

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №5

по дисциплине «Операционная система Linux»

Программирование в ОС семейства Linux. Программирование на SHELL.

Использование командных файлов.

Студент

Посаднев В.В.

Группа АС-18

Руководитель

Кургасов В.В.

Липецк 2020 г.

Цель работы

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

Задание кафедры

Написать скрипты, при запуске которых выполняются следующие действия:

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, посмотреть 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 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

Задание 1.

Используя команды ECHO, PRINTF вывести информационные сообщения на экран.

Полученный код скрипта:

Задание: Скрипт вывода информации на экран

name='Vitaly_Posadnev'

echo "Информационное сообщение, лабораторная работа 5, скрипт 1, \$name" #вывод информации на экран с помощью команды echo

*printf "Информационное сообщение, лабораторная работа 5, скрипт 1, %s \n" \$name
#вывод информации на экран с помощью команды printf*

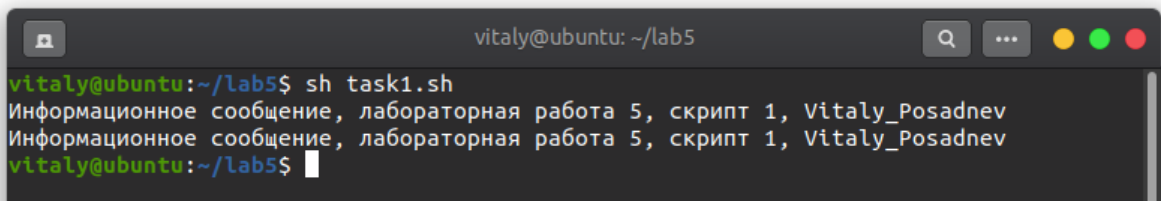


Рисунок 1 – Пример выполнения скрипта вывода информационного сообщения

Задание 2.

Присвоить переменной A целочисленное значение. Просмотреть значение переменной A.

Полученный код скрипта:

Задание: Присвоить переменной A целочисленное значение. Посмотреть значение переменной A

A=2020 #Присвоение переменной A числа 2020

printf "Значение переменной A = %d \n" \$A #Вывод установленного числа

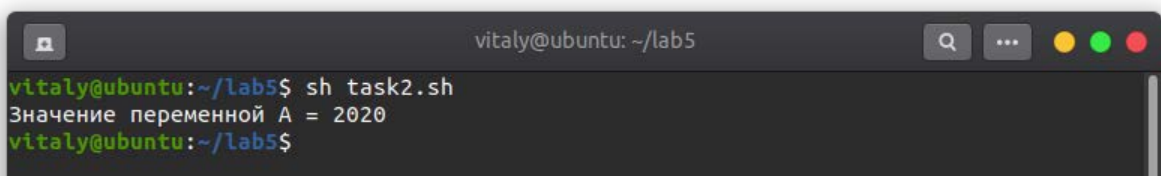


Рисунок 2 – Пример выполнения скрипта вывода целочисленной переменной

Задание 3.

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

Полученный код скрипта:

```
# Задание: Присвоить переменной В значение переменной А. Посмотреть содержимое переменной В.  
A=2020      #Целочисленная переменная равная 2020  
B=$A #В значение В помещаем значение А  
printf "Значение А = %d, значение В = %d\n" $A $B    #Выведем с помощью функции printf значения переменных
```

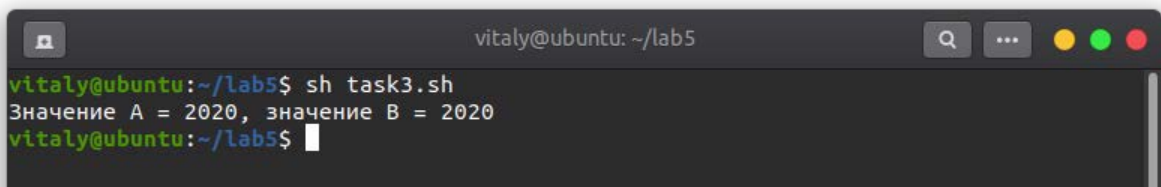


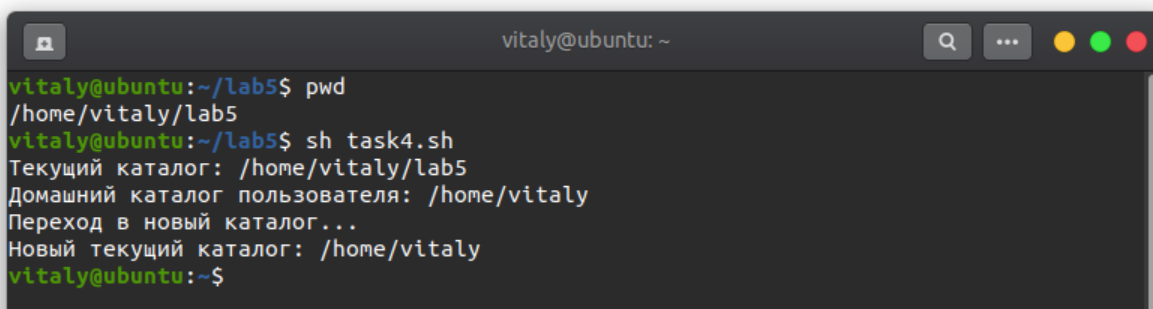
Рисунок 3 – Пример выполнения скрипта присвоения значения другой переменной

Задание 4.

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

Полученный код скрипта:

```
# Задание: Присвоить переменной С значение “путь до своего каталога”. Перейти в этот каталог с использованием переменной.  
printf "Текущий каталог: " # Выводим текущий каталог  
pwd  
home_dir=$HOME # Получаем домашний каталог пользователя  
printf "Домашний каталог пользователя: %s\n" $home_dir # Выводим домашний каталог пользователя  
printf "Переход в новый каталог...\n"  
cd $home_dir # Переходим в домашний каталог пользователя  
printf "Новый текущий каталог: " # Выводим новый текущий каталог  
pwd  
exec bash
```



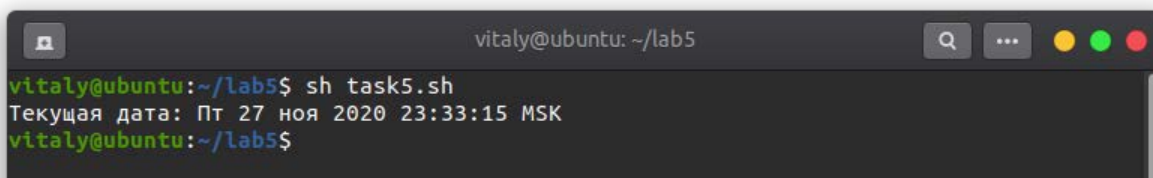
```
vitaly@ubuntu: ~  
vitaly@ubuntu:~/lab5$ pwd  
/home/vitaly/lab5  
vitaly@ubuntu:~/lab5$ sh task4.sh  
Текущий каталог: /home/vitaly/lab5  
Домашний каталог пользователя: /home/vitaly  
Переход в новый каталог...  
Новый текущий каталог: /home/vitaly  
vitaly@ubuntu:~$
```

Рисунок 4 – Пример выполнения скрипта перехода в домашний каталог
Задание 5.

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

Полученный код скрипта:

*# Задание: Присвоить переменной D значение “имя команды”, а именно, команды DATE.
Выполнить эту команду, используя значение переменной.
cur_date="date" # Сохраняем в переменную команду DATE
printf "Текущая дата: " # Выводим текущую дату
\$cur_date*



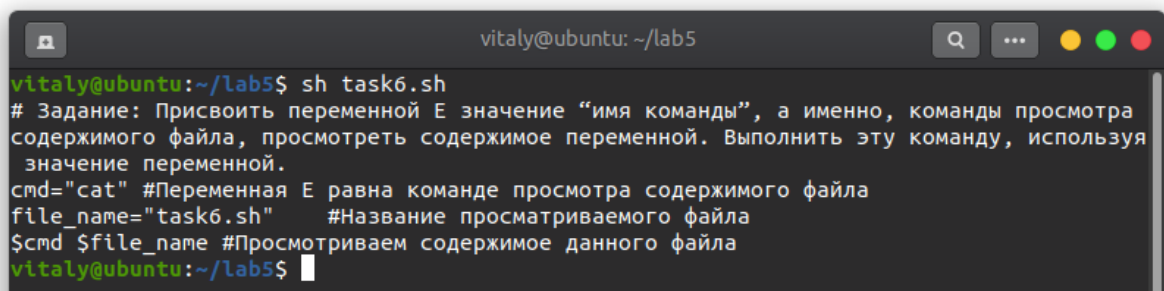
```
vitaly@ubuntu: ~/lab5  
vitaly@ubuntu:~/lab5$ sh task5.sh  
Текущая дата: Пт 27 ноя 2020 23:33:15 MSK  
vitaly@ubuntu:~/lab5$
```

Рисунок 5 – Пример выполнения скрипта вывода текущей даты
Задание 6.

