

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

Студент

Фетисов В. Д.

Группа ПИ-19

Руководитель

Кургасов В. В.

К.П.Н.

Липецк 2021

Оглавление

Цель работы	2
Задание кафедры	3
Ход работы.....	6
Вывод	25

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.

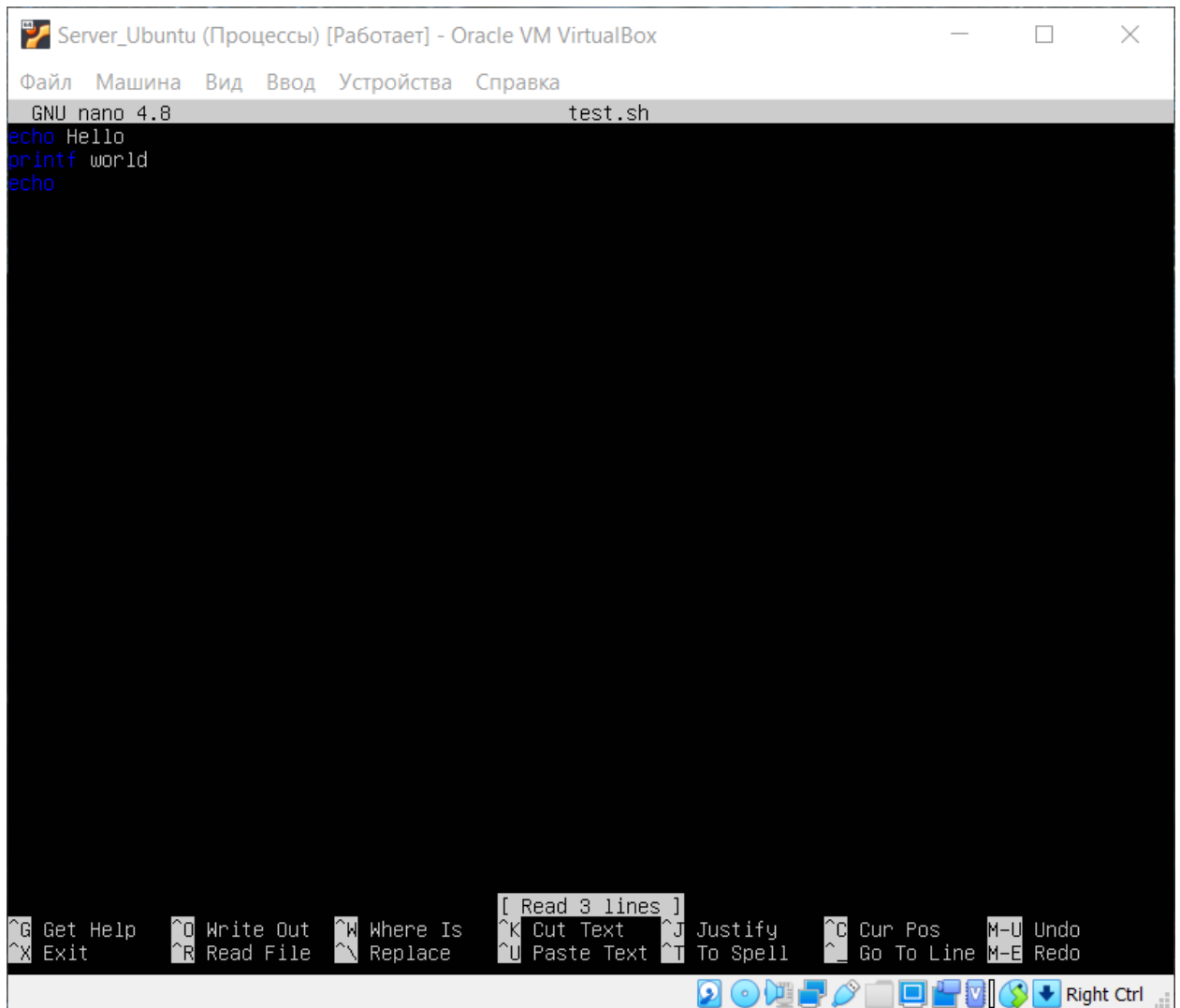
21. Анализируются атрибуты файла. Если первый файл существует и используется для чтения, а второй файл существует и используется для записи, то содержимое первого файла перенаправляется во второй файл. В случае несовпадений указанных атрибутов или отсутствия файлов на экран выдаются соответствующие сообщения (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 test.sh` и сделаем его исполнимым с помощью `chmod +x test.sh`, в этом файле и будем

