

# Липецкий государственный технический университет

Кафедра прикладной математики

## Отчет по лабораторной работе №5 «Программирование в ОС семейства Linux.»

Студент

\_\_\_\_\_  
подпись, дата

Стукановский А.О.  
фамилия, инициалы

Группа

ПМ-18

Руководитель

доц., к.п.н. кафедры АСУ  
ученая степень, ученое звание

\_\_\_\_\_  
подпись, дата

Кургасов В. В.  
фамилия, инициалы

Липецк 2020 г.

# Содержание

Цель работы	4
Практическое задание	4
Выполнение практического задания.	6
1) Используя команды ECHO, PRINTF вывести информационные сообщения на экран. . . . .	6
2) Присвоить переменной A целочисленное значение. Просмотреть значение переменной A. . .	6
3) Присвоить переменной B значение переменной A. Просмотреть значение переменной B. . .	6
4) Присвоить переменной C значение “путь до своего каталога”. Перейти в этот каталог с использованием переменной. . . . .	6
5) Присвоить переменной D значение “имя команды”, а именно команды DATE. Выполнить эту команду, используя значение переменной. . . . .	7
6) Присвоить переменной E значение “имя команды”, а именно команды просмотра содержимого файла, просмотреть содержимое переменной. Выполнить эту команду, используя значение переменной. . . . .	7
7) Присвоить переменной F значение “имя команды”, а именно сортировки содержимого текстового файла. Выполнить эту команду, используя значение переменной. . . . .	7
8) Программа запрашивает значение переменной, а затем выводит значение этой переменной.	8
9) Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введенной переменной. . . . .	8
10) Программа запрашивает значения двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на экран (использовать команды а) EXPR; б) BC). . . . .	8
11) Вычислить объем цилиндра. Исходные данные запрашиваются программой. Результат выводится на экран. . . . .	9
12) Используя позиционные параметры, отобразить имя программы, количество аргументов командной строки, значение каждого аргумента командной строки. . . . .	9
13) Используя позиционный параметр, отобразить содержимое текстового файла, указанного в качестве аргумента командной строки. После паузы экран очищается. . . . .	10
14) Используя оператор FOR, отобразить содержимое текстовых файлов текущего каталога поэкранно. . . . .	10
15) Программой запрашивается ввод числа, значение которого затем сравнивается с допустимым значением. В результате этого сравнения на экран выдаются соответствующие сообщения. . . . .	12
16) Программой запрашивается год, определяется, високосный ли он. Результат выдается на экран. . . . .	13
17) Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значения инкрементируются.	13
18) В качестве аргумента командной строки указывается пароль. Если пароль введен верно, постранично отображается в длинном формате с указанием скрытых файлов содержимое каталога /etc. . . . .	14
19) Проверить, существует ли файл. Если да, выводится на экран его содержимое, если нет - выдается соответствующее сообщение. . . . .	15
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. . . . .	15

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). . . . .	15
22) Если файл запуска программы найден, программа запускается (по выбору). . . . .	16
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. . . . .	17
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. . . . .	17
25) Написать скрипт с использованием функции, например, функции, суммирующей значения двух переменных. . . . .	18
<b>Вывод</b>	<b>19</b>
<b>Список литературы</b>	<b>20</b>

## Цель работы

Изучить основные возможности языка программирования Shell с целью автоматизации процесса администрирования системы за счет написания и использования командных файлов.

Написать скрипты, при запуске которых выполняются следующие действия:

## Практическое задание

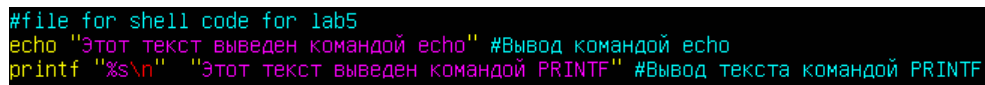
- 1) Используя команды ECHO, PRINTF, вывести информационные сообщения на экран.
- 2) Присвоить переменной A целочисленное значение. Просмотреть значение переменной A.
- 3) Присвоить переменной B значение переменной A. Просмотреть значение переменной B.
- 4) Присвоить переменной C значение “путь до своего 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.
- 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.

