

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №5

по Операционной системе Linux

Программирование на SHELL. Использование командных файлов

Студент

Лобов М.Ю.

Группа АИ-18

Руководитель

Кургасов В.В.

Доцент, к.п.н.

Липецк 2020 г.

Оглавление

Цель работы	3
Задание кафедры.....	4
Ход работы.....	7
Вывод.....	26

Цель работы

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

Задание кафедры

1. Используя команды ECHO, PRINTF вывести информационные сообщения на экран.
2. Присвоить переменной А целочисленное значение. Просмотреть значение переменной А.
3. Присвоить переменной В значение переменной А. Просмотреть значение переменной В.
4. Присвоить переменной С 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 Е 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.
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 имена файлов и 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.

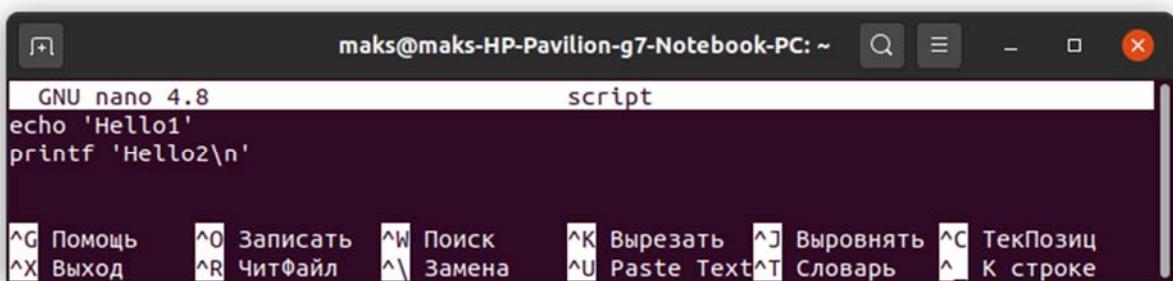
25. Написать скрипт с использованием функции, например, функции, суммирующей значения двух переменных.

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:

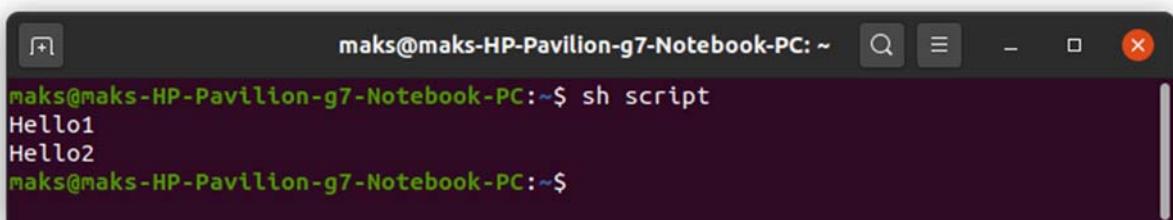


```
GNU nano 4.8                                     script
echo 'Hello1'
printf 'Hello2\n'

^G Помощь   ^O Записать   ^W Поиск   ^K Вырезать   ^J Выровнять   ^C ТекПозиц
^X Выход   ^R ЧитФайл   ^\| Замена   ^U Paste Text ^T Словарь   ^_ К строке
```

Рисунок 1.1 – Скрипт для задания 1

После запуска данного сценария увидим:

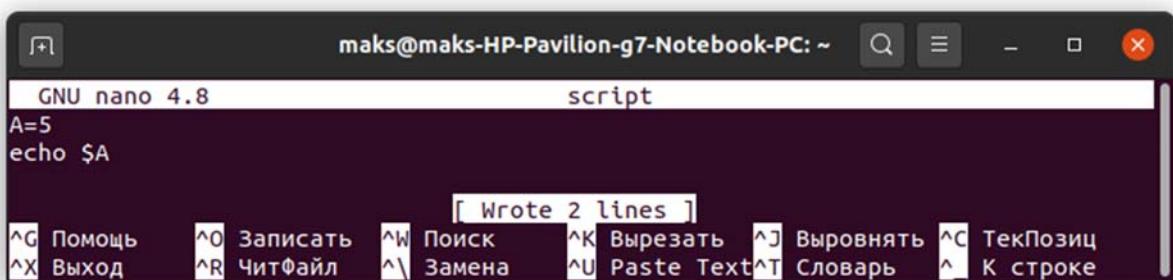


```
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$ sh script
Hello1
Hello2
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$
```

Рисунок 1.2 – Результат выполнения сценария

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

Напишем сценарий:

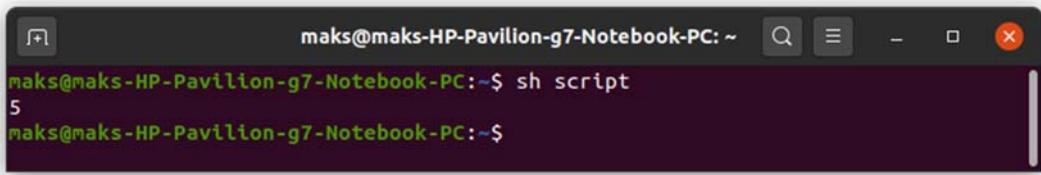


```
GNU nano 4.8                                     script
A=5
echo $A

^G Помощь   ^O Записать   ^W Поиск   ^K Вырезать   ^J Выровнять   ^C ТекПозиц
^X Выход   ^R ЧитФайл   ^\| Замена   ^U Paste Text ^T Словарь   ^_ К строке
```

Рисунок 2.1 – Скрипт для задания 2

Запустим сценарий:

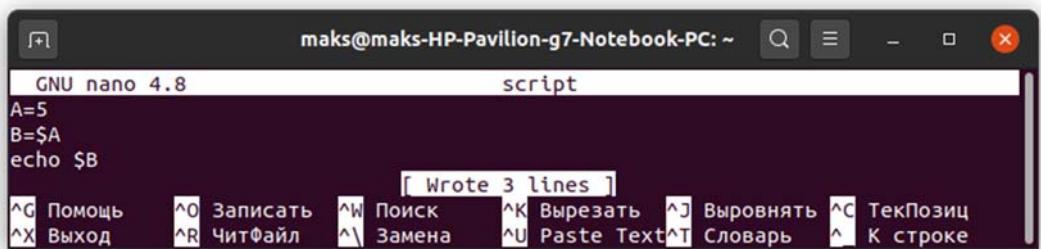


```
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$ sh script
5
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$
```

Рисунок 2.2 – Результат выполнения сценария

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

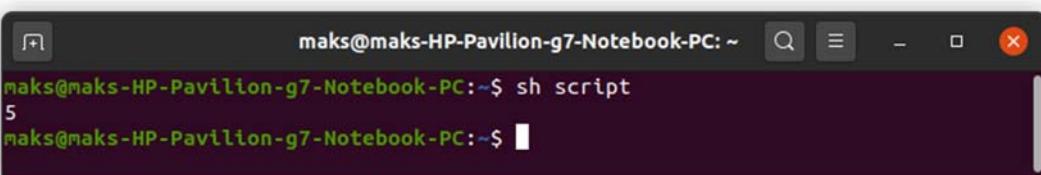
Напишем сценарий:



```
GNU nano 4.8
script
A=5
B=$A
echo $B
^G Помощь      ^O Записать   ^W Поиск      ^K Вырезать   ^J Выровнять  ^C ТекПозиц
^X Выход       ^R ЧитФайл    ^\ Замена     ^U Paste Text^T Словарь   ^ К строке
```

Рисунок 3.1 – Скрипт для задания 3

Выполним сценарий:

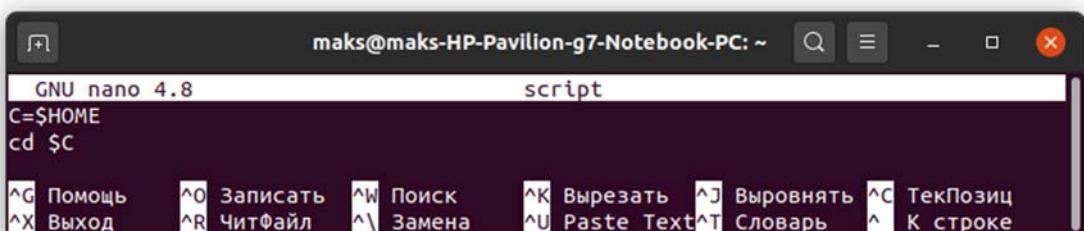


```
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$ sh script
5
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$
```

Рисунок 3.2 – Результат выполнения сценария

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

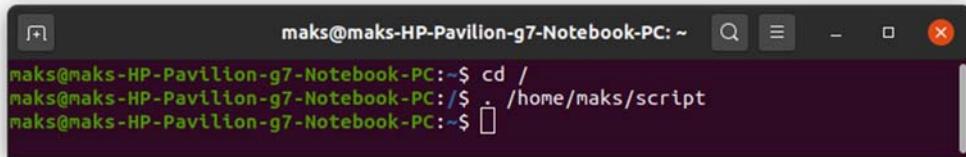
Сделаем это с использованием встроенной переменной **HOME**:



```
GNU nano 4.8
script
C=$HOME
cd $C
^G Помощь      ^O Записать   ^W Поиск      ^K Вырезать   ^J Выровнять  ^C ТекПозиц
^X Выход       ^R ЧитФайл    ^\ Замена     ^U Paste Text^T Словарь   ^ К строке
```

Рисунок 4.1 – Скрипт для задания 4

Проверим работу сценария:

