

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине «Операционная система Linux»

Программирование на SHELL. Использование командных файлов

Студентка

Пехова А.А.

Группа ПИ-19

Руководитель

Кургасов В.В.

К.П.Н.

Липецк 2021

Оглавление

Цель работы	2
Задание кафедры	3
Ход работы.....	5
Вывод	54
Контрольные вопросы	54

Цель работы

Изучение основных возможностей языка программирования Shell с целью автоматизации процесса администрирования системы за счет написания и использования командных файлов.

Задание кафедры

1. Используя команды ECHO, PRINTF вывести информационные сообщения на экран.
2. Присвоить переменной A целочисленное значение. Просмотреть значение переменной A.
3. Присвоить переменной B значение переменной A. Просмотреть значение переменной B.
4. Присвоить переменной C значение “путь до своего каталога”. Перейти в этот каталог с использованием переменной.
5. Присвоить переменной D значение “имя команды”, а именно, команды DATE. Выполнить эту команду, используя значение переменной.
6. Присвоить переменной E значение “имя команды”, а именно, команды просмотра содержимого файла, просмотреть содержимое переменной. Выполнить эту команду, используя значение переменной.
7. Присвоить переменной F значение “имя команды”, а именно сортировки содержимого текстового файла. Выполнить эту команду, используя значение переменной.

Написать скрипты, при запуске которых выполняются следующие действия:

8. Программа запрашивает значение переменной, а затем выводит значение этой переменной.
9. Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введенной переменной.
10. Программа запрашивает значения двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на экран (использовать команды а) EXPR; б) BC)..

11. Вычислить объем цилиндра. Исходные данные запрашиваются программой. Результат выводится на экран.
12. Используя позиционные параметры, отобразить имя программы, количество аргументов командной строки, значение каждого аргумента командной строки.
13. Используя позиционный параметр, отобразить содержимое текстового файла, указанного в качестве аргумента командной строки. После паузы экран очищается.
14. Используя оператор FOR, отобразить содержимое текстовых файлов текущего каталога поэкранно.
15. Программой запрашивается ввод числа, значение которого затем сравнивается с допустимым значением. В результате этого сравнения на экран выдаются соответствующие сообщения.
16. Программой запрашивается год, определяется, високосный ли он. Результат выдается на экран.
17. Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значения инкрементируются.
18. В качестве аргумента командной строки указывается пароль. Если пароль введен верно, постранично отображается в длинном формате с указанием скрытых файлов содержимое каталога /etc.
19. Проверить, существует ли файл. Если да, выводится на экран его содержимое, если нет - выдается соответствующее сообщение.
20. Если файл есть каталог и этот каталог можно читать, просматривается содержимое этого каталога. Если каталог отсутствует, он создается. Если файл не есть каталог, просматривается содержимое файла.

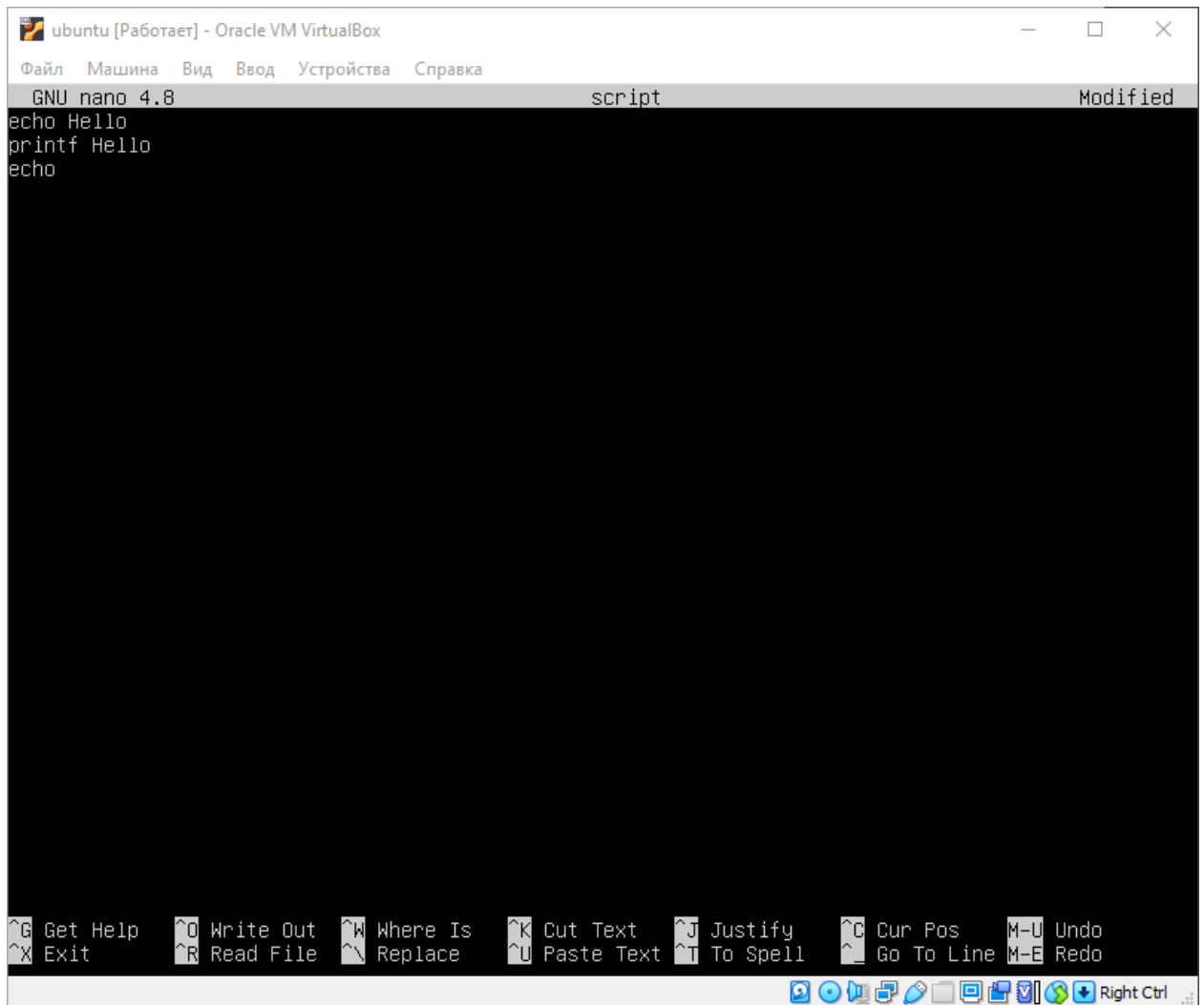
21. Анализируются атрибуты файла. Если первый файл существует и используется для чтения, а второй файл существует и используется для записи, то содержимое первого файла перенаправляется во второй файл. В случае несовпадений указанных атрибутов или отсутствия файлов на экран выдаются соответствующие сообщения (использовать а) имена файлов; б) позиционные параметры).
22. Если файл запуска программы найден, программа запускается (по выбору).
23. В качестве позиционного параметра задается файл, анализируется его размер. Если размер файла больше нуля, содержимое файла сортируется по первому столбцу по возрастанию, отсортированная информация помещается в другой файл, содержимое которого затем отображается на экране.
24. Командой TAR осуществляется сборка всех текстовых файлов текущего каталога в один архивный файл `my.tar`, после паузы просматривается содержимое файла `my.tar`, затем командой GZIP архивный файл `my.tar` сжимается.
25. Написать скрипт с использованием функции, например, функции, суммирующей значения двух переменных.

Ход работы

Создадим файл с помощью команды `nano script`, в этом файле и будем записывать сценарии для выполнения заданий.

1. Используя команды `ECHO`, `PRINTF` вывести информационные сообщения на экран.

Напишем следующий скрипт:



```
ubuntu [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
GNU nano 4.8                                script                                Modified
echo Hello
printf Hello
echo

^G Get Help      ^O Write Out    ^W Where Is    ^K Cut Text    ^J Justify    ^C Cur Pos    M-U Undo
^X Exit          ^R Read File    ^_ Replace     ^U Paste Text  ^T To Spell   ^G Go To Line M-E Redo
Right Ctrl
```

Рисунок 1.1 – Скрипт для задания 1

После запуска данного сценария увидим:

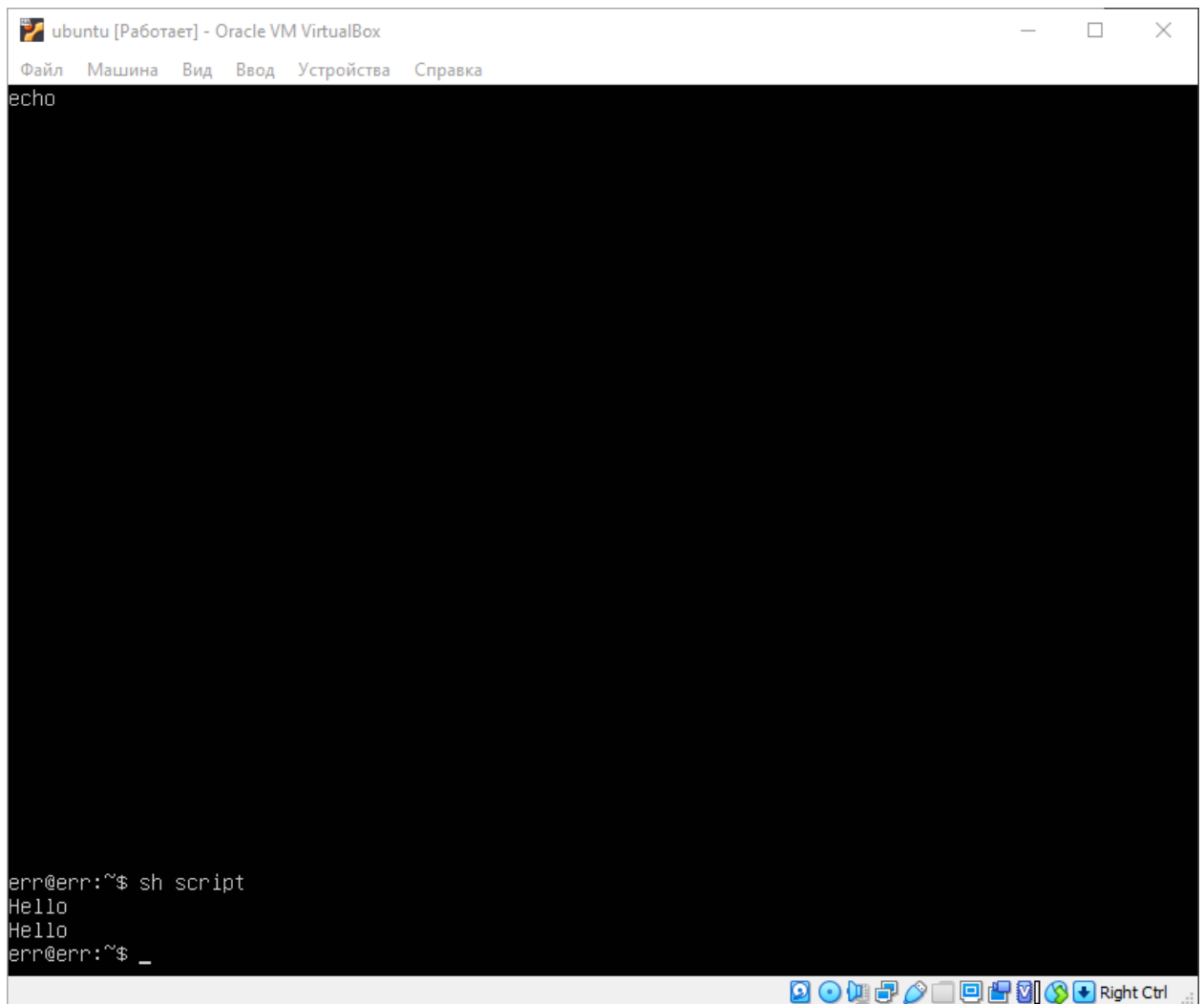


Рисунок 1.2 – Результат выполнения сценария

2. Присвоить переменной А целочисленное значение. Просмотреть значение переменной А.
3. Присвоить переменной В значение переменной А. Просмотреть значение переменной В.

Напишем сценарий:

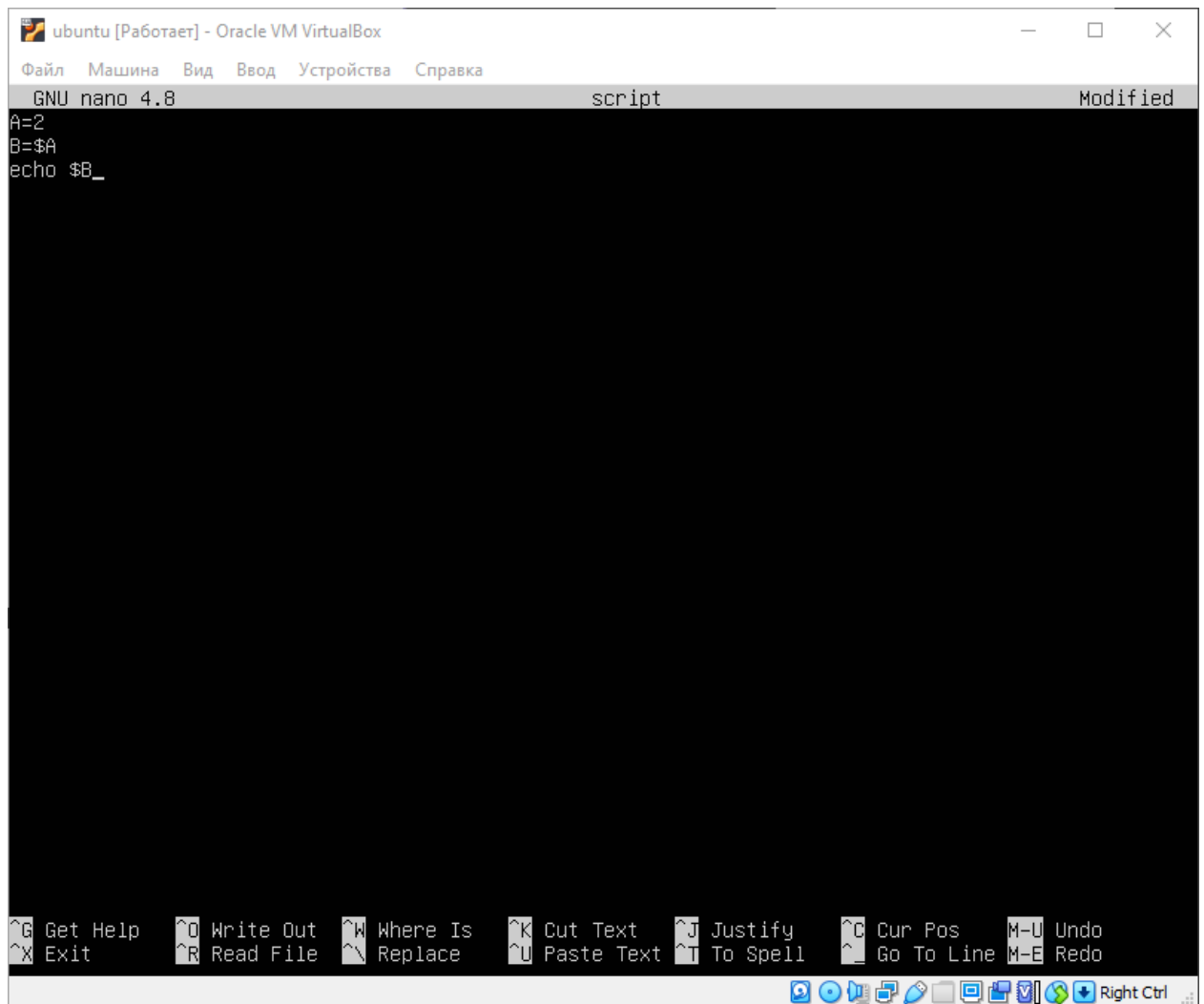


Рисунок 2 – Скрипт для заданий 2, 3

Выполним сценарий:

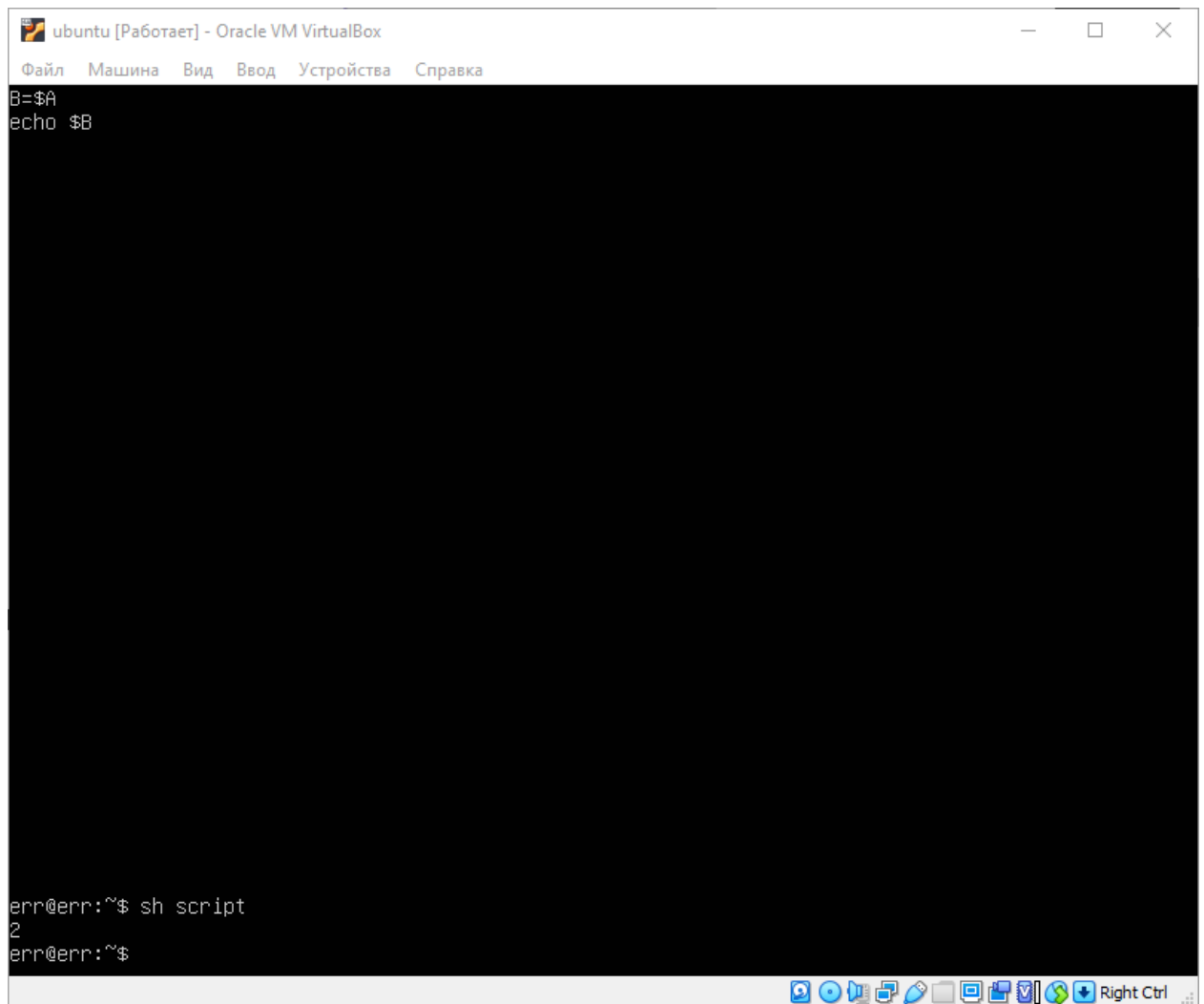


Рисунок 3 – Результат выполнения сценария

4. Присвоить переменной С значение “путь до своего каталога”. Перейти в этот каталог с использованием переменной.