## Выполнение практического задания.

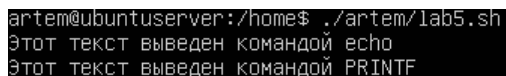
Используя команды **ECHO**, **PRINTF** вывести информационные сообщения на экран.

Код скрипта изображён на рисунке 1. Пример работы скрипта можно увидеть на рисунке 2.



```
#file for shell code for lab5
echo "Этот текст выведен командой echo" #Вывод командой echo
printf "%s\\n" "Этот текст выведен командой PRINTF" #Вывод текста командой PRINTF
```

Рисунок 1.



```
artem@ubuntu-server:~/home$ ./artem/lab5.sh
Этот текст выведен командой echo
Этот текст выведен командой PRINTF
```

Рисунок 2.

Присвоить переменной **A** целочисленное значение. Просмотреть значение переменной


**A.**

Код скрипта изображён на рисунке 3. Пример работы скрипта можно увидеть на рисунке 4.



```
A=101 #Присваиваю переменной A значение 101
printf "Значение A = %d\\n" $A #Вывожу на экран значение A
```

Рисунок 3.



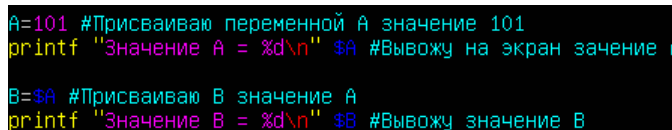
```
Значение A = 101
```

Рисунок 4.

Присвоить переменной **B** значение переменной **A**. Просмотреть значение переменной

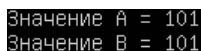
**B.**

Код скрипта изображён на рисунке 5. Пример работы скрипта можно увидеть на рисунке 6.



```
A=101 #Присваиваю переменной A значение 101
printf "Значение A = %d\\n" $A #Вывожу на экран значение A
B=$A #Присваиваю B значение A
printf "Значение B = %d\\n" $B #Вывожу значение B
```

Рисунок 5.



```
Значение A = 101
Значение B = 101
```

Рисунок 6.

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

Код скрипта изображён на рисунке 7. Пример работы скрипта можно увидеть на рисунке 8.

```
C=$HOME #Присваиваем C "путь до своего каталога"
echo Директория до перехода: $PWD
cd $C #Переход в свой каталог
echo Директория после перехода: $PWD
```

Рисунок 7.

```
Директория до перехода: /home
Директория после перехода: /home/artem
```

Рисунок 8.

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

Код скрипта изображён на рисунке 9. Пример работы скрипта можно увидеть на рисунке 10.

```
D="date" #Присваиваем переменной имя команды date
$D # Исполняем команду, записанную в D
```

Рисунок 9.

```
Fri Nov 27 17:41:45 UTC 2020
```

Рисунок 10.

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

Код скрипта изображён на рисунке 11. Пример работы скрипта можно увидеть на рисунке 12.

```
E="less text.txt" #Присваиваем переменной команду просмотра файла text.txt
echo $E #Выводим содержимое переменной E
$E #Исполняем команду переменной E
```

Рисунок 11.

```
less text.txt
UID      PID    PPID  C  STIME TTY          TIME CMD
artem    930      1    0 13:34 ?        00:00:00 /lib/systemd/systemd --user
artem    932      930    0 13:34 ?        00:00:00 (sd-pam)
artem    943      657    0 13:34 tty1      00:00:00 -bash
artem   2819      943    0 14:46 tty1      00:00:00 -bash
artem   2821     2819    0 14:46 tty1      00:00:00 ps -f -u artem
```

Рисунок 12.

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

Код скрипта изображён на рисунке 13. Пример работы скрипта можно увидеть на рисунке 14.

```
F="sort -b text.txt" #Присваиваем переменной команду сортировки файла text.txt
$F #Исполнение команды переменной F
~
~
~
```

Рисунок 13.

UID	PID	PPID	C	STIME	TTY	TIME	CMD
artem	930	1	0	13:34	?	00:00:00	/lib/systemd/systemd --user
artem	932	930	0	13:34	?	00:00:00	(sd-pam)
artem	943	657	0	13:34	tty1	00:00:00	-bash
artem	2819	943	0	14:46	tty1	00:00:00	-bash
artem	2821	2819	0	14:46	tty1	00:00:00	ps -f -u artem

```
artem@ubuntuserver:/home$ _
```

Рисунок 14.