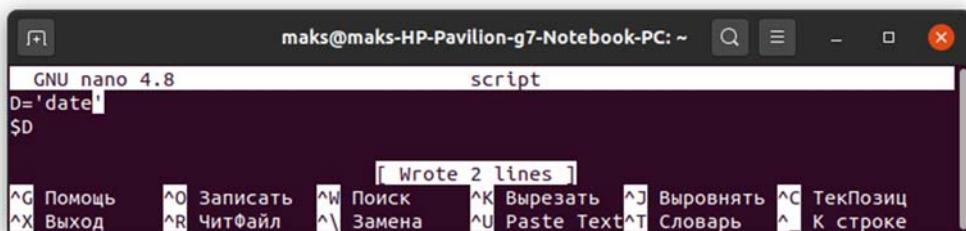


```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ cd /  
maks@maks-HP-Pavilion-g7-Notebook-PC:/$ . /home/maks/script  
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

Рисунок 4.2 – Результат выполнения сценария

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

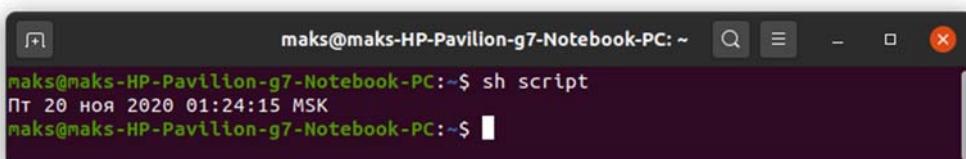
Выполним задание с использованием встроенной переменной DATE:



```
GNU nano 4.8                                     script  
D='date'  
$D  
[ Wrote 2 lines ]  
^G Помощь   ^O Записать   ^W Поиск   ^K Вырезать   ^J Выровнять   ^C ТекПозиц  
^X Выход   ^R ЧитФайл   ^\ Замена   ^U Paste Text ^T Словарь   ^ К строке
```

Рисунок 5.1 – Скрипт для задания 5

Исполним сценарий:

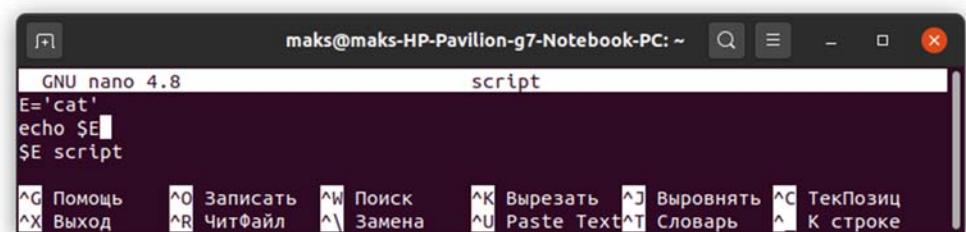


```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script  
Пт 20 ноя 2020 01:24:15 MSK  
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

Рисунок 5.2 – Результат выполнения сценария

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

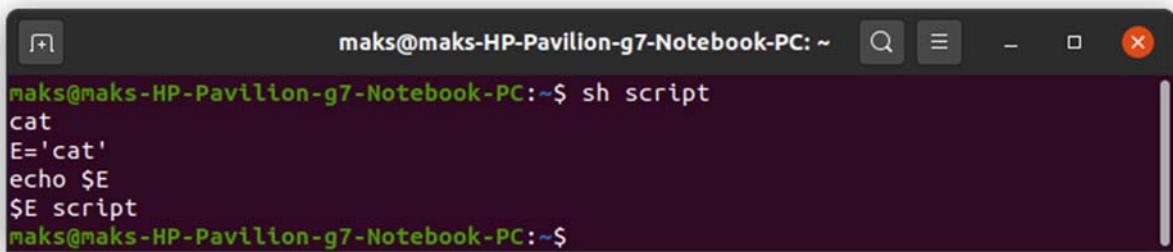
Присвоим переменной значение команды cat:



```
GNU nano 4.8                                     script  
E='cat'  
echo $E  
$E script  
^G Помощь   ^O Записать   ^W Поиск   ^K Вырезать   ^J Выровнять   ^C ТекПозиц  
^X Выход   ^R ЧитФайл   ^\ Замена   ^U Paste Text ^T Словарь   ^ К строке
```

Рисунок 6.1 – Скрипт для задания 6

Выполним сценарий:

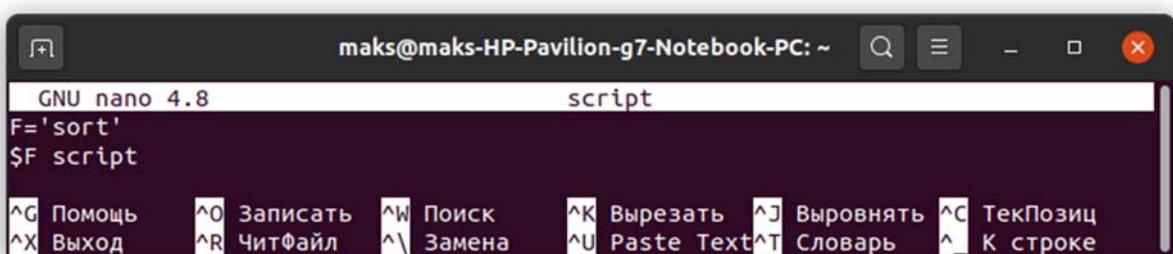


```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script
cat
$E='cat'
echo $E
$E script
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

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:

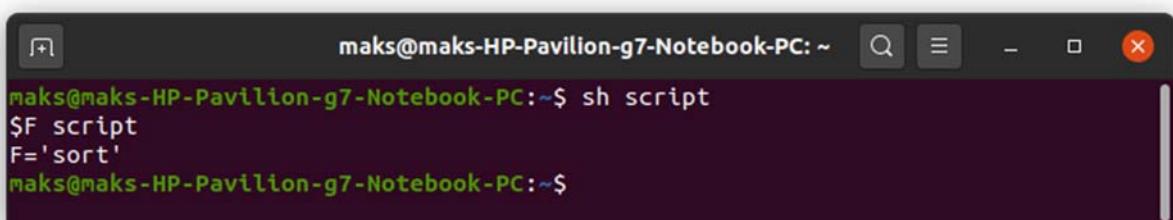


```
GNU nano 4.8
script
F='sort'
$F script

^G Помощь ^O Записать ^W Поиск ^K Вырезать ^J Выровнять ^C ТекПозиц
^X Выход ^R ЧитФайл ^\| Замена ^U Paste Text ^T Словарь ^_ К строке
```

Рисунок 7.1 – Скрипт для задания 7

Исполним сценарий:



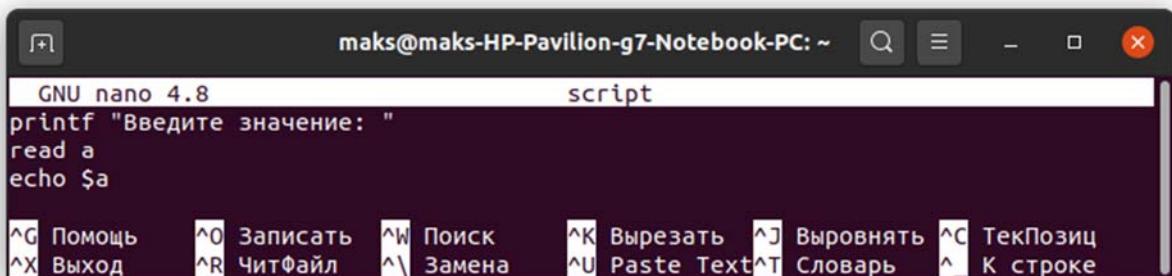
```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script
$F script
F='sort'
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

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:



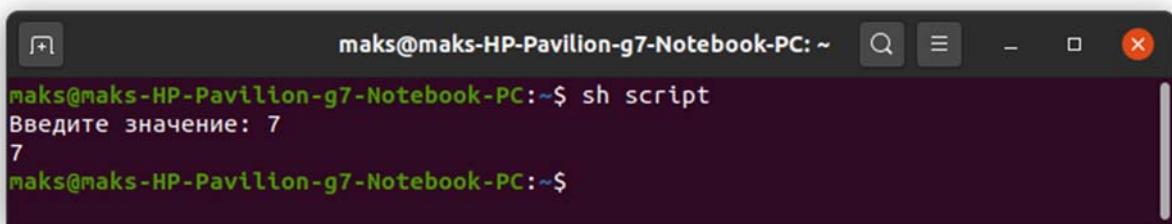
The screenshot shows a terminal window titled "maks@maks-HP-Pavilion-g7-Notebook-PC: ~". The window contains the following text:

```
GNU nano 4.8                                     script
printf "Введите значение: "
read a
echo $a
```

At the bottom of the terminal window, there is a menu bar with Russian labels: Помощь (Help), Записать (Save), Поиск (Search), Вырезать (Cut), Выровнять (Justify), ТекПозиц (TextPos), Выход (Exit), ЧитФайл (ReadFile), Замена (Replace), Paste Text, Словарь (Dictionary), and К строке (To Line).