Сделаем это с использованием встроенной переменной HOME:

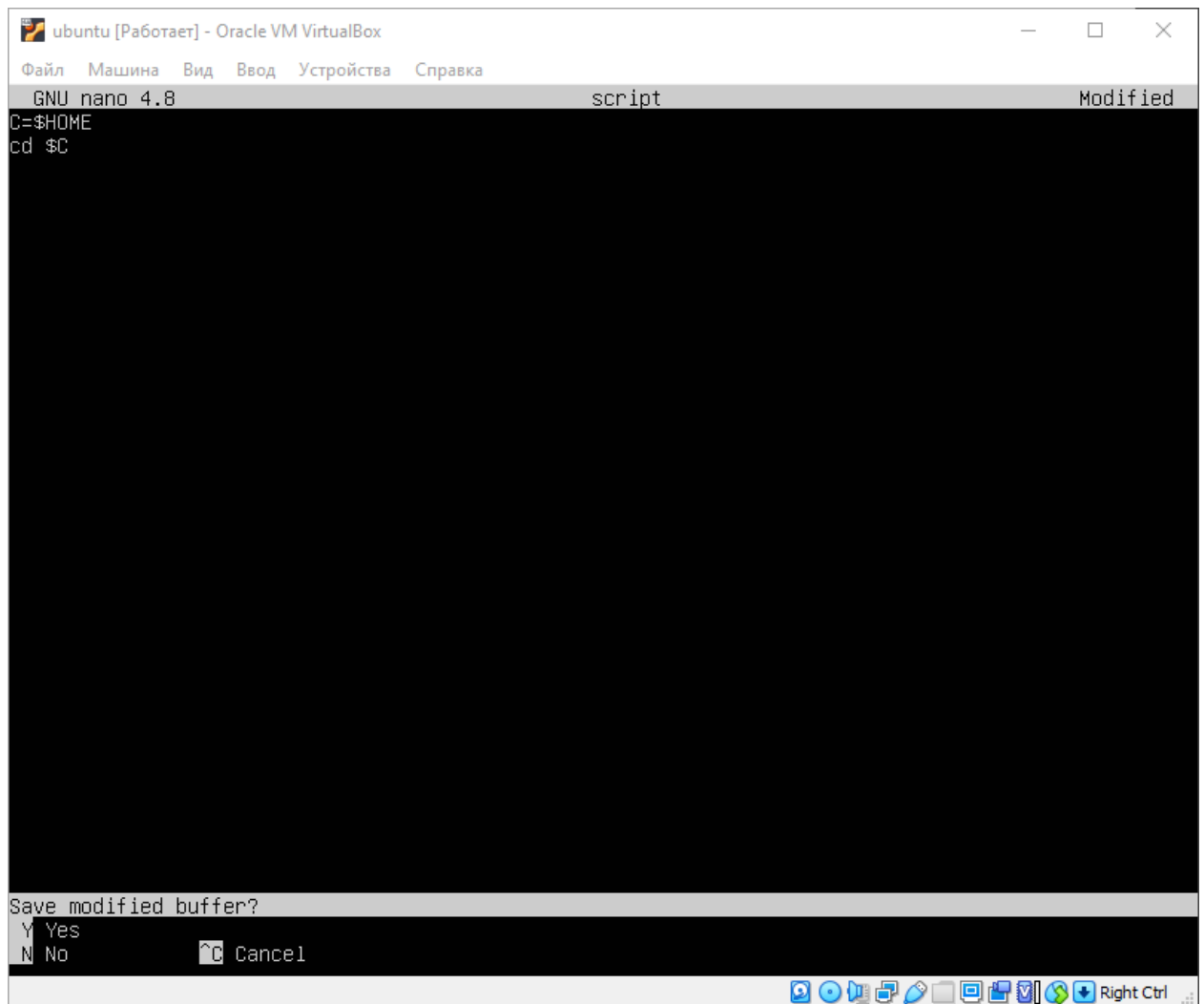


Рисунок 4.1 – Скрипт для задания 4

Проверим работу сценария:

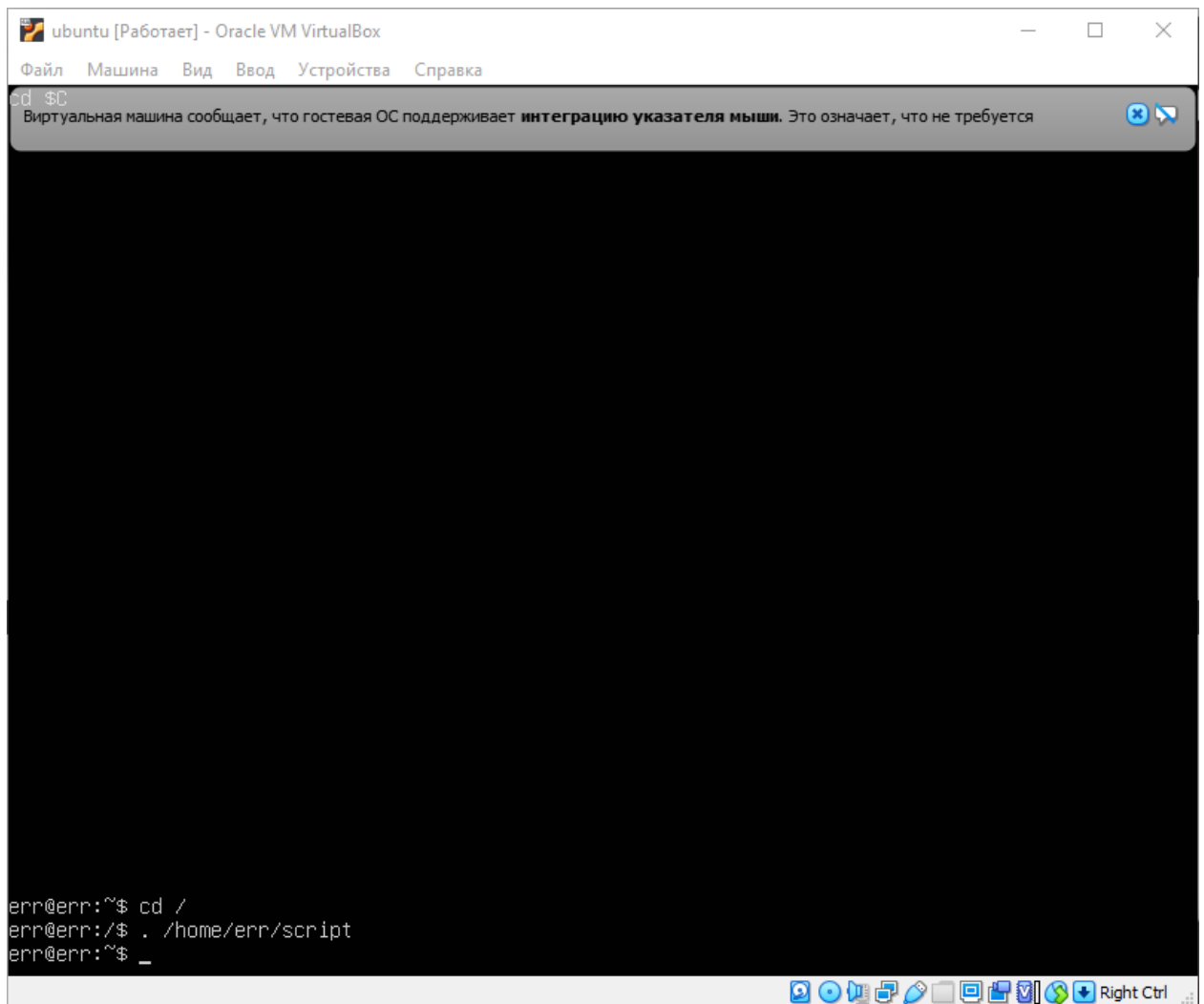


Рисунок 4.2 – Результат выполнения сценария

5. Присвоить переменной D значение “имя команды”, а именно, команды DATE. Выполнить эту команду, используя значение переменной.

Выполним задание с использованием встроенной переменной DATE:

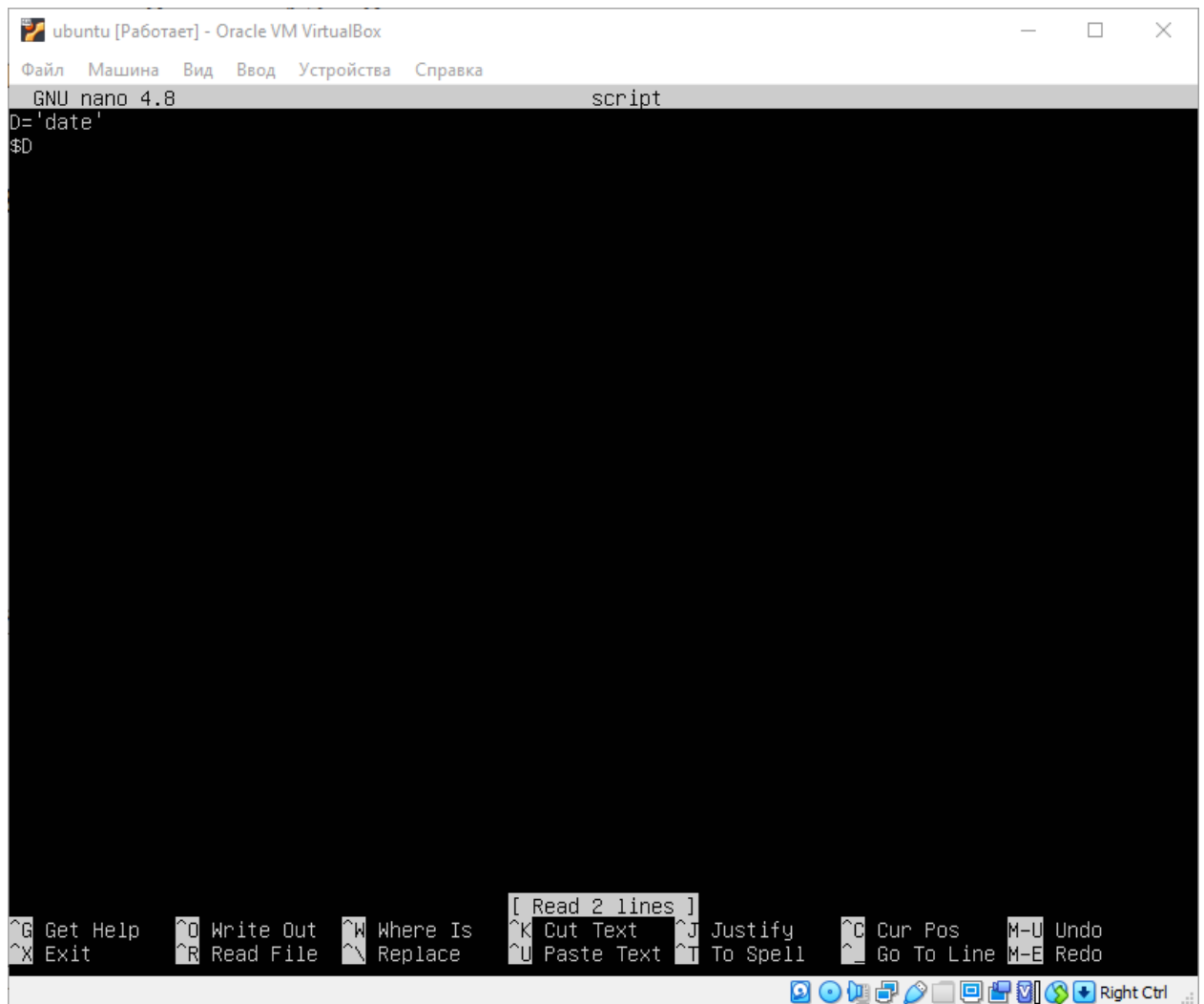


Рисунок 5.1 – Скрипт для задания 5

Исполним сценарий:

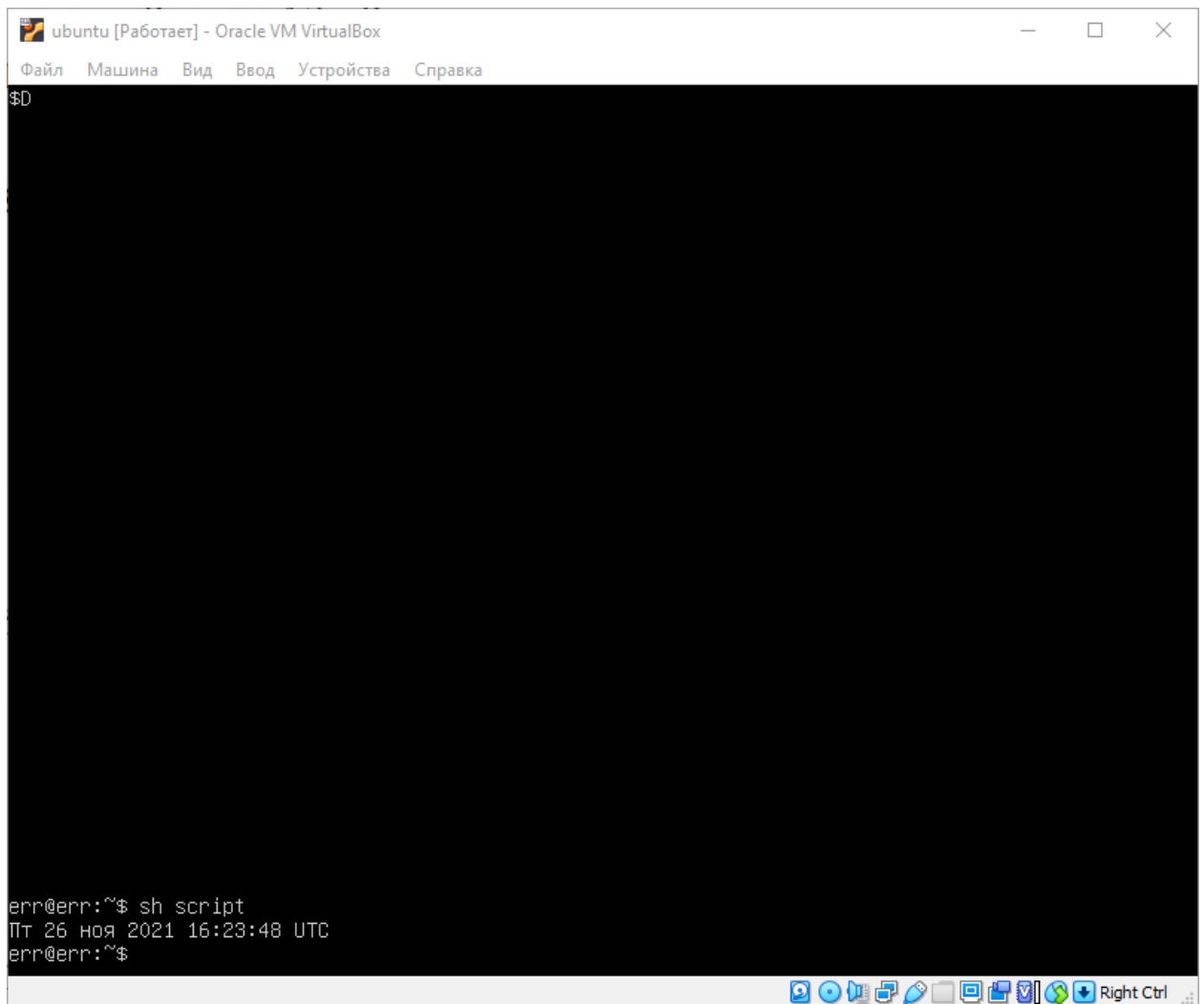


Рисунок 5.2 – Результат выполнения сценария

6. Присвоить переменной Е значение “имя команды”, а именно, команды просмотра содержимого файла, просмотреть содержимое переменной. Выполнить эту команду, используя значение переменной.

Присвоим переменной значение команды cat:

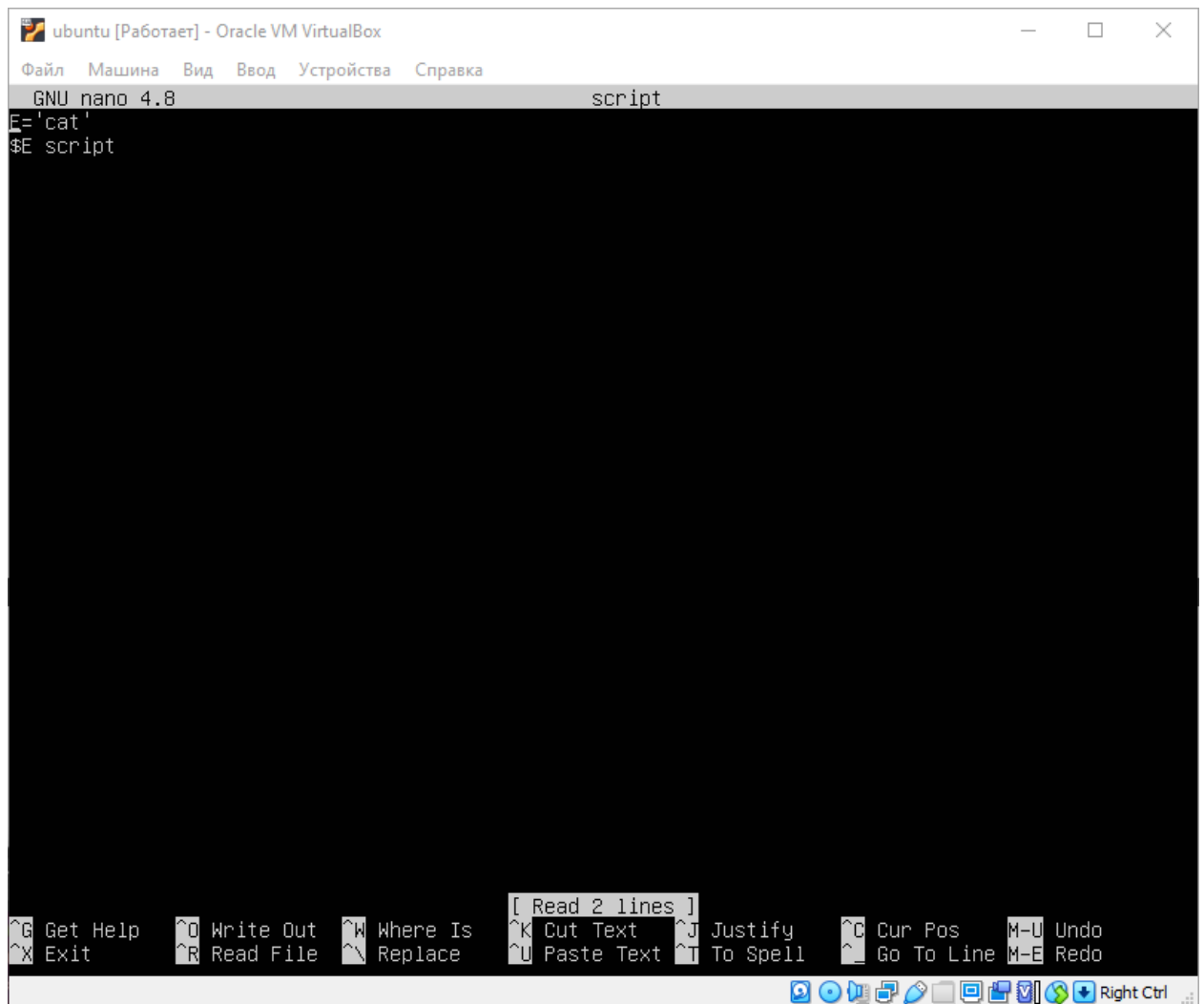


Рисунок 6.1 – Скрипт для задания 6

Выполним сценарий:

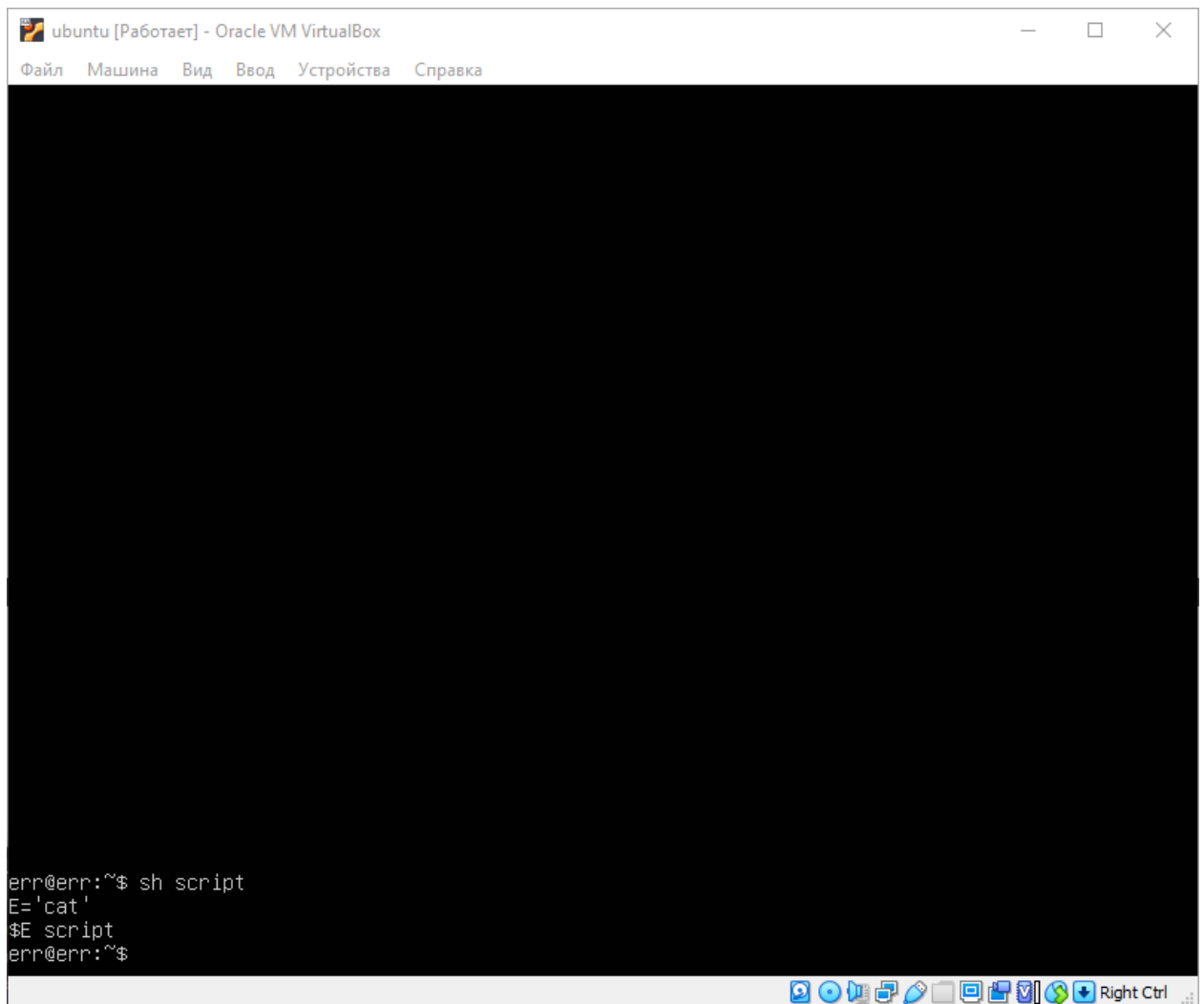


Рисунок 6.2 – Результат выполнения сценария

7. Присвоить переменной F значение “имя команды”, а именно сортировки содержимого текстового файла. Выполнить эту команду, используя значение переменной.