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

Код скрипта изображён на рисунке 15. Пример работы скрипта можно увидеть на рисунке 16.

```
printf "%s: " "Введите переменную"
read a
printf "Введённая переменная: %d\n" $a
```

Рисунок 15.

```
artem@ubuntuserver:~$ ./lab5.v2.sh
Введите переменную: 1470
Введённая переменная: 1470
artem@ubuntuserver:~$ _
```

Рисунок 16.

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

Код скрипта изображён на рисунке 17. Пример работы скрипта можно увидеть на рисунке 18.

```
printf "%s\n" "Как Вас зовут?"
read a
printf "Привет, %s\n" $a
```

Рисунок 17.

```
artem@ubuntuserver:~$ ./lab5.v2.sh
Как Вас зовут?
Артём
Привет, Артём!
artem@ubuntuserver:~$ _
```

Рисунок 18.

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

Код скрипта изображён на рисунке 19. Пример работы скрипта можно увидеть на рисунке 20.



```

printf "%s: " "Введите переменную A"
read A
printf "%s: " "Введите переменную B"
read B
printf "A + B = %d + %d = %d\n" $A $B $(expr $A + $B)
printf "A - B = %d - %d = " $A $B
echo $A - $B | bc
printf "A * B = %d * %d = " $A $B
echo $A '*' $B | bc
printf "A / B = %d / %d = %d\n" $A $B $(expr $A / $B)

```

Рисунок 19.

```

artem@ubuntu:~$ ./lab5.v2.sh
Введите переменную A: 8
Введите переменную B: 4
A + B = 8 + 4 = 12
A - B = 8 - 4 = 4
A * B = 8 * 4 = 32
A / B = 8 / 4 = 2
artem@ubuntu:~$

```

Рисунок 20.

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

Код скрипта изображён на рисунке 21. Пример работы скрипта можно увидеть на рисунке 22.

```

printf "%s: " "Введите высоту цилиндра"
read H
printf "%s: " "Введите радиус цилиндра"
read R
PI=3.1415926535
printf "Объём цилиндра V = пир^2*H = "
echo $PI \* $R \* $R \* $H | bc

```

Рисунок 21.

```

artem@ubuntu:~$ ./lab5.v2.sh
Введите высоту цилиндра: 14
Введите радиус цилиндра: 70
Объём цилиндра V = пир^2*H = 215513.2560301000
artem@ubuntu:~$ _

```

Рисунок 22.

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

Код скрипта изображён на рисунке 23. Пример работы скрипта можно увидеть на рисунке 24.

```

printf "Имя скрипта: $0 \n"
printf "Количество позиционных параметров: $#\n"
list="$@"
a=1
printf "Список позиционных аргументов:\n"
for i in $list; do
    printf "%d) $i:\n" $a
    a=$(expr $a + 1)
done

```

Рисунок 23.

```

artem@ubuntu:~$ ./lab5.v2.sh 12 13 5 7
Имя скрипта: ./lab5.v2.sh
Количество позиционных параметров: 4
Список позиционных аргументов:
1) 12;
2) 13;
3) 5;
4) 7;
artem@ubuntu:~$ _

```

Рисунок 24.

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

Код скрипта изображён на рисунке 25. Пример работы скрипта можно увидеть на рисунках 26 и 27.

```

less $1
clear
~
~
~

```

Рисунок 25.

```

artem@ubuntu:~$ ./lab5.v2.sh text.txt
UID      PID    PPID  C  STIME TTY          TIME CMD
artem    930     1    0  13:34 ?        00:00:00 /lib/systemd/systemd --user
artem    932     930    0  13:34 ?        00:00:00 (sd-pam)
artem    943     657    0  13:34 tty1    00:00:00 -bash
artem   2819     943    0  14:46 tty1    00:00:00 -bash
artem   2821    2819    0  14:46 tty1    00:00:00 ps -f -u artem
text.txt (END)

```

Рисунок 26.

```

artem@ubuntu:~$ _

```

Рисунок 27.