записывать сценарии для выполнения заданий.

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



```
Server_Ubuntu (Процессы) [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
GNU nano 4.8                                     test.sh
echo Hello
printf world
echo
```

The screenshot shows a terminal window titled "Server_Ubuntu (Процессы) [Работает] - Oracle VM VirtualBox". Inside the terminal, the nano text editor is open, editing a file named "test.sh". The editor's status bar at the top indicates "GNU nano 4.8" and the filename "test.sh". The content of the file is visible as three lines: "echo Hello", "printf world", and "echo". The bottom of the terminal displays a comprehensive list of nano editor shortcuts, including "Get Help", "Write Out", "Where Is", "Cut Text", "Paste Text", "Justify", "Cur Pos", "Go To Line", "Undo", "Exit", "Read File", "Replace", "To Spell", and "Redo". A status bar at the very bottom of the window shows various system icons and a "Right Ctrl" indicator.

Рисунок 1.1 – использование команды `echo` и `printf`

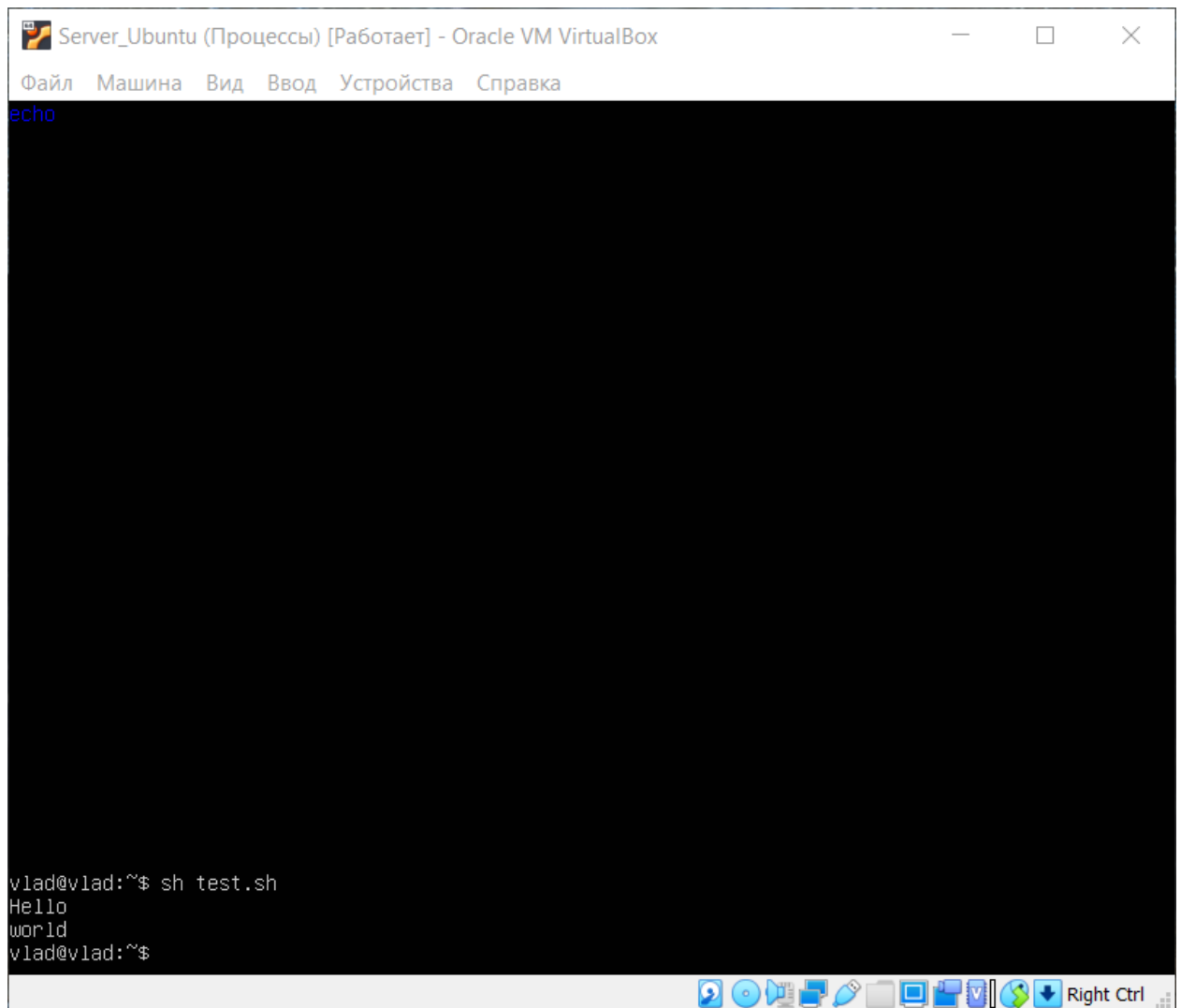


Рисунок 1.2 – использование команды echo и printf

2. Присвоить переменной A целочисленное значение. Просмотреть значение переменной A.
3. Присвоить переменной B значение переменной A. Просмотреть значение переменной B.


```
GNU nano 4.8                                test.sh
A=25
echo $A
B=$A
echo $B
echo
```

