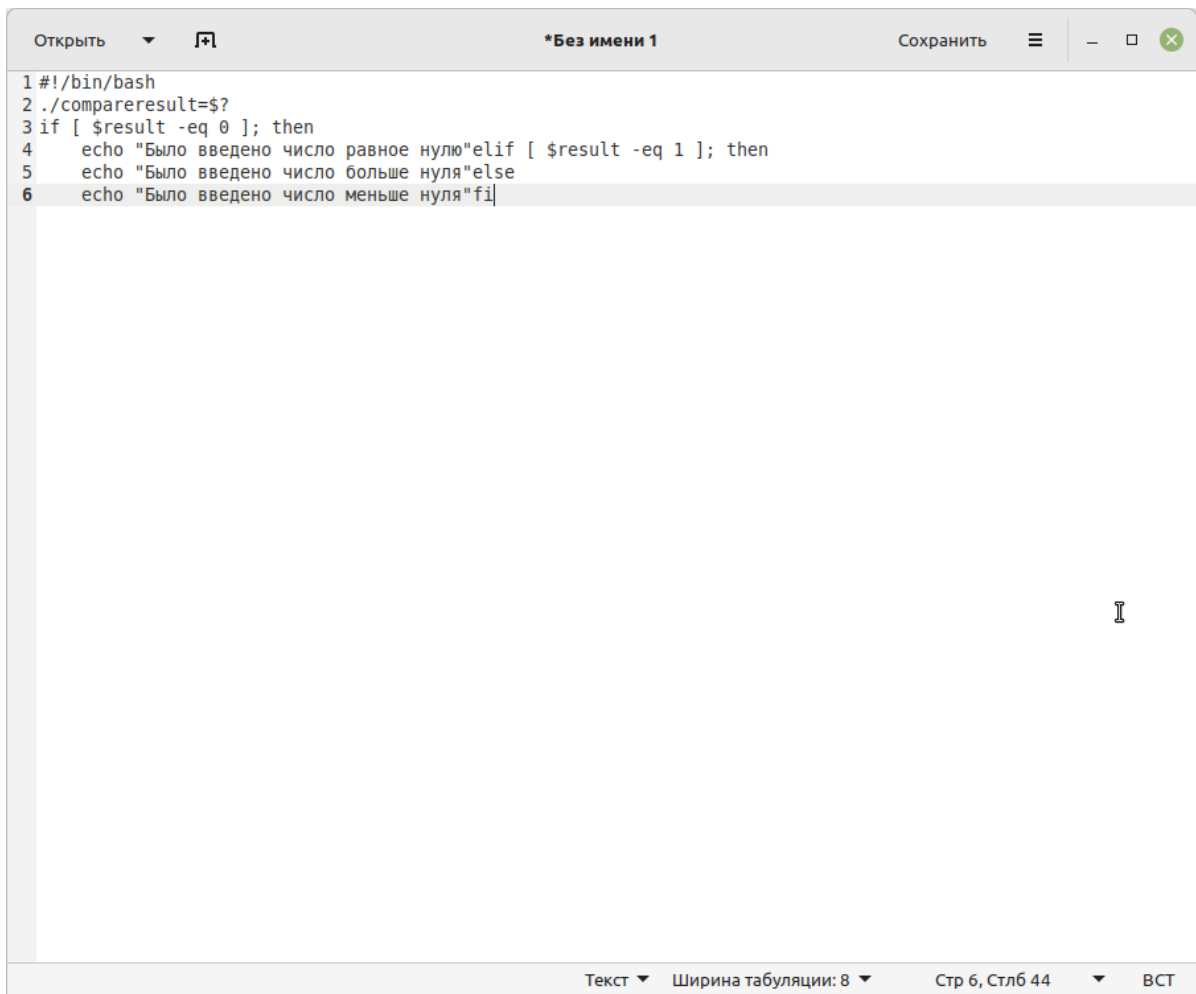


Рис. 4.7: рис.1.7

Шаг 8. даю файлу права на выполнение