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

Код скрипта изображён на рисунке 28. Пример работы скрипта можно увидеть на рисунках с 29 по 33.

```

list=$(ls -l $PWD | awk '{print $9}' | grep txt)
for k in $list
do
    less $k
done

```

Рисунок 28.

```

artem@ubuntu:~$ ./lab5.v2.sh
-l
1.txt (END)

```

Рисунок 29.

```

artem@ubuntuserver:~$ ./lab5.v2.sh
-1
  PID TTY          TIME CMD
  915 tty1        00:00:00 bash
 1060 tty1        00:00:00 top
 1313 tty1        00:00:00 ps
2.txt (END)

```

Рисунок 30.

```

artem@ubuntuserver:~$ ./lab5.v2.sh
-1
  PID TTY          TIME CMD
  915 tty1        00:00:00 bash
 1060 tty1        00:00:00 top
 1313 tty1        00:00:00 ps
F S  UID      PID   PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000      915    648   0  80   0 - 1768 do_wai tty1        00:00:00 bash
0 T  1000     1060    915   0  80   0 - 1979 do_sig tty1        00:00:00 top
0 R  1000     1285    915   0  80   0 - 1888 -      tty1        00:00:00 ps
pvv.txt (END)

```

Рисунок 31.

```

artem@ubuntuserver:~$ ./lab5.v2.sh
-1
  PID TTY          TIME CMD
  915 tty1        00:00:00 bash
 1060 tty1        00:00:00 top
 1313 tty1        00:00:00 ps
F S  UID      PID   PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000      915    648   0  80   0 - 1768 do_wai tty1        00:00:00 bash
0 T  1000     1060    915   0  80   0 - 1979 do_sig tty1        00:00:00 top
0 R  1000     1285    915   0  80   0 - 1888 -      tty1        00:00:00 ps
ps -f -u username
UID      PID   PPID  C STIME TTY          TIME CMD
artem      930     1    0 13:34 ?           00:00:00 /lib/systemd/systemd --user
artem      932     930   0 13:34 ?           00:00:00 (sd-pam)
artem      943     657   0 13:34 tty1        00:00:00 -bash
artem     2819     943   0 14:46 tty1        00:00:00 -bash
artem     2821    2819   0 14:46 tty1        00:00:00 ps -f -u artem
text.txt (END)

```

Рисунок 32.

```

PID TTY          TIME CMD
 1 ?            00:00:01 systemd
 2 ?            00:00:00 kthreadd
 3 ?            00:00:00 rcu_gp
 4 ?            00:00:00 rcu_par_gp
 6 ?            00:00:00 kworker/0:0H-kblockd
 8 ?            00:00:00 kworker/u2:0-events_power_efficient
 9 ?            00:00:00 mm_percpu_wq
10 ?            00:00:00 ksoftirqd/0
11 ?            00:00:00 rcu_sched
12 ?            00:00:00 migration/0
13 ?            00:00:00 idle_inject/0
14 ?            00:00:00 cpuhp/0
15 ?            00:00:00 kdevtmpfs
16 ?            00:00:00 netns
17 ?            00:00:00 rcu_tasks_kthre
18 ?            00:00:00 kauditd
19 ?            00:00:00 khungtaskd
20 ?            00:00:00 oom_reaper
21 ?            00:00:00 writeback
22 ?            00:00:00 kcompactd0
23 ?            00:00:00 ksmd
24 ?            00:00:00 khugepaged
70 ?            00:00:00 kintegrityd
71 ?            00:00:00 kblockd
72 ?            00:00:00 blkcg_punt_bio
73 ?            00:00:00 tpm_dev_wq
74 ?            00:00:00 ata_sff
75 ?            00:00:00 md
76 ?            00:00:00 edac-poller
77 ?            00:00:00 devfreq_wq
78 ?            00:00:00 watchdogd
81 ?            00:00:00 kswapd0
82 ?            00:00:00 ecryptfs-kthrea
84 ?            00:00:00 kthrotld
85 ?            00:00:00 acpi_thermal_pm
txt2.txt

```

Рисунок 33.

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

Код скрипта изображён на рисунке 34. Пример работы скрипта можно увидеть на рисунке 35.

```

printf "Введите число для сравнения: "
read a
b=1470
if test $a -gt $b
then
    echo Введённое число больше допустимого
else
    if test $a -eq $b
    then
        echo Введённое число равно допустимому
    else
        echo Введённое число меньше допустимого
    fi
fi

```

Рисунок 34.

```

artem@ubuntuserver:~$ ./lab5.v2.sh
Введите число для сравнения: 5
Введённое число меньше допустимого
artem@ubuntuserver:~$ ./lab5.v2.sh
Введите число для сравнения: 2000
Введённое число больше допустимого
artem@ubuntuserver:~$ ./lab5.v2.sh
Введите число для сравнения: 1470
Введённое число равно допустимому
artem@ubuntuserver:~$

```

Рисунок 35.