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

Полученный код скрипта:

*# Задание: Присвоить переменной E значение “имя команды”, а именно, команды просмотра содержимого файла, просмотреть содержимое переменной. Выполнить эту команду, используя значение переменной.
cmd="cat" #Переменная E равна команде просмотра содержимого файла
file_name="task6.sh" #Название просматриваемого файла
\$cmd \$file_name #Просматриваем содержимое данного файла*



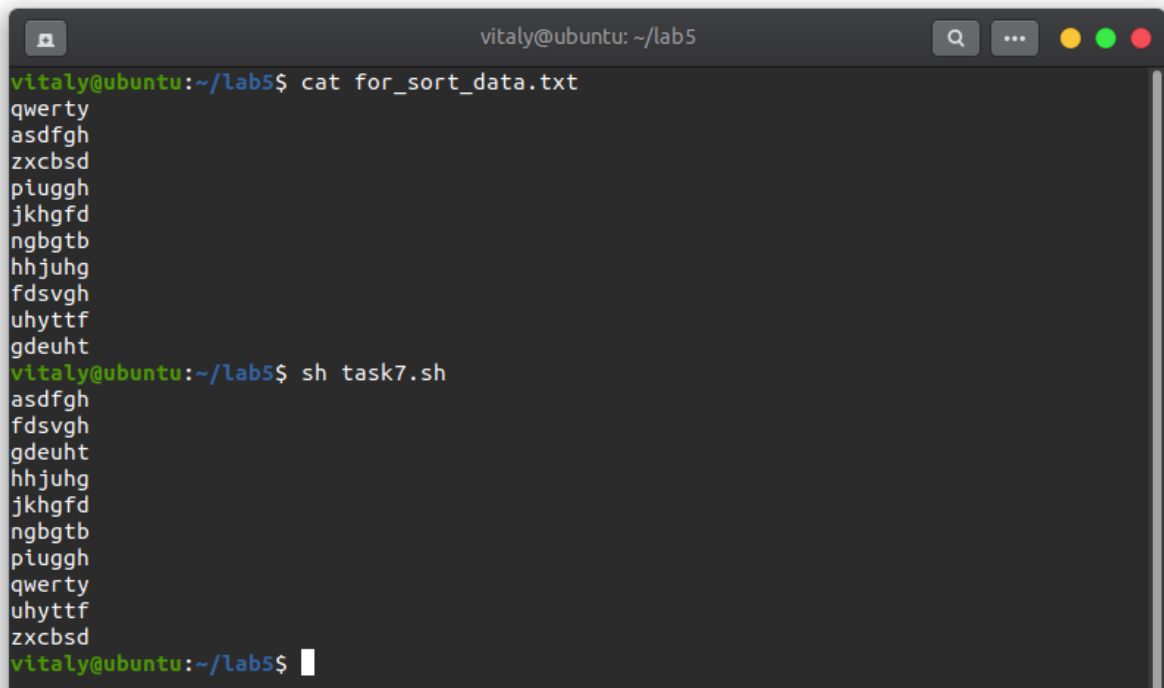
```
vitaly@ubuntu: ~/lab5
vitaly@ubuntu:~/lab5$ sh task6.sh
# Задание: Присвоить переменной E значение "имя команды", а именно, команды просмотра
содержимого файла, просмотреть содержимое переменной. Выполнить эту команду, используя
значение переменной.
cmd="cat" #Переменная E равна команде просмотра содержимого файла
file_name="task6.sh" #Название просматриваемого файла
$cmd $file_name #Просматриваем содержимое данного файла
vitaly@ubuntu:~/lab5$
```

Рисунок 6 – Пример выполнения скрипта вывода содержимого файла
Задание 7.

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

Полученный код скрипта:

```
# Задание: Присвоить переменной F значение “имя команды”, а именно сортировки
содержимого текстового файла. Выполнить эту команду, используя значение переменной.
cmd="sort" # Название команды для сортировки
file_name="for_sort_data.txt" # Название файла откуда будут браться данные для
сортировки
$cmd $file_name # Вызов функции сортировки из указанного файла
```



```
vitaly@ubuntu: ~/lab5
vitaly@ubuntu:~/lab5$ cat for_sort_data.txt
qwerty
asdfgh
zxcbsd
piuggh
jkhgfd
ngbgtd
hhjuhg
fdsvgh
uhyttf
gdeuht
vitaly@ubuntu:~/lab5$ sh task7.sh
asdfgh
fdsvgh
gdeuht
hhjuhg
jkhgfd
ngbgtd
piuggh
qwerty
uhyttf
zxcbsd
vitaly@ubuntu:~/lab5$
```

Рисунок 7 – Пример выполнения скрипта сортировки файла

Задание 8.

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

Полученный код скрипта:

Задание: Программа запрашивает значение переменной, а затем выводит значение этой переменной.

printf "Введите какое-либо значение: " # Вывод сообщения в консоль

read value # Считывание значения из консоли

echo "Введенное значение = \$value" # Вывод результат считывания

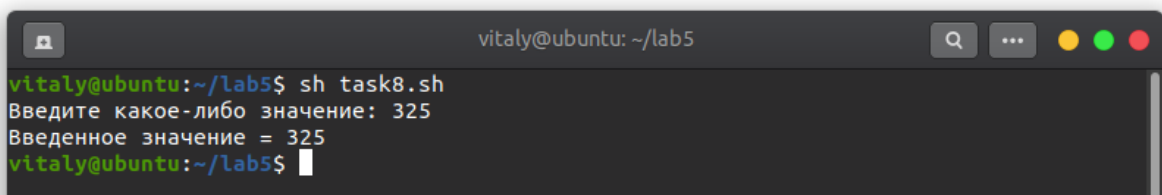


Рисунок 8 – Пример выполнения скрипта считывания введенных данных

Задание 9.

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

Полученный код скрипта:

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

printf "Введите имя пользователя: " #Выводим сообщение на консоль

read user_name #Считываем введенное значение в переменную user_name

echo "Здравствуйте, \$user_name!" #Приветствуем пользователя

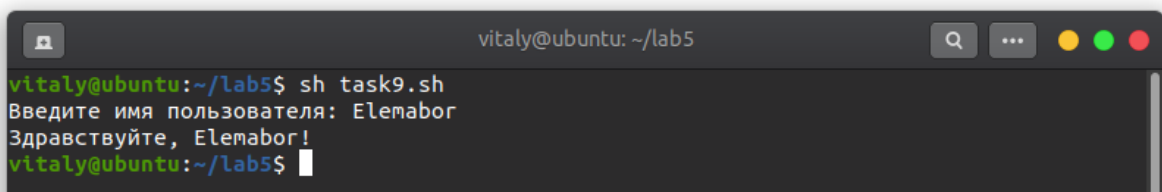


Рисунок 9 – Пример выполнения скрипта приветствия пользователя

Задание 10.

Программа запрашивает значения двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на

экран (использовать команды а) EXPR; б) BC).

Полученный код скрипта:

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

printf "Введите значение первой переменной: " #Выводим сообщение на экран для
считывания первого числа
read first_value #Считываем первое число
printf "Введите значение второй переменной: " #Выводим сообщение на экран для
считывания второго числа
read second_value #Считываем второе число

sum_expr=$(expr $first_value + $second_value) #Рассчитываем сумму через EXPR
dif_expr=$(expr $first_value - $second_value) #Рассчитываем разность через EXPR
mul_expr=$(expr $first_value \* $second_value) #Рассчитываем произведение через EXPR
div_expr=$(expr $first_value / $second_value) #Рассчитываем деление через EXPR

sum_bc=$(echo $first_value + $second_value | bc) #Рассчитываем сумму через BC
dif_bc=$(echo $first_value - $second_value | bc) #Рассчитываем разность через BC
mul_bc=$(echo $first_value \* $second_value | bc) #Рассчитываем произведение через BC
div_bc=$(echo $first_value / $second_value | bc) #Рассчитываем деление через BC

printf "\nСумма вычисленная с помощью EXPR: %s + %s = %s\n" $first_value $second_value
$sum_expr
printf "Разность вычисленная с помощью EXPR: %s - %s = %s\n" $first_value $second_value
$dif_expr
printf "Произведение вычисленное с помощью EXPR: %s * %s = %s\n" $first_value
$second_value $mul_expr
printf "Деление вычисленное с помощью EXPR: %s / %s = %s\n\n" $first_value $second_value
```

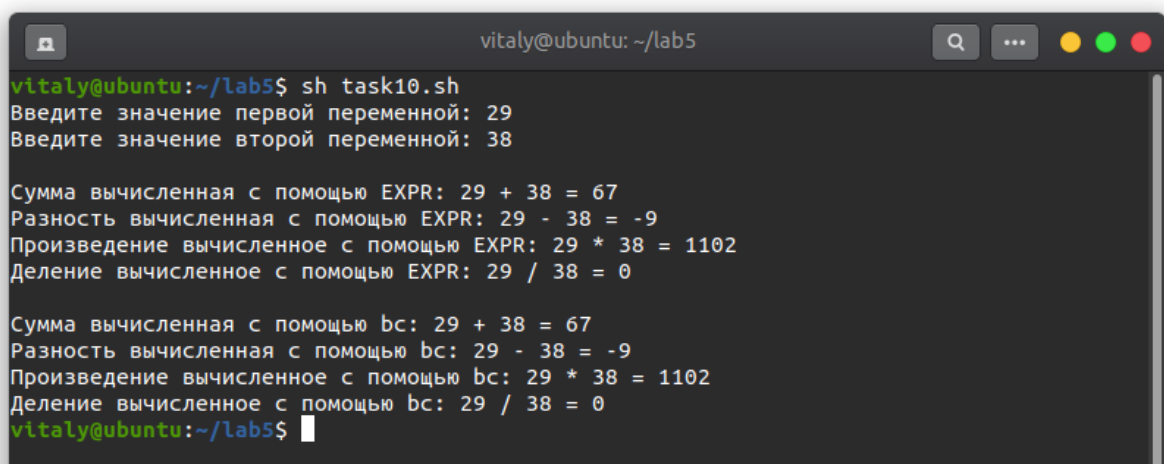
\$div_expr

printf "Сумма вычисленная с помощью bc: %s + %s = %s\n" \$first_value \$second_value \$sum_bc

printf "Разность вычисленная с помощью bc: %s - %s = %s\n" \$first_value \$second_value \$dif_bc

*printf "Произведение вычисленное с помощью bc: %s * %s = %s\n" \$first_value \$second_value \$mul_bc*

printf "Деление вычисленное с помощью bc: %s / %s = %s\n" \$first_value \$second_value \$div_bc