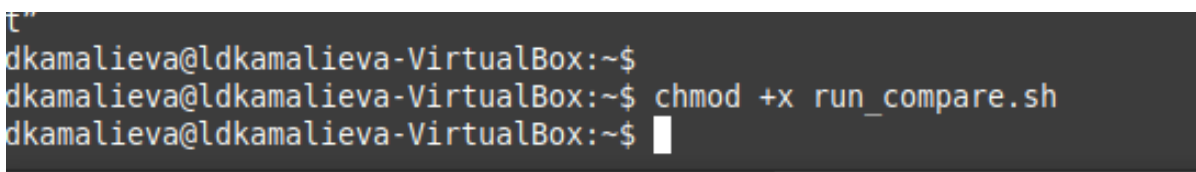
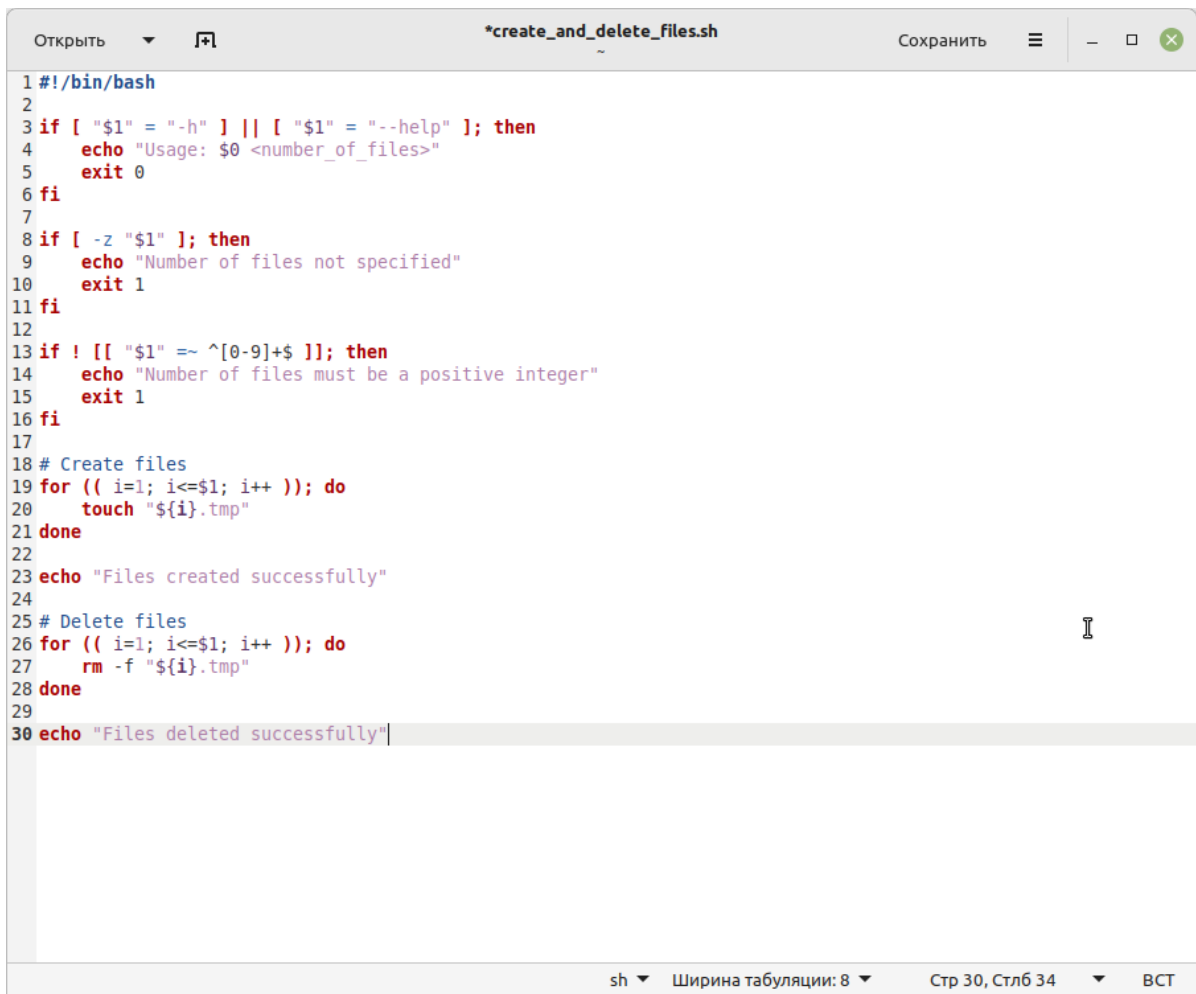