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

Код скрипта изображён на рисунке 36. Пример работы скрипта можно увидеть на рисунке 37.

```

printf "Введите год: "
read year
if test $(expr $year % 4) -eq 0 -a $(expr $year % 100) -ne 0 -o $(expr $year % 400) -eq 0
then
    echo Високосный
else
    echo Невисокосный
fi

```

Рисунок 36.

```

artem@ubuntuserver:~$ ./lab5.v2.sh
Введите год: 2000
Високосный
artem@ubuntuserver:~$ ./lab5.v2.sh
Введите год: 2003
Невисокосный
artem@ubuntuserver:~$ ./lab5.v2.sh
Введите год: 1900
Невисокосный
artem@ubuntuserver:~$

```

Рисунок 37.

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

Код скрипта изображён на рисунке 38. Пример работы скрипта можно увидеть на рисунке 39.

```

printf "Введите первое значение: "
read x1
printf "Введите второе значение: "
read x2
printf "Введите левую границу диапазона: "
read a
printf "Введите правую границу диапазона: "
read b
while test $x1 -ge $a -a $x1 -le $b -o $x2 -ge $a -a $x2 -le $b
do
    if test $x1 -ge $a -a $x1 -le $b
    then x1=$(expr $x1 + 1)
    fi
    if test $x2 -ge $a -a $x2 -le $b
    then x2=$(expr $x2 + 1)
    fi
done
printf "первое значение равно: %d\n" $x1
printf "второе значение равно: %d\n" $x2

```

Рисунок 38.

```

artem@ubuntuuserver:~$ ./lab5.v2.sh
Введите первое значение: 5
Введите второе значение: 15
Введите левую границу диапазона: 14
Введите правую границу диапазона: 70
первое значение равно: 5
второе значение равно: 71
artem@ubuntuuserver:~$

```

Рисунок 39.

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

Код скрипта изображён на рисунке 40. Пример работы скрипта можно увидеть на рисунках 41 и 42.

```

pass=1470
if test $1 -eq $pass
then ls -al /etc | less
fi

```

Рисунок 40.

```

artem@ubuntuuserver:~$ ./lab5.v2.sh 147
artem@ubuntuuserver:~$ ./lab5.v2.sh 1470

```

Рисунок 41.

```

-rw-r--r-- 1 root root      45 Jan 26  2020 bash_completion
drwxr-xr-x 2 root root    4096 Oct  3  07:30 bash_completion.d
-rw-r--r-- 1 root root    367 Apr 14  2020 bindresvport.blacklist
drwxr-xr-x 2 root root    4096 Apr 22  2020 binfmt.d
drwxr-xr-x 2 root root    4096 Jul 31 16:29 byobu
drwxr-xr-x 3 root root    4096 Jul 31 16:28 ca-certificates
-rw-r--r-- 1 root root   6505 Oct 31 06:21 ca-certificates.conf
-rw-r--r-- 1 root root   5714 Jul 31 16:29 ca-certificates.conf.dpkg-old
drwxr-xr-x 2 root root    4096 Jul 31 16:29 calendar
drwxr-xr-x 4 root root    4096 Oct  3  07:25 cloud
drwxr-xr-x 2 root root    4096 Oct  3  07:31 console-setup
drwxr-xr-x 2 root root    4096 Jul 31 16:29 cron.d
drwxr-xr-x 2 root root    4096 Oct 16 16:53 cron.daily
drwxr-xr-x 2 root root    4096 Jul 31 16:28 cron.hourly
drwxr-xr-x 2 root root    4096 Jul 31 16:28 cron.monthly
drwxr-xr-x 2 root root    4096 Jul 31 16:30 cron.weekly
-rw-r--r-- 1 root root   1042 Feb 13  2020 crontab
drwxr-xr-x 2 root root    4096 Oct  3  07:30 cryptsetup-initramfs
-rw-r--r-- 1 root root     54 Jul 31 16:29 crypttab
drwxr-xr-x 4 root root    4096 Jul 31 16:28 dbus-1
drwxr-xr-x 3 root root    4096 Jul 31 16:29 dconf
-rw-r--r-- 1 root root   2969 Aug  3  2019 debconf.conf
-rw-r--r-- 1 root root     13 Dec  5  2019 debian_version
drwxr-xr-x 3 root root    4096 Nov 13 20:40 default
-rw-r--r-- 1 root root    604 Sep 15  2018 deluser.conf
drwxr-xr-x 2 root root    4096 Jul 31 16:28 depmod.d
drwxr-xr-x 4 root root    4096 Jul 31 16:28 dhcp
drwxr-xr-x 4 root root    4096 Jul 31 16:28 dpkg
-rw-r--r-- 1 root root    685 Feb 14  2020 e2scrub.conf
-rw-r--r-- 1 root root     96 Jul 31 16:28 environment
-rw-r--r-- 1 root root   1816 Dec 27  2019 ethertypes
drwxr-xr-x 4 root root    4096 Jul 31 16:29 fonts
-rw-r--r-- 1 root root    630 Oct  3  07:24 fstab
-rw-r--r-- 1 root root    280 Jun 20  2014 fuse.conf
drwxr-xr-x 3 root root    4096 Jul 31 16:30 fwupd
-rw-r--r-- 1 root root   2584 Feb  1  2020 gai.conf
:~