```
vitaly@ubuntu: ~/lab5
vitaly@ubuntu:~/lab5$ sh task10.sh
Введите значение первой переменной: 29
Введите значение второй переменной: 38

Сумма вычисленная с помощью EXPR: 29 + 38 = 67
Разность вычисленная с помощью EXPR: 29 - 38 = -9
Произведение вычисленное с помощью EXPR: 29 * 38 = 1102
Деление вычисленное с помощью EXPR: 29 / 38 = 0

Сумма вычисленная с помощью bc: 29 + 38 = 67
Разность вычисленная с помощью bc: 29 - 38 = -9
Произведение вычисленное с помощью bc: 29 * 38 = 1102
Деление вычисленное с помощью bc: 29 / 38 = 0
vitaly@ubuntu:~/lab5$
```

Рисунок 10 – Пример выполнения скрипта математического вычисления
Задание 11.

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

Полученный код скрипта:

*# Задание: Вычислить объем цилиндра. Исходные данные запрашиваются программой.
Результат выводится на экран.*

printf "Вычисление объема цилиндра происходит по следующей формуле: $V=\pi \cdot h \cdot r^2$ \n"

#Вывод информационного сообщения о вычислении объема

printf "Введите значение радиуса: " #Спрашиваем значение радиуса

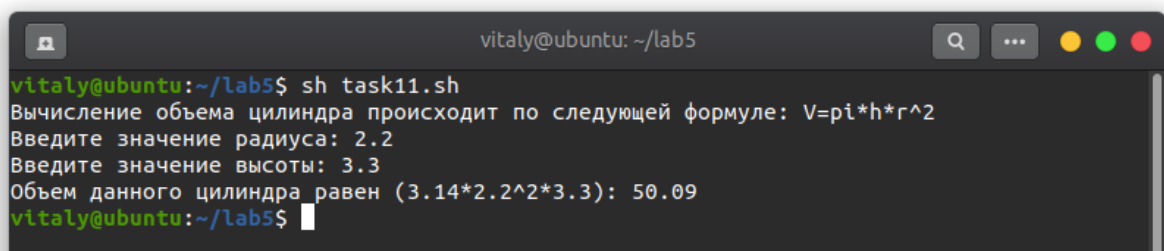
read r #Считываем значение радиуса

printf "Введите значение высоты: " #Спрашиваем значение высоты

read h #Считываем значение высоты

v=\$(echo 3.14\$r*\$r*\$h | bc) #Рассчитываем объем цилиндра*

```
printf "Объем данного цилиндра равен (3.14*%s^2*%s): %s\n" $r $h $v
```



```
vitaly@ubuntu: ~/lab5
vitaly@ubuntu:~/lab5$ sh task11.sh
Вычисление объема цилиндра происходит по следующей формуле: V=pi*h*r^2
Введите значение радиуса: 2.2
Введите значение высоты: 3.3
Объем данного цилиндра равен (3.14*2.2^2*3.3): 50.09
vitaly@ubuntu:~/lab5$
```

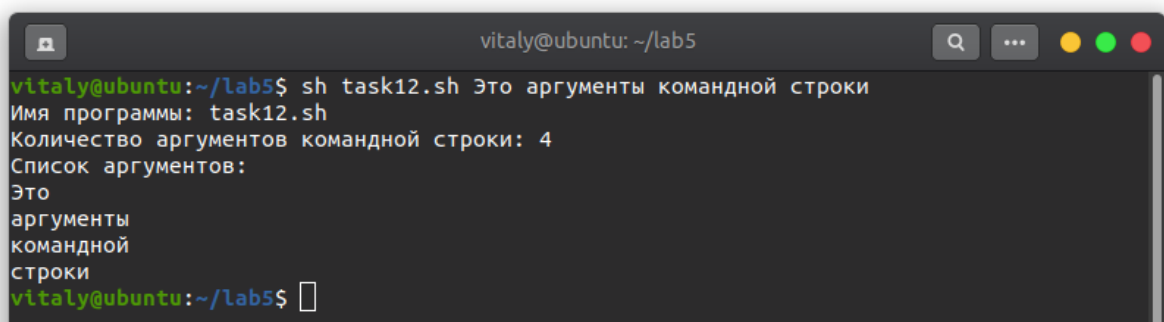
Рисунок 11 – Пример выполнения скрипта вычисления объема цилиндра

Задание 12.

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

Полученный код скрипта:

```
# Задание: Используя позиционные параметры, отобразить имя программы, количество
аргументов командной строки, значение каждого аргумента командной строки.
printf "Имя программы: %s\n" $0 #Выводим имя программы (самый первый аргумент)
printf "Количество аргументов командной строки: %s\n" $#      #Выводим количество
аргументов командной строки
printf "Список аргументов: \n" #Начало вывода каждого аргумента
args="$@" #Считываем все аргумент строки
for arg in $args; #Проходим по всем аргументам и записываем их значение в переменную
arg
do #В цикле выводим
echo "$arg" #Выводим значение текущего аргумента
done #Заканчиваем цикл вывода
```



```
vitaly@ubuntu: ~/lab5
vitaly@ubuntu:~/lab5$ sh task12.sh Это аргументы командной строки
Имя программы: task12.sh
Количество аргументов командной строки: 4
Список аргументов:
Это
аргументы
командной
строки
vitaly@ubuntu:~/lab5$
```

Рисунок 12 – Пример выполнения скрипта вывода аргументов командной строки

Задание 13.

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

Полученный код скрипта:

```
# Задание: Используя позиционный параметр, отобразить содержимое текстового файла,  
указанного в качестве аргумента командной строки. После паузы экран очищается.  
cmd_cnt=$#  
if test $cmd_cnt -ne 1    #Если количество аргументов равно 1 (указано только название  
исполняемого файла т.е. не указан файл для чтения)  
then    #Тогда выводим ошибку  
echo "Ошибка: Укажите аргумент!"  
exit 1 #Завершаем программу с ошибкой  
else cat $1 #Иначе считываем первый аргумент (название необходимого для открытия  
файла) и выводим его содержимое через cat  
fi #Заканчиваем проверка условия  
read _ #Считываем значение из консоли  
clear #Как только введем любой символ (прекратим паузу) очищаем экран
```

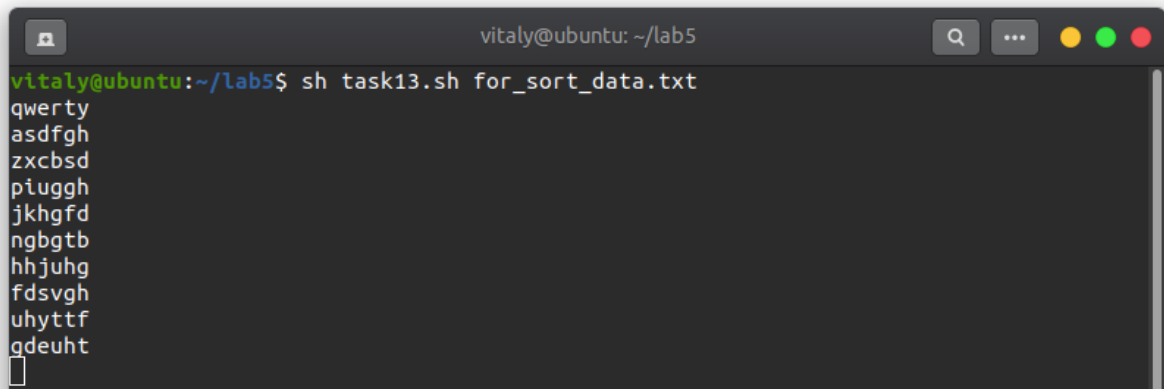


Рисунок 13 – Пример выполнения скрипта вывода содержимого файла и ожидания ввода с клавиатуры

Задание 14.

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

Полученный код скрипта:

Задание: Используя оператор FOR, отобразить содержимое текстовых файлов текущего каталога построчно.

```
list_files_dir=$(ls | grep .sh) #Получаем список всех файлов с расширением .sh
for file in $list_files_dir; #Проходимся по всем элементам списка файлов
do #Начинаем цикл
echo "Содержимое файла $file: " #Выводим название считываемого файла
echo $(cat $file) #Выводим содержимое файла
read _ #Считываем введенное значение консоли
clear #Очищаем экран после вывода содержимого файла
done #Заканчиваем цикл
echo "Конец работы скрипта" #Выводим информационное сообщение
```

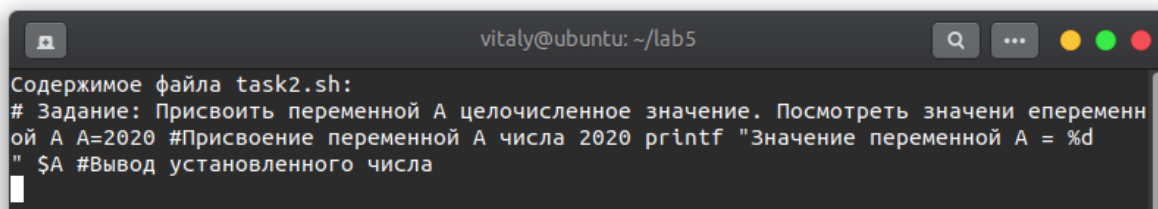


Рисунок 14 – Пример выполнения скрипта постраничного вывода
содержимого файлов каталога

Задание 15.

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

Полученный код скрипта:

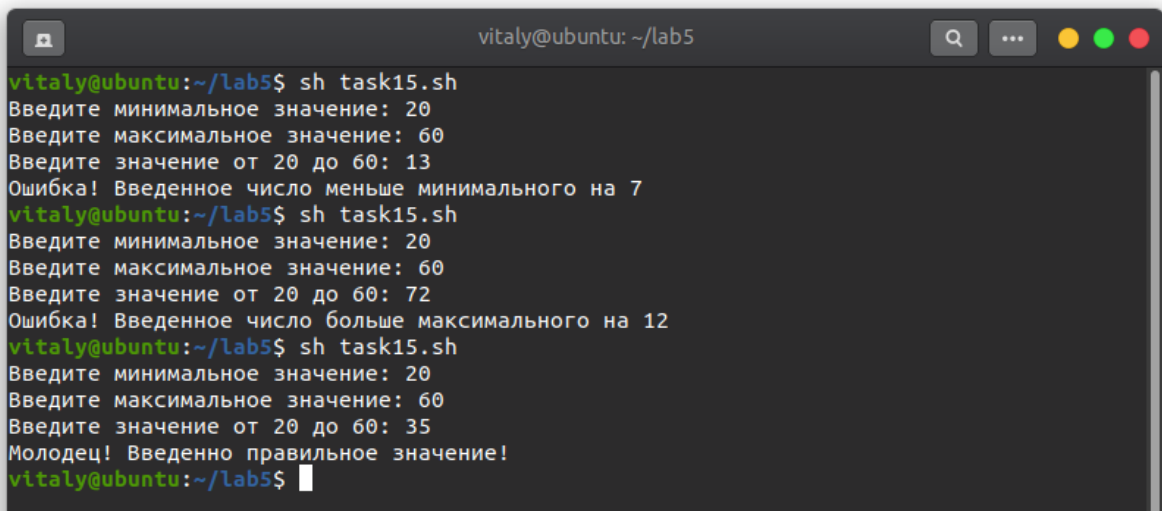
Задание: Программой запрашивается ввод числа, значение которого затем сравнивается с допустимым значением. В результате этого сравнения на экран выдаются соответствующие сообщения.