Рисунок 8.1 – Скрипт для задания 8

Исполним сценарий:



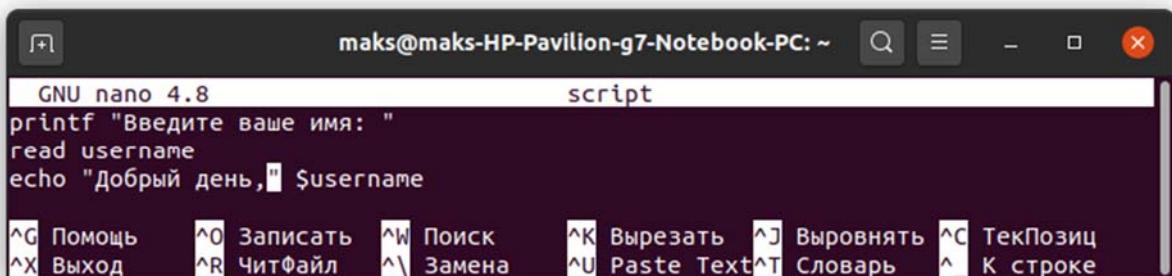
The screenshot shows a terminal window titled "maks@maks-HP-Pavilion-g7-Notebook-PC: ~\$". The window contains the following text:

```
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$ sh script
Введите значение: 7
7
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$
```

Рисунок 8.2 – Результат выполнения сценария

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

Сделаем по примеру предыдущего задания:



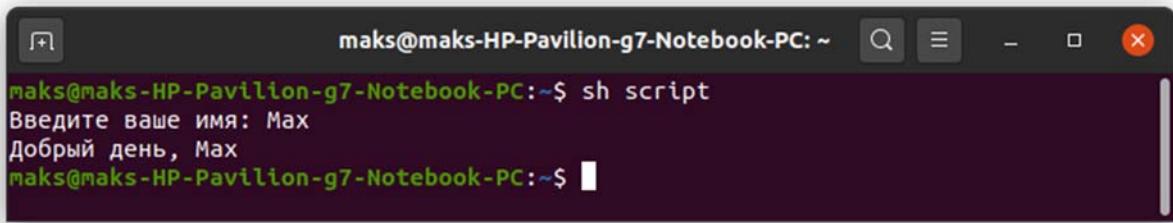
The screenshot shows a terminal window titled "maks@maks-HP-Pavilion-g7-Notebook-PC: ~". The window contains the following text:

```
GNU nano 4.8                                     script
printf "Введите ваше имя: "
read username
echo "добрый день," $username
```

At the bottom of the terminal window, there is a menu bar with Russian labels: Помощь (Help), Записать (Save), Поиск (Search), Вырезать (Cut), Выровнять (Justify), ТекПозиц (TextPos), Выход (Exit), ЧитФайл (ReadFile), Замена (Replace), Paste Text, Словарь (Dictionary), and К строке (To Line).

Рисунок 9.1 – Скрипт для задания 9

Запустим файл сценария:

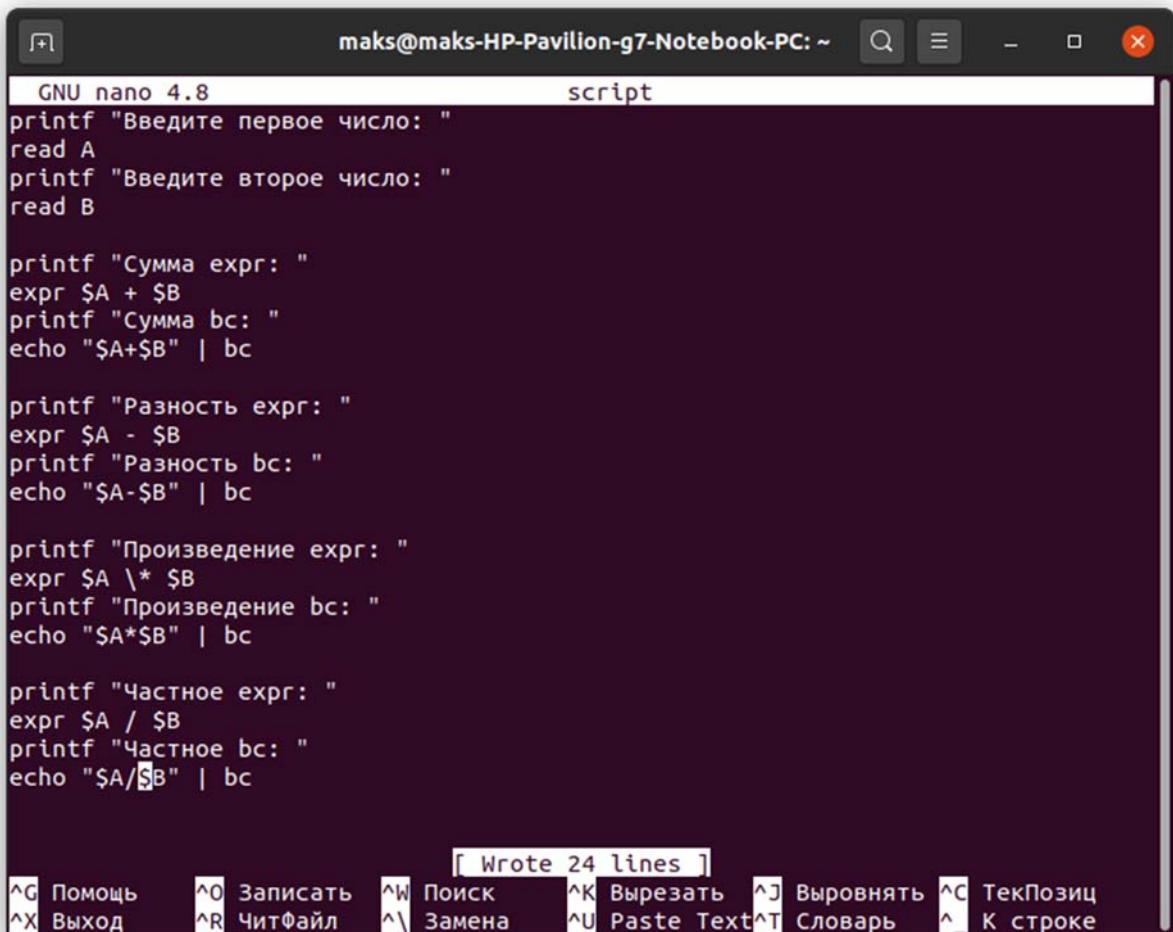


```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script
Введите ваше имя: Max
Добрый день, Max
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

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:



```
GNU nano 4.8                                     script
printf "Введите первое число: "
read A
printf "Введите второе число: "
read B

printf "Сумма expr: "
expr $A + $B
printf "Сумма bc: "
echo "$A+$B" | bc

printf "Разность expr: "
expr $A - $B
printf "Разность bc: "
echo "$A-$B" | bc

printf "Произведение expr: "
expr $A \* $B
printf "Произведение bc: "
echo "$A*$B" | bc

printf "Частное expr: "
expr $A / $B
printf "Частное bc: "
echo "$A/$B" | bc

[ Wrote 24 lines ]
```

^G Помощь ^O Записать ^W Поиск ^K Вырезать ^J Выровнять ^C ТекПозиц
^X Выход ^R ЧитФайл ^\ Замена ^U Paste Text ^T Словарь ^ K строке

Рисунок 10.1 – Скрипт для задания 10

Исполним скрипт:

```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script
Введите первое число: 7
Введите второе число: 4
Сумма expr: 11
Сумма bc: 11
Разность expr: 3
Разность bc: 3
Произведение expr: 28
Произведение bc: 28
Частное expr: 1
Частное bc: 1
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

Рисунок 10.2 – Результат выполнения сценария

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

Формула объема цилиндра имеет вид $V = \pi R^2 h$. Реализуем её:

```
GNU nano 4.8
script
printf "Введите радиус основания: "
read R
printf "Введите высоту: "
read h
printf "Объем цилиндра равен "
echo "3.14*$R*$R*$h" | bc
```

Рисунок 11.1 – Скрипт для выполнения задания 11

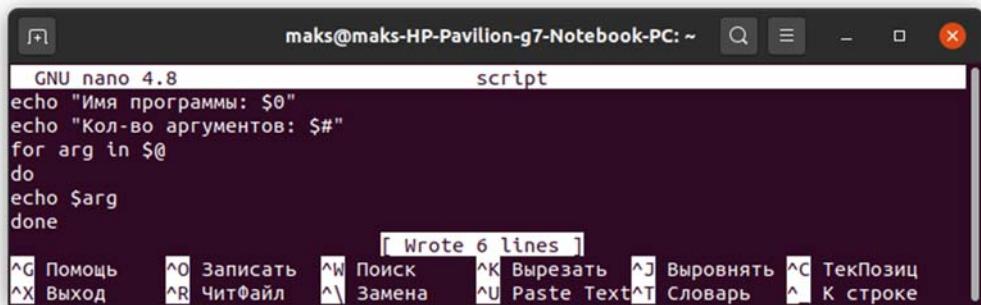
Запустим сценарий:

```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script
Введите радиус основания: 2
Введите высоту: 5
Объем цилиндра равен 62.80
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

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:

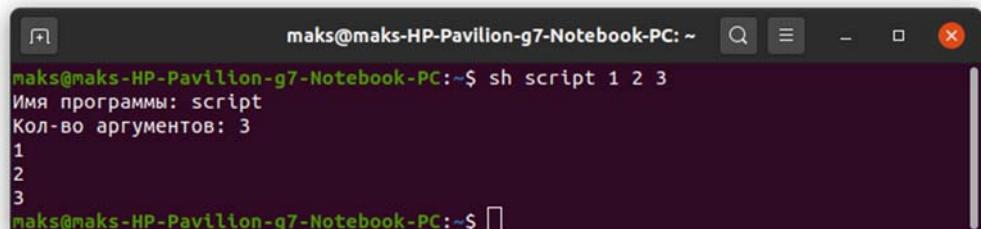


```
GNU nano 4.8                               script
echo "Имя программы: $@"
echo "Кол-во аргументов: ${#}"
for arg in $@
do
echo $arg
done
[ Wrote 6 lines ]
```

Terminal menu bar: Помощь, Записать, Поиск, Вырезать, Выровнять, ТекПозиц, Выход, ЧитФайл, Замена, Paste Text, Словарь, К строке

Рисунок 12.1 – Скрипт для задания 12

Исполним сценарий:

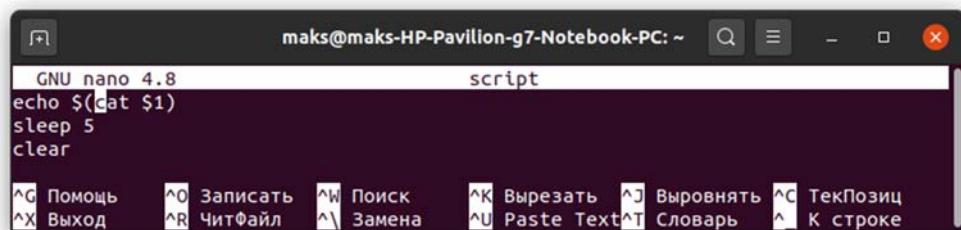