```

Рисунок 42.

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

Код скрипта изображён на рисунке 43. Пример работы скрипта можно увидеть на рисунке 44.

```
p=$PWD/'$1
if test -e $p
then less $p
else
echo Файла не существует
fi
```

Рисунок 43.

```
artem@ubuntuserver:~$ ./lab5.v2.sh text.poema
Файла не существует
artem@ubuntuserver:~$ ./lab5.v2.sh text.txt
UID      PID    PPID  C  STIME TTY      TIME CMD
artem    930     1    0  13:34 ?        00:00:00 /lib/systemd/systemd --user
artem    932     930    0  13:34 ?        00:00:00 (sd-pam)
artem    943     657    0  13:34 tty1    00:00:00 -bash
artem   2819     943    0  14:46 tty1    00:00:00 -bash
artem   2821    2819    0  14:46 tty1    00:00:00 ps -f -u artem
/home/artem/text.txt (END)
```

Рисунок 44.

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.

Код скрипта изображён на рисунке 45. Пример работы скрипта можно увидеть на рисунке 46.

```
if test -e $1
then
    if test -d $1 -a -r $1
    then
        ls $1
    else
        less $1
    fi
else
    mkdir $1
fi
```

Рисунок 45.

```
artem@ubuntuserver:~$ ./lab5.v2.sh /home/artem/test.txt
artem@ubuntuserver:~$ ./lab5.v2.sh /home/artem/text.txt
UID      PID    PPID  C  STIME TTY      TIME CMD
artem    930     1    0  13:34 ?        00:00:00 /lib/systemd/systemd --user
artem    932     930    0  13:34 ?        00:00:00 (sd-pam)
artem    943     657    0  13:34 tty1    00:00:00 -bash
artem   2819     943    0  14:46 tty1    00:00:00 -bash
artem   2821    2819    0  14:46 tty1    00:00:00 ps -f -u artem
artem@ubuntuserver:~$ ls
1.txt  echo  lab5.v2.sh  loop2  pvv.txt  snap  test.txt  txt2.txt
2.txt  lab5.sh  loop       pipe  result.txt  test.sh  text.txt
artem@ubuntuserver:~$ _
```

Рисунок 46.

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).

Код скрипта изображён на рисунке 47. Пример работы скрипта можно увидеть на рисунке 48.

```
if test -e $1 -a -r $1
then
    if test -e $2 -a -w $2
    then
        cat $1 >> $2
        echo Файл 1 записан в файл 2
    else
        if test -e $2
        then
            echo Второй файл недоступен для записи
        else
            echo Второй файл не существует
        fi
    fi
else
    if test -e $1
    then
        echo Первый файл недоступен для чтения
    else
        echo Первого файла не существует
    fi
fi
```

Рисунок 47.

```
artem@ubuntuuser:~$ ./lab5.v2.sh /home/artem/2.txt /home/artem/1.txt
Файл 1 записан в файл 2
artem@ubuntuuser:~$ ./lab5.v2.sh /home/artem/2.txt /home/artem/122222.txt
Второй файл не существует
artem@ubuntuuser:~$ _
```

Рисунок 48.