```
printf "Введите минимальное значение: "
read min_digit # Вводим левое значение диапазона
printf "Введите максимальное значение: "
read max_digit # Вводим левое значение диапазона
printf "Введите значение от %d до %d: " $min_digit $max_digit # Выводим интервал
read digit # Считываем число для диапазона
if test $digit -gt $max_digit; then # Проверяем число на границы диапазона
```

```

printf "Ошибка! Введенное число больше максимального на %d\n" $(echo $digit - $max_digit
/ bc)
elif test $digit -lt $min_digit; then
printf "Ошибка! Введенное число меньше минимального на %d\n" $(echo $min_digit - $digit /
bc)
else
printf "Молодец! Введено правильное значение!\n"
fi

```



```

vitaly@ubuntu: ~/lab5
vitaly@ubuntu:~/lab5$ sh task15.sh
Введите минимальное значение: 20
Введите максимальное значение: 60
Введите значение от 20 до 60: 13
Ошибка! Введенное число меньше минимального на 7
vitaly@ubuntu:~/lab5$ sh task15.sh
Введите минимальное значение: 20
Введите максимальное значение: 60
Введите значение от 20 до 60: 72
Ошибка! Введенное число больше максимального на 12
vitaly@ubuntu:~/lab5$ sh task15.sh
Введите минимальное значение: 20
Введите максимальное значение: 60
Введите значение от 20 до 60: 35
Молодец! Введено правильное значение!
vitaly@ubuntu:~/lab5$

```

Рисунок 15 – Пример выполнения скрипта допустимых значения для введенного числа

Задание 16.

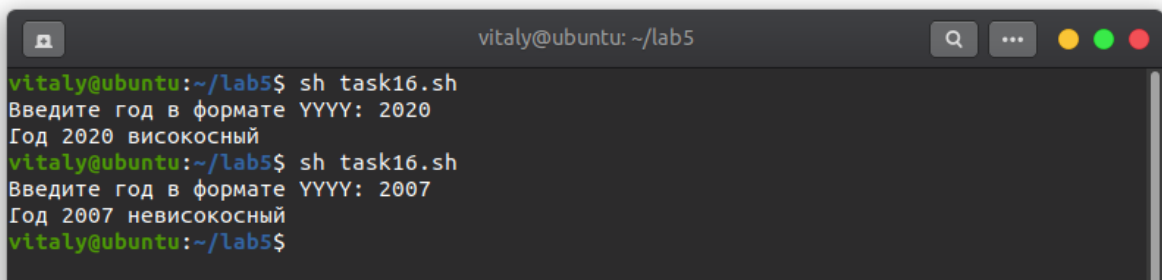
Программой запрашивается год, определяется, високосный ли он. Результат выдается на экран.

Полученный код скрипта:

```

# Задание: Программой запрашивается год, определяется, високосный ли он. Результат в
ыдается на экран.
printf "Введите год в формате YYYY: "
read input_year # Считываем введенный год
if [ $((input_year % 4)) -eq 0 ] # Проверяем остаток деления на 4
then
printf "Год %s високосный\n" $input_year
else
printf "Год %s невисокосный\n" $input_year

```

```
vitaly@ubuntu: ~/lab5
vitaly@ubuntu:~/lab5$ sh task16.sh
Введите год в формате YYYY: 2020
Год 2020 високосный
vitaly@ubuntu:~/lab5$ sh task16.sh
Введите год в формате YYYY: 2007
Год 2007 невисокосный
vitaly@ubuntu:~/lab5$
```

Рисунок 16 – Пример выполнения скрипта проверки високосного года

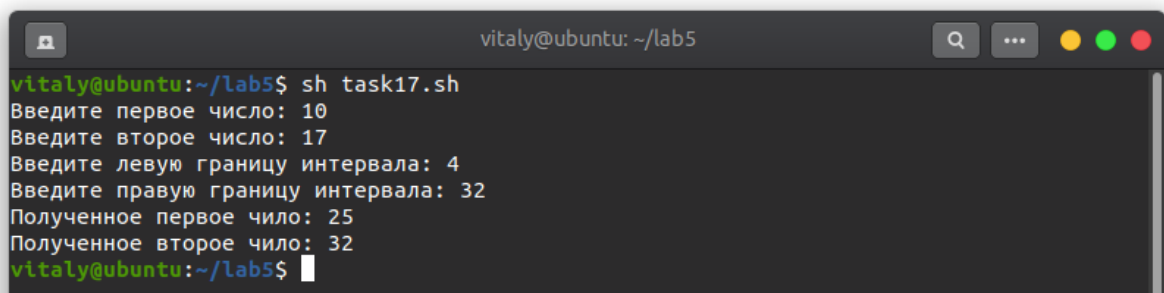
Задание 17.

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

Полученный код скрипта:

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

```
printf "Введите первое число: "
read first_digit # Считываем первое считанное число
printf "Введите второе число: "
read second_digit # Считываем второе считанное число
printf "Введите левую границу интервала: "
read start_interval # Считываем начало интервала
printf "Введите правую границу интервала: "
read end_interval # Считываем конец интервала
while [ $first_digit -ge $start_interval -a $first_digit -lt $end_interval -a $second_digit -ge $start_interval -a $second_digit -lt $end_interval ]
do # Пока выполняется цикл
first_digit=$((first_digit+1)) # Увеличиваем на единицу первое число
second_digit=$((second_digit+1)) # Увеличиваем на единицу второе число
done # Когда выполнение условия закончилось
echo "Полученное первое число: $first_digit" # Выводим полученное первое число
echo "Полученное второе число: $second_digit" # Выводим полученное второе число
```

A screenshot of a terminal window titled 'vitaly@ubuntu: ~/lab5'. The prompt is 'vitaly@ubuntu:~/lab5\$'. The user has entered 'sh task17.sh'. The script outputs: 'Введите первое число: 10', 'Введите второе число: 17', 'Введите левую границу интервала: 4', 'Введите правую границу интервала: 32', 'Полученное первое число: 25', and 'Полученное второе число: 32'. The prompt is now 'vitaly@ubuntu:~/lab5\$' with a cursor.