Присвоим переменной F значение sort:

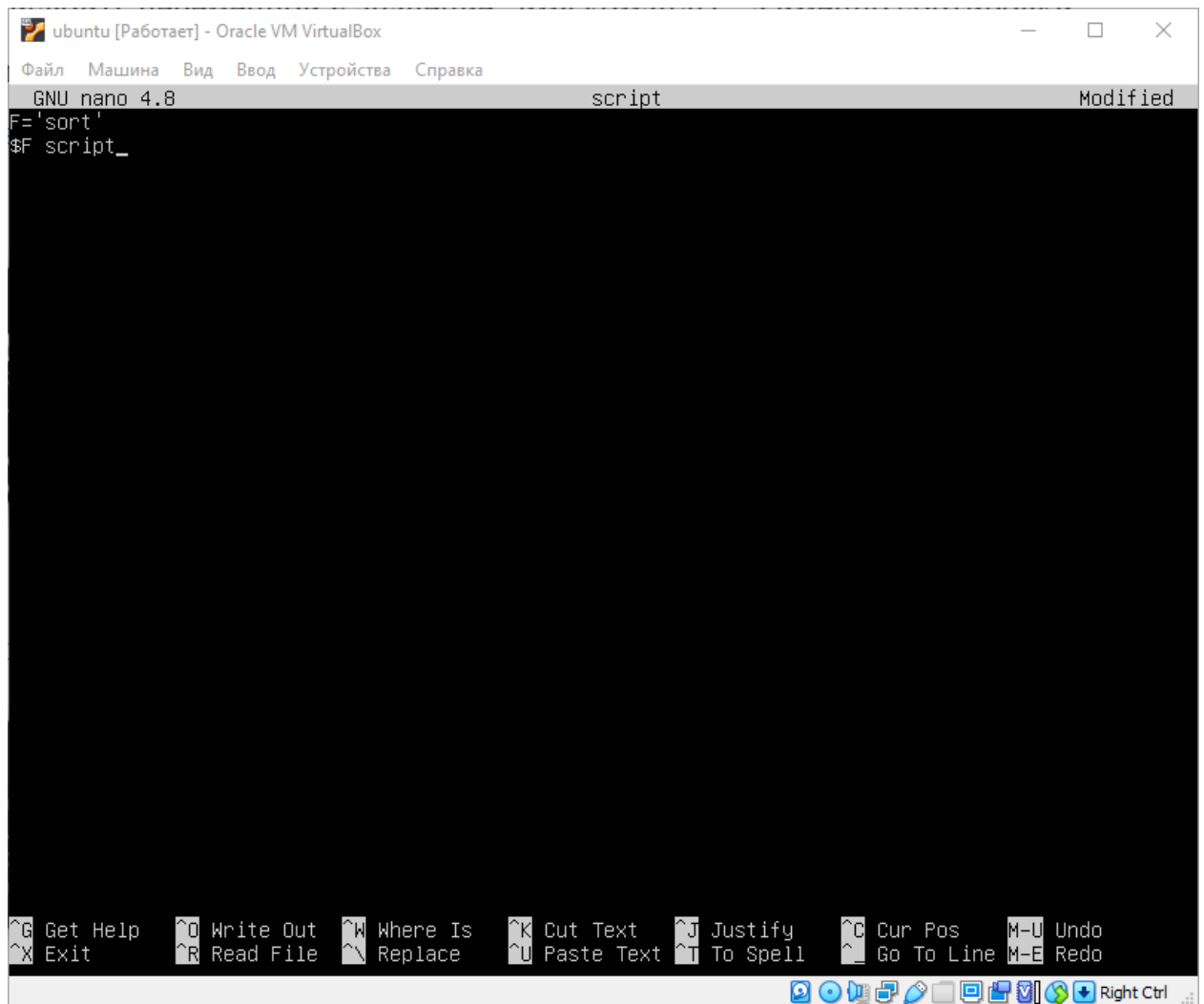


Рисунок 7.1 – Скрипт для задания 7

Исполним сценарий:

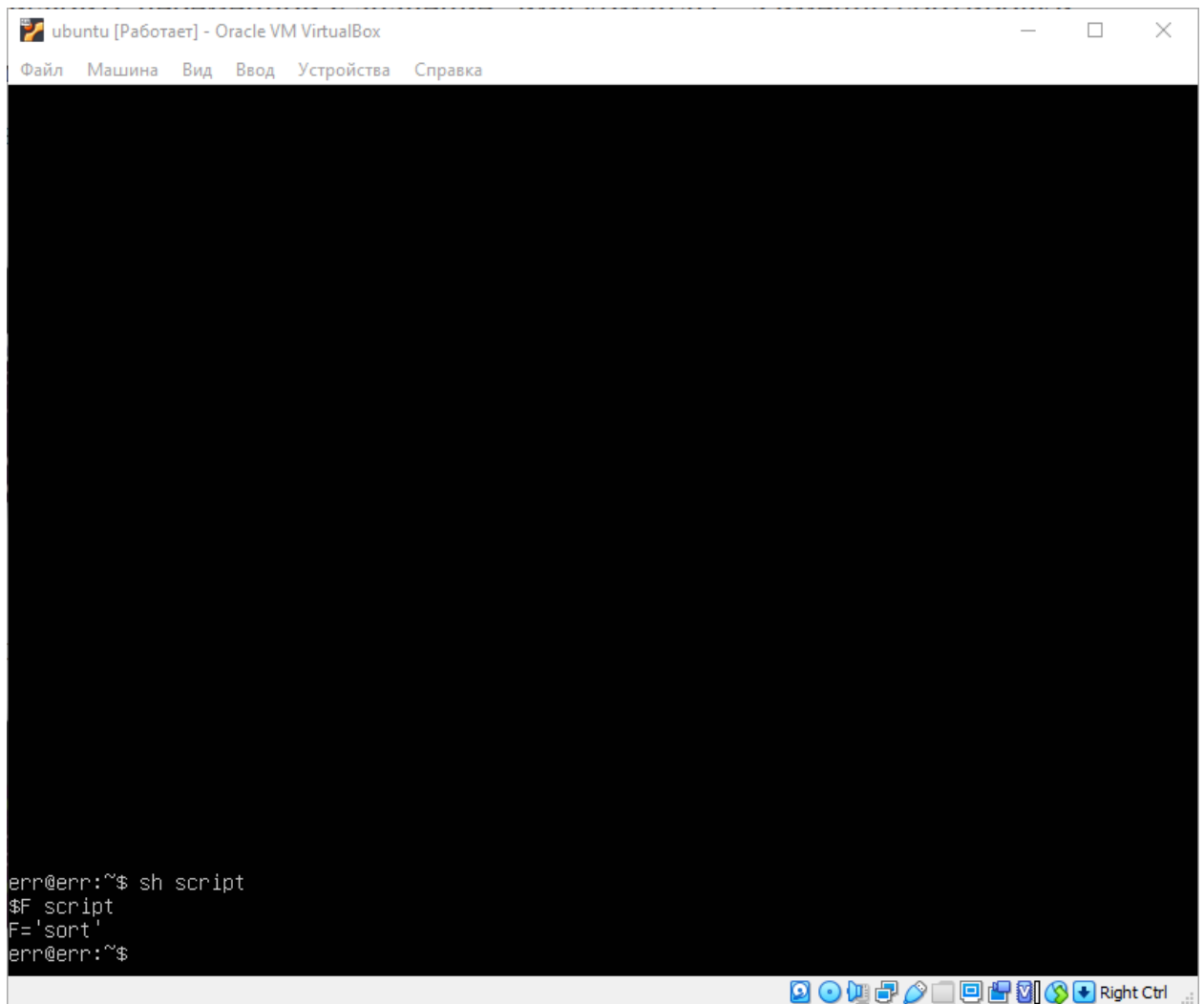


Рисунок 7.2 – Результат выполнения сценария

8. Программа запрашивает значение переменной, а затем выводит значение этой переменной.

Запросить у пользователя значение переменной позволяет команда `read`.

Используем эту команду в написании скрипта:

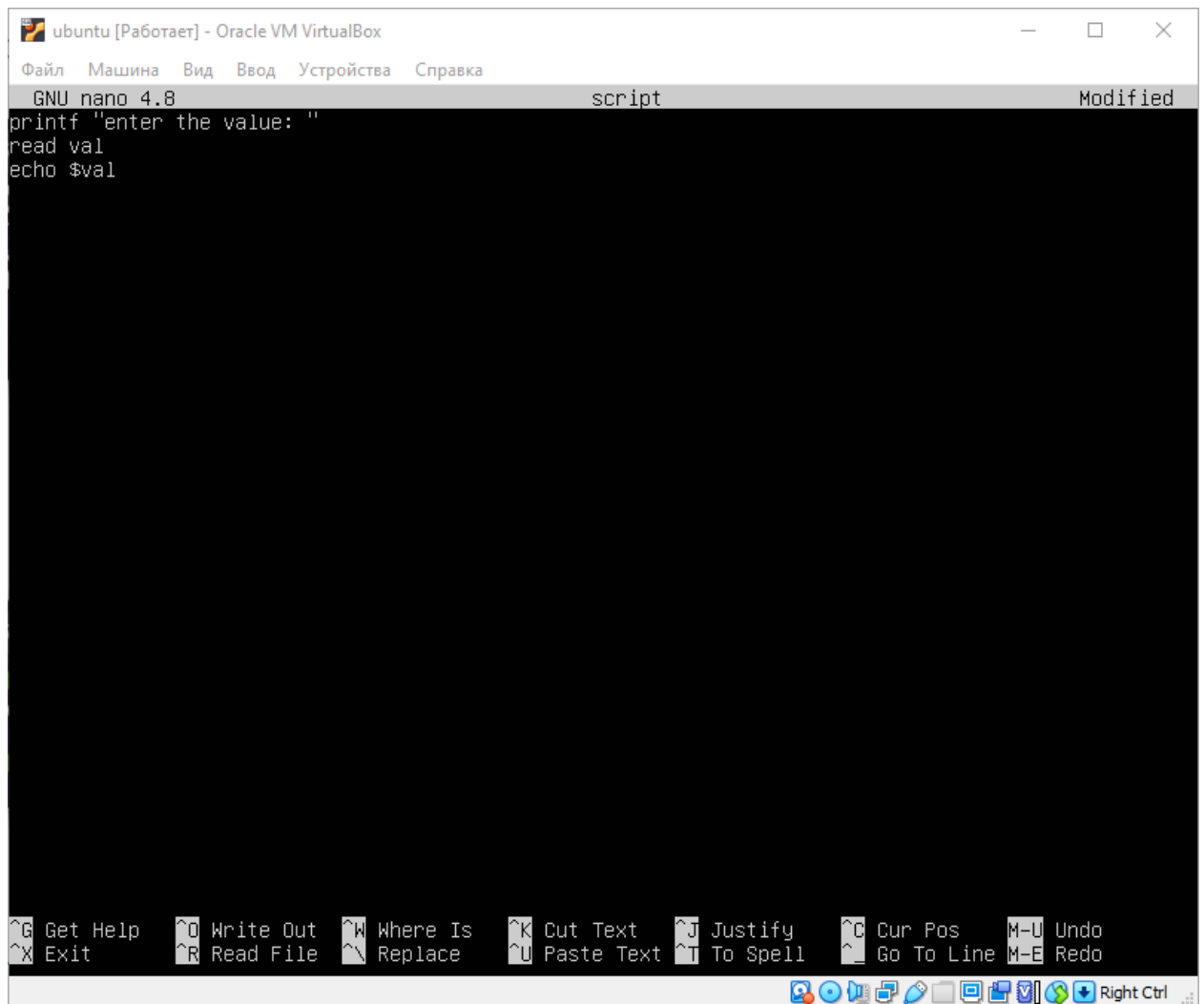


Рисунок 8.1 – Скрипт для задания 8

Исполним сценарий:

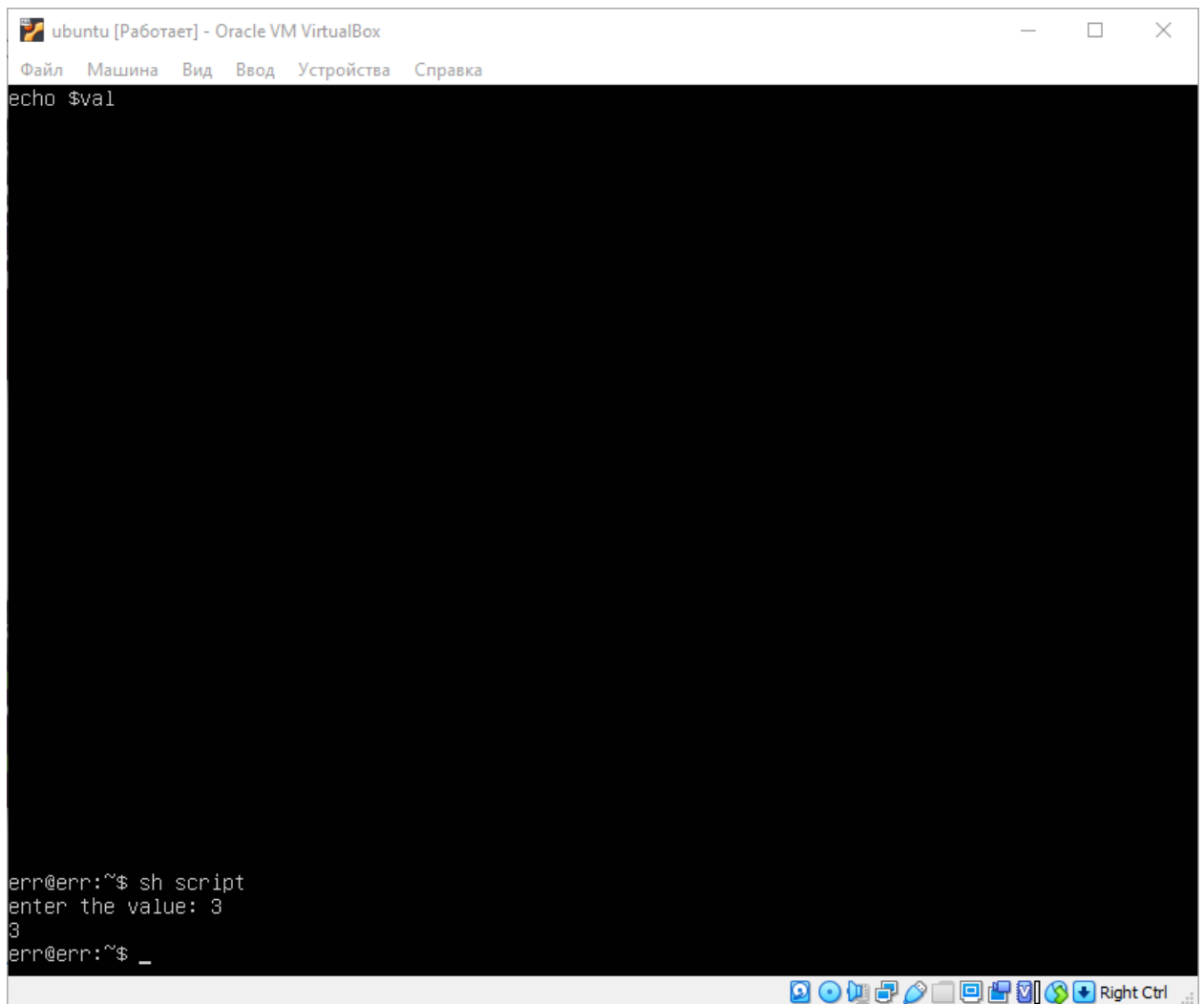


Рисунок 8.2 – Результат выполнения сценария

9. Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введенной переменной.

Сделаем по примеру предыдущего задания:

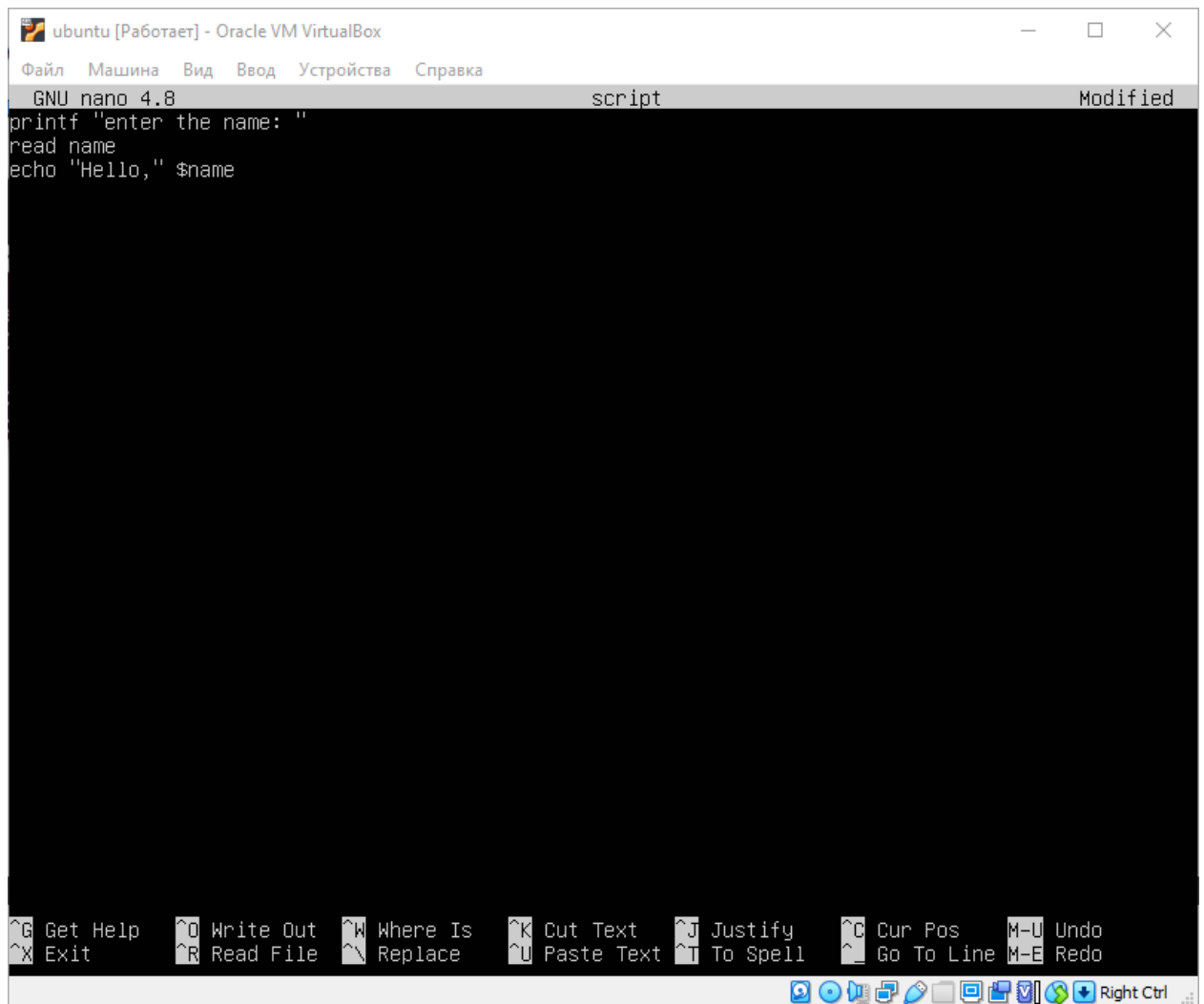


Рисунок 9.1 – Скрипт для задания 9

Запустим файл сценария:

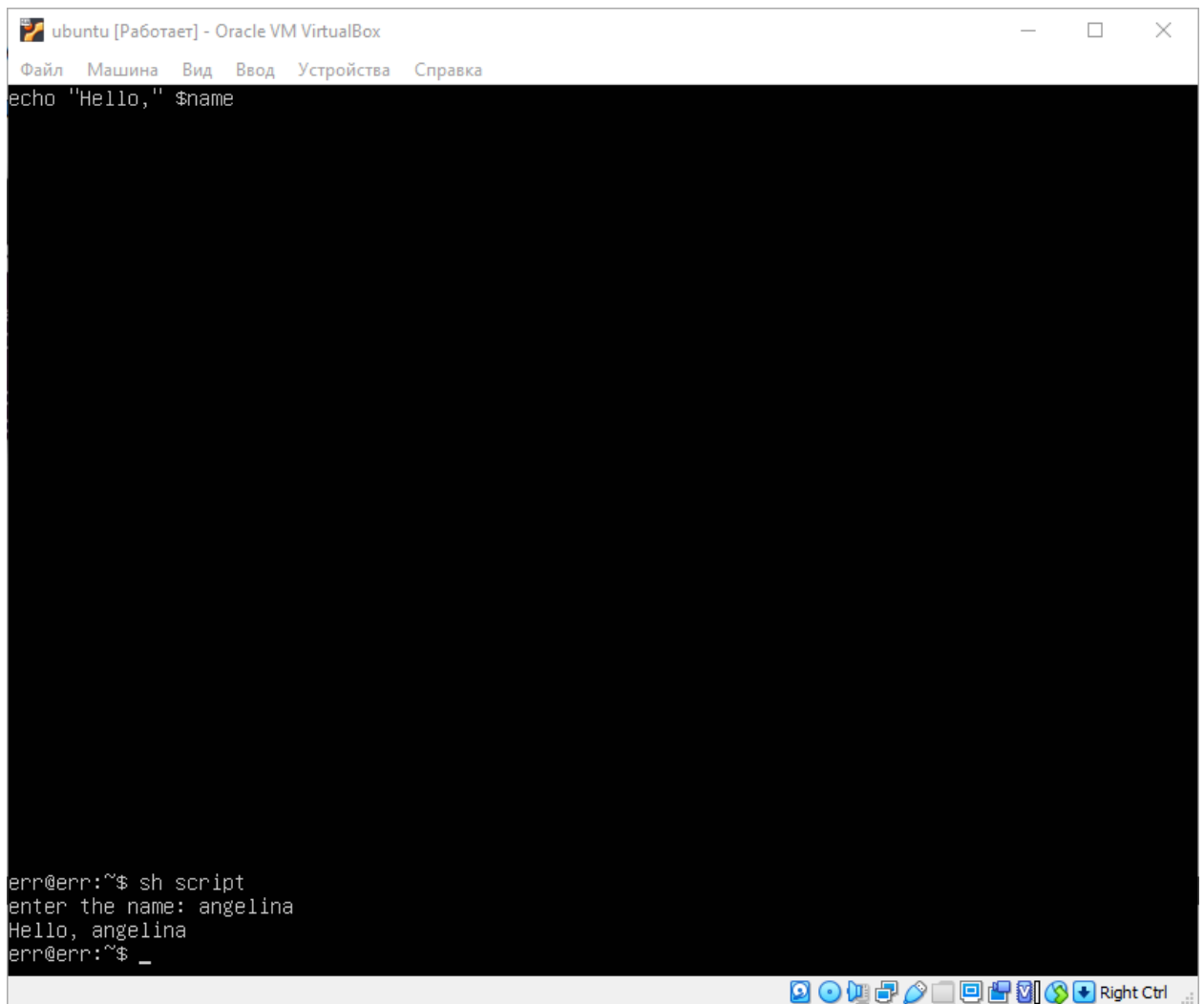
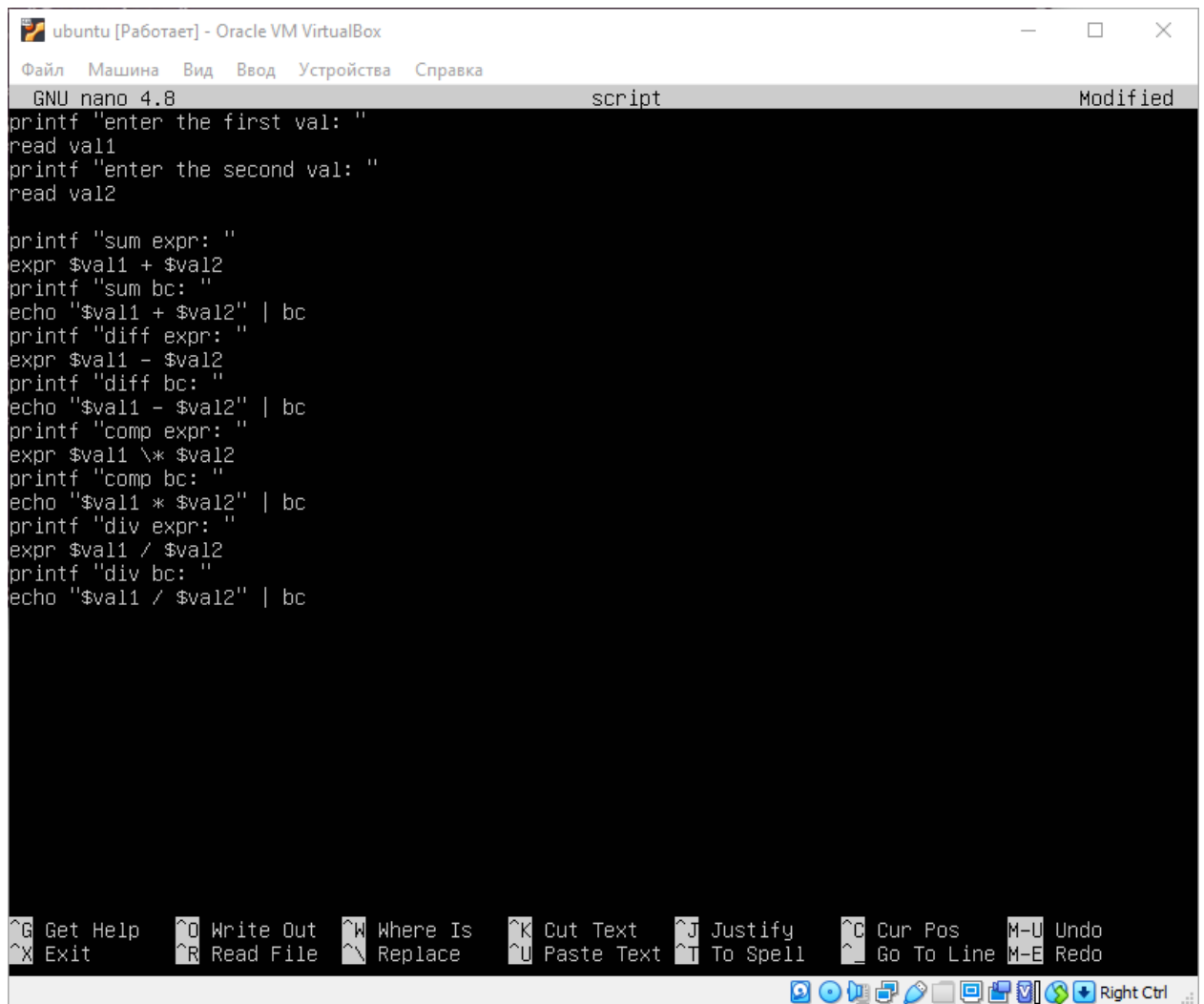


Рисунок 9.2 – Результат выполнения сценария

10. Программа запрашивает значения двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на экран (использовать команды а) EXPR; б) BC).