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

Код скрипта изображён на рисунке 49. Пример работы скрипта можно увидеть на рисунке 50.

```
list=$(ls -l $PWD | awk '{print $9}' | grep sh)
for k in $list
do
    echo Запустить $k "(Y/N)"
    read A
    if test $A = 'Y'
    then
        echo Запуск скрипта
        ./$k
    else
        echo Скрипт $k пропущен
    fi
done
echo Список скриптов закончился
```

Рисунок 49.



```

artem@ubuntuserver:~$ ./lab5.v2.sh
Запустить lab5.sh (Y/N)
Y
Запуск скрипта
Этот текст выведен командой echo
Этот текст выведен командой PRINTF
Значение A = 101
Значение B = 101
Директория до перехода: /home/artem
Директория после перехода: /home/artem
Fri Nov 27 22:05:59 UTC 2020
less text.txt
UID      PID      PPID    C  STIME TTY          TIME CMD
artem    930        1    0  13:34 ?        00:00:00 /lib/systemd/systemd --user
artem    932        930    0  13:34 ?        00:00:00 (sd-pam)
artem    943        657    0  13:34 tty1    00:00:00 -bash
artem   2819        943    0  14:46 tty1    00:00:00 -bash
artem   2821        943    0  14:46 tty1    00:00:00 ps -f -u artem
UID      PID      PPID    C  STIME TTY          TIME CMD
artem    930        1    0  13:34 ?        00:00:00 /lib/systemd/systemd --user
artem    932        930    0  13:34 ?        00:00:00 (sd-pam)
artem    943        657    0  13:34 tty1    00:00:00 -bash
artem   2819        943    0  14:46 tty1    00:00:00 -bash
artem   2821        943    0  14:46 tty1    00:00:00 ps -f -u artem
Запустить lab5.v2.sh (Y/N)
N
Скрипт lab5.v2.sh пропущен
Запустить test.sh (Y/N)
N
Скрипт test.sh пропущен
Список скриптов закончился
artem@ubuntuserver:~$

```

Рисунок 50.

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.

Код скрипта изображён на рисунке 51. Пример работы скрипта можно увидеть на рисунке 52.

```

f=$PWD/'$1
if test -s $f
then
    sort -k1 $f > $PWD/'prom.txt
    less $PWD/'prom.txt
fi

```

Рисунок 51.

```

artem@ubuntuserver:~$ ./lab5.v2.sh text.txt
UID      PID      PPID    C  STIME TTY          TIME CMD
artem    930        1    0  13:34 ?        00:00:00 /lib/systemd/systemd --user
artem    932        930    0  13:34 ?        00:00:00 (sd-pam)
artem    943        657    0  13:34 tty1    00:00:00 -bash
artem   2819        943    0  14:46 tty1    00:00:00 -bash
artem   2821        943    0  14:46 tty1    00:00:00 ps -f -u artem
artem@ubuntuserver:~$ ls
1.txt  echo      lab5.v2.sh  loop2  prom.txt  result.txt  test.sh  text.txt
2.txt  lab5.sh  loop       pipe   pvw.txt   snap        test.txt  txt2.txt
artem@ubuntuserver:~$

```

Рисунок 52.

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.

Код скрипта изображён на рисунке 53. Пример работы скрипта можно увидеть на рисунке 54.

```
list=$(ls -l $PWD | awk '{ print "'$PWD'/"$9}' | grep txt)
tar -cvf my.tar $list

tar -tvf my.tar
echo Архив собран, результат my.tar
gzip my.tar > my.tar.gz
echo Архив сжат, результат my.tar.gz
```

Рисунок 53.

```
artem@ubuntuserver:~$ ./lab5.v2.sh
tar: Removing leading `/' from member names
/home/artem/1.txt
tar: Removing leading `/' from hard link targets
/home/artem/2.txt
/home/artem/prom.txt
/home/artem/pvv.txt
/home/artem/result.txt
/home/artem/test.txt/
/home/artem/text.txt
/home/