```
vitaly@ubuntu:~/lab5$ sh task17.sh
Введите первое число: 10
Введите второе число: 17
Введите левую границу интервала: 4
Введите правую границу интервала: 32
Полученное первое число: 25
Полученное второе число: 32
vitaly@ubuntu:~/lab5$
```

Рисунок 17 – Пример выполнения скрипта инкрементирования в диапазоне
Задание 18.

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

Полученный код скрипта:

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

password=hardPassword # Пароль для получения доступа

pswd=\$1 # Берем первый аргумент (после названия скрипта)

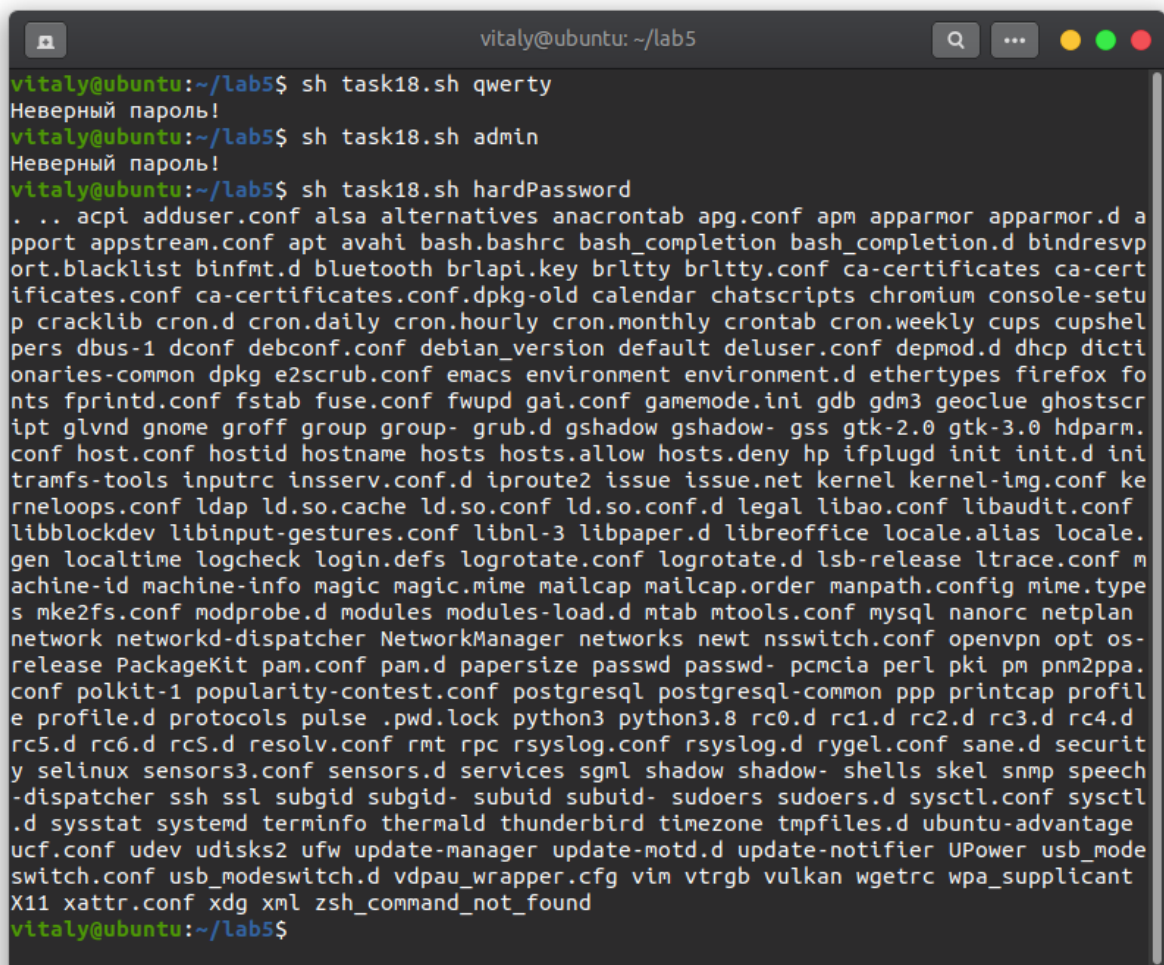
*if ["\$password" = "\$pswd"]; # Сравниваем переданное значение с проверочным
then*

cd /etc # Переходим в нужный каталог

echo \$(ls -a | less) # Выводим содержимое каталога

else

*echo 'Неверный пароль!' # Если пароль оказался неправильным, то выводим сообщение
fi;*

A screenshot of a terminal window titled 'vitaly@ubuntu: ~/lab5'. The user runs a script 'task18.sh' three times with different arguments: 'qwerty', 'admin', and 'hardPassword'. The first two attempts result in the message 'Неверный пароль!' (Incorrect password!). The third attempt lists a long directory of system files and configuration directories, such as 'acpi', 'adduser.conf', 'alsa', 'alternatives', 'anacrontab', 'apc.conf', 'apm', 'apparmor', 'apparmor.d', 'apport', 'appstream.conf', 'apt', 'avahi', 'bash.bashrc', 'bash_completion', 'bash_completion.d', 'bindresvport.blacklist', 'binfmt.d', 'bluetooth', 'brlapi.key', 'brltty', 'brltty.conf', 'ca-certificates', 'ca-certificates.conf', 'ca-certificates.conf.dpkg-old', 'calendar', 'chatscripts', 'chromium', 'console-setup', 'cracklib', 'cron.d', 'cron.daily', 'cron.hourly', 'cron.monthly', 'crontab', 'cron.weekly', 'cups', 'cupshelpers', 'dbus-1', 'dconf', 'debconf.conf', 'debian_version', 'default', 'deluser.conf', 'depmod.d', 'dhcp', ' dictionaries-common', 'dpkg', 'e2scrub.conf', 'emacs', 'environment', 'environment.d', 'ethertypes', 'firefox', 'fonts', 'fprintd.conf', 'fstab', 'fuse.conf', 'fwupd', 'gai.conf', 'gamemode.ini', 'gdb', 'gdm3', 'geoclue', 'ghostscript', 'glvnd', 'gnome', 'groff', 'group', 'group-', 'grub.d', 'gshadow', 'gshadow-', 'gss', 'gtk-2.0', 'gtk-3.0', 'hdparm.conf', 'host.conf', 'hostid', 'hostname', 'hosts', 'hosts.allow', 'hosts.deny', 'hp', 'ifplugd', 'init', 'init.d', 'initramfs-tools', 'inputrc', 'insserv.conf.d', 'iproute2', 'issue', 'issue.net', 'kernel', 'kernel-img.conf', 'kerneloops.conf', 'ldap', 'ld.so.cache', 'ld.so.conf', 'ld.so.conf.d', 'legal', 'libao.conf', 'libaudit.conf', 'libblockdev', 'libinput-gestures.conf', 'libnl-3', 'libpaper.d', 'libreoffice', 'locale.alias', 'locale.gen', 'localtime', 'logcheck', 'login.defs', 'logrotate.conf', 'logrotate.d', 'lsb-release', 'ltrace.conf', 'machine-id', 'machine-info', 'magic', 'magic.mime', 'mailcap', 'mailcap.order', 'manpath.config', 'mime.types', 'mke2fs.conf', 'modprobe.d', 'modules', 'modules-load.d', 'mtab', 'mtools.conf', 'mysql', 'nanorc', 'netplan', 'network', 'networkd-dispatcher', 'NetworkManager', 'networks', 'newt', 'nsswitch.conf', 'openvpn', 'opt', 'os-release', 'PackageKit', 'pam.conf', 'pam.d', 'papersize', 'passwd', 'passwd-', 'pcmcia', 'perl', 'pki', 'pm', 'pnm2ppa.conf', 'polkit-1', 'popularity-contest.conf', 'postgresql', 'postgresql-common', 'ppp', 'printcap', 'profile.d', 'protocols', 'pulse', '.pwd.lock', 'python3', 'python3.8', 'rc0.d', 'rc1.d', 'rc2.d', 'rc3.d', 'rc4.d', 'rc5.d', 'rc6.d', 'rcS.d', 'resolv.conf', 'rmt', 'rpc', 'rsyslog.conf', 'rsyslog.d', 'rygel.conf', 'sane.d', 'security', 'selinux', 'sensors3.conf', 'sensors.d', 'services', 'sgml', 'shadow', 'shadow-', 'shells', 'skel', 'snmp', 'speech-dispatcher', 'ssh', 'ssl', 'subgid', 'subgid-', 'subuid', 'subuid-', 'sudoers', 'sudoers.d', 'sysctl.conf', 'sysctl.d', 'sysstat', 'systemd', 'terminfo', 'thermald', 'thunderbird', 'timezone', 'tmpfiles.d', 'ubuntu-advantage', 'ucf.conf', 'udev', 'udisks2', 'ufw', 'update-manager', 'update-motd.d', 'update-notifier', 'UPower', 'usb_mode switch.conf', 'usb_modeswitch.d', 'vdpa_wrapper.cfg', 'vim', 'vtrgb', 'vulkan', 'wgetrc', 'wpa_supplicant', 'X11', 'xattr.conf', 'xdg', 'xml', 'zsh_command_not_found'.

```
vitaly@ubuntu:~/lab5$ sh task18.sh qwerty
Неверный пароль!
vitaly@ubuntu:~/lab5$ sh task18.sh admin
Неверный пароль!
vitaly@ubuntu:~/lab5$ sh task18.sh hardPassword
. . . acpi adduser.conf alsa alternatives anacrontab apc.conf apm apparmor apparmor.d a
pport appstream.conf apt avahi bash.bashrc bash_completion bash_completion.d bindresvp
ort.blacklist binfmt.d bluetooth brlapi.key brltty brltty.conf ca-certificates ca-cert
ificates.conf ca-certificates.conf.dpkg-old calendar chatscripts chromium console-setu
p cracklib cron.d cron.daily cron.hourly cron.monthly crontab cron.weekly cups cupshel
pers dbus-1 dconf debconf.conf debian_version default deluser.conf depmod.d dhcp dicti
onaries-common dpkg e2scrub.conf emacs environment environment.d ethertypes firefox fo
nts fprintd.conf fstab fuse.conf fwupd gai.conf gamemode.ini gdb gdm3 geoclue ghostscr
ipt glvnd gnome groff group group- grub.d gshadow gshadow- gss gtk-2.0 gtk-3.0 hdparm.
conf host.conf hostid hostname hosts hosts.allow hosts.deny hp ifplugd init init.d ini
tramsfs-tools inputrc insserv.conf.d iproute2 issue issue.net kernel kernel-img.conf ke
rneloops.conf ldap ld.so.cache ld.so.conf ld.so.conf.d legal libao.conf libaudit.conf
libblockdev libinput-gestures.conf libnl-3 libpaper.d libreoffice locale.alias locale.
gen localtime logcheck login.defs logrotate.conf logrotate.d lsb-release ltrace.conf m
achine-id machine-info magic magic.mime mailcap mailcap.order manpath.config mime.type
s mke2fs.conf modprobe.d modules modules-load.d mtab mtools.conf mysql nanorc netplan
network networkd-dispatcher NetworkManager networks newt nsswitch.conf openvpn opt os-
release PackageKit pam.conf pam.d papersize passwd passwd- pcmcia perl pki pm pnm2ppa.
conf polkit-1 popularity-contest.conf postgresql postgresql-common ppp printcap profil
e profile.d protocols pulse .pwd.lock python3 python3.8 rc0.d rc1.d rc2.d rc3.d rc4.d
rc5.d rc6.d rcS.d resolv.conf rmt rpc rsyslog.conf rsyslog.d rygel.conf sane.d securit
y selinux sensors3.conf sensors.d services sgml shadow shadow- shells skel snmp speech
-dispatcher ssh ssl subgid subgid- subuid subuid- sudoers sudoers.d sysctl.conf sysctl
.d sysstat systemd terminfo thermald thunderbird timezone tmpfiles.d ubuntu-advantage
ucf.conf udev udisks2 ufw update-manager update-motd.d update-notifier UPower usb_mode
switch.conf usb_modeswitch.d vdpau_wrapper.cfg vim vtrgb vulkan wgetrc wpa_supplicant
X11 xattr.conf xdg xml zsh_command_not_found
vitaly@ubuntu:~/lab5$
```

Рисунок 18 – Пример выполнения скрипта с проверкой пароля

Задание 19.

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

Полученный код скрипта:

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

fileName=\$1 # Считываем первый аргумент команды (после названия скрипта)

if [-f "\$fileName"] # Проверяем существование данного файла

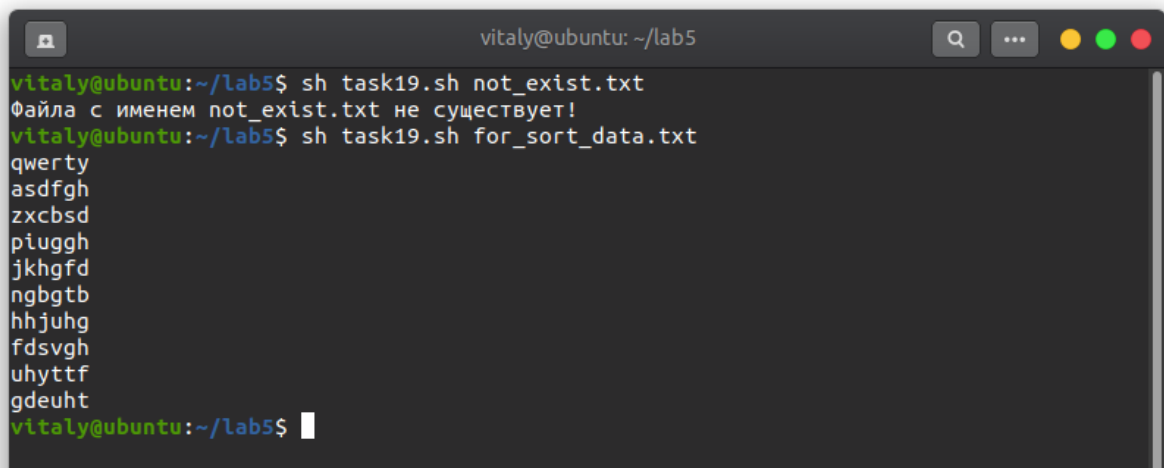
then

cat \$fileName # Выводим содержимое файла

else

printf "Файла с именем %s не существует!\n" \$fileName # В случае ошибки, выводим сообщение об ошибке

fi