Рис. 4.8: рис.1.8

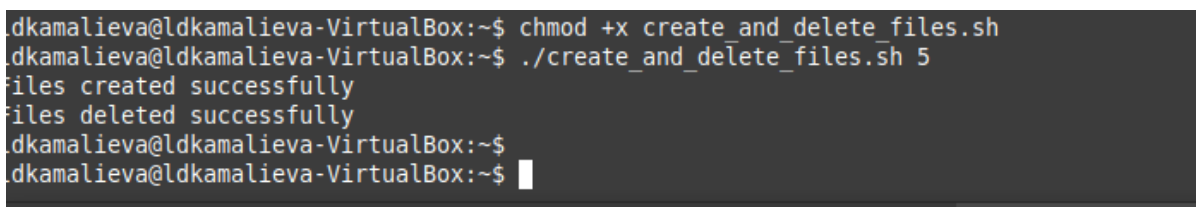
Шаг 9. создаю файл и прописываю в нем скрипт



```
1 #!/bin/bash
2
3 if [ "$1" = "-h" ] || [ "$1" = "--help" ]; then
4     echo "Usage: $0 <number_of_files>"
5     exit 0
6 fi
7
8 if [ -z "$1" ]; then
9     echo "Number of files not specified"
10    exit 1
11 fi
12
13 if ! [[ "$1" =~ ^[0-9]+$ ]]; then
14     echo "Number of files must be a positive integer"
15     exit 1
16 fi
17
18 # Create files
19 for (( i=1; i<=$1; i++ )); do
20     touch "${i}.tmp"
21 done
22
23 echo "Files created successfully"
24
25 # Delete files
26 for (( i=1; i<=$1; i++ )); do
27     rm -f "${i}.tmp"
28 done
29
30 echo "Files deleted successfully"
```

Рис. 4.9: рис.1.9

Шаг 9. проверяю работу кода



```
ldkamalieva@ldkamalieva-VirtualBox:~$ chmod +x create_and_delete_files.sh
ldkamalieva@ldkamalieva-VirtualBox:~$ ./create_and_delete_files.sh 5
files created successfully
files deleted successfully
ldkamalieva@ldkamalieva-VirtualBox:~$
ldkamalieva@ldkamalieva-VirtualBox:~$
```

Рис. 4.10: рис.1.10

4.1 Контрольные вопросы

1 Каково предназначение команды `getopts`? 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 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 `getopts` может распознать аргумент, то она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные 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 введенных пользователем данных.

2 Какое отношение метасимволы имеют к генерации имён файлов? При перечислении имён файлов текущего каталога можно использовать следующие символы: 1 соответствует произвольной, в том числе и пустой строке; 2 `?` соответствует любому одинарному символу; 3 `[c1-c2]` соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`. Например, 1.1 `echo` выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; 1.2. `ls.c` выведет все файлы с последними двумя символами, совпадающими с `c`. 1.3. `echo prog.?` выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.` 1.4. `[a-z]` соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

3 Какие операторы управления действиями вы знаете? 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 Какие операторы используются для прерывания цикла? 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 Для чего нужны команды `false` и `true`? 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` и `until false do echo hello mike done`.

6 Что означает строка `if test -f mans/i.$s`, встречаемая в командном файле? Строка `if test -f mans/i.s` проверяет, существует ли файл `mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то коман-

да вернет нулевое значение (ложь). 7 Объясните различия между конструкциями `while` и `until`. 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.

5 Выводы

я ознакомилась с функциями etacs

Список литературы