^G Get Help	^O Write Out	^W Where Is	[Read 5 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 2.1-Задание 2 и 3, скрипт

```
echo $B  
echo
```

```
vlad@vlad:~$ sh test.sh  
25  
25  
vlad@vlad:~$ _
```

Рисунок 2.2-Задание 2 и 3, вывод

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

```
GNU nano 4.8                                test.sh
C=~vlad
cd $C
```

[Read 2 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 ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo

Рисунок 3.1-Задание 4, скрипт

```
vlad@vlad:~$ cd /
vlad@vlad:/$ ls
bin  cdrom  etc  lib  lib64  lost+found  mnt  proc  run  snap  swap.img  tmp  var
boot  dev  home  lib32  libx32  media  opt  root  sbin  srv  sys  usr
vlad@vlad:/$ . /home/vlad/test.sh
vlad@vlad:~$ ls
loop  loop2  test.sh
vlad@vlad:~$ _
```

Рисунок 3.2-Задание 4, вывод

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

```
GNU nano 4.8                                test.sh
D=date
$D
```

Рисунок 4.1-Задание 5, скрипт

```
vlad@vlad:~$ . test.sh
Пт 03 дек 2021 20:36:56 UTC
vlad@vlad:~$ _
```

Рисунок 4.2-Задание 5, вывод

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

```
GNU nano 4.8 test.sh
E='cat test.sh'
$E
```

Рисунок 5.1-Задание 6, скрипт

```
vlad@vlad:~$ sh test.sh
E='cat test.sh'
$E
vlad@vlad:~$ _
```

Рисунок 5.2-Задание 6, вывод

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

```
GNU nano 4.8 test.sh
F='sort test.sh'
$F
```

Рисунок 6.1 – Задание 7, скрипт

```
vlad@vlad:~$ sh test.sh
$F
F='sort test.sh'
vlad@vlad:~$ _
```