```
vitaly@ubuntu: ~/lab5
vitaly@ubuntu:~/lab5$ sh task19.sh not_exist.txt
Файла с именем not_exist.txt не существует!
vitaly@ubuntu:~/lab5$ sh task19.sh for_sort_data.txt
qwerty
asdfgh
zxcbsd
piuggh
jkhgfd
ngbgfb
hhjuhg
fdsvgh
uhyttf
gdeuht
vitaly@ubuntu:~/lab5$
```

Рисунок 19 – Пример выполнения скрипта проверки существования файла с последующем выводом

Задание 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.

Полученный код скрипта:

Задание: 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.

fileName=\$1 # Считываем первый аргумент команды (после названия скрипта)

if [-d "\$fileName" -a -r "\$fileName"] # Проверяем, что указанный аргумент является каталогом и доступен для чтения

then

ls \$fileName # Выводим содержимого каталога

elif [-f "\$fileName"] # Проверяем, что указанный аргумент является файлом

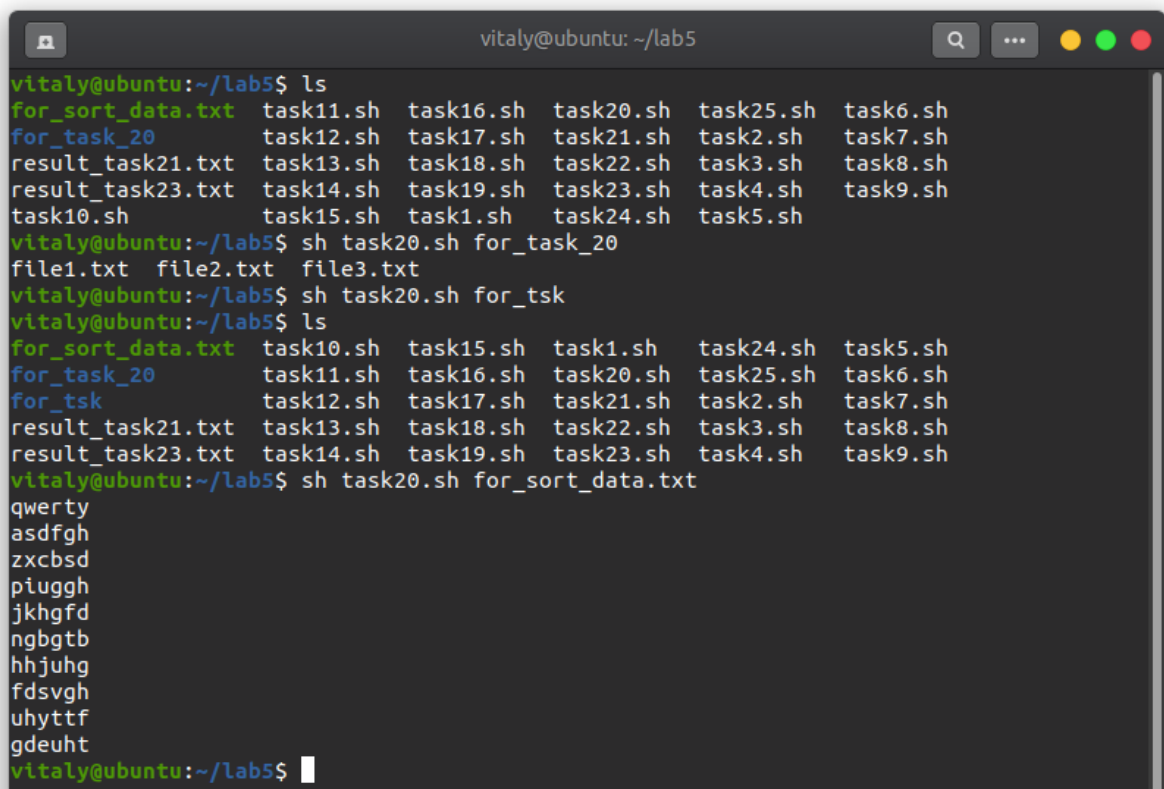
then

cat \$fileName # Выводим содержимое файла

else

mkdir \$fileName # В случае невыполнения вышеуказанных условий создаем данный каталог

fi



```
vitaly@ubuntu: ~/lab5
vitaly@ubuntu:~/lab5$ ls
for_sort_data.txt  task11.sh  task16.sh  task20.sh  task25.sh  task6.sh
for_task_20        task12.sh  task17.sh  task21.sh  task2.sh   task7.sh
result_task21.txt  task13.sh  task18.sh  task22.sh  task3.sh   task8.sh
result_task23.txt  task14.sh  task19.sh  task23.sh  task4.sh   task9.sh
task10.sh          task15.sh  task1.sh   task24.sh  task5.sh
vitaly@ubuntu:~/lab5$ sh task20.sh for_task_20
file1.txt file2.txt file3.txt
vitaly@ubuntu:~/lab5$ sh task20.sh for_tsk
vitaly@ubuntu:~/lab5$ ls
for_sort_data.txt  task10.sh  task15.sh  task1.sh   task24.sh  task5.sh
for_task_20        task11.sh  task16.sh  task20.sh  task25.sh  task6.sh
for_tsk            task12.sh  task17.sh  task21.sh  task2.sh   task7.sh
result_task21.txt  task13.sh  task18.sh  task22.sh  task3.sh   task8.sh
result_task23.txt  task14.sh  task19.sh  task23.sh  task4.sh   task9.sh
vitaly@ubuntu:~/lab5$ sh task20.sh for_sort_data.txt
qwerty
asdfgh
zxcbsd
piuggh
jkhgfd
ngbgfb
hhjuhg
fdsvgh
uhyttf
gdeuht
vitaly@ubuntu:~/lab5$
```

Рисунок 20 – Пример выполнения скрипта проверки файла на каталог
Задание 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).

Полученный код скрипта:

Задание: 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).

echo "Пункт а (имена файлов for_sort_data.txt и result_task21.txt)"

firstFile=for_sort_data.txt # Название первого файла

```

secondFile=result_task21.txt  # Название второго файла
if [ -f $firstFile -a -r $firstFile -a -f $secondFile -a -w $secondFile ] # Проверяем необходимые
условия
then
    cat $firstFile > $secondFile  # Перенаправляем содержимое первого файла во второй
else
    printf "Файл %s и(или) %s не существуют, либо обладают неправильными
атрибутами!\n" $firstFile $secondFile
fi

```