Исполним оба варианта выполнения задания в одном файле:



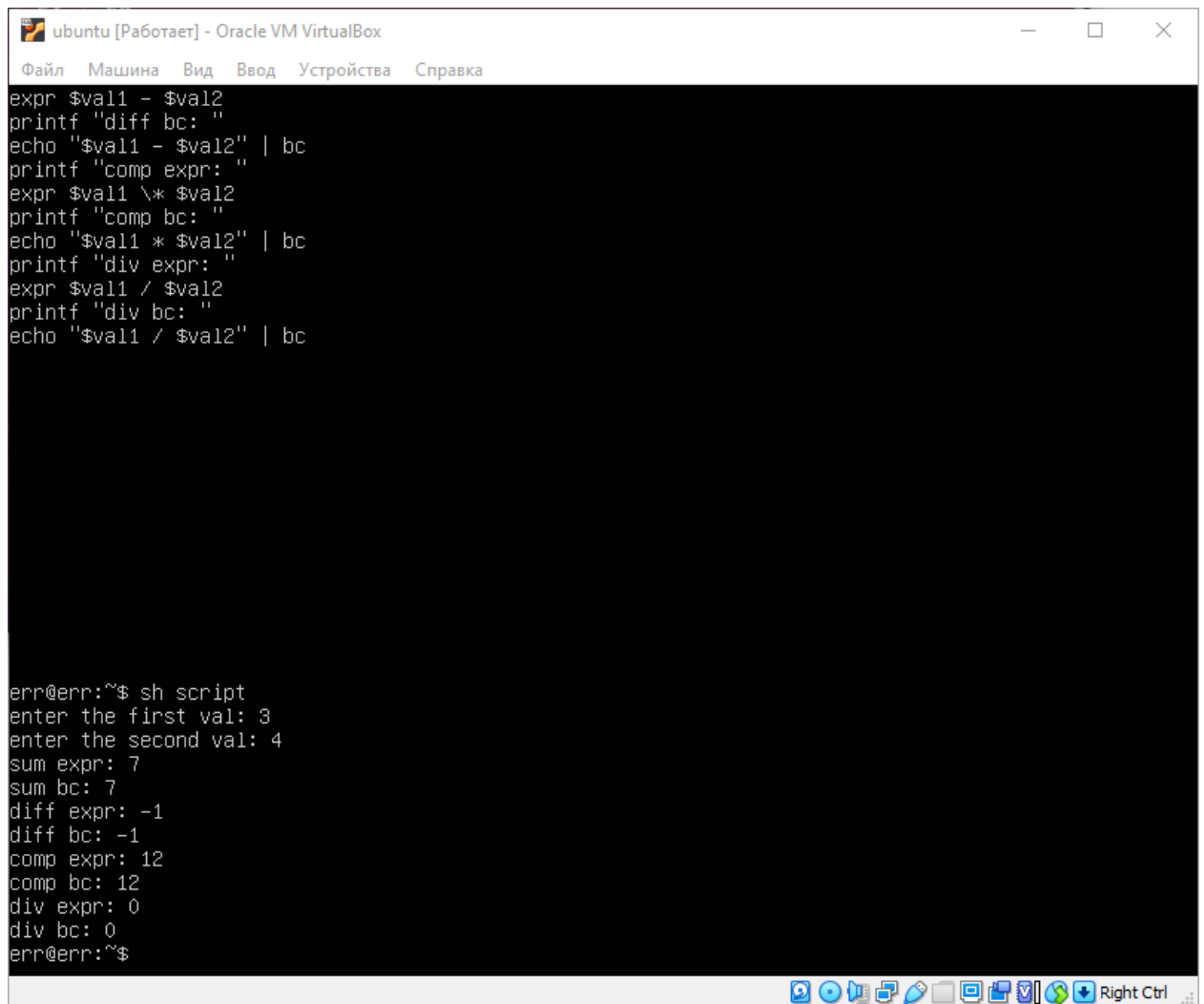
The screenshot shows a window titled "ubuntu [Работает] - Oracle VM VirtualBox". Inside is a terminal window running the GNU nano 4.8 editor. The editor is editing a file named "script". The script contains the following code:

```
printf "enter the first val: "  
read val1  
printf "enter the second val: "  
read val2  
  
printf "sum expr: "  
expr $val1 + $val2  
printf "sum bc: "  
echo "$val1 + $val2" | bc  
printf "diff expr: "  
expr $val1 - $val2  
printf "diff bc: "  
echo "$val1 - $val2" | bc  
printf "comp expr: "  
expr $val1 \* $val2  
printf "comp bc: "  
echo "$val1 * $val2" | bc  
printf "div expr: "  
expr $val1 / $val2  
printf "div bc: "  
echo "$val1 / $val2" | bc
```

The bottom of the window shows a menu bar with various shortcuts and a status bar with icons and "Right Ctrl".

Рисунок 10.1 – Скрипт для задания 10

Исполним скрипт:



```
ubuntu [Работаer] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка

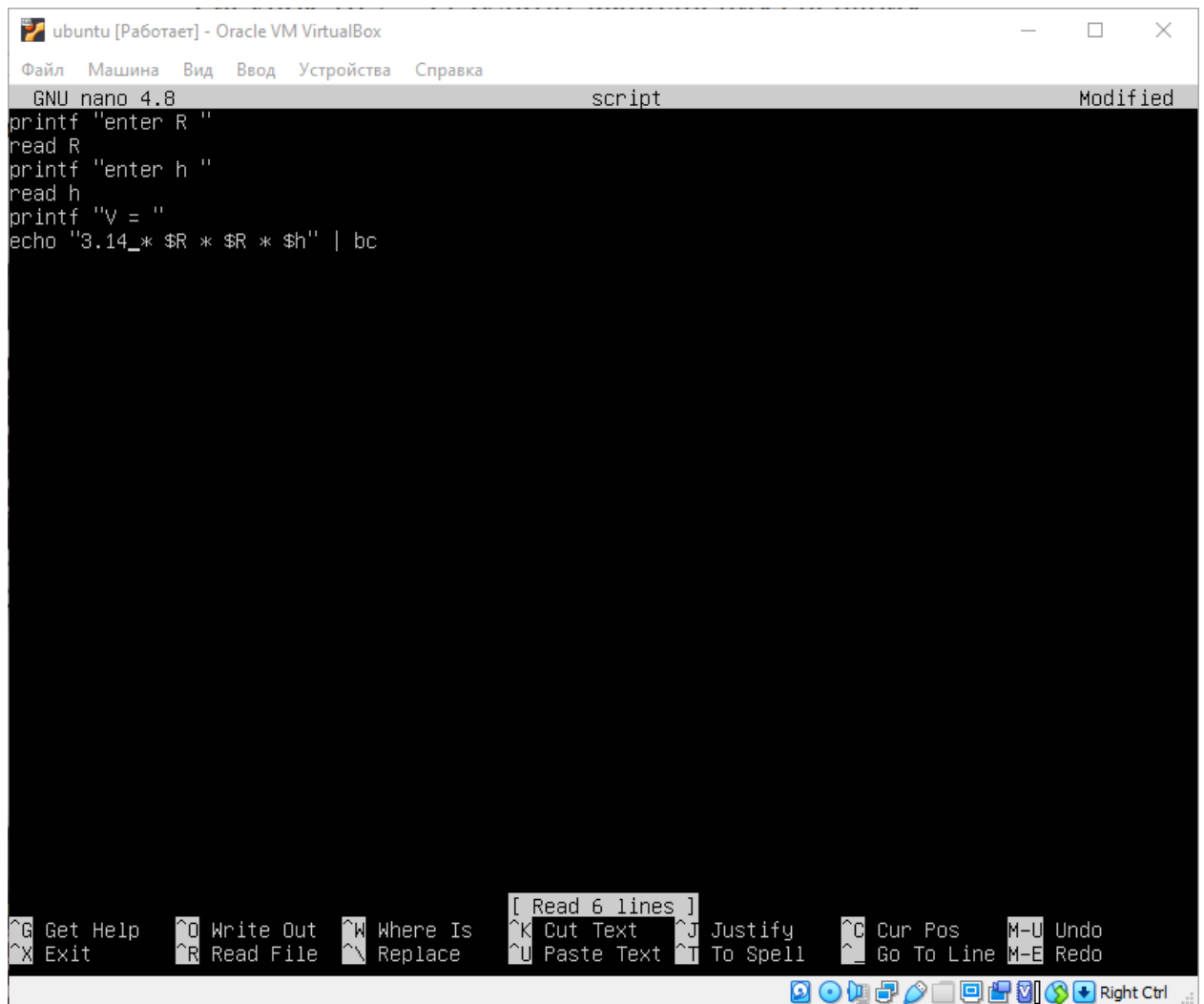
expr $val1 - $val2
printf "diff bc: "
echo "$val1 - $val2" | bc
printf "comp expr: "
expr $val1 \* $val2
printf "comp bc: "
echo "$val1 * $val2" | bc
printf "div expr: "
expr $val1 / $val2
printf "div bc: "
echo "$val1 / $val2" | bc

err@err:~$ sh script
enter the first val: 3
enter the second val: 4
sum expr: 7
sum bc: 7
diff expr: -1
diff bc: -1
comp expr: 12
comp bc: 12
div expr: 0
div bc: 0
err@err:~$
```

Рисунок 10.2 – Результат выполнения сценария

11. Вычислить объем цилиндра. Исходные данные запрашиваются программой. Результат выводится на экран.

Формула объёма цилиндра имеет вид $h \cdot \pi \cdot R^2$. Реализуем её:



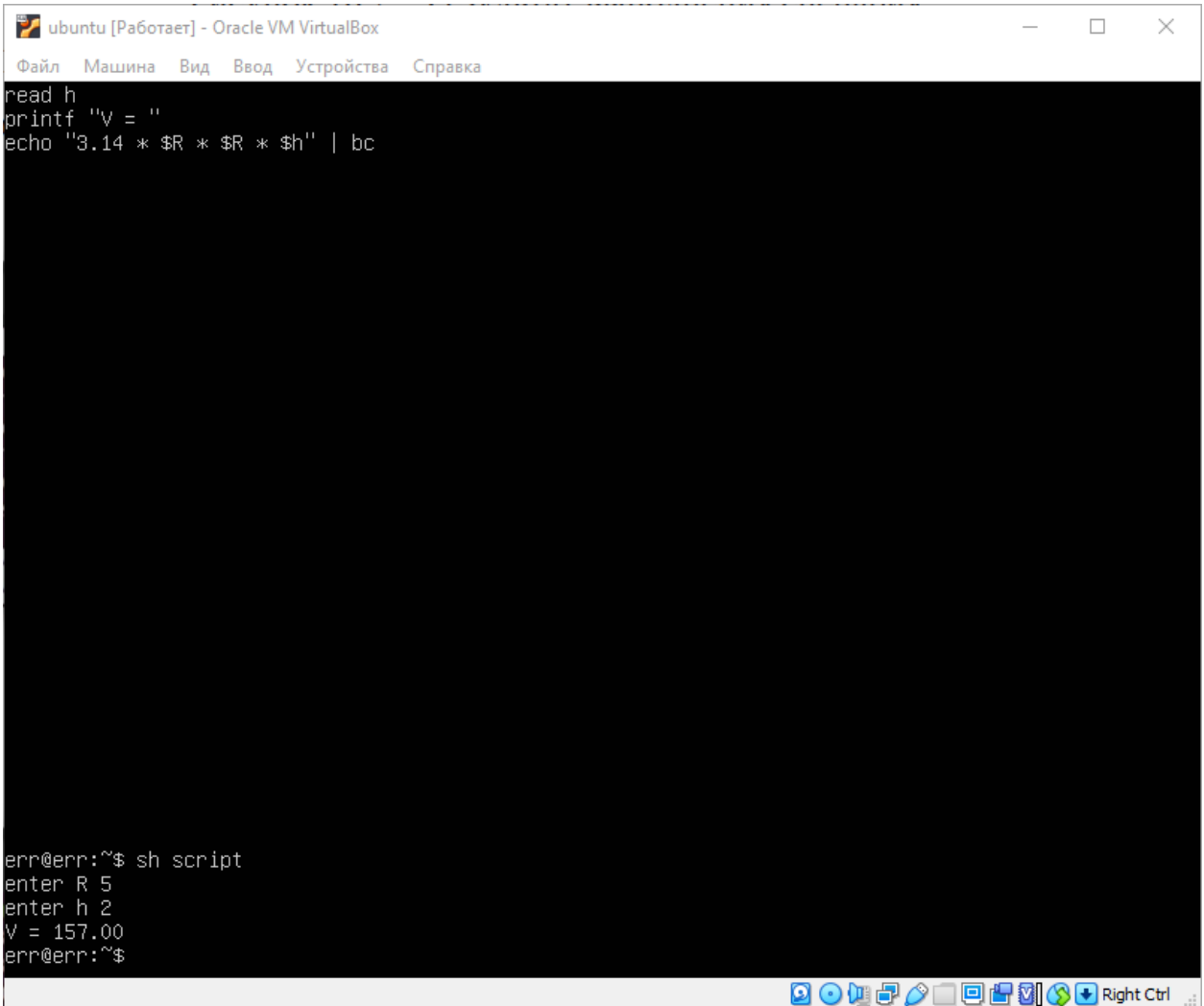
The screenshot shows a window titled "ubuntu [Работает] - Oracle VM VirtualBox". Inside, the GNU nano 4.8 editor is open with a file named "script". The script contains the following code:

```
printf "enter R "  
read R  
printf "enter h "  
read h  
printf "v = "  
echo "3.14_* $R * $R * $h" | bc
```

The bottom of the window displays a menu with various shortcuts and actions, including "Get Help", "Exit", "Write Out", "Read File", "Where Is", "Replace", "Cut Text", "Paste Text", "Justify", "To Spell", "Cur Pos", "Go To Line", "Undo", and "Redo". A status bar at the bottom right indicates "Right Ctrl".

Рисунок 11.1 – Скрипт для выполнения задания 11

Запустим сценарий:



The screenshot shows a terminal window titled "ubuntu [Работает] - Oracle VM VirtualBox". The menu bar includes "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". The terminal content is as follows:

```
read h
printf "V = "
echo "3.14 * $R * $R * $h" | bc
```



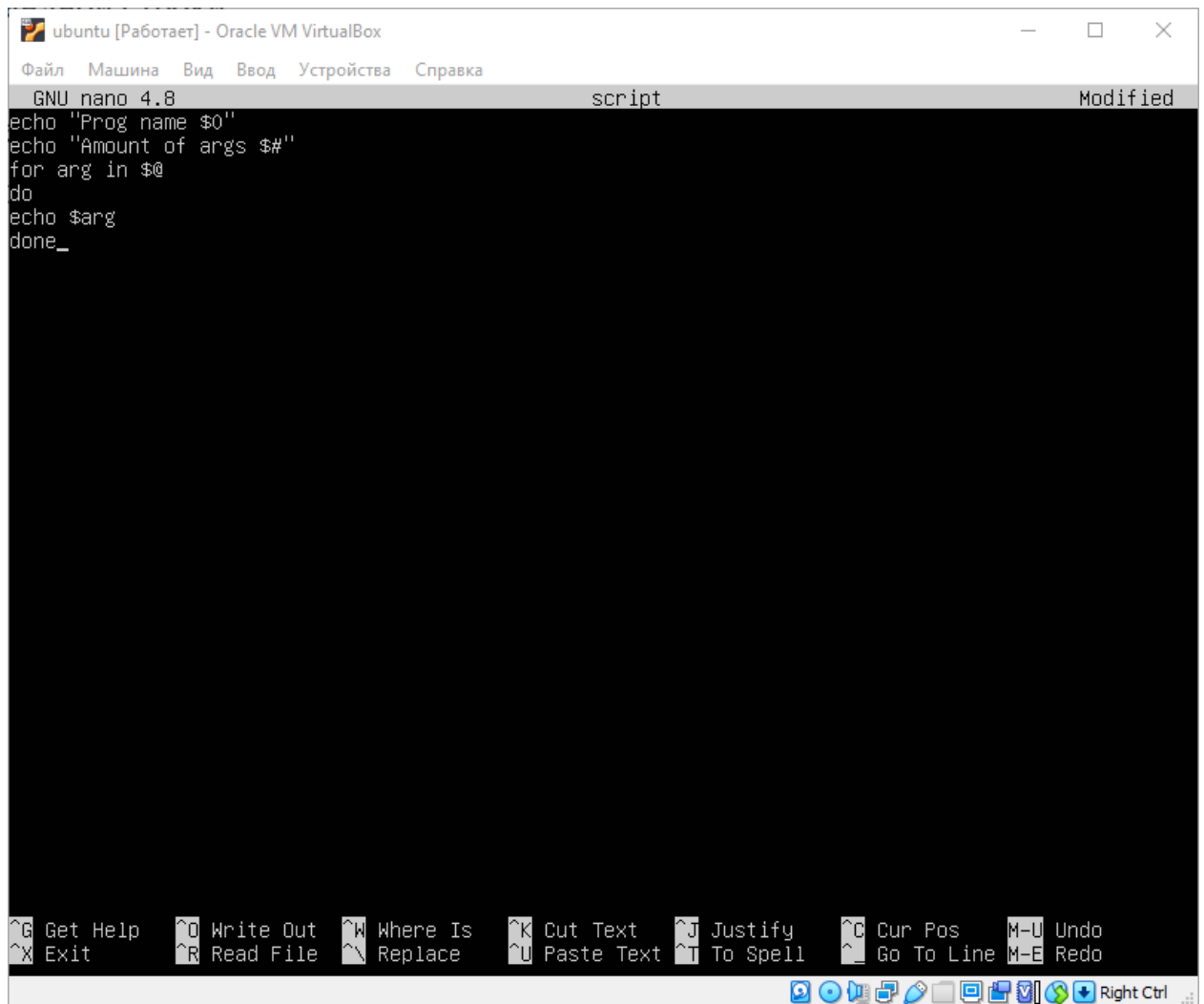
```
err@err:~$ sh script
enter R 5
enter h 2
V = 157.00
err@err:~$
```

The bottom of the window features a taskbar with various icons and a "Right Ctrl" label.

Рисунок 11.2 – Результат выполнения сценария

12. Используя позиционные параметры, отобразить имя программы, количество аргументов командной строки, значение каждого аргумента командной строки.

Для обозначения количества аргументов командной строки используем переменную \$#. А для обращения к аргументу командной строки будем использовать \$<num>, где num – номер аргумента. Напишем скрипт:



The screenshot shows a window titled "ubuntu [Работает] - Oracle VM VirtualBox". Inside the window is a terminal running the GNU nano 4.8 editor. The editor is editing a file named "script". The script content is as follows:

```
GNU nano 4.8 script Modified
echo "Prog name $0"
echo "Amount of args $#"
```

The script continues with a loop structure, but the text is partially cut off in the image. The visible lines are:

```
for arg in $@
do
echo $arg
done_
```

The bottom of the window shows the nano editor's command shortcuts, such as ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^C Cur Pos, M-U Undo, ^X Exit, ^R Read File, ^_ Replace, ^U Paste Text, ^T To Spell, ^_ Go To Line, M-E Redo, and a Right Ctrl key icon.

Рисунок 12.1 – Скрипт для задания 12

Исполним сценарий:

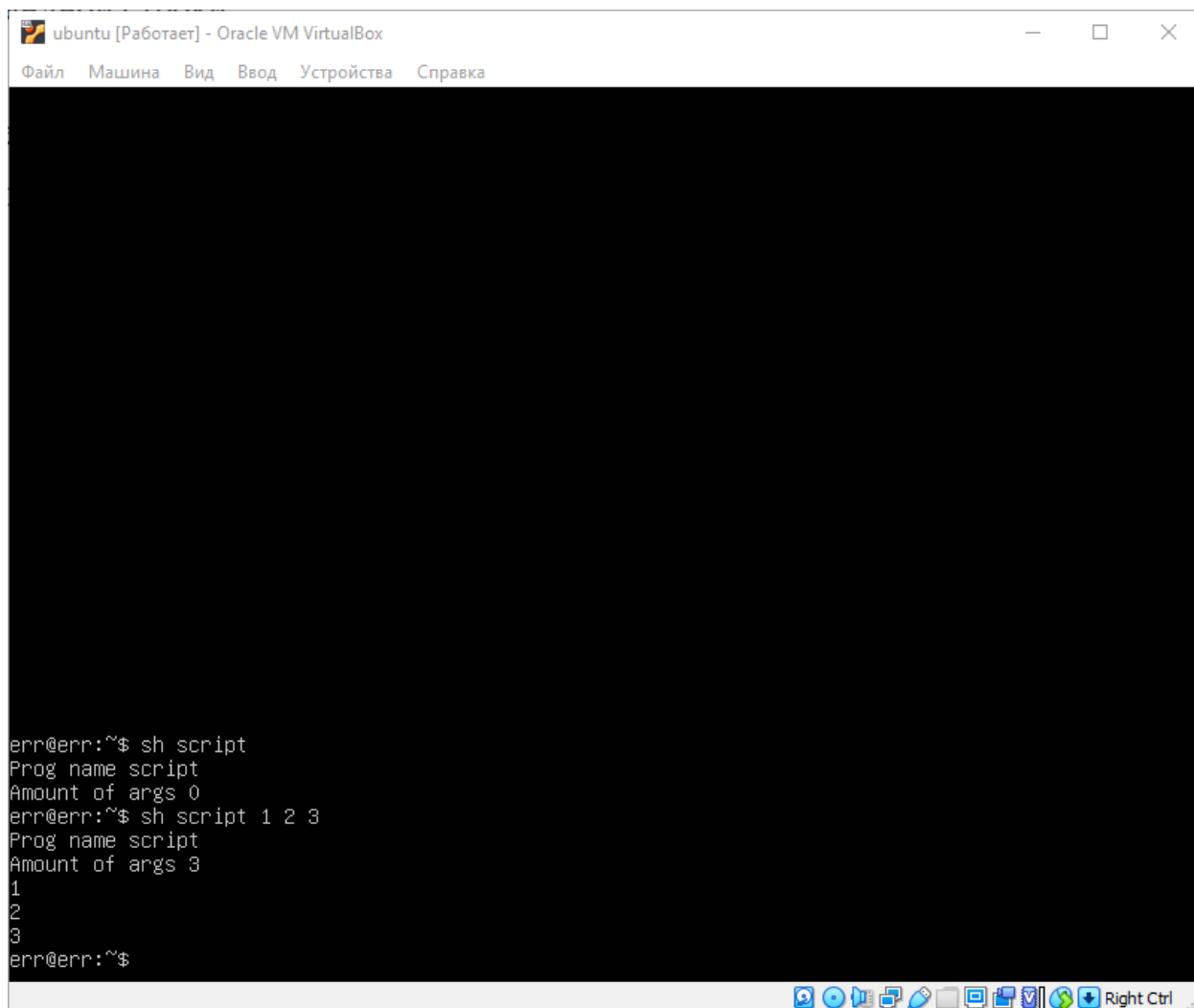


Рисунок 12.2 – Результат выполнения сценария

13. Используя позиционный параметр, отобразить содержимое текстового файла, указанного в качестве аргумента командной строки. После паузы экран очищается.