Рисунок 6.2 – Задание 7, вывод

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

```
GNU nano 4.8                                test.sh
A
read A
echo $A
```

Рисунок 7.1 – Задание 8, скрипт

```
vlad@vlad:~$ sh test.sh
test.sh: 1: A: not found
25
25
vlad@vlad:~$ _
```

Рисунок 7.2 – Задание 8, вывод

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

```
GNU nano 4.8
echo 'Hey, enter you name'
read name
echo 'Hello ' $name ' _ you cool'
```

Рисунок 8.1 - Задание 9, скрипт

```
vlad@vlad:~$ sh test.sh
Hey, enter you name
Vlad
Hello Vlad , you cool
vlad@vlad:~$ _
```

Рисунок 8.2 – Задание 9, вывод

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

```
GNU nano 4.8
echo "enter A"
read A
echo "enter B"
read B

echo expr
echo "A+B=" $(expr $A + $B)
echo "A-B=" $(expr $A - $B)
echo "A*B=" $(expr $A \* $B)
echo "A/B=" $(expr $A / $B)

echo bc
printf "A+B= "
echo "$A + $B" | bc
printf "A-B= "
echo "$A - $B" | bc
printf "A*B= "
echo "$A * $B" | bc
printf "A/B= "
echo "$A / $B" | bc
```

Рисунок 9.1 - Задание 10, скрипт

```
vlad@vlad:~$ sh test.sh
enter A
10
enter B
2
expr
A+B= 12
A-B= 8
A*B= 20
A/B= 5
bc
A+B= 12
A-B= 8
A*B= 20
A/B= 5
vlad@vlad:~$
```

Рисунок 9.2 – Задание 10, вывод

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

```
GNU nano 4.8
echo "enter h"
read H
echo "enter r"
read R
printf "pi*r^2*h= "
echo "3.14 * $R * $R * $H" |bc
```

Рисунок 10.1 – Задание 11, скрипт

```
vlad@vlad:~$ sh test.sh
enter h
2
enter r
2
pi*r^2*h= 25.12
vlad@vlad:~$ _
```

Рисунок 10.2 – Задание 11, вывод

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

\$* – все аргументы;

\$@ – все аргументы;

\$# – количество аргументов;

\$0 – имя скрипта;

\$\$ – PID процесса;

#! – PID последнего процесса в *background-e*;

\$? – результат выполнения выражения или скрипта (0 – если успешно, 1 – если ошибка);

\$_ – последний аргумент.

```
GNU nano 4.8
echo "Program name is $0"
echo "Kolvo arg $#"
```

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

Рисунок 11.1 – Задание 12, скрипт

```
vlad@vlad:~$ sh test.sh 1 2 3 4 5
Program name is test.sh
Kolvo arg 5
1
2
3
4
5
vlad@vlad:~$
```

Рисунок 11.2 – Задание 12, вывод

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

```
Файл  Машина  Вид  DE
GNU nano 4.8
echo $(cat $1)
sleep 5
clear
```

Рисунок 12.1 – Задание 13, скрипт

```
Файл  Машина  Вид  Ввод  Устройства
vlad@vlad:~$ sh test.sh test.sh
echo $(cat $1) sleep 5 clear
_
```

Рисунок 12.2 – Задание 13, вывод

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:


```

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

```

Рисунок 13.1 – Задание 14, скрипт

```

vlad@vlad:~$ sh test.sh
test.sh: 3: [-f: not found
test.sh: 3: [-f: not found
test.sh: 3: [-f: not found
vlad@vlad:~$

```

Рисунок 13.2 – Задание 14, вывод

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`.

```

echo vvedite chislo
read C
if [ $C -eq 1 ]
then echo "chislo ravno 1"
else
echo "chislo ne ravno 1"
fi