```

# Позиционные параметры
echo "Пункт б (считываем названия файлов из аргументов)"
firstFile=$1  # Считываем первый аргумент команды (после названия скрипта)
secondFile=$2  # Считываем второй аргумент команды (после названия первого файла)
if [ -f $firstFile -a -r $firstFile -a -f $secondFile -a -w $secondFile ] # Проверяем необходимые
условия
then
    cat $firstFile > $secondFile  # Перенаправляем содержимого первого файла во второй
else
    printf "Файл %s и(или) %s не существуют, либо обладают неправильными
атрибутами!\n" $firstFile $secondFile
fi

```

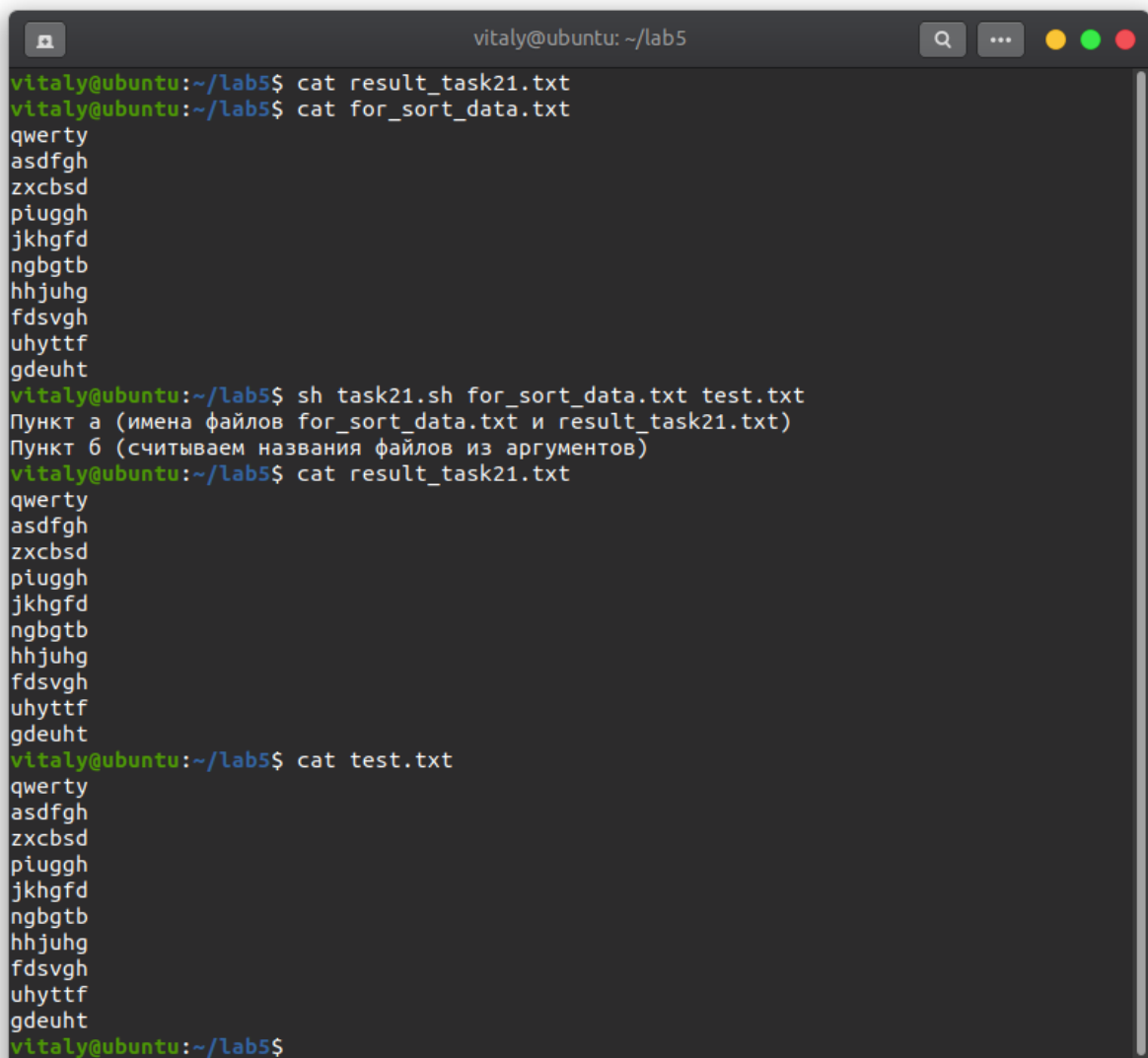
A screenshot of a terminal window titled 'vitaly@ubuntu: ~/lab5'. The terminal shows a series of commands and their outputs. The user first runs 'cat result_task21.txt' and 'cat for_sort_data.txt', both displaying a list of keyboard layouts: qwerty, asdfgh, zxcbsd, piuggh, jkhgfd, ngbgfb, hhjuhg, fdsvgh, uhyttf, gdeuht. Then, the user runs 'sh task21.sh for_sort_data.txt test.txt'. Below this, there are two lines of Russian text: 'Пункт а (имена файлов for_sort_data.txt и result_task21.txt)' and 'Пункт б (считываем названия файлов из аргументов)'. The user then runs 'cat result_task21.txt' and 'cat test.txt', both displaying the same list of keyboard layouts. The terminal ends with the prompt 'vitaly@ubuntu:~/lab5\$'.

Рисунок 21 – Пример выполнения скрипта анализа атрибутов файлов
Задание 22.

Если файл запуска программы найден, программа запускается (по выбору).

Полученный код скрипта:

```
# Задание: Если файл запуска программы найден, программа запускается (по выбору).  
if [ $# -eq 0 ] #Проверка, что передан параметр  
then #Если параметров не передано, то выводим сообщение  
    echo "Передайте название программы аргументом!"  
else #Если аргумент передан  
    progName=$1 # Считываем первый аргумент (название открываемой программы)  
    cd /bin #Переходим в папку с программами
```

```

find $progName && $progName # Пытаемся найти программу и открываем её
fi

```

```

vitaly@ubuntu: ~/lab5
vitaly@ubuntu:~/lab5$ sh task22.sh
Передайте название программы аргументом!
vitaly@ubuntu:~/lab5$ sh task22.sh ip
ip
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename
where  OBJECT := { link | address | addrlabel | route | rule | neigh | ntable |
                  tunnel | tuntap | maddress | mroute | mrule | monitor | xfrm |
                  netns | l2tp | fou | macsec | tcp_metrics | token | netconf | ila |
                  vrf | sr | nexthop }
OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
             -h[uman-readable] | -iec | -j[son] | -p[retty] |
             -f[amily] { inet | inet6 | mpls | bridge | link } |
             -4 | -6 | -I | -D | -M | -B | -O |
             -l[oops] { maximum-addr-flush-attempts } | -br[ief] |
             -o[neline] | -t[imestamp] | -ts[hort] | -b[atch] [filename] |
             -rc[vbuf] [size] | -n[etns] name | -N[umeric] | -a[ll] |
             -c[olor]}
vitaly@ubuntu:~/lab5$ sh task22.sh not_exist
find: 'not_exist': No such file or directory
vitaly@ubuntu:~/lab5$

```

Рисунок 22 – Пример выполнения скрипта проверки и запуска программы

Задание 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.

Полученный код скрипта:

Задание: 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.

```
if [ $# -gt 1 ] # Проверяем, что переданы 2 аргумента
```

```
then # Если переданы 2 аргумента
```

```
    srcFile=$1 # Считываем название первого файла
```

```
    dstFile=$2 # Считываем название второго файла
```

```
    if [ -s "$srcFile" ] # Проверяем, что файл существует и не пуст
```

```
    then # Если существует и не пуст
```

```
        sort -k 1 $srcFile > $dstFile # Сортируем по первому столбцу файл srcFile и перенаправляем в файл dstFile
```



```

        cat $dstFile  # Выводим отсортированный файл
    else  # Если файл не существует или пуст
        printf "Файл %s не существует или пуст\n" $srcFile
    fi
else  # Если указаны не все файлы
    echo "Ошибка! Файлы не указаны!"
fi

```

```

vitaly@ubuntu: ~/lab5
vitaly@ubuntu:~/lab5$ touch result_task23.txt
vitaly@ubuntu:~/lab5$ cat result_task23.txt
vitaly@ubuntu:~/lab5$ sh task23.sh for_sort_data.txt result_task23.txt
asdfgh
fdsvgh
gdeuht
hhjuhg
jkhgfd
ngbgfb
piuggh
qwerty
uhyttf
zxcbsd
vitaly@ubuntu:~/lab5$ cat result_task23.txt
asdfgh
fdsvgh
gdeuht
hhjuhg
jkhgfd
ngbgfb
piuggh
qwerty
uhyttf
zxcbsd
vitaly@ubuntu:~/lab5$

```

Рисунок 23 – Пример выполнения скрипта сортировки по первому столбцу
Задание 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.

Полученный код скрипта:

```

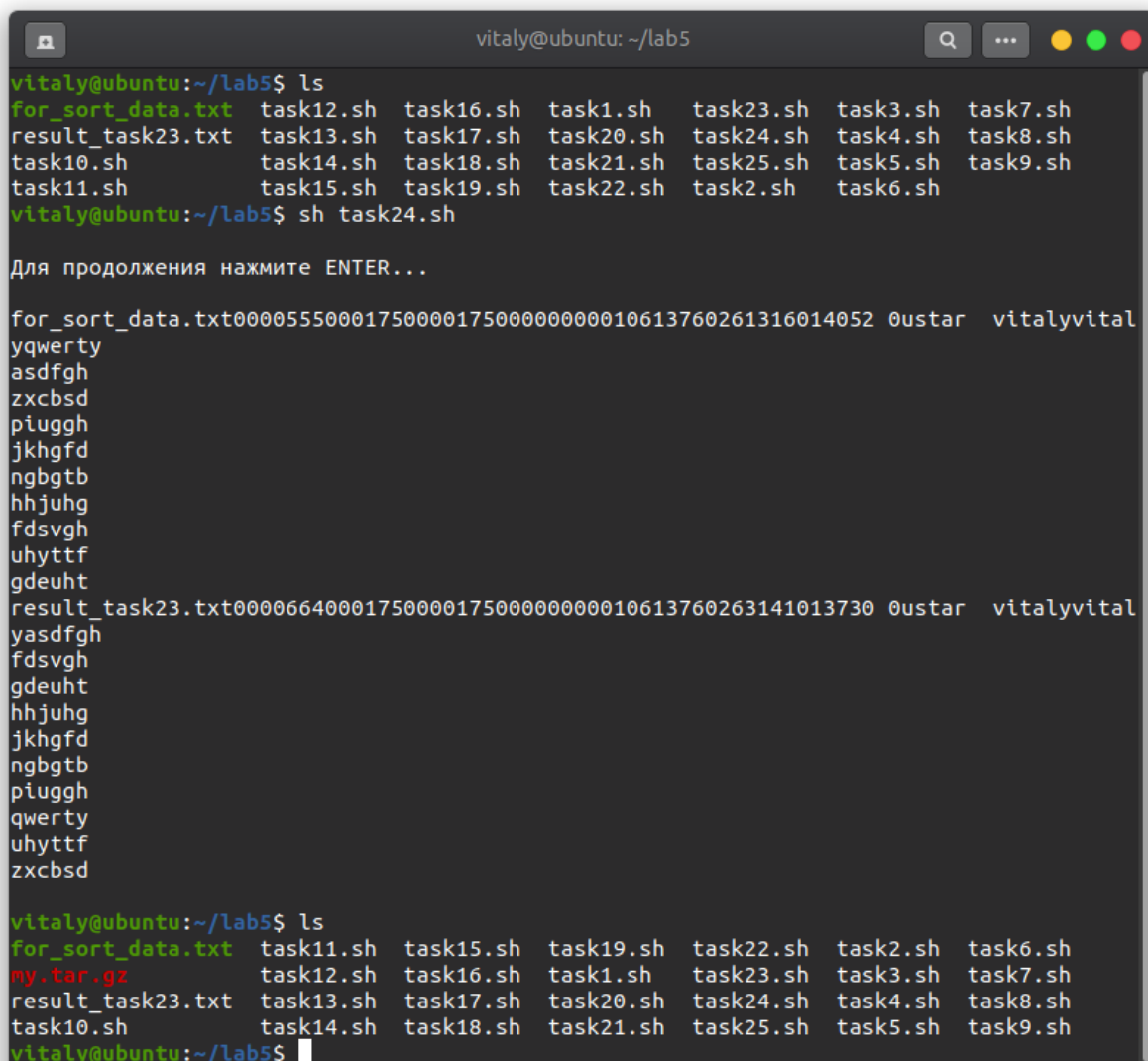
# Задание: Командой TAR осуществляется сборка всех текстовых файлов текущего
каталога в один архивный файл my.tar, после паузы просматривается содержимое файла
my.tar, затем командой GZIP архивный файл my.tar сжимается.
tar -cf my.tar *.txt  # Создаем архив и указываем ему название (это делается с помощью
ключа -c),

```

```

# Для упаковки содержимого в один файл используем ключ -f
printf "\nДля продолжения нажмите ENTER...\n" # Ожидаем нажатие кнопки ENTER
read key # Считываем введенное значение
cat my.tar # Выводим содержимое файла
gzip my.tar # Сжимаем файл my.tar
echo # Выводим пустую строку

```



```

vitaly@ubuntu: ~/lab5
vitaly@ubuntu:~/lab5$ ls
for_sort_data.txt  task12.sh  task16.sh  task1.sh  task23.sh  task3.sh  task7.sh
result_task23.txt  task13.sh  task17.sh  task20.sh  task24.sh  task4.sh  task8.sh
task10.sh          task14.sh  task18.sh  task21.sh  task25.sh  task5.sh  task9.sh
task11.sh          task15.sh  task19.sh  task22.sh  task2.sh   task6.sh
vitaly@ubuntu:~/lab5$ sh task24.sh

Для продолжения нажмите ENTER...

for_sort_data.txt000055500017500001750000000010613760261316014052 0ustar  vitalyvital
yqwerty
asdfgh
zxcbsd
piuggh
jkhgfd
ngbgfb
hhjuhg
fdsvgh
uhyttf
gdeuht
result_task23.txt000066400017500001750000000010613760263141013730 0ustar  vitalyvital
yasdfgh
fdsvgh
gdeuht
hhjuhg
jkhgfd
ngbgfb
piuggh
qwerty
uhyttf
zxcbsd

vitaly@ubuntu:~/lab5$ ls
for_sort_data.txt  task11.sh  task15.sh  task19.sh  task22.sh  task2.sh  task6.sh
my.tar.gz          task12.sh  task16.sh  task1.sh  task23.sh  task3.sh  task7.sh
result_task23.txt  task13.sh  task17.sh  task20.sh  task24.sh  task4.sh  task8.sh
task10.sh          task14.sh  task18.sh  task21.sh  task25.sh  task5.sh  task9.sh
vitaly@ubuntu:~/lab5$

```

Рисунок 24 – Пример выполнения скрипта создания архива из текстовых файлов с последующим сжатием

Задание 25.

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

Полученный код скрипта:

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

Функция вычисления суммы двух чисел

```
sum_func() {  
    if [ $# -gt 1 ]  
    then  
        res=$(expr $1 + $2)  
        printf "Функция вычисления суммы: %s + %s = %s\n" $1 $2 $res  
    else  
        echo "Ошибка! В функцию переданы не все аргументы!"  
    fi  
}
```

Функция вычисления разности двух чисел

```
minus_func() {  
    if [ $# -gt 1 ]  
    then  
        res=$(expr $1 - $2)  
        printf "Функция вычисления разности: %s - %s = %s\n" $1 $2 $res  
    else  
        echo "Ошибка! В функцию переданы не все аргументы!"  
    fi  
}
```

Функция вычисления произведения двух чисел

```
mult_func() {  
    if [ $# -gt 1 ]  
    then  
        res=$(expr $1 \* $2)  
        printf "Функция вычисления произведения: %s * %s = %s\n" $1 $2 $res  
    else  
        echo "Ошибка! В функцию переданы не все аргументы!"  
    fi  
}
```

```

# Функция нахождения частного двух чисел
div_func() {
    if [ $# -gt 1 ]
    then
        res=$(expr $1 / $2)
        printf "Функция вычисления деления: %s / %s = %s\n" $1 $2 $res
    else
        echo "Ошибка! В функцию переданы не все аргументы!"
    fi
}

```

```

echo "=====МЕНЮ===== "
echo "1. Сумма двух чисел"
echo "2. Разность двух чисел"
echo "3. Произведение двух чисел"
echo "4. Деление двух чисел"
printf "Введите номер желаемой команды:"
read numFunc # Считываем номер введенной команды
echo # Выводим пустую строку

```

```

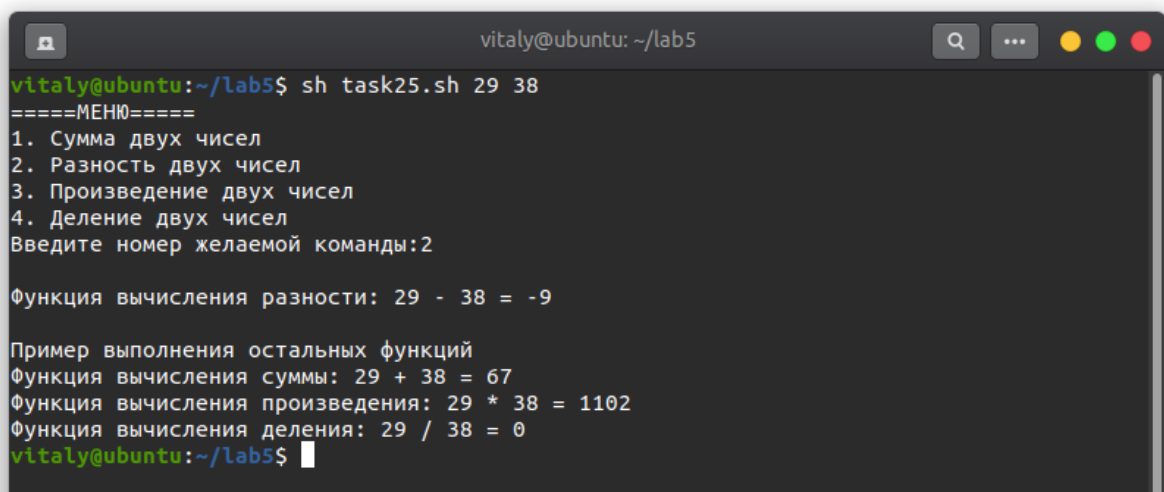
case "$numFunc" in
    "1") sum_func $1 $2 # Если введена 1
        echo "\nПример выполнения остальных функций"
        minus_func $1 $2
        mult_func $1 $2
        div_func $1 $2;;
    "2") minus_func $1 $2 # Если введена 2
        echo "\nПример выполнения остальных функций"
        sum_func $1 $2
        mult_func $1 $2
        div_func $1 $2;;
    "3") mult_func $1 $2 # Если введена 3
        echo "\nПример выполнения остальных функций"
        sum_func $1 $2

```

```

minus_func $1 $2
div_func $1 $2;;
"4") div_func $1 $2 # Если введена 4
echo "\nПример выполнения остальных функций"
sum_func $1 $2
minus_func $1 $2
mult_func $1 $2;;
*) echo "Указан неправильный пункт";; # В любом другом случае
esac

```



```

vitaly@ubuntu: ~/lab5
vitaly@ubuntu:~/lab5$ sh task25.sh 29 38
=====МЕНЮ=====
1. Сумма двух чисел
2. Разность двух чисел
3. Произведение двух чисел
4. Деление двух чисел
Введите номер желаемой команды:2

Функция вычисления разности: 29 - 38 = -9

Пример выполнения остальных функций
Функция вычисления суммы: 29 + 38 = 67
Функция вычисления произведения: 29 * 38 = 1102
Функция вычисления деления: 29 / 38 = 0
vitaly@ubuntu:~/lab5$

```

Рисунок 25 – Пример выполнения скрипта использующий собственные функции

Вывод

В ходе выполнения данной лабораторной работы я изучил основные возможности языка программирования Shell с целью автоматизации процесса администрирования системы за счет написания и использования командных файлов.