```
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$ sh script 1 2 3
Имя программы: script
Кол-во аргументов: 3
1
2
3
```

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:

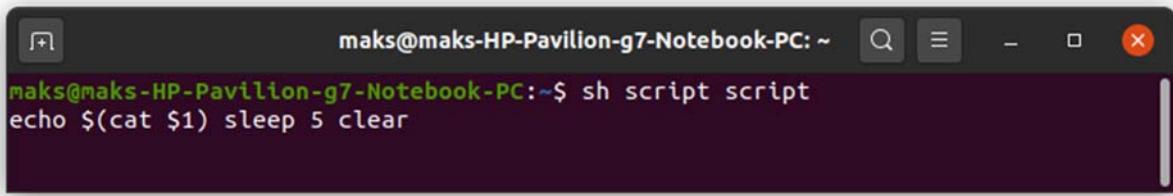


```
GNU nano 4.8                               script
echo $(cat $1)
sleep 5
clear
```

Terminal menu bar: Помощь, Записать, Поиск, Вырезать, Выровнять, ТекПозиц, Выход, ЧитФайл, Замена, Paste Text, Словарь, К строке

Рисунок 13.1 – Скрипт для задания 13

Запустим сценарий на выполнение:

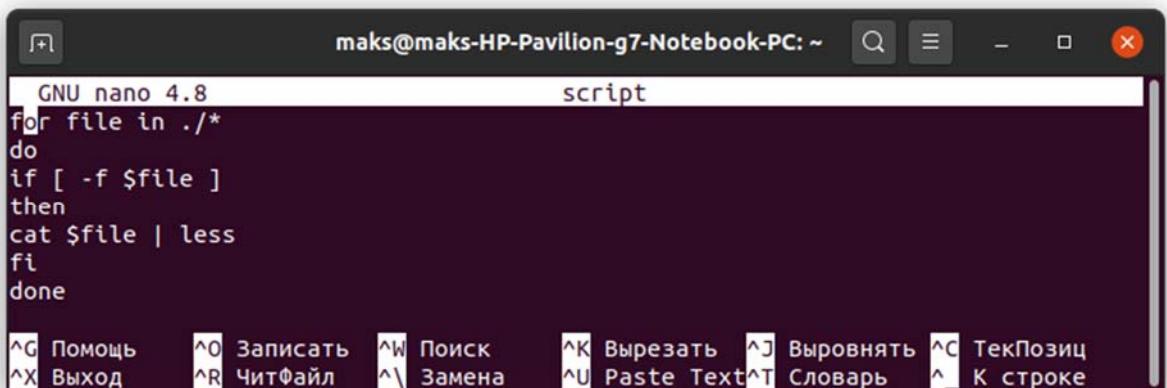


```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script script
echo $(cat $1) sleep 5 clear
```

Рисунок 13.2 – Результат выполнения сценария

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

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

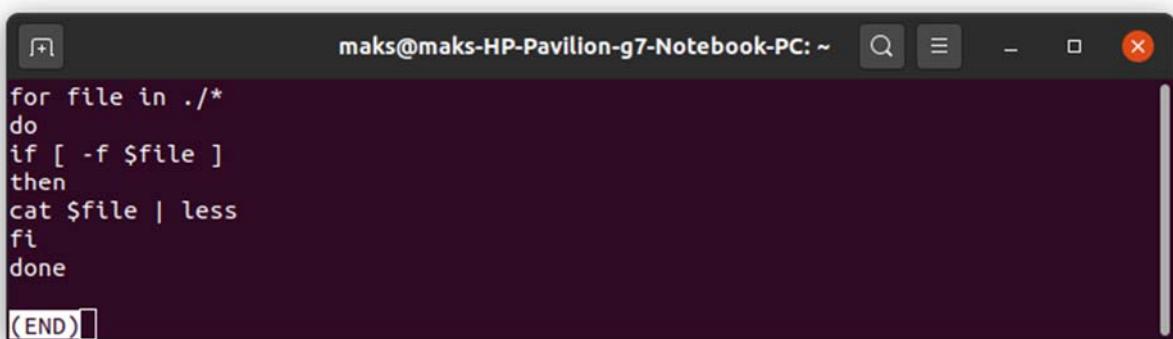


```
GNU nano 4.8
script
for file in ./*
do
if [ -f $file ]
then
cat $file | less
fi
done
```

^G Помощь ^O Записать ^W Поиск ^K Вырезать ^J Выровнять ^C ТекПозиц
^X Выход ^R ЧитФайл ^A Замена ^U Paste Text ^T Словарь ^ ^ К строке

Рисунок 14.1 – Скрипт для задания 14

Запустим скрипт:

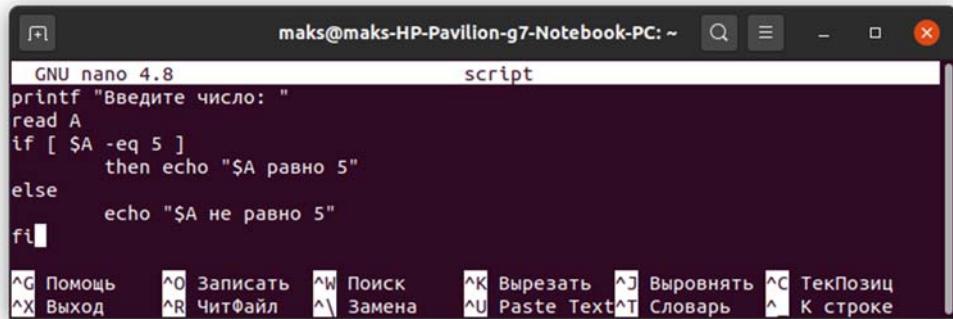


```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ ./script
for file in ./*
do
if [ -f $file ]
then
cat $file | less
fi
done
(END)
```

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** (от англ. *equal – равно*). Используем это в написании сценария:

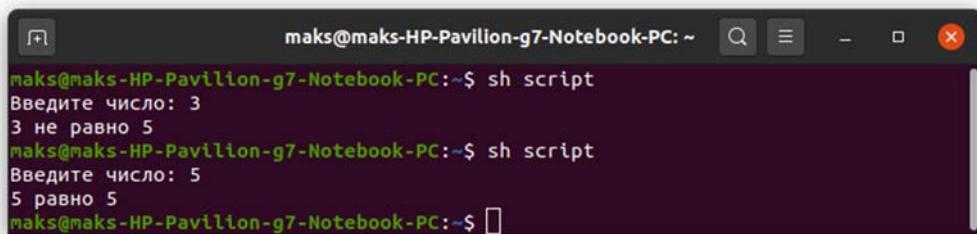


```
GNU nano 4.8                                     script
printf "Введите число: "
read A
if [ $A -eq 5 ]
    then echo "$A равно 5"
else
    echo "$A не равно 5"
fi
```

^G Помощь ^O Записать ^W Поиск ^K Вырезать ^J Выровнять ^C ТекПозиц
^X Выход ^R ЧитФайл ^\ Замена ^U Paste Text ^T Словарь ^ К строке

Рисунок 15.1 – Скрипт для задания 15

Выполним скрипт:

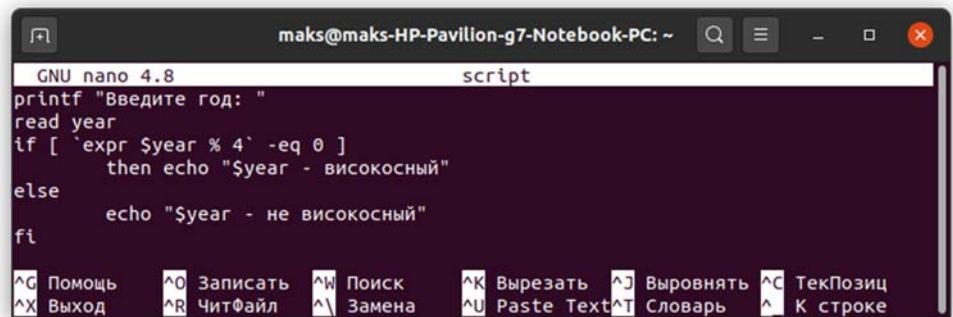


```
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$ sh script
Введите число: 3
3 не равно 5
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$ sh script
Введите число: 5
5 равно 5
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$
```

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:

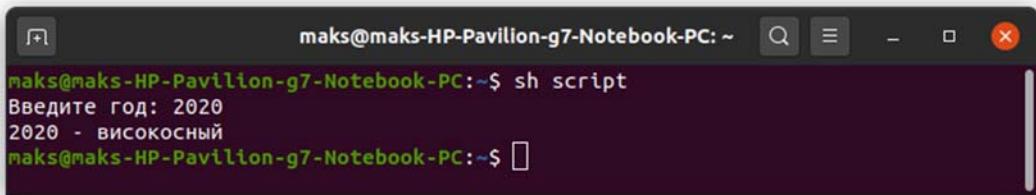