```

Рисунок 14.1 – Задание 15, скрипт

```

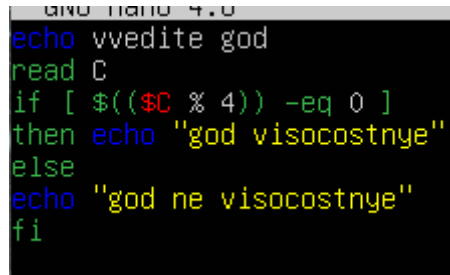
vlad@vlad:~$ sh test.sh
vvedite chislo
1
chislo ravno 1
vlad@vlad:~$ sh test.sh
vvedite chislo
2
chislo ne ravno 1
vlad@vlad:~$

```

Рисунок 14.2 – Задание 15, вывод

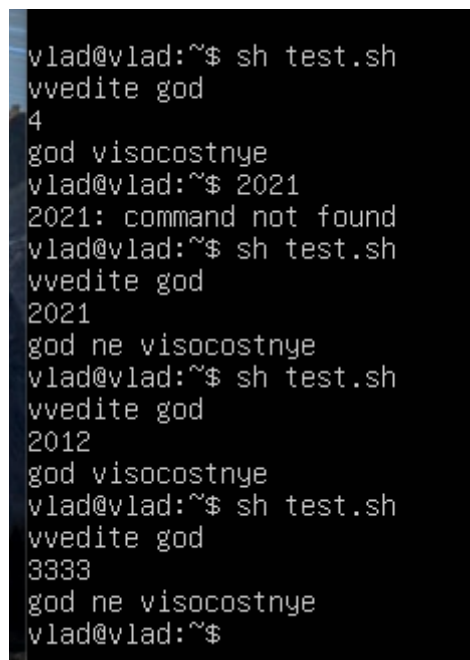
16. Программой запрашивается год, определяется, високосный ли он. Результат выдается на экран.

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



```
GNU nano 4.0
echo vvedite god
read C
if [ $((C % 4)) -eq 0 ]
then echo "god visocostnye"
else
echo "god ne visocostnye"
fi
```

Рисунок 15.1 – Задание 16, скрипт



```
vlad@vlad:~$ sh test.sh
vvedite god
4
god visocostnye
vlad@vlad:~$ 2021
2021: command not found
vlad@vlad:~$ sh test.sh
vvedite god
2021
god ne visocostnye
vlad@vlad:~$ sh test.sh
vvedite god
2012
god visocostnye
vlad@vlad:~$ sh test.sh
vvedite god
3333
god ne visocostnye
vlad@vlad:~$
```

Рисунок 15.2 – Задание 16, вывод

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

```

echo vvedite chislo 1
read A
echo vvedite chislo 2
read B
echo vvedite 1 chislo v diapason
read L
echo vvedite 2 chislo v diapason
read R
if [ $A -gt $L ] || [ $B -gt $L ]
then
while [ $A -lt $R ] || [ $B -lt $R ]
do
A=$(expr $A + 1)
B=$(expr $B + 1)
done
fi
echo $A
echo $B

```

Рисунок 16.1 – Задание 17, скрипт

```

vlad@vlad:~$ sh test.sh
vvedite chislo 1
3
vvedite chislo 2
3
vvedite 1 chislo v diapason
1
vvedite 2 chislo v diapason
5
5
5
vlad@vlad:~$

```

Рисунок 16.2 – Задание 17, вывод

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:

```

read p
if [ $p = "0000" ]
then ls -la /etc | less
else echo "Neverny pwd"
fi

```

Рисунок 17.1 – Задание 18, скрипт

```
total 820
drwxr-xr-x 97 root root      4096 дек 10 19:07 .
drwxr-xr-x 20 root root      4096 окт 16 07:31 ..
-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 22 13:33 apport
drwxr-xr-x  7 root root      4096 окт 16 07:28 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 22 13:33 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 окт 16 07:33 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 окт 16 07:31 cloud
drwxr-xr-x  2 root root      4096 окт 16 07:34 console-setup
drwxr-xr-x  2 root root      4096 авг 24 08:47 cron.d
drwxr-xr-x  2 root root      4096 ноя 22 13:33 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 22 13:33 default
-rw-r--r--  1 root root      604 сен 15 2018 deluser.conf
:
```