Напишем сценарий:

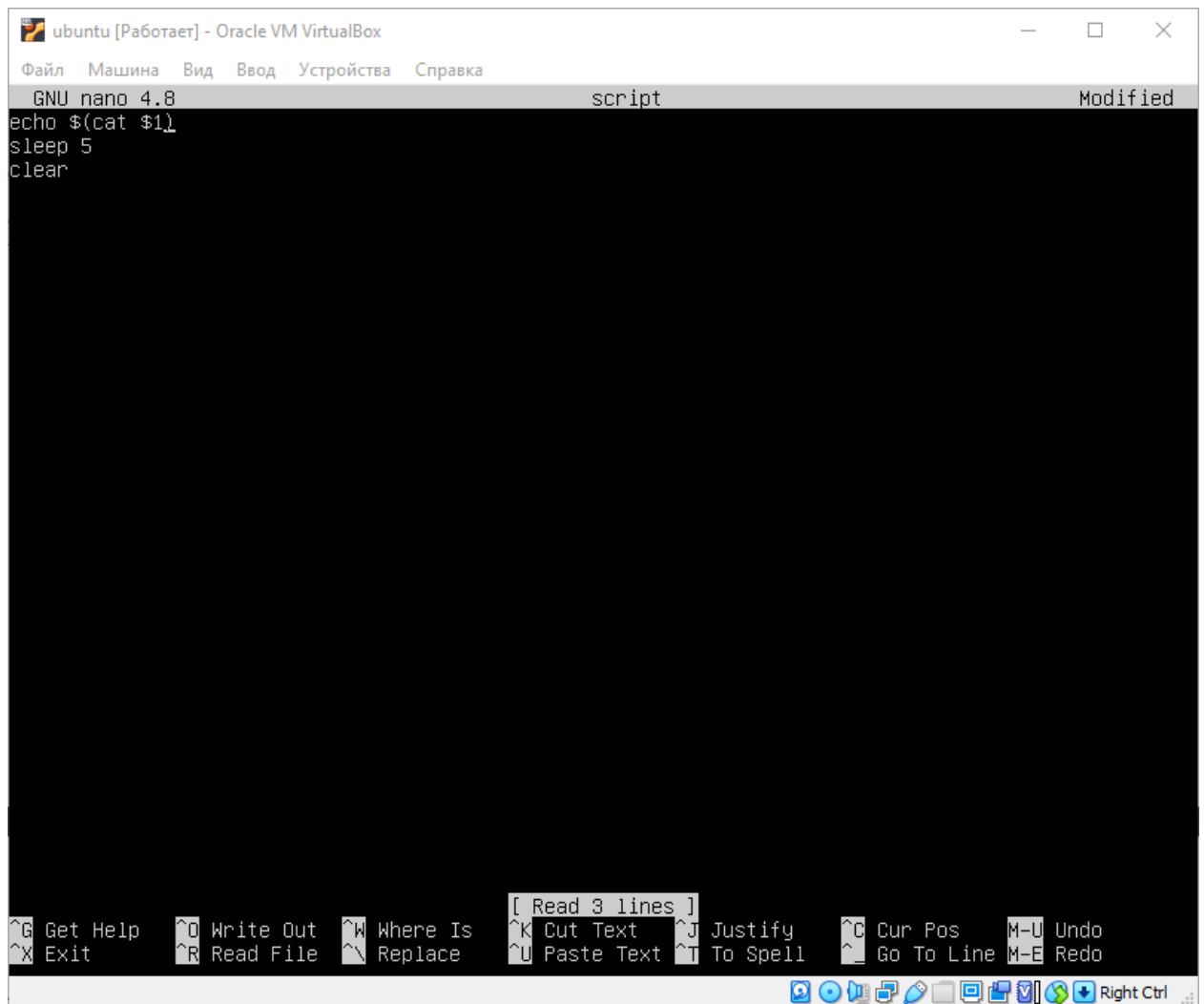


Рисунок 13.1 – Скрипт для задания 13

Запустим сценарий на выполнение:

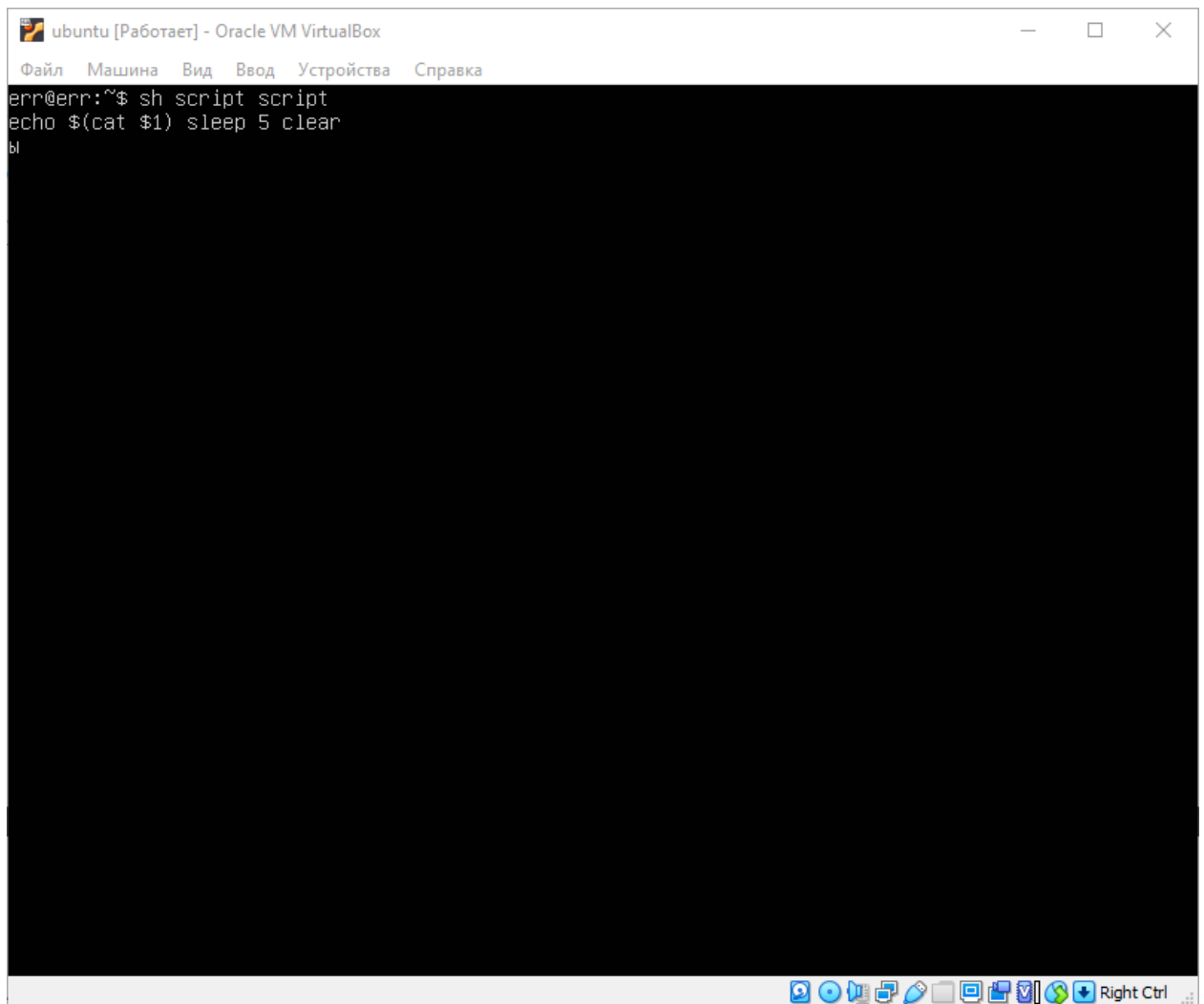


Рисунок 13.2 – Результат выполнения сценария

14. Используя оператор FOR, отобразить содержимое текстовых файлов текущего каталога поэкранно.

Для поиска в данной директории используем конструкцию `./*`. Затем проверим, не является ли выбранный файл директорией, чтобы наш скрипт сработал для файлов:

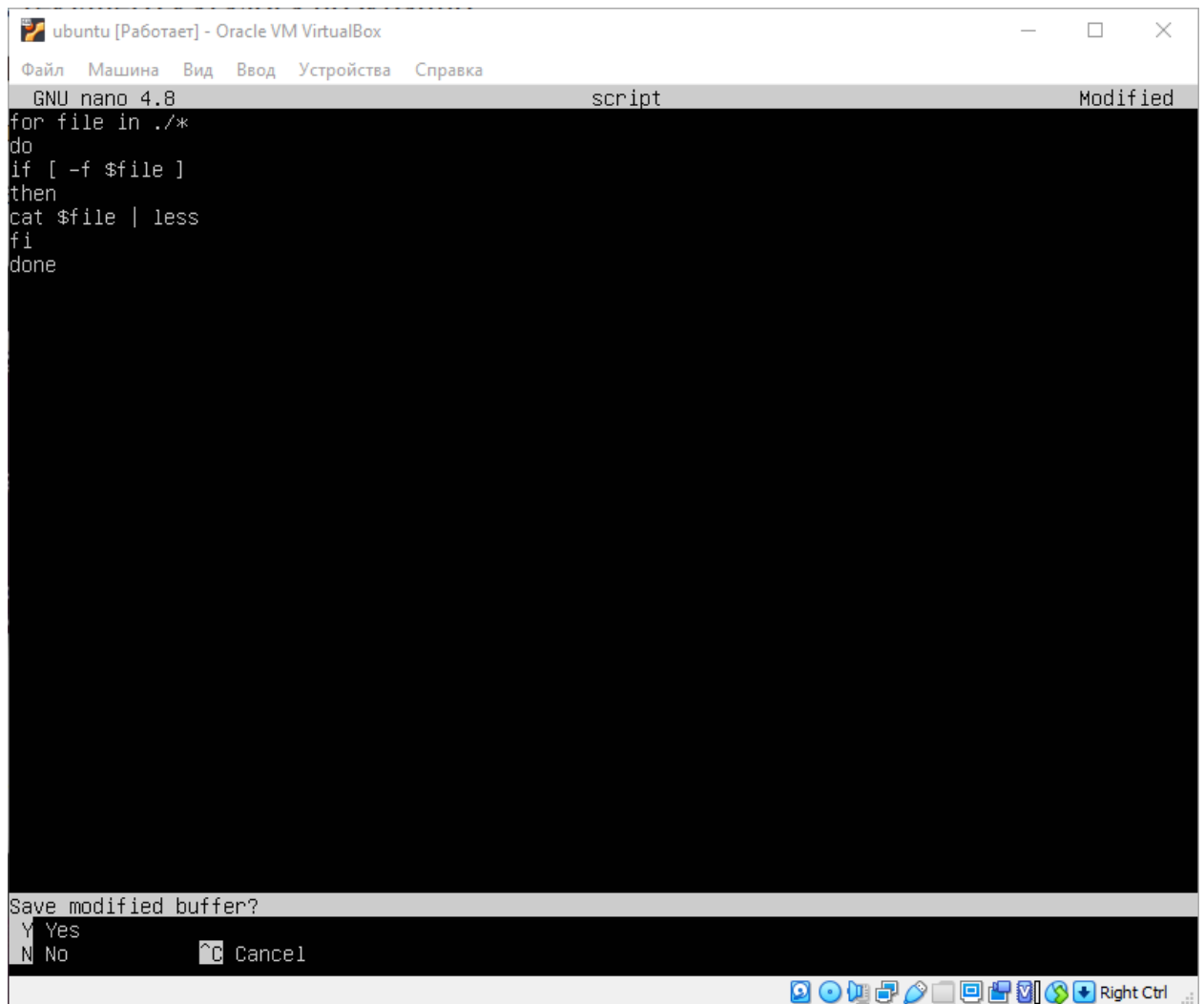


Рисунок 14.1 – Скрипт для задания 14

Запустим скрипт:

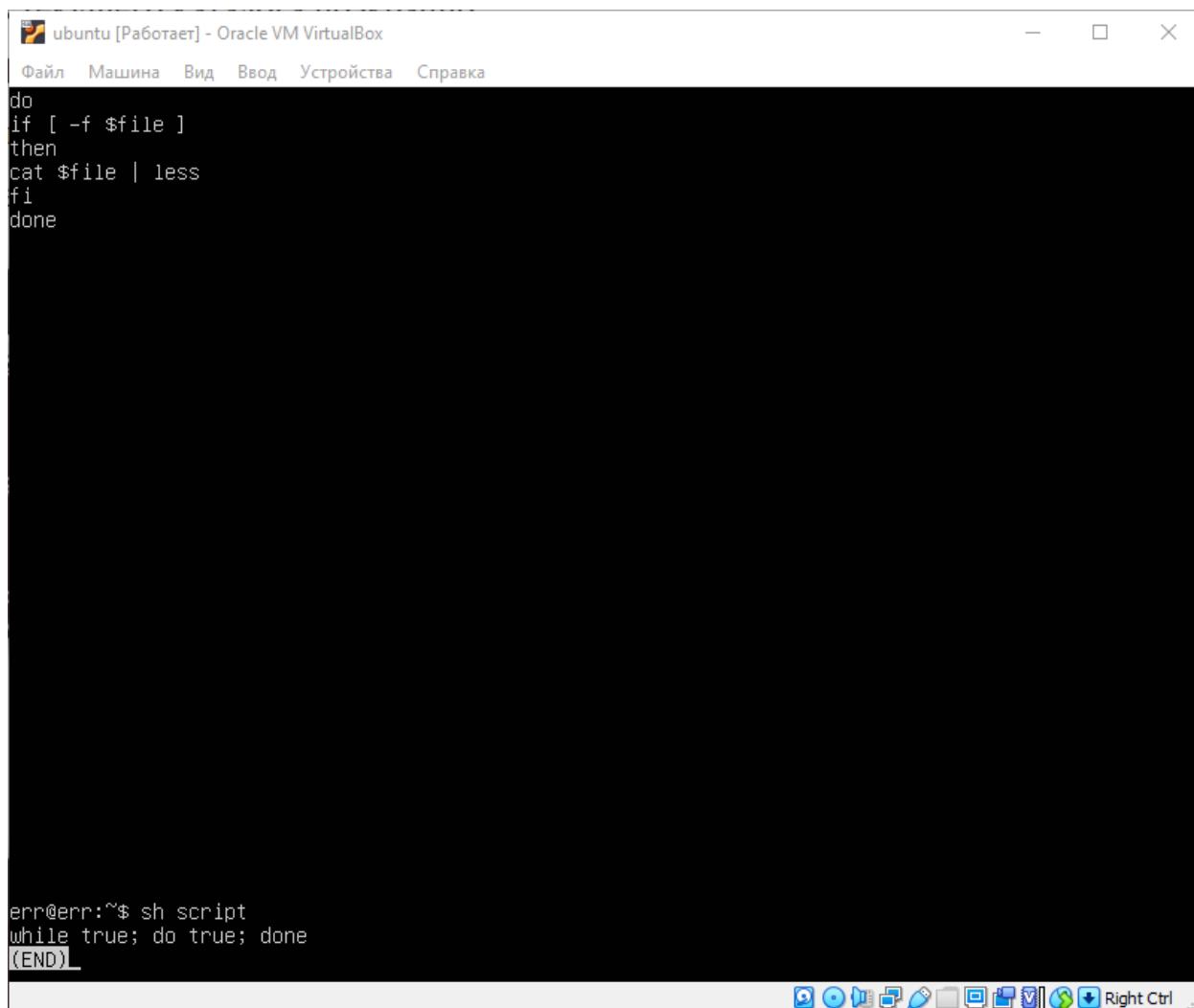
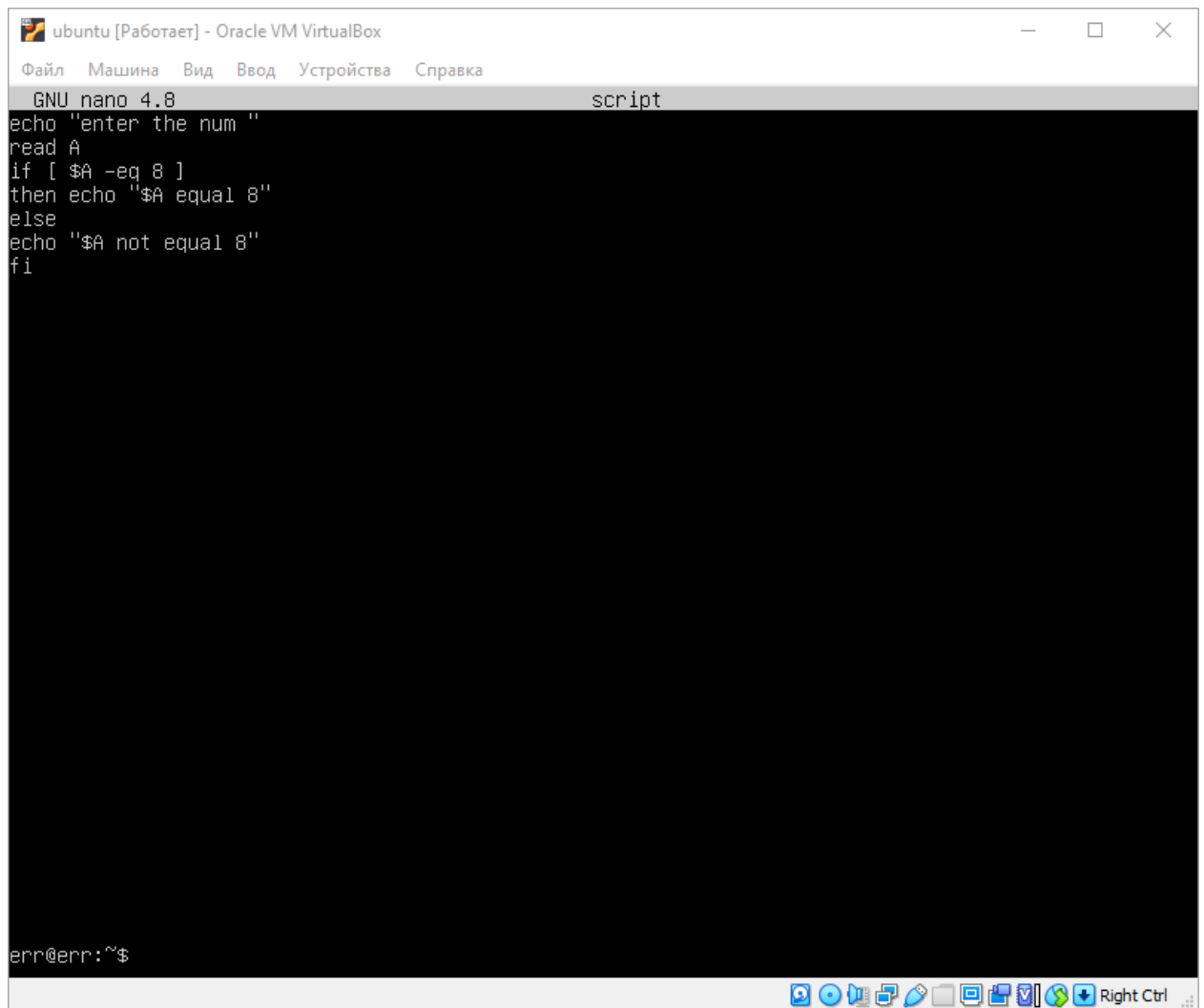


Рисунок 14.2 – Результат выполнения сценария

15. Программой запрашивается ввод числа, значение которого затем сравнивается с допустимым значением. В результате этого сравнения на экран выдаются соответствующие сообщения.

Для сравнения чисел в оболочке `bash` используется конструкция `—eq`. Используем это в написании сценария:



```
ubuntu [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
GNU nano 4.8                                script
echo "enter the num "
read A
if [ $A -eq 8 ]
then echo "$A equal 8"
else
echo "$A not equal 8"
fi

err@err:~$
```

Рисунок 15.1 – Скрипт для задания 15

Выполним скрипт:

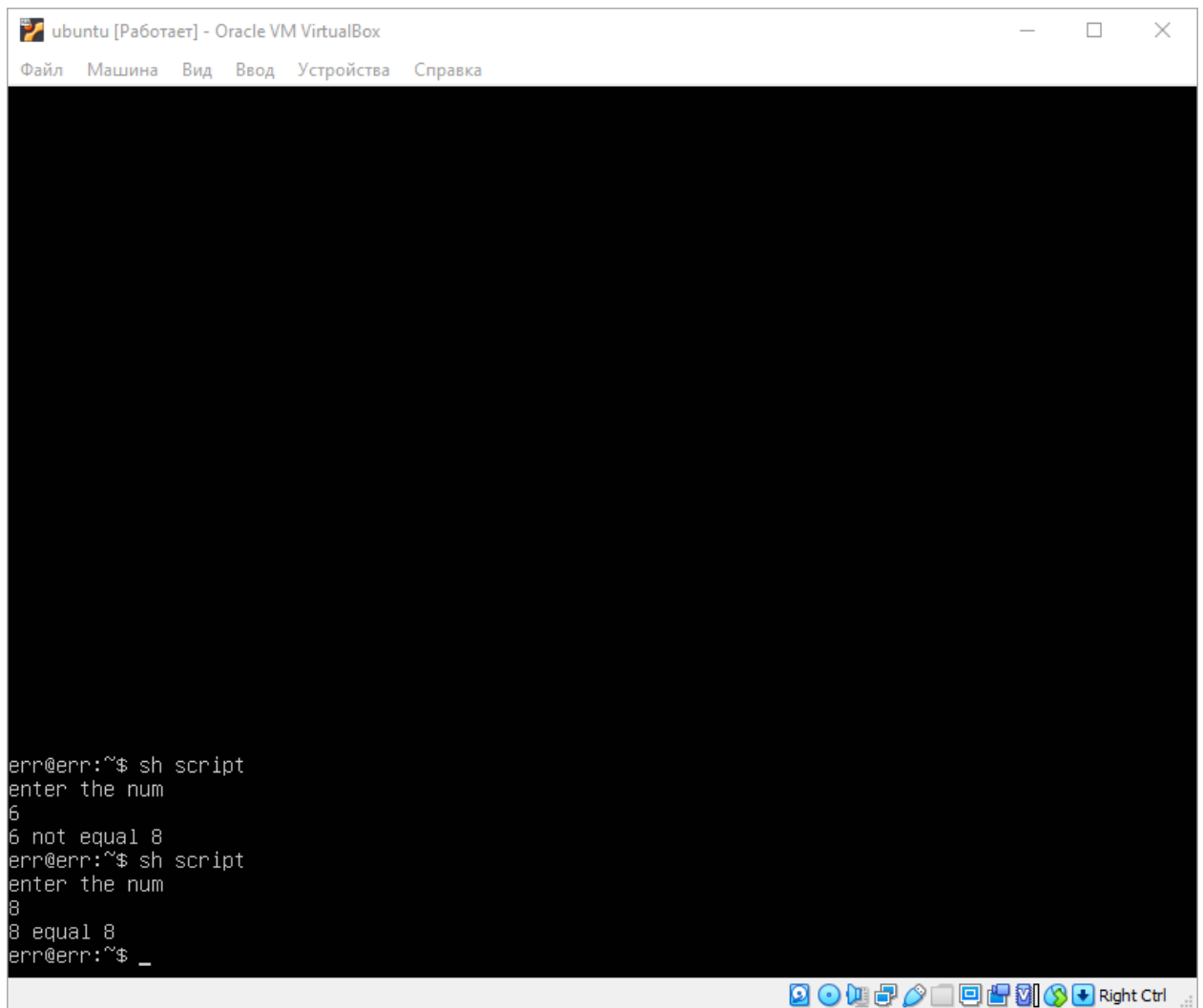


Рисунок 15.2 – Результат выполнения сценария

16. Программой запрашивается год, определяется, високосный ли он. Результат выдается на экран.

Для того, чтобы год был високосным, он должен делиться без остатка на 4. Реализуем это:

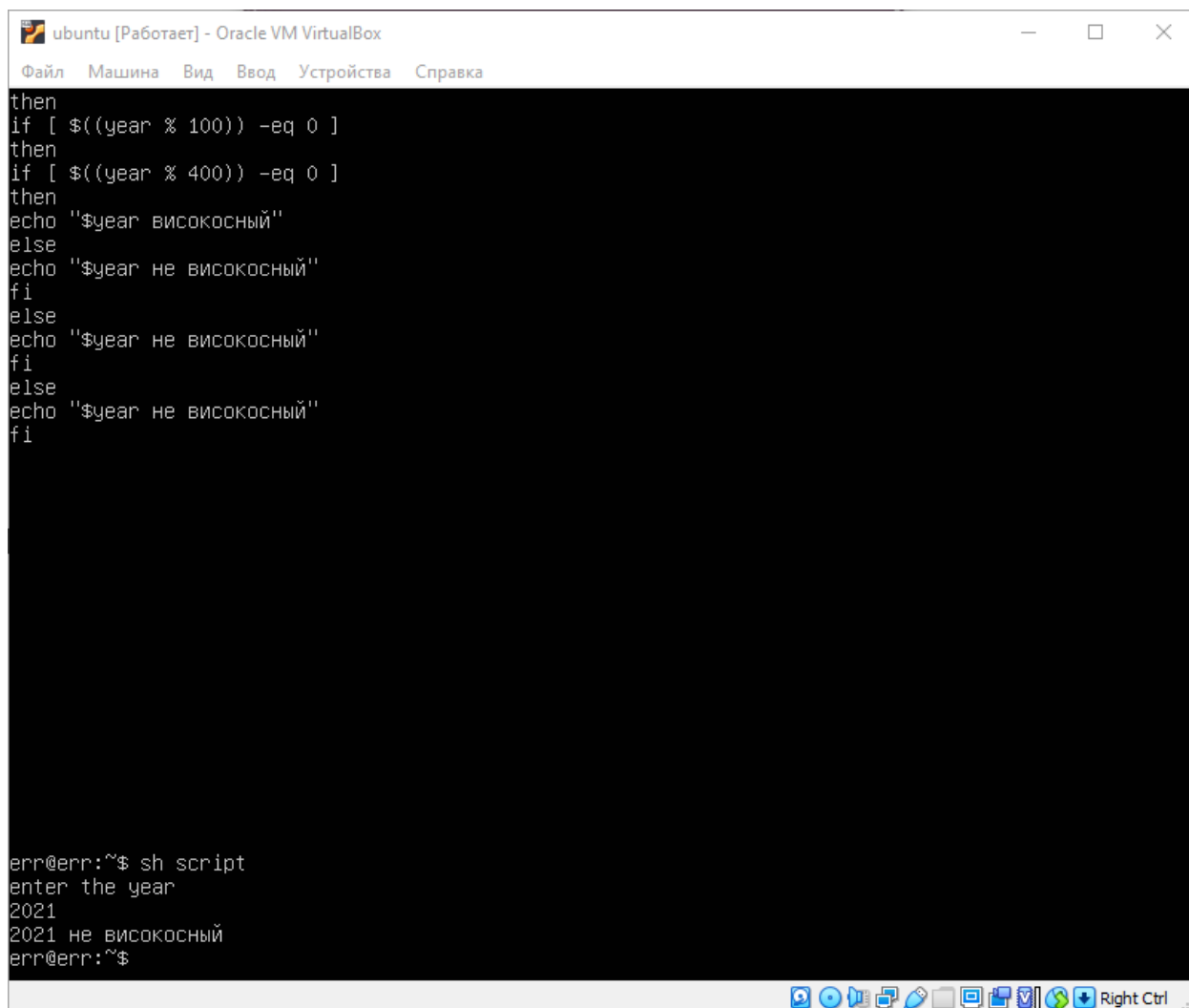
The screenshot shows a terminal window titled "ubuntu [Работает] - Oracle VM VirtualBox". Inside the terminal, the GNU nano 4.8 editor is open, editing a file named "script". The script is a shell script that prompts the user to enter a year and then checks if it is a leap year. The script uses a series of if-then-else statements to check the year modulo 4, 100, and 400. The output of the script is in Russian: "високосный" for leap years and "не високосный" for non-leap years. The nano editor's status bar at the bottom shows various keyboard shortcuts and the current cursor position.

```
GNU nano 4.8 script
echo "enter the year "
read year
if [  $$(year \% 4)$  -eq 0 ]
then
if [  $$(year \% 100)$  -eq 0 ]
then
if [  $$(year \% 400)$  -eq 0 ]
then
echo "$year високосный"
else
echo "$year не високосный"
fi
else
echo "$year не високосный"
fi
else
echo "$year не високосный"
fi
fi
```

^G Get Help ^O Write Out ^W Where Is [Read 18 lines] ^K Cut Text ^J Justify ^C Cur Pos M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo

Рисунок 16.1 – Скрипт для задания 16

Исполним данный сценарий:



The screenshot shows a terminal window titled "ubuntu [Работает] - Oracle VM VirtualBox". The menu bar includes "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". The terminal content is as follows:

```
then
if [ $((year % 100)) -eq 0 ]
then
if [ $((year % 400)) -eq 0 ]
then
echo "$year високосный"
else
echo "$year не високосный"
fi
else
echo "$year не високосный"
fi
else
echo "$year не високосный"
fi
fi

err@err:~$ sh script
enter the year
2021
2021 не високосный
err@err:~$
```

The terminal window has a standard Linux desktop environment at the bottom with various icons and a "Right Ctrl" button.

Рисунок 16.2 – Результат выполнения сценария

17. Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значения инкрементируются.

Напишем сценарий:

```
GNU nano 4.8                                test
read a
read b
read A
read B
if [ $a -gt $A ] || [ $b -gt $A ]
then
while [ $a -lt $B ] || [ $b -lt $B ]
do
a=$(expr $a + 1)
b=$(expr $b + 1)
done
fi
echo $a
echo $b
```

[Read 14 lines]

^G Get Help	^O Write Out	^W Where Is	^K Cut Text	^J Justify	^C Cur Pos	M-U Undo
^X Exit	^R Read File	^N Replace	^U Paste Text	^T To Spell	^_ Go To Line	M-E Redo

Рисунок 17.1 – Скрипт для задания 17

Запустим скрипт на выполнение:

```
while [ $a -lt $B ] || [ $b -lt $B ]
do
a=$(expr $a + 1)
b=$(expr $b + 1)
done
fi
echo $a
echo $b
```

```
err@err:~$ sh test
```

```
4
```

```
6
```

```
3
```

```
5
```

```
5
```

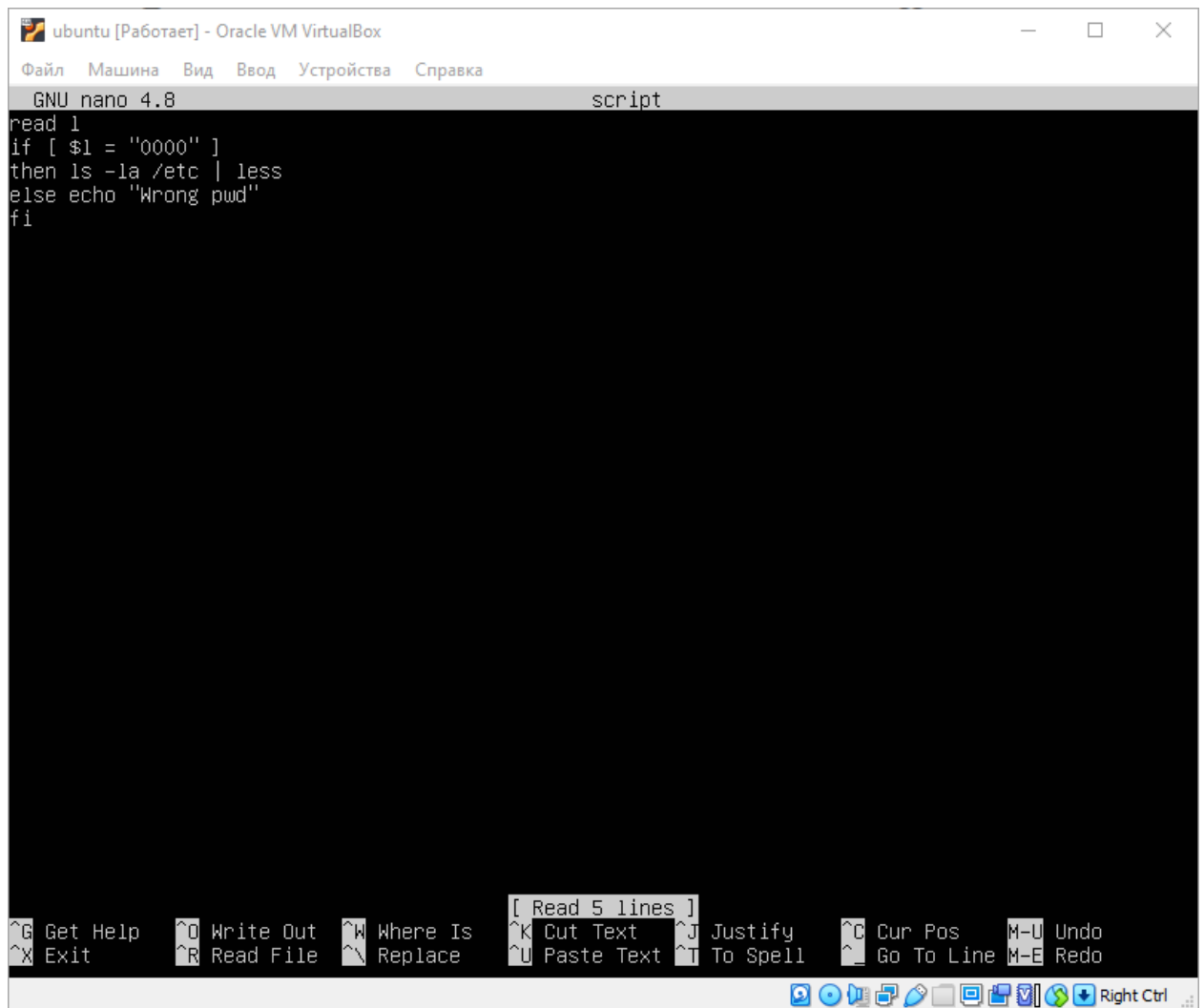
```
7
```

```
err@err:~$
```

Рисунок 17.2 – Результат выполнения сценария

18. В качестве аргумента командной строки указывается пароль. Если пароль введен верно, постранично отображается в длинном формате с указанием скрытых файлов содержимое каталога /etc.

Для сравнения двух строк используется оператор =. Напишем скрипт:



The image shows a screenshot of a nano editor window titled "ubuntu [Работает] - Oracle VM VirtualBox". The window has a menu bar with options: "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". The editor is editing a file named "script". The content of the script is as follows:

```
read l
if [ $1 = "0000" ]
then ls -la /etc | less
else echo "Wrong pwd"
fi
```

The bottom of the window displays a status bar with various keyboard shortcuts and a toolbar with icons for navigation and editing. The status bar includes:

- ^G Get Help
- ^O Write Out
- ^W Where Is
- [Read 5 lines]
- ^K Cut Text
- Justify
- ^C Cur Pos
- M-U Undo
- ^X Exit
- ^R Read File
- ^N Replace
- ^U Paste Text
- To Spell
- Go To Line
- M-E Redo

The toolbar at the bottom right includes icons for undo, redo, find, and other functions, along with a "Right Ctrl" button.

Рисунок 18.1 – Скрипт для задания 18

Исполним сценарий:

```
total 812
drwxr-xr-x 97 root root      4096 ноя 19 18:57 .
drwxr-xr-x 20 root root      4096 ноя 19 18:49 ..
-rw-r--r--  1 root root      3028 авг 24 08:42 adduser.conf
drwxr-xr-x  2 root root      4096 авг 24 08:47 alternatives
drwxr-xr-x  3 root root      4096 авг 24 08:47 apparmor
drwxr-xr-x  7 root root      4096 авг 24 08:47 apparmor.d
drwxr-xr-x  3 root root      4096 ноя 19 18:54 apport
drwxr-xr-x  7 root root      4096 ноя 19 18:48 apt
-rw-r----- 1 root daemon    144 ноя 12 2018 at.deny
-rw-r--r--  1 root root     2319 фев 25 2020 bash.bashrc
-rw-r--r--  1 root root      45 янв 26 2020 bash_completion
drwxr-xr-x  2 root root      4096 ноя 19 18:54 bash_completion.d
-rw-r--r--  1 root root      367 апр 14 2020 bindresvport.blacklist
drwxr-xr-x  2 root root      4096 апр 22 2020 binfmt.d
drwxr-xr-x  2 root root      4096 авг 24 08:47 byobu
drwxr-xr-x  3 root root      4096 авг 24 08:42 ca-certificates
-rw-r--r--  1 root root     6570 ноя 19 18:54 ca-certificates.conf
-rw-r--r--  1 root root     6569 авг 24 08:45 ca-certificates.conf.dpkg-old
drwxr-xr-x  2 root root      4096 авг 24 08:47 calendar
drwxr-xr-x  4 root root      4096 ноя 19 18:50 cloud
drwxr-xr-x  2 root root      4096 ноя 19 18:55 console-setup
drwxr-xr-x  2 root root      4096 авг 24 08:47 cron.d
drwxr-xr-x  2 root root      4096 ноя 19 18:54 cron.daily
drwxr-xr-x  2 root root      4096 авг 24 08:43 cron.hourly
drwxr-xr-x  2 root root      4096 авг 24 08:43 cron.monthly
-rw-r--r--  1 root root     1042 фев 13 2020 crontab
drwxr-xr-x  2 root root      4096 авг 24 08:47 cron.weekly
drwxr-xr-x  2 root root      4096 авг 24 08:47 cryptsetup-initramfs
-rw-r--r--  1 root root      54 авг 24 08:46 crypttab
drwxr-xr-x  4 root root      4096 авг 24 08:42 dbus-1
drwxr-xr-x  3 root root      4096 авг 24 08:46 dconf
-rw-r--r--  1 root root     2969 авг 3 2019 debconf.conf
-rw-r--r--  1 root root      13 дек 5 2019 debian_version
drwxr-xr-x  3 root root      4096 ноя 19 18:54 default
-rw-r--r--  1 root root      604 сен 15 2018 deluser.conf
:
```

Рисунок 18.2 – Результат выполнения сценария

19. Проверить, существует ли файл. Если да, выводится на экран его содержимое, если нет - выдается соответствующее сообщение.

Чтобы проверить файл на существование, используем конструкцию `-e`:

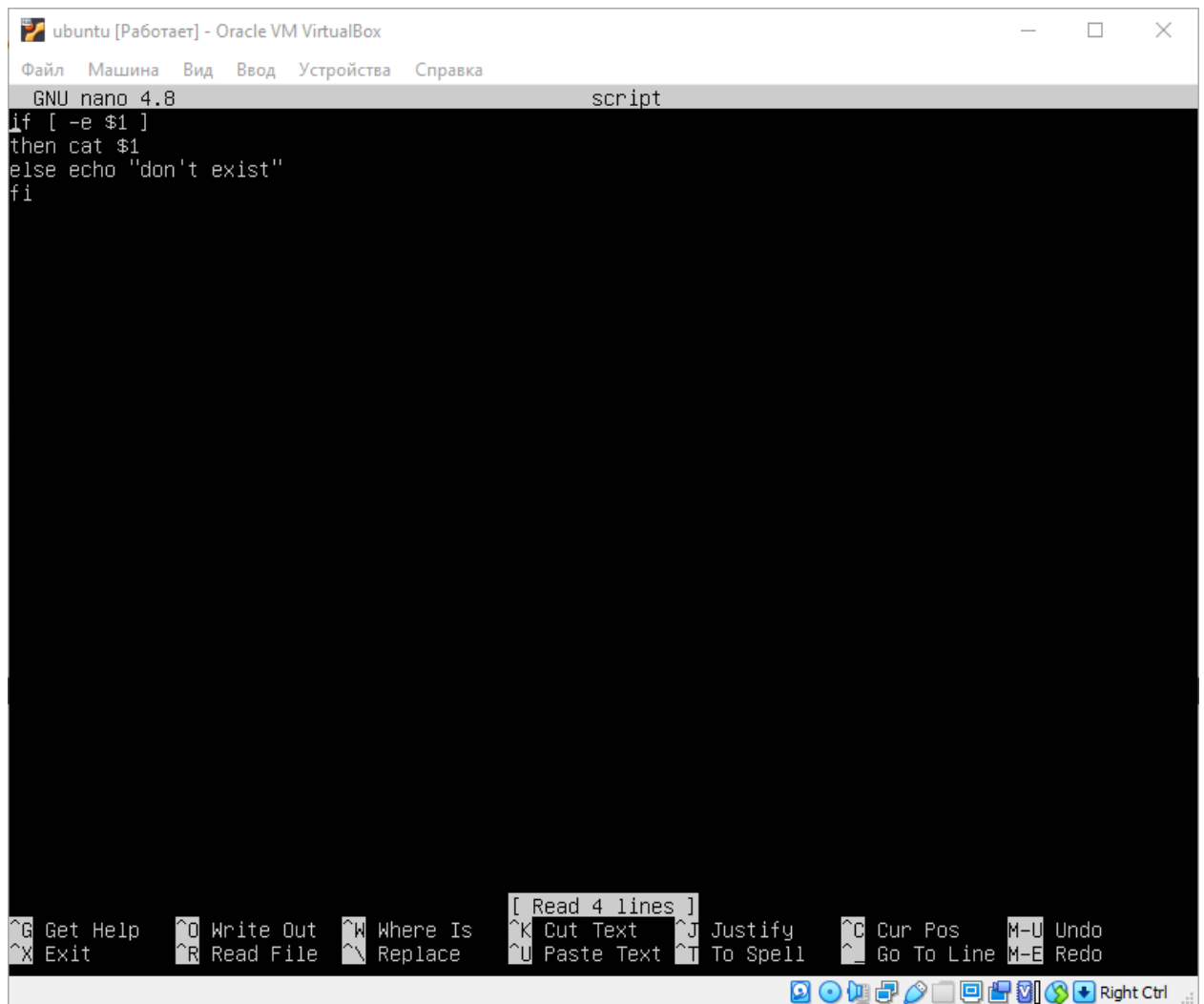


Рисунок 19.1 – Скрипт для задания 19

Запустим сценарий:

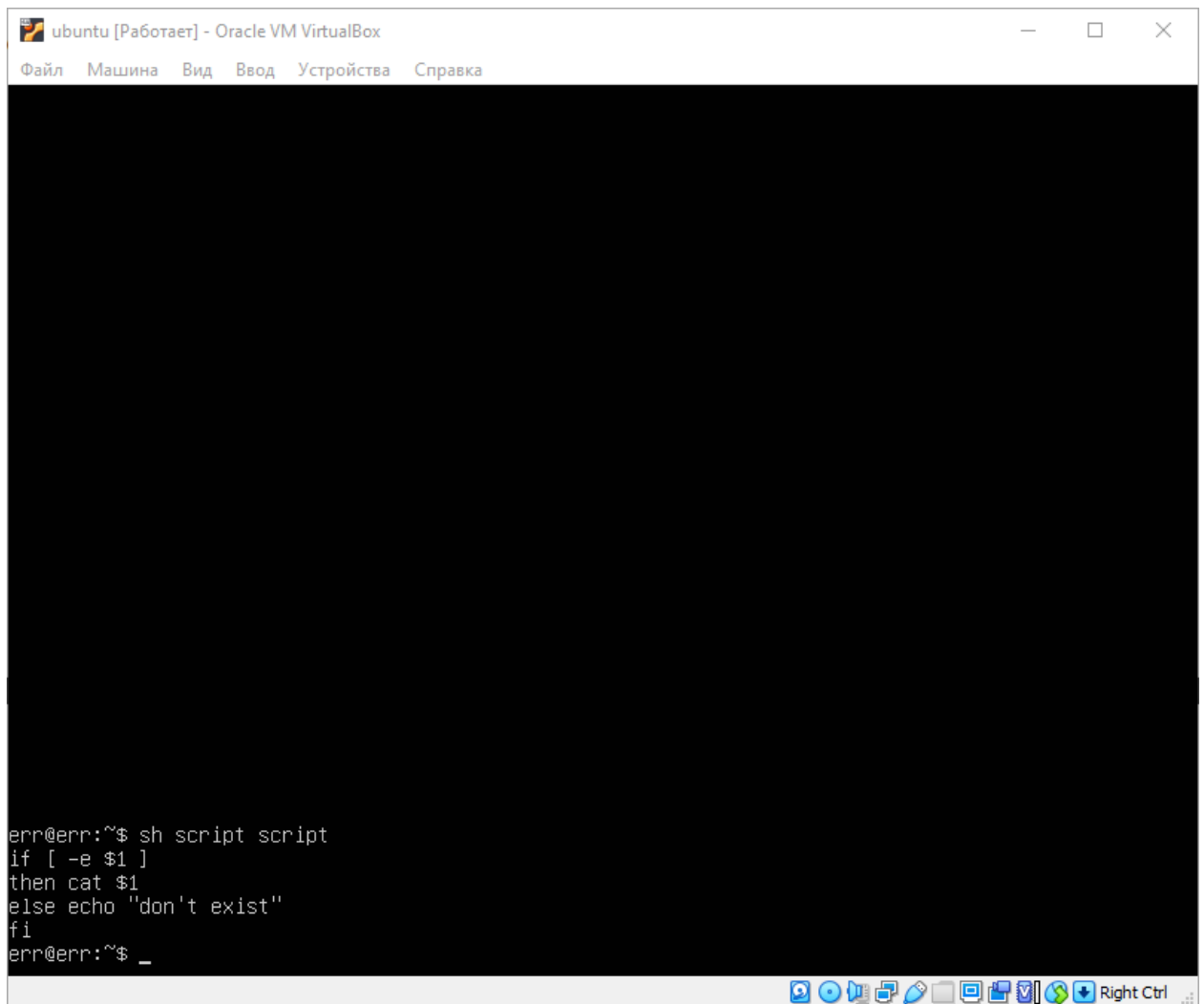
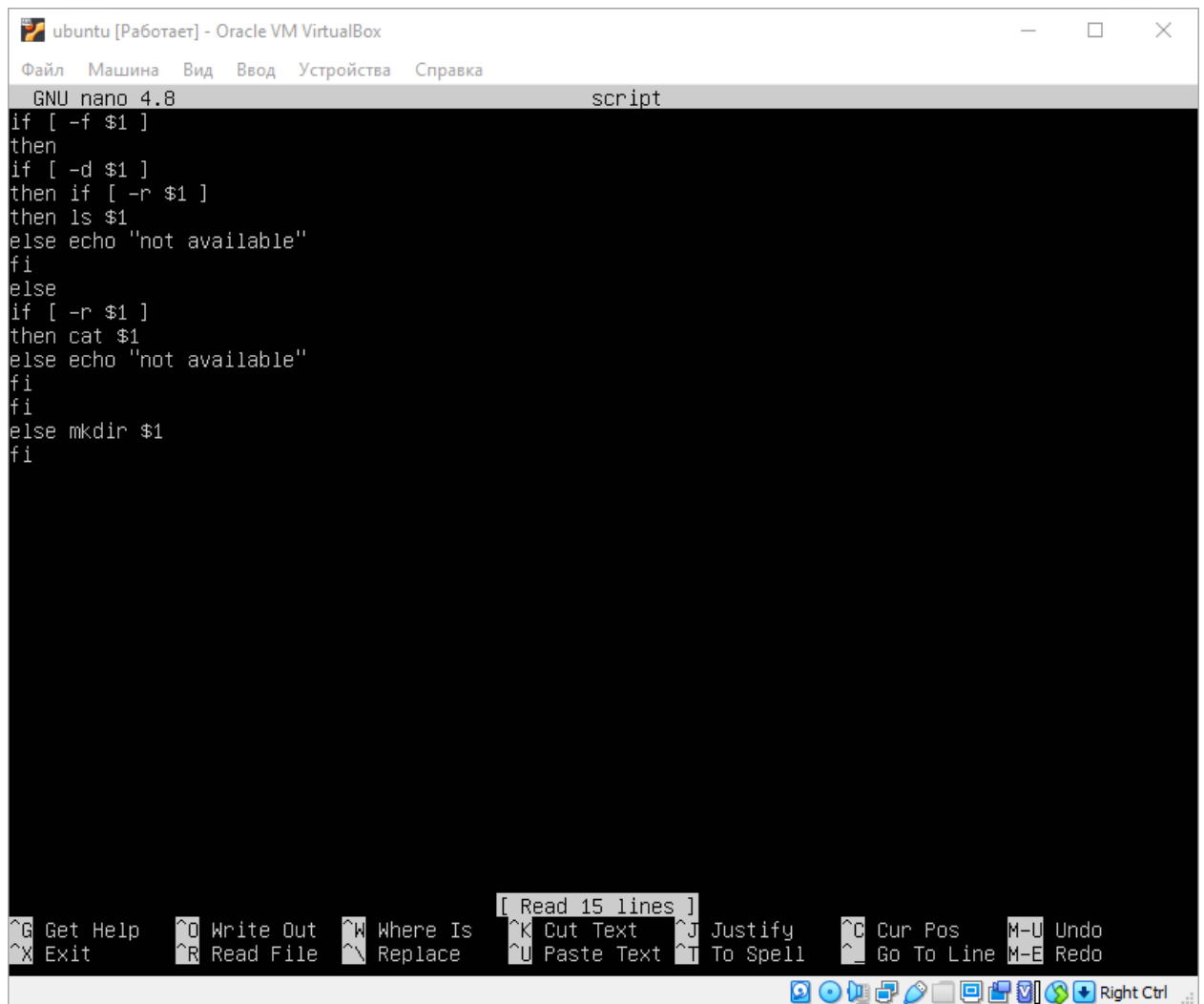


Рисунок 19.2 – Результат выполнения сценария

20. Если файл есть каталог и этот каталог можно читать, просматривается содержимое этого каталога. Если каталог отсутствует, он создается. Если файл не есть каталог, просматривается содержимое файла.

Чтобы проверить, доступен ли файл или каталог для чтения, используется опция `-r`. Напишем скрипт с использованием данной конструкции:



The screenshot shows a window titled "ubuntu [Работает] - Oracle VM VirtualBox". Inside the window is a terminal running the GNU nano 4.8 editor. The editor is editing a file named "script". The script contains the following code:

```
if [ -f $1 ]
then
if [ -d $1 ]
then if [ -r $1 ]
then ls $1
else echo "not available"
fi
else
if [ -r $1 ]
then cat $1
else echo "not available"
fi
fi
else mkdir $1
fi
```

At the bottom of the window, there is a menu bar with various options: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^C Cur Pos, M-U Undo, ^X Exit, ^R Read File, ^_ Replace, ^U Paste Text, ^T To Spell, ^_ Go To Line, M-E Redo. Below the menu bar is a toolbar with icons for various functions, including a "Right Ctrl" button.

Рисунок 20.1 – Скрипт для задания 20

Теперь проверим наш сценарий:

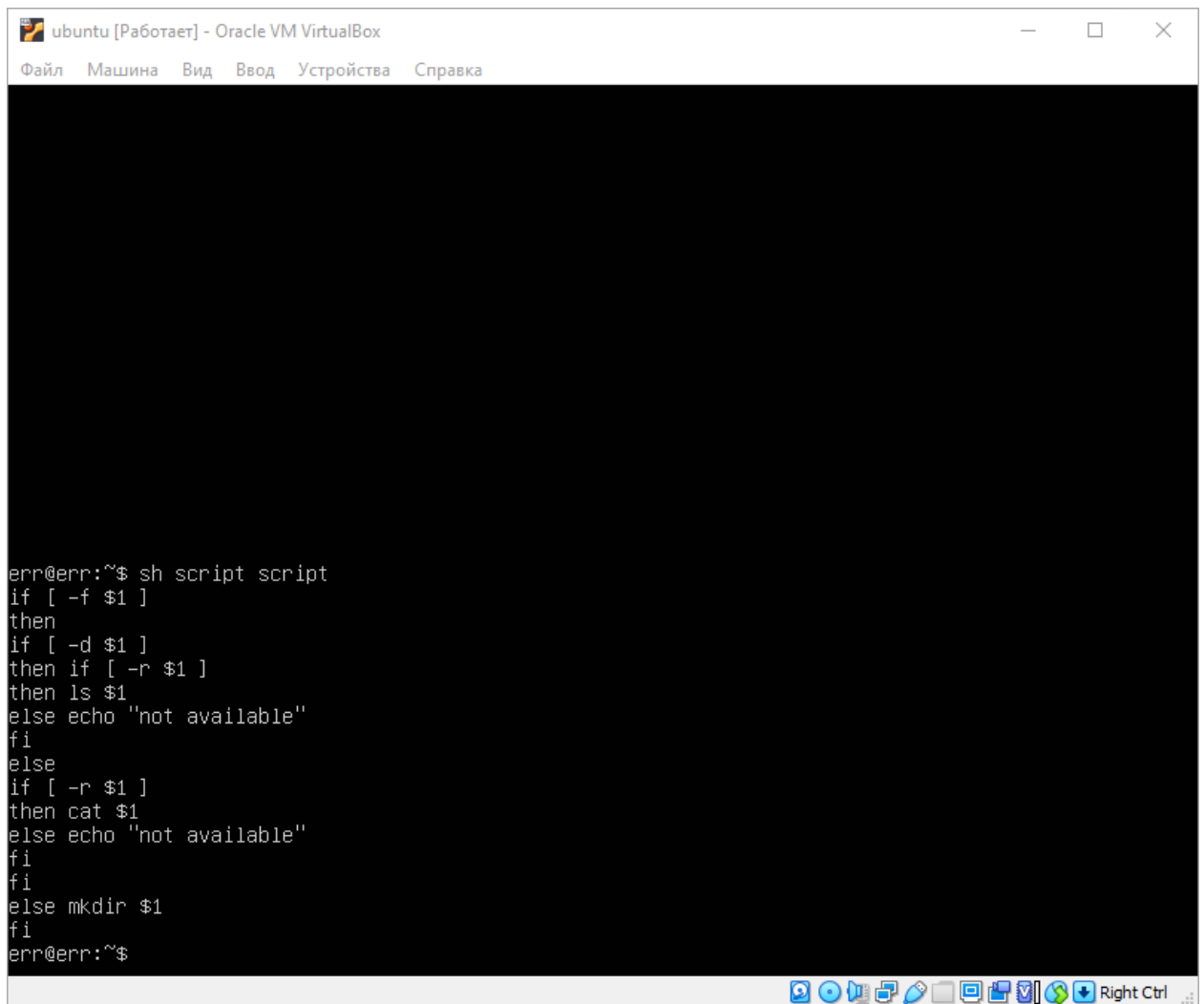
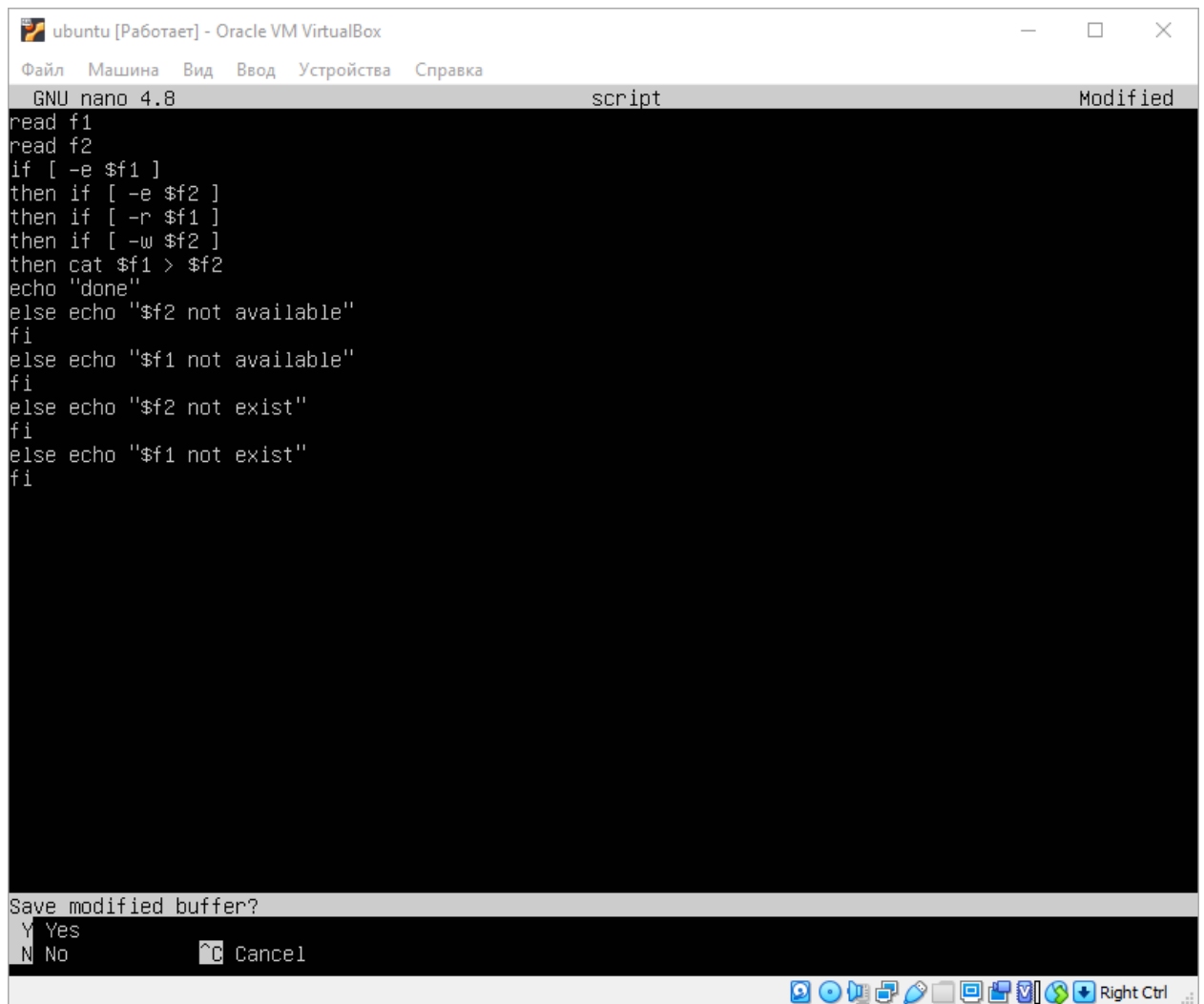


Рисунок 20.2 – Результат выполнения сценария

21. Анализируются атрибуты файла. Если первый файл существует и используется для чтения, а второй файл существует и используется для записи, то содержимое первого файла перенаправляется во второй файл. В случае несовпадений указанных атрибутов или отсутствия файлов на экран выдаются соответствующие сообщения (использовать имена файлов и позиционные параметры).

Для проверки доступности файла для записи используется конструкция `–t`. Сначала реализуем задание, спросив названия файлов у пользователя:



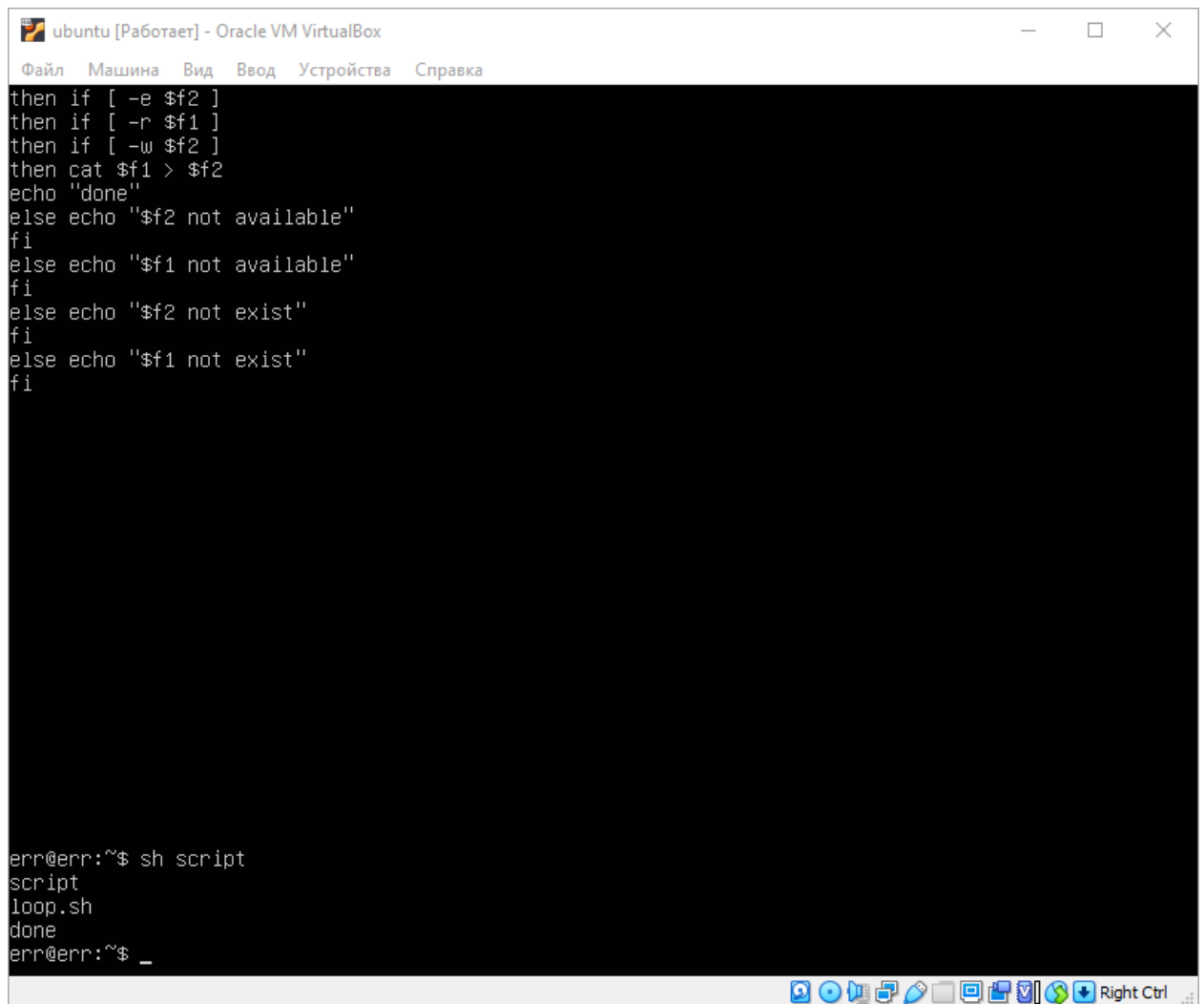
The screenshot shows a window titled "ubuntu [Работаает] - Oracle VM VirtualBox". Inside, the GNU nano 4.8 editor is open, editing a file named "script". The script contains the following code:

```
read f1
read f2
if [ -e $f1 ]
then if [ -e $f2 ]
then if [ -r $f1 ]
then if [ -w $f2 ]
then cat $f1 > $f2
echo "done"
else echo "$f2 not available"
fi
else echo "$f1 not available"
fi
else echo "$f2 not exist"
fi
else echo "$f1 not exist"
fi
```

At the bottom of the editor, a prompt "Save modified buffer?" is displayed with options "Y Yes", "N No", and a "Cancel" button. The window's taskbar at the bottom shows various icons and the text "Right Ctrl".

Рисунок 21.1 – Скрипт для задания 21 (через имена файлов)


Выполним скрипт:



```
ubuntu [Работаает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
then if [ -e $f2 ]
then if [ -r $f1 ]
then if [ -w $f2 ]
then cat $f1 > $f2
echo "done"
else echo "$f2 not available"
fi
else echo "$f1 not available"
fi
else echo "$f2 not exist"
fi
else echo "$f1 not exist"
fi

err@err:~$ sh script
script
loop.sh
done
err@err:~$ _
```

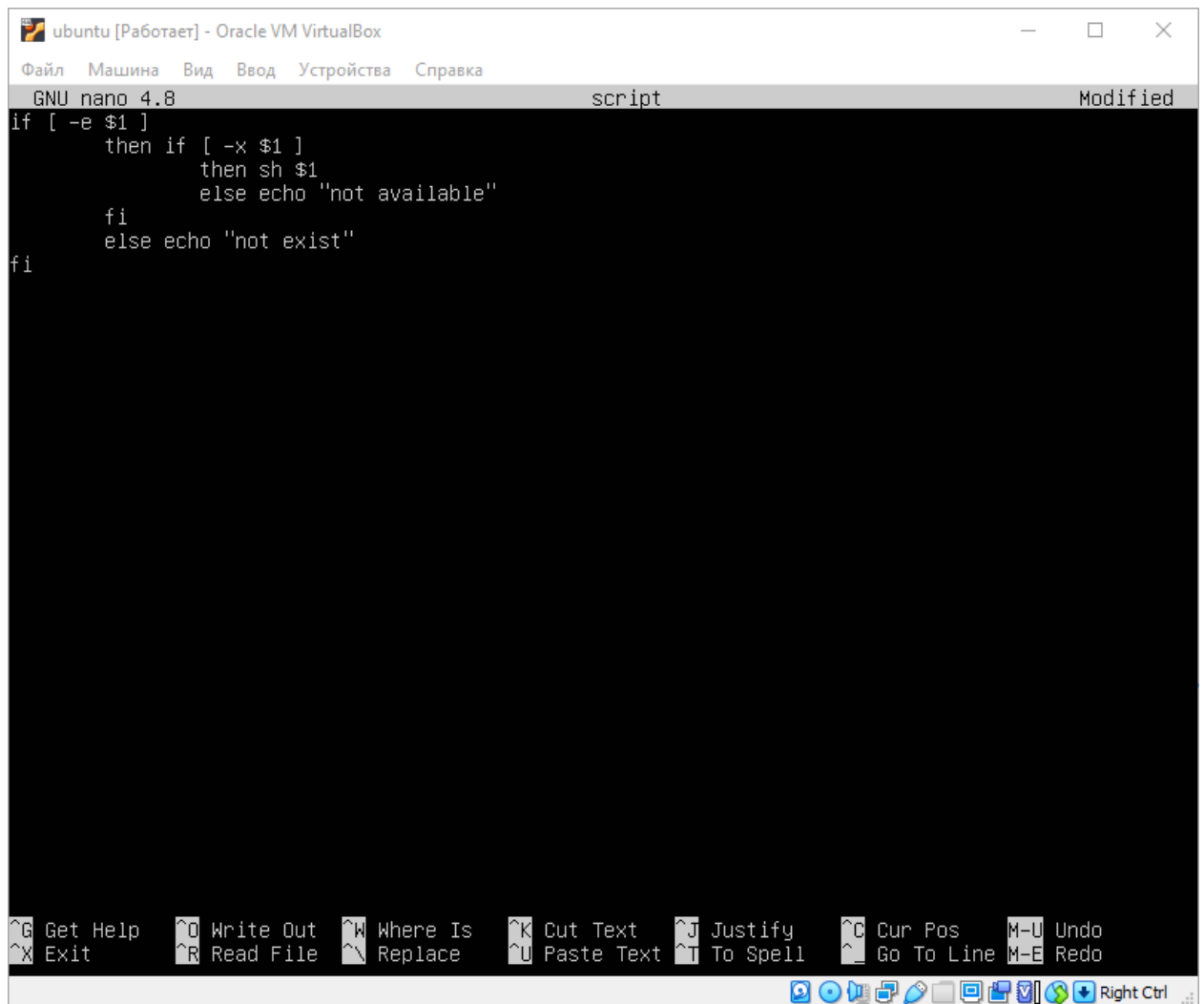
Рисунок 21.2 – Результат выполнения сценария (через имена файлов)



```
ubuntu [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
GNU nano 4.8                                loop.sh
read f1
read f2
if [ -e $f1 ]
then if [ -e $f2 ]
then if [ -r $f1 ]
then if [ -w $f2 ]
then cat $f1 > $f2
fi
fi
fi
fi
fi
fi
```

Рисунок 21.3 – Результат выполнения сценария (через имена файлов)

22. Если файл запуска программы найден, программа запускается (по выбору).
Проверим файл, переданный в качестве аргумента командной строки, на существование и доступность для исполнения (опция `-x`):



The image shows a terminal window titled "ubuntu [Работает] - Oracle VM VirtualBox". Inside the terminal, the GNU nano 4.8 text editor is open, editing a file named "script". The script contains the following code:

```
if [ -e $1 ]
then if [ -x $1 ]
then sh $1
else echo "not available"
fi
else echo "not exist"
fi
```

The bottom of the terminal window displays a menu with various keyboard shortcuts for nano, such as ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^C Cur Pos, M-U Undo, ^X Exit, ^R Read File, ^_ Replace, ^U Paste Text, ^T To Spell, ^_ Go To Line, M-E Redo, and a Right Ctrl button.

Рисунок 22.1 – Скрипт для задания 22

Затем создадим новый файл сценария, который нам придётся запустить:

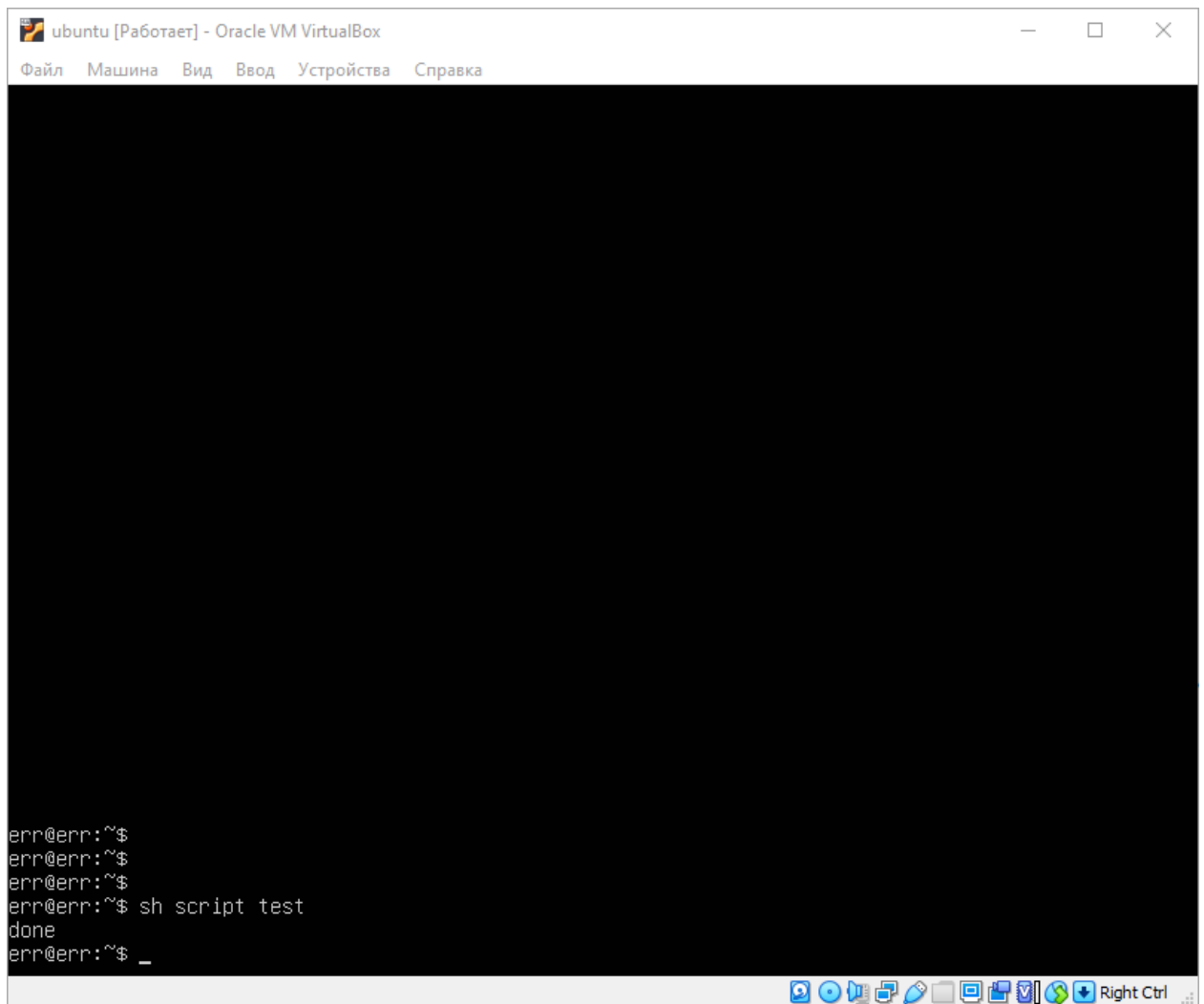


Рисунок 22.2 – Содержание файла test

23. В качестве позиционного параметра задается файл, анализируется его размер. Если размер файла больше нуля, содержимое файла сортируется по первому столбцу по возрастанию, отсортированная информация помещается в другой файл, содержимое которого затем отображается на экране.

Для проверки файла на содержание в нём хотя бы одного символа используется конструкция `-s`. Чтобы отсортировать данные в файле по определённому столбцу, используем опцию `-k<num>` (где `num` – номер столбца) команды `sort`:

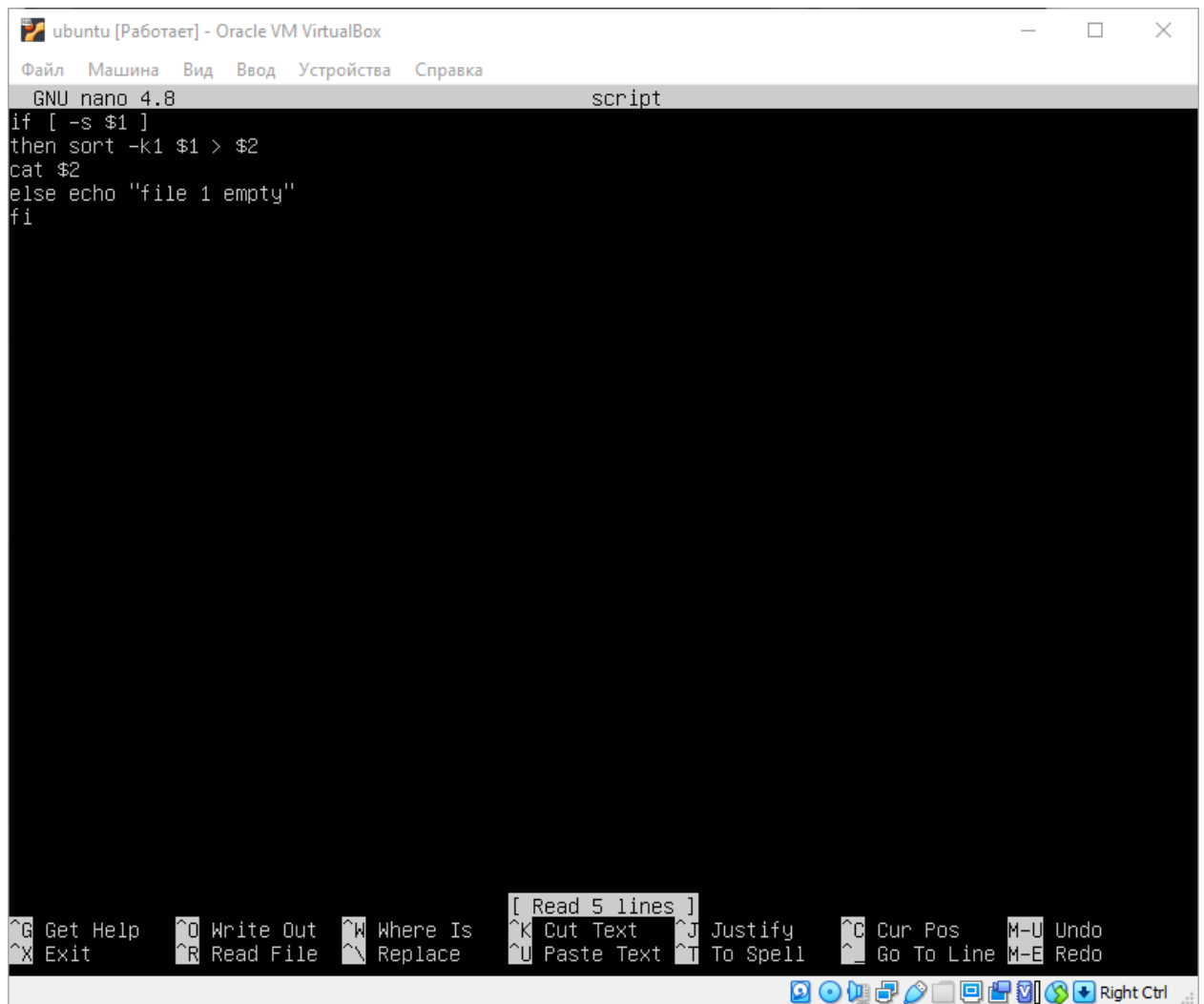


Рисунок 23.1 – Скрипт для задания 23

Исполним скрипт:

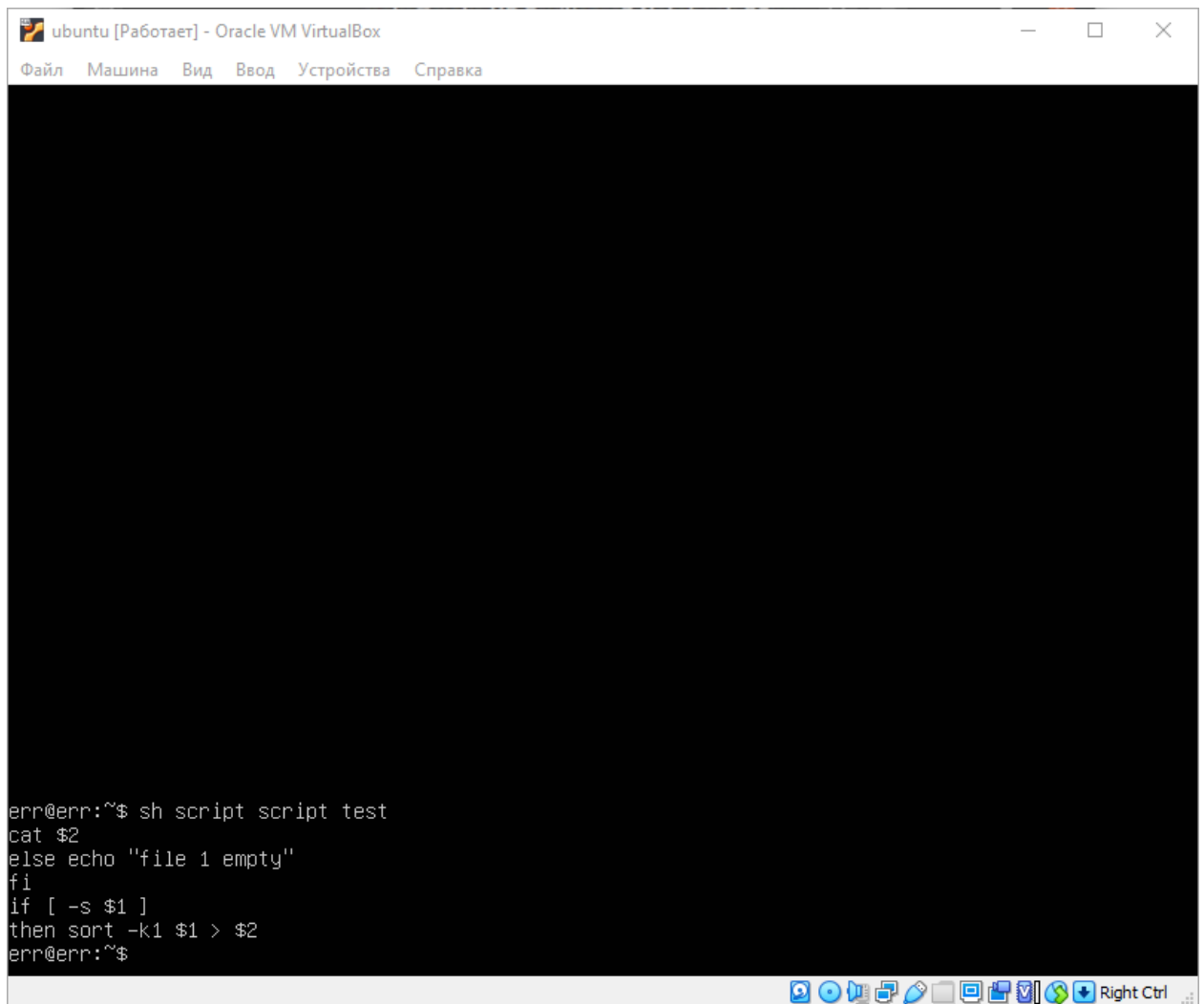
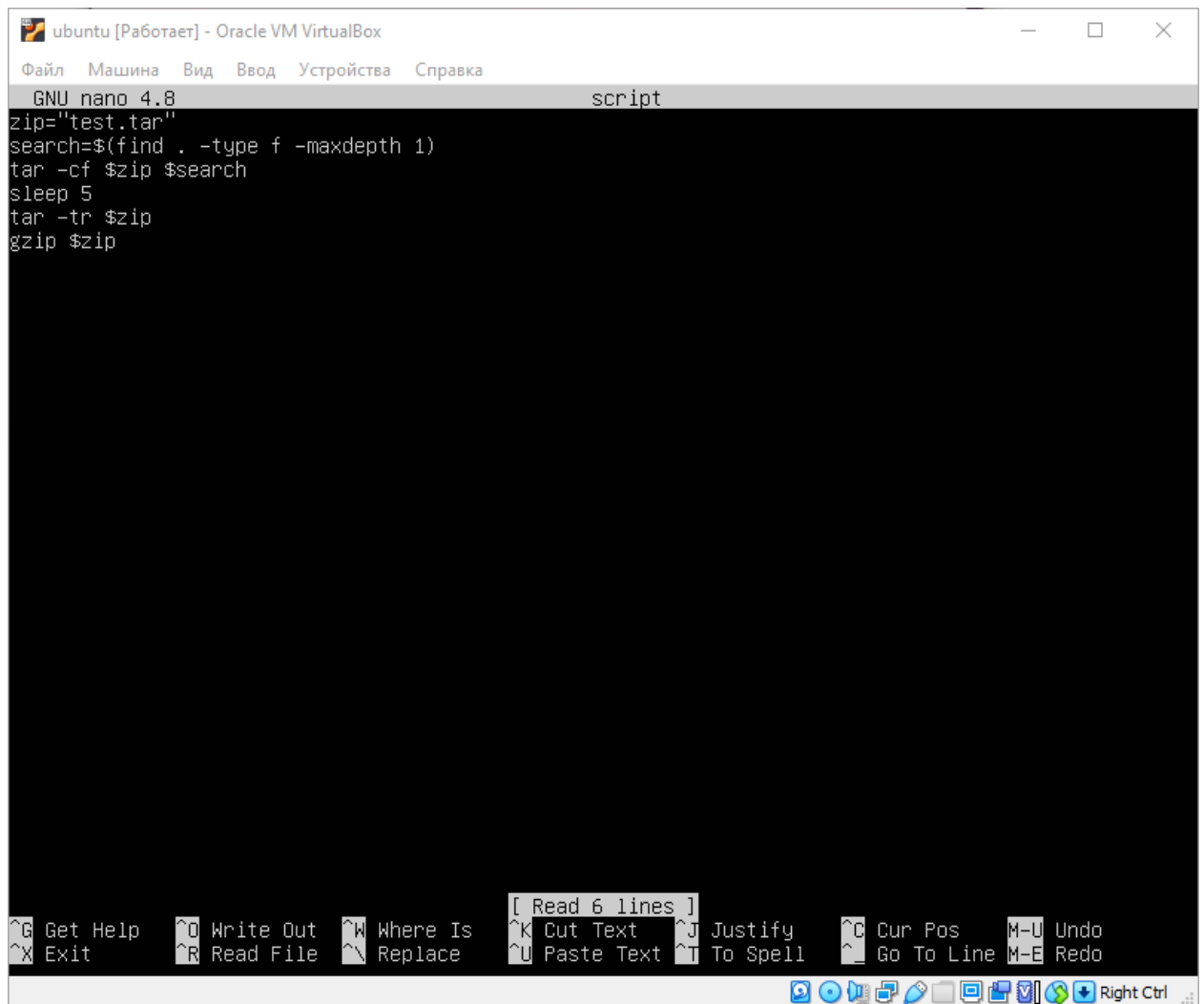


Рисунок 23.2 – Результат выполнения сценария

24. Командой TAR осуществляется сборка всех текстовых файлов текущего каталога в один архивный файл my.tar, после паузы просматривается содержимое файла my.tar, затем командой GZIP архивный файл my.tar сжимается.

Сначала найдём в данном каталоге с помощью команды find нужные файлы (параметр `–maxdepth 1` говорит о том, что в подкаталогах искать не нужно, а параметр `–type f` – о том, что ищем только файлы, а не каталоги). Затем с помощью команды `tar –cf` собираем найденные файлы в один архив. Ну и выводим содержимое с помощью команды `tar –tf`:



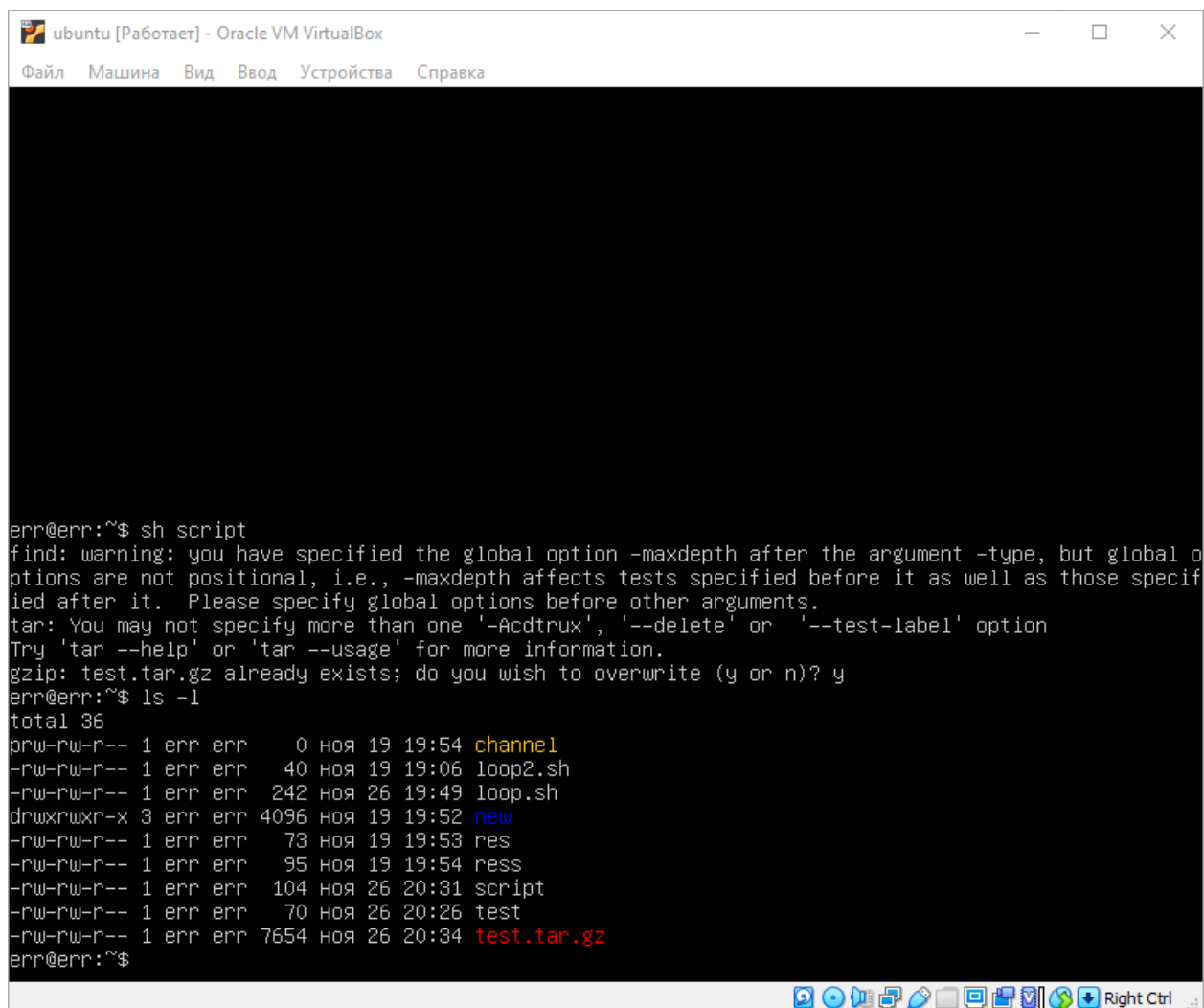
The screenshot shows a terminal window titled "ubuntu [Работает] - Oracle VM VirtualBox". Inside the terminal, the GNU nano 4.8 editor is open, editing a file named "script". The script contains the following lines:

```
zip="test.tar"
search=$(find . -type f -maxdepth 1)
tar -cf $zip $search
sleep 5
tar -tr $zip
gzip $zip
```

The nano editor's status bar at the bottom displays various keyboard shortcuts and a message "[Read 6 lines]". The shortcuts include: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^C Cur Pos, M-U Undo, ^X Exit, ^R Read File, ^_ Replace, ^U Paste Text, ^T To Spell, ^_ Go To Line, M-E Redo. A "Right Ctrl" button is also visible on the right side of the status bar.

Рисунок 24.1 – Скрипт для задания 24

Выполним скрипт:



```
ubuntu [Работаer] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка

err@err:~$ sh script
find: warning: you have specified the global option -maxdepth after the argument -type, but global options are not positional, i.e., -maxdepth affects tests specified before it as well as those specified after it. Please specify global options before other arguments.
tar: You may not specify more than one '-Acctrux', '--delete' or '--test-label' option
Try 'tar --help' or 'tar --usage' for more information.
gzip: test.tar.gz already exists; do you wish to overwrite (y or n)? y
err@err:~$ ls -l
total 36
-rw-rw-r-- 1 err err  0 ноя 19 19:54 channel
-rw-rw-r-- 1 err err  40 ноя 19 19:06 loop2.sh
-rw-rw-r-- 1 err err 242 ноя 26 19:49 loop.sh
drwxrwxr-x 3 err err 4096 ноя 19 19:52 new
-rw-rw-r-- 1 err err  73 ноя 19 19:53 res
-rw-rw-r-- 1 err err  95 ноя 19 19:54 ress
-rw-rw-r-- 1 err err 104 ноя 26 20:31 script
-rw-rw-r-- 1 err err  70 ноя 26 20:26 test
-rw-rw-r-- 1 err err 7654 ноя 26 20:34 test.tar.gz
err@err:~$
```

Рисунок 24.2 – Результат выполнения сценария

25. Написать скрипт с использованием функции, например, функции, суммирующей значения двух переменных.

Напишем сценарий:

```
GNU nano 4.8                                script                                Modified
read a
read b
sum(){
res=$(expr $a + $b)
echo $res
}
sum
```

[Read 7 lines]

^G Get Help	^O Write Out	^W Where Is	^K Cut Text	^J Justify	^C Cur Pos	M-U Undo
^X Exit	^R Read File	^N Replace	^U Paste Text	^T To Spell	^_ Go To Line	M-E Redo

Рисунок 25.1 – Скрипт для задания 25

Выполним скрипт:

```
res=$(expr $a + $b)
echo $res
}
sum
```

```
err@err:~$ sh script
3
4
7
err@err:~$ _
```

Рисунок 25.2 – Результат выполнения сценария

Вывод

В ходе выполнения лабораторной работы были изучены основные возможности языка программирования Shell с целью автоматизации процесса администрирования системы за счёт написания и использования командных файлов.