```
GNU nano 4.8                                     script
printf "Введите год: "
read year
if [ `expr $year % 4` -eq 0 ]
    then echo "$year - високосный"
else
    echo "$year - не високосный"
fi
```

^G Помощь ^O Записать ^W Поиск ^K Вырезать ^J Выровнять ^C ТекПозиц
^X Выход ^R ЧитФайл ^\ Замена ^U Paste Text ^T Словарь ^ К строке

Рисунок 15.1 – Скрипт для задания 16

Исполним данный сценарий:

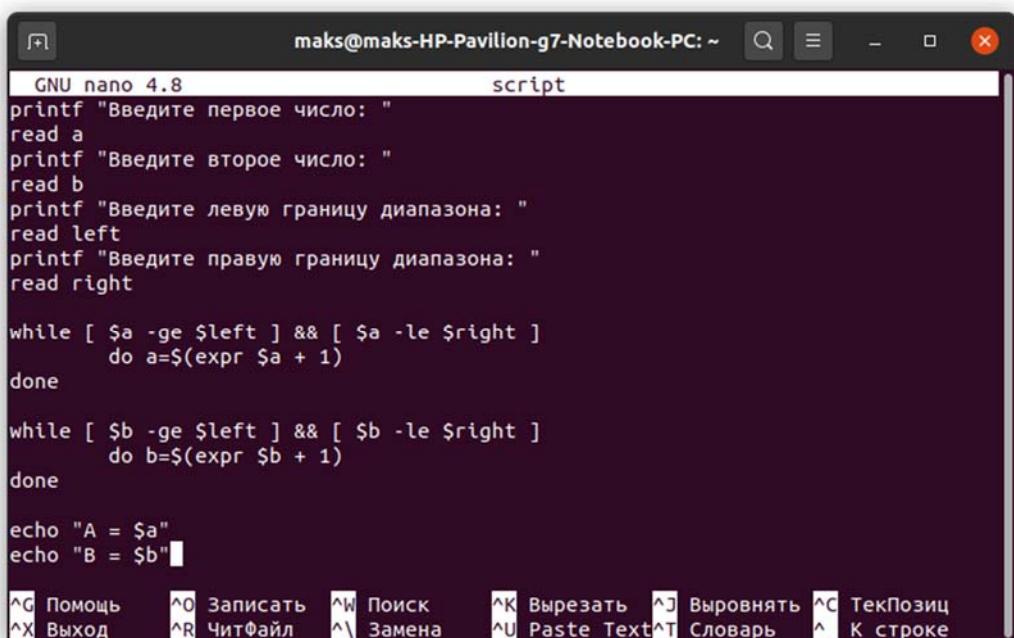


```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script
Введите год: 2020
2020 - високосный
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

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                               script
printf "Введите первое число: "
read a
printf "Введите второе число: "
read b
printf "Введите левую границу диапазона: "
read left
printf "Введите правую границу диапазона: "
read right

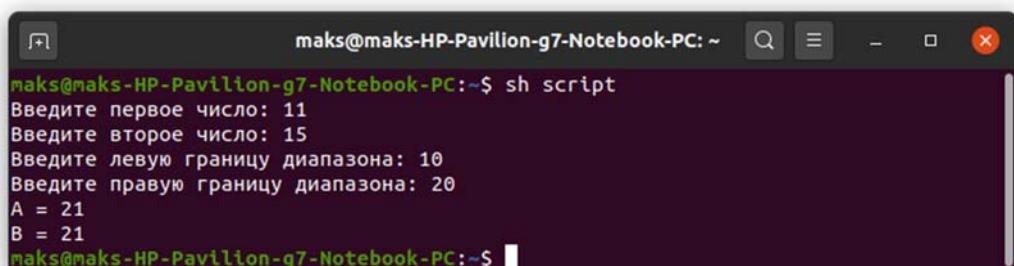
while [ $a -ge $left ] && [ $a -le $right ]
    do a=$(expr $a + 1)
done

while [ $b -ge $left ] && [ $b -le $right ]
    do b=$(expr $b + 1)
done

echo "A = $a"
echo "B = $b"
```

Рисунок 17.1 – Скрипт для задания 17

Запустим скрипт на выполнение:

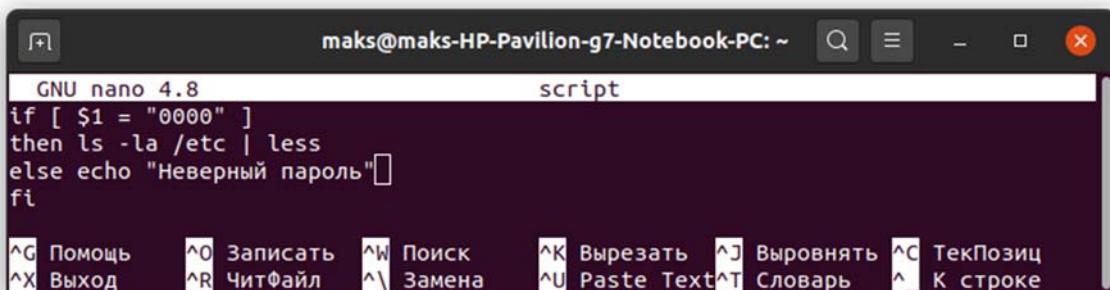


```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script
Введите первое число: 11
Введите второе число: 15
Введите левую границу диапазона: 10
Введите правую границу диапазона: 20
A = 21
B = 21
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

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:

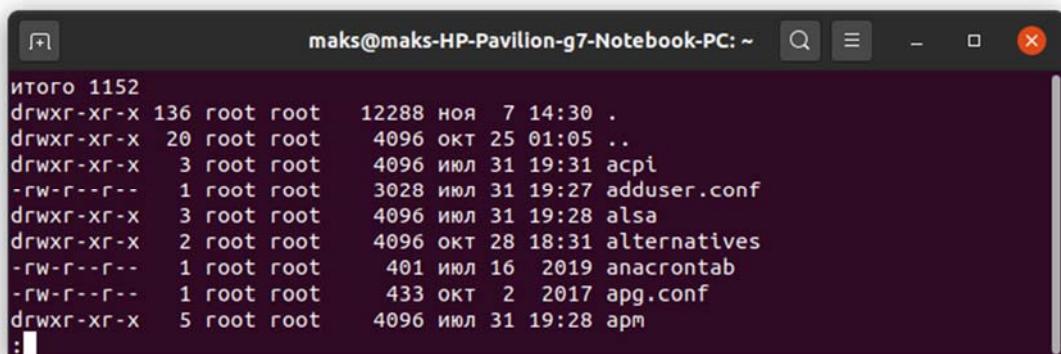


```
GNU nano 4.8                                     script
if [ $1 = "0000" ]
then ls -la /etc | less
else echo "Неверный пароль"
fi
```

Terminal menu bar: Помощь, Записать, Поиск, Вырезать, Выровнять, ТекПозиц, Выход, ЧитФайл, Замена, Paste Text, Словарь, К строке

Рисунок 18.1 – Скрипт для задания 18

Исполним сценарий:

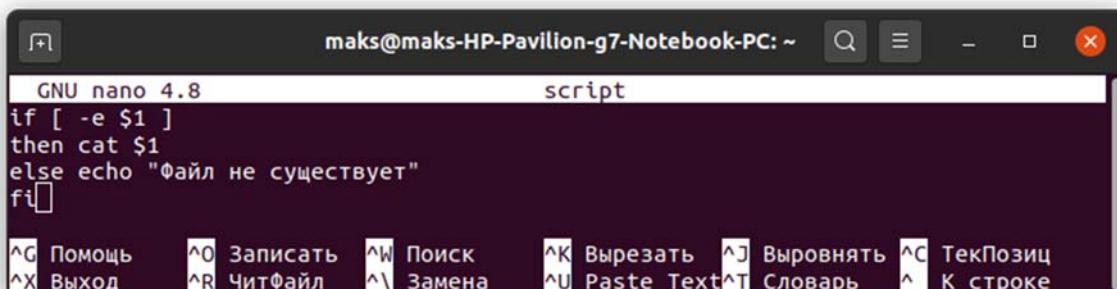


```
макс@maks-HP-Pavilion-g7-Notebook-PC: ~
итого 1152
drwxr-xr-x 136 root root 12288 ноя 7 14:30 .
drwxr-xr-x 20 root root 4096 окт 25 01:05 ..
drwxr-xr-x 3 root root 4096 июл 31 19:31 acpi
-rw-r--r-- 1 root root 3028 июл 31 19:27 adduser.conf
drwxr-xr-x 3 root root 4096 июл 31 19:28 alsa
drwxr-xr-x 2 root root 4096 окт 28 18:31 alternatives
-rw-r--r-- 1 root root 401 июл 16 2019 anacrontab
-rw-r--r-- 1 root root 433 окт 2 2017 apg.conf
drwxr-xr-x 5 root root 4096 июл 31 19:28 apt
:
```

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**:

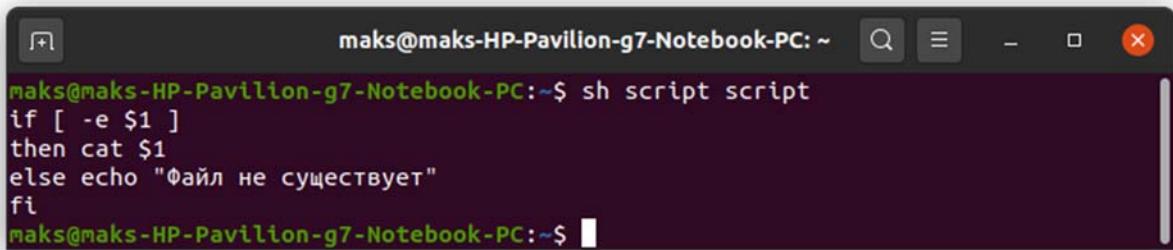


```
GNU nano 4.8                                     script
if [ -e $1 ]
then cat $1
else echo "Файл не существует"
fi
```

Terminal menu bar: Помощь, Записать, Поиск, Вырезать, Выровнять, ТекПозиц, Выход, ЧитФайл, Замена, Paste Text, Словарь, К строке

Рисунок 19.1 – Скрипт для задания 19

Запустим сценарий:

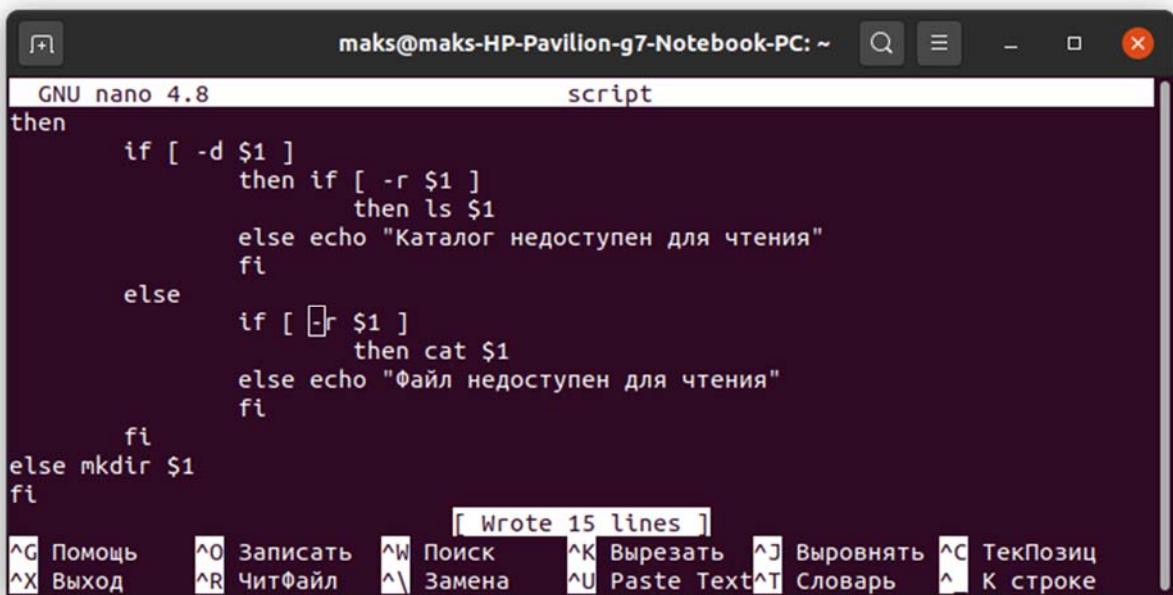


```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script script
if [ -e $1 ]
then cat $1
else echo "Файл не существует"
fi
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

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:

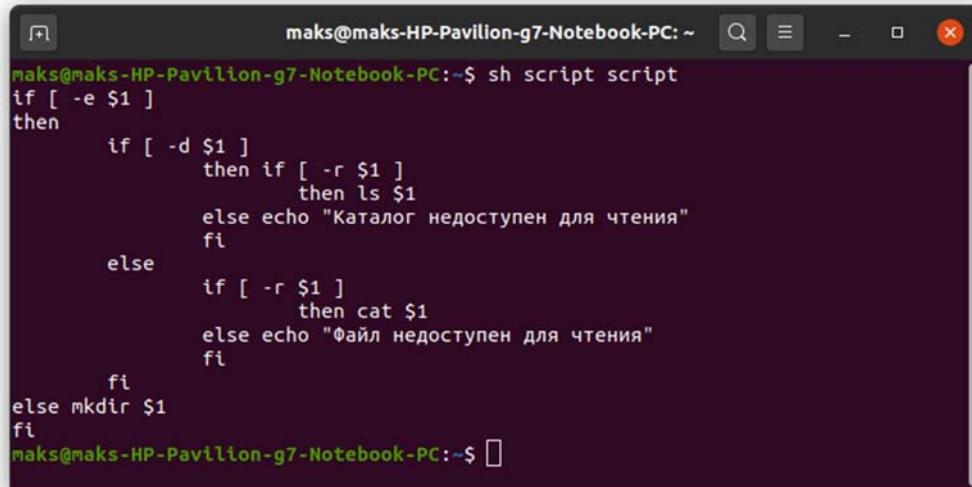


```
GNU nano 4.8                                     script
then
    if [ -d $1 ]
        then if [ -r $1 ]
            then ls $1
        else echo "Каталог недоступен для чтения"
        fi
    else
        if [ -r $1 ]
            then cat $1
        else echo "Файл недоступен для чтения"
        fi
    fi
else mkdir $1
fi
[ Wrote 15 lines ]
```

^G Помощь ^O Записать ^W Поиск ^K Вырезать ^J Выровнять ^C ТекПозиц
^X Выход ^R ЧитФайл ^\| Замена ^U Paste Text ^T Словарь ^ К строке

Рисунок 20.1 – Скрипт для задания 20

Теперь проверим наш сценарий:

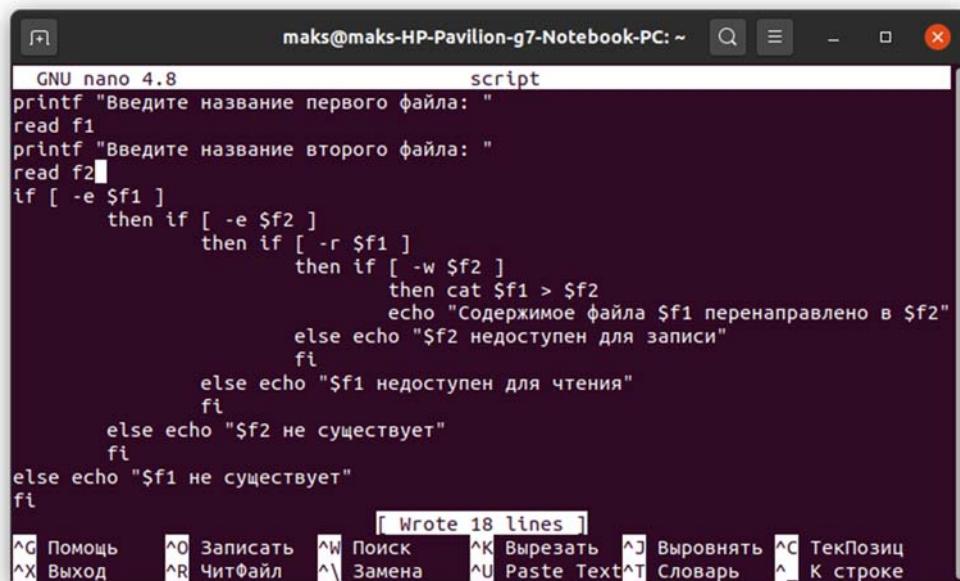


```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script script
if [ -e $1 ]
then
    if [ -d $1 ]
        then if [ -r $1 ]
            then ls $1
            else echo "Каталог недоступен для чтения"
        fi
    else
        if [ -r $1 ]
            then cat $1
            else echo "Файл недоступен для чтения"
        fi
    fi
else mkdir $1
fi
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

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 **-r**. Сначала реализуем задание, спросив названия файлов у пользователя:

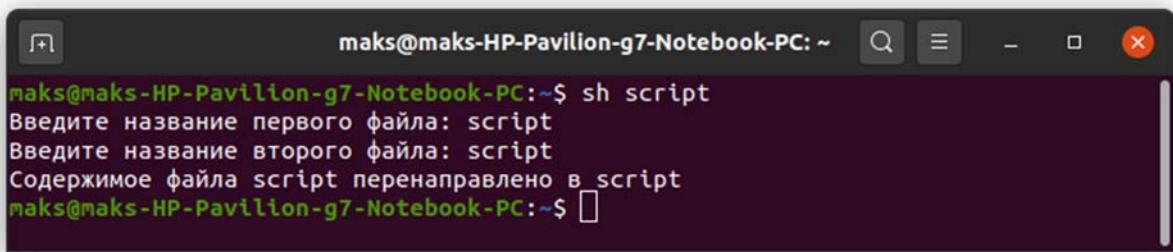


```
GNU nano 4.8          script
Введите название первого файла: "
read f1
Введите название второго файла: "
read f2
if [ -e $f1 ]
    then if [ -e $f2 ]
        then if [ -r $f1 ]
            then if [ -w $f2 ]
                then cat $f1 > $f2
                echo "Содержимое файла $f1 перенаправлено в $f2"
                else echo "$f2 недоступен для записи"
            fi
        else echo "$f1 недоступен для чтения"
    fi
    else echo "$f2 не существует"
    fi
else echo "$f1 не существует"
fi
[ Wrote 18 lines ]
```

^G Помощь ^O Записать ^W Поиск ^K Вырезать ^J Выровнять ^C ТекПозиц
^X Выход ^R ЧитФайл ^\ Замена ^U Paste Text ^T Словарь ^_ К строке