Рисунок 17.2 – Задание 18, вывод

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`:

```
if [ -e $1 ]
then
cat $1
else
echo "Not found"
fi
```

Рисунок 18.1 – Задание 19, скрипт

```
vlad@vlad:~$ sh test.sh test.sh
if [ -e $1 ]
then
cat $1
else
echo "Not found"
fi
vlad@vlad:~$ _
```

Рисунок 18.2 – Задание 19, вывод

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:

```
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
```

Рисунок 19.1 – Задание 20, скрипт

```

vlad@vlad:~$ ls
loop loop2 test.sh
vlad@vlad:~$ sh test.sh test.sh
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 $1
fi
vlad@vlad:~$ _

```

Рисунок 19.2 – Задание 20, вывод

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 `-w`. Сначала реализуем задание, спросив названия файлов у пользователя:

```

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

```

Рисунок 20.1 – Задание 21, скрипт

```
vlad@vlad:~$ sh test.sh
test.sh
loop
done
vlad@vlad:~$ sh test.sh test.sh
```

Рисунок 20.2 – Задание 21, вывод

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`):

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

Рисунок 21.1 – Задание 22, скрипт

Затем создадим новый файл сценария, который нам придётся запустить:

```
vlad@vlad:~$ sh test.sh script
not exist
vlad@vlad:~$
```

Рисунок 21.2 – Задание 22, вывод

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`:

```
if [ -s $1 ]
then sort -k1 $1 > $2
cat $2
else echo "file 1 empty"
fi
```

Рисунок 22.1 – Задание 23, скрипт

```
vlad@vlad:~$ sh test.sh test.sh script
cat $2
else echo "file 1 empty"
fi
if [ -s $1 ]
then sort -k1 $1 > $2
vlad@vlad:~$
```

Рисунок 22.2 – Задание 23, вывод

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`:

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

Рисунок 23.1 – Задание 24, скрипт


```

vlad@vlad:~$ sh test.sh
find: warning: you have specified the global option -maxdepth after the argument -type, but global o
ptions are not positional, i.e., -maxdepth affects tests specified before it as well as those specif
ied 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)? ls -l
not overwritten
vlad@vlad:~$ sh test.sh
find: warning: you have specified the global option -maxdepth after the argument -type, but global o
ptions are not positional, i.e., -maxdepth affects tests specified before it as well as those specif
ied after it. Please specify global options before other arguments.
tar: ./test.tar: file is the archive; not dumped
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
vlad@vlad:~$ ls -l
total 24
-rw-rw-r-- 1 vlad vlad 242 дек 10 21:23 loop
-rw-rw-r-- 1 vlad vlad 40 окт 23 18:57 loop2
-rw-rw-r-- 1 vlad vlad 71 дек 10 21:33 script
-rwxrwxr-x 1 vlad vlad 105 дек 10 21:39 test.sh
-rw-rw-r-- 1 vlad vlad 7371 дек 10 21:40 test.tar.gz
vlad@vlad:~$

```

Рисунок 23.2 – Задание 24, вывод

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

```

#!/bin/bash
echo "Vvedite stepen"
read a
echo "Vvedite chislo"
read b
c=$b
pow() {
while [ $a -ne 1 ]
do
c=$(expr $c \* $b)
a=$(expr $a - 1)
echo $c
done
}
pow

```

Рисунок 24.1 – Задание 25, скрипт

```

vlad@vlad:~$ sh test.sh
Vvedite stepen
3
Vvedite chislo
3
9
27
vlad@vlad:~$

```

Рисунок 24.2 – Задание 25, вывод

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.