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

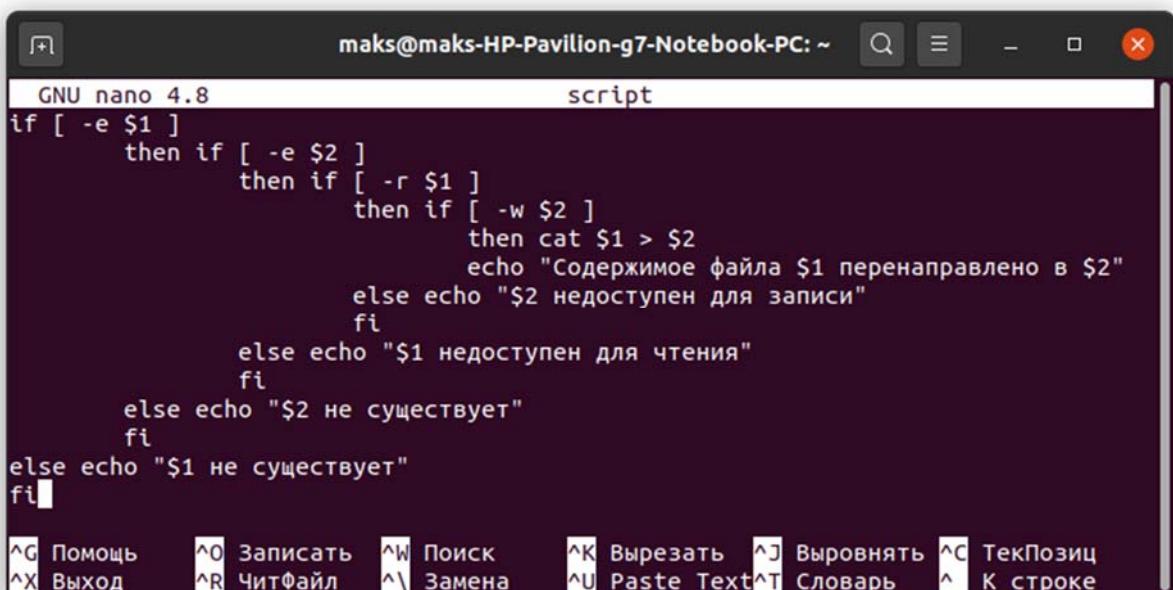
Выполним скрипт:



```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script
Введите название первого файла: script
Введите название второго файла: script
Содержимое файла script перенаправлено в script
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

Рисунок 21.2 – Результат выполнения сценария (через имена файлов)

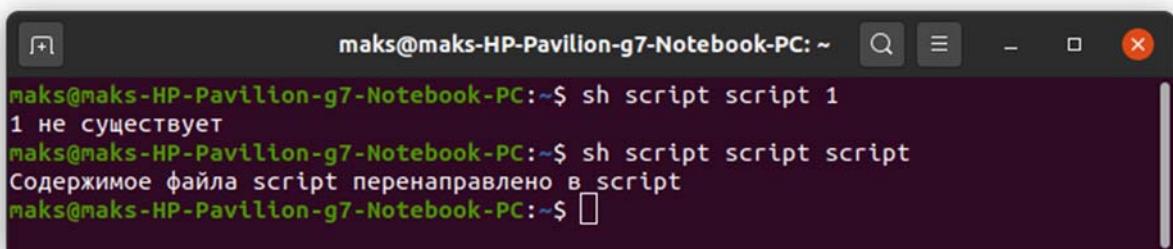
Теперь передадим имена файлов в качестве аргументов командной строки и немного модифицируем наш сценарий:



```
GNU nano 4.8
script
if [ -e $1 ]
    then if [ -e $2 ]
        then if [ -r $1 ]
            then if [ -w $2 ]
                then cat $1 > $2
                echo "Содержимое файла $1 перенаправлено в $2"
                else echo "$2 недоступен для записи"
                fi
            else echo "$1 недоступен для чтения"
            fi
        else echo "$2 не существует"
        fi
    else echo "$1 не существует"
fi
^G Помощь ^O Записать ^W Поиск ^K Вырезать ^J Выровнять ^C ТекПозиц
^X Выход ^R ЧитФайл ^\ Замена ^U Paste Text ^T Словарь ^_ К строке
```

Рисунок 21.3 – Скрипт для задания 21 (через позиционные параметры)

Запустим скрипт:

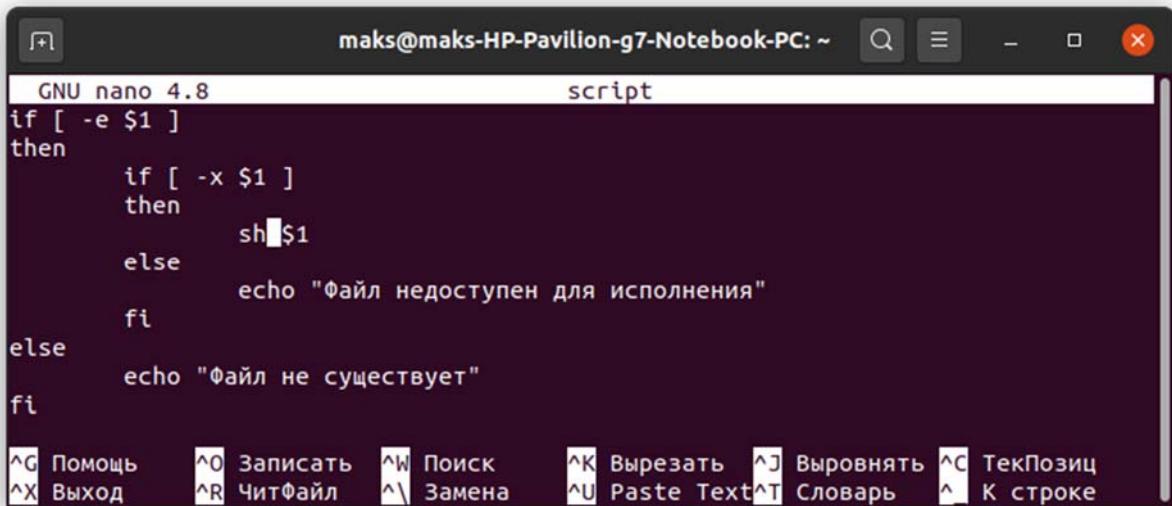


```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script script 1
1 не существует
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script script script
Содержимое файла script перенаправлено в script
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

Рисунок 21.4 – Результат выполнения сценария (через позиционные параметры)

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**):

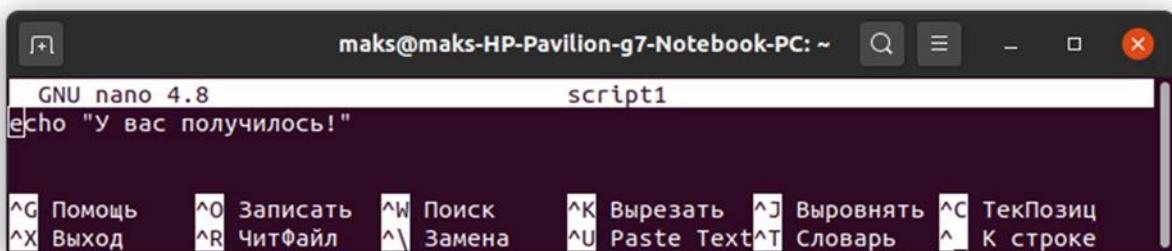


```
GNU nano 4.8          script
if [ -e $1 ]
then
    if [ -x $1 ]
    then
        sh $1
    else
        echo "Файл недоступен для исполнения"
    fi
else
    echo "Файл не существует"
fi
```

^G Помощь ^O Записать ^W Поиск ^K Вырезать ^J Выровнять ^C ТекПозиц
^X Выход ^R ЧитФайл ^\ Замена ^U Paste Text ^T Словарь ^_ К строке

Рисунок 22.1 – Скрипт для задания 22

Затем создадим новый файл сценария, который нам придётся запустить:

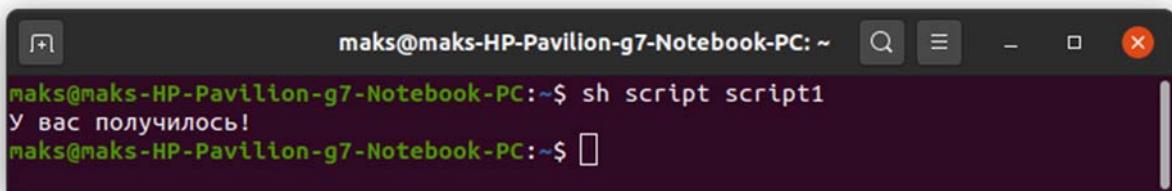


```
GNU nano 4.8          script1
echo "У вас получилось!"
```

^G Помощь ^O Записать ^W Поиск ^K Вырезать ^J Выровнять ^C ТекПозиц
^X Выход ^R ЧитФайл ^\ Замена ^U Paste Text ^T Словарь ^_ К строке

Рисунок 22.2 – Содержание файла script1

Ну и, наконец, запустим наш основной сценарий:

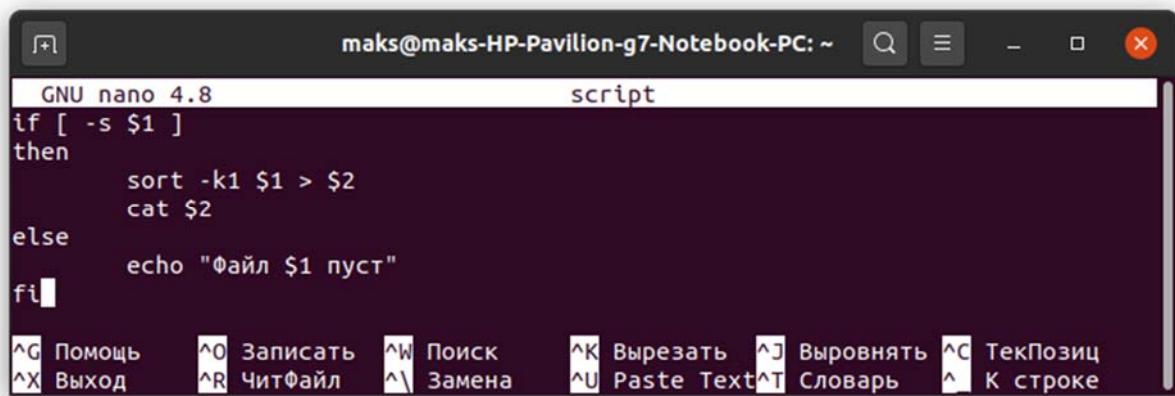


```
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$ sh script1
У вас получилось!
maks@maks-HP-Pavilion-g7-Notebook-PC: ~$
```

Рисунок 22.3 – Результат выполнения сценария

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`:

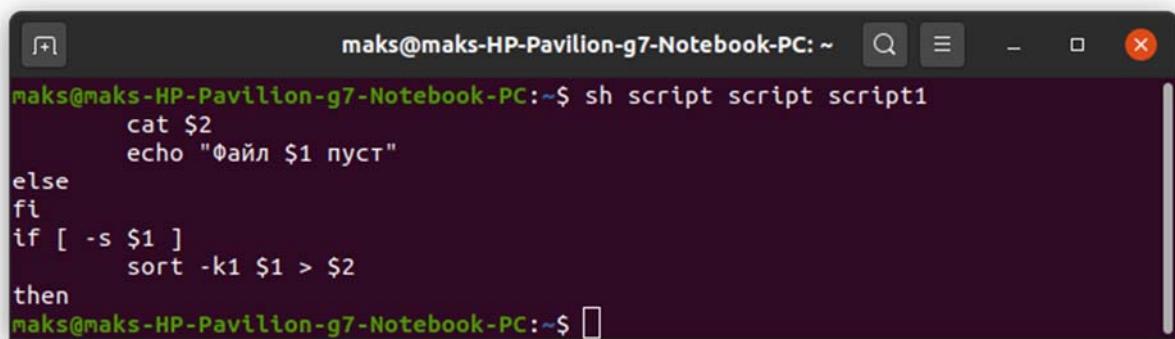


```
GNU nano 4.8                               script
if [ -s $1 ]
then
    sort -k1 $1 > $2
    cat $2
else
    echo "Файл $1 пуст"
fi

^G Помощь      ^O Записать      ^W Поиск      ^K Вырезать      ^J Выровнять      ^C ТекПозиц
^X Выход      ^R ЧитФайл      ^\ Замена      ^U Paste Text ^T Словарь      ^_ К строке
```

Рисунок 23.1 – Скрипт для задания 23

Исполним скрипт:

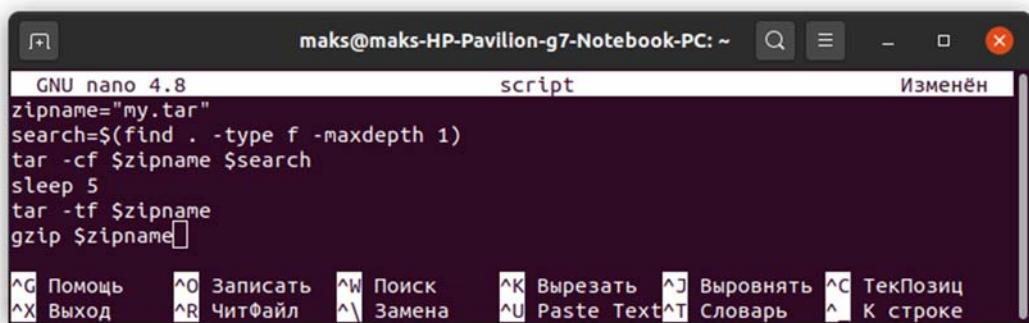


```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script script1
cat $2
echo "Файл $1 пуст"
else
fi
if [ -s $1 ]
    sort -k1 $1 > $2
then
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

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.

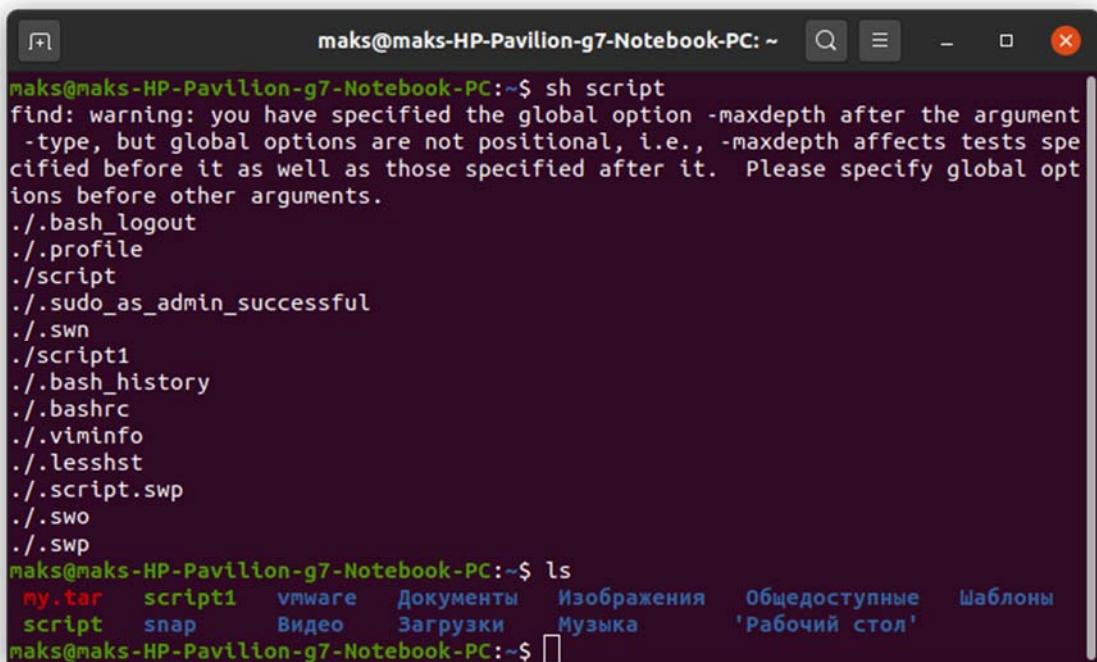
Сначала найдём в данном каталоге с помощью команды **find** нужные файлы (параметр **-maxdepth 1** говорит о том, что в подкаталогах искать не нужно, а параметр **-type f** – о том, что ищем только файлы, а не каталоги). Затем с помощью команды **tar -cf** собираем найденные файлы в один архив. Ну и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



```
GNU nano 4.8
zipname="my.tar"
search=$(find . -type f -maxdepth 1)
tar -cf $zipname $search
sleep 5
tar -tf $zipname
gzip $zipname
```

Рисунок 24.1 – Скрипт для задания 24

Выполним скрипт:

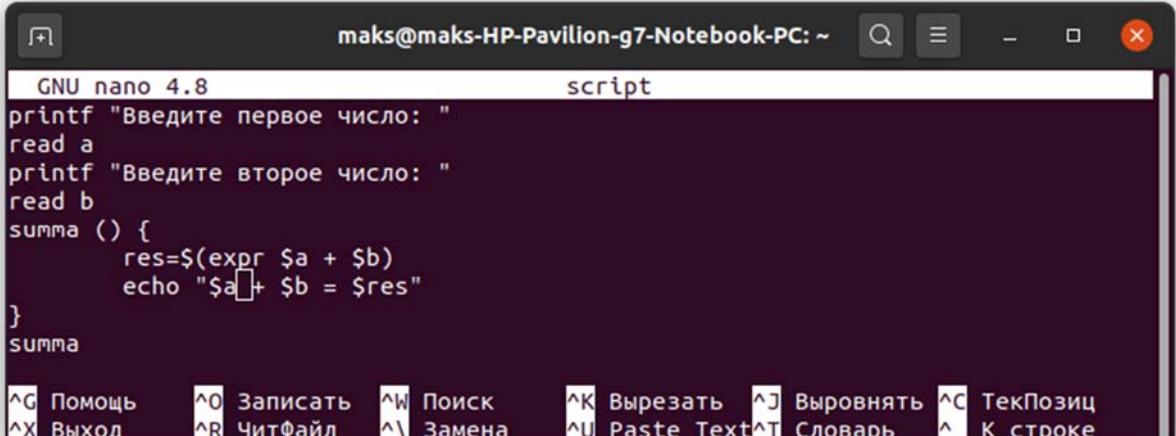


```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script
find: warning: you have specified the global option -maxdepth after the argument
         -type, but global options are not positional, i.e., -maxdepth affects tests spe
cified before it as well as those specified after it. Please specify global opt
ions before other arguments.
./.bash_logout
./.profile
./script
./sudo_as_admin_successful
./swn
./script1
./.bash_history
./.bashrc
./viminfo
./.lessht
./.script.swp
./.swo
./.swp
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ ls
my.tar  script1  vmware  Документы  Изображения  Общедоступные  Шаблоны
script  snap     Видео   Загрузки  Музыка       'Рабочий стол'
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

Рисунок 24.2 – Результат выполнения сценария

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

Напишем сценарий:

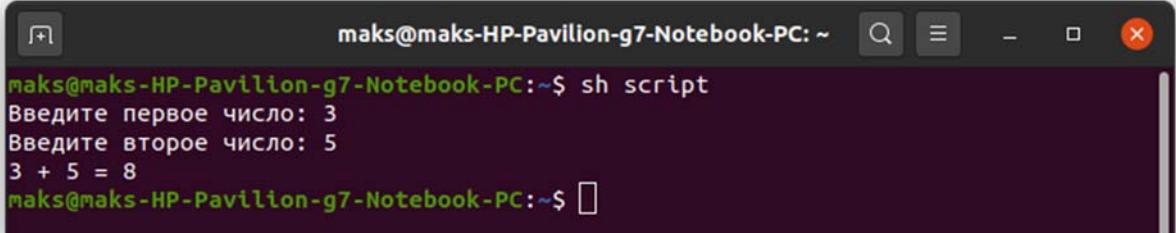


```
GNU nano 4.8                               script
printf "Введите первое число: "
read a
printf "Введите второе число: "
read b
summa () {
    res=$(expr $a + $b)
    echo "$a + $b = $res"
}
summa
```

^G Помощь ^O Записать ^W Поиск ^K Вырезать ^J Выровнять ^C ТекПозиц
^X Выход ^R ЧитФайл ^\| Замена ^U Paste Text ^T Словарь ^ К строке

Рисунок 25.1 – Скрипт для задания 25

Выполним скрипт:



```
maks@maks-HP-Pavilion-g7-Notebook-PC:~$ sh script
Введите первое число: 3
Введите второе число: 5
3 + 5 = 8
maks@maks-HP-Pavilion-g7-Notebook-PC:~$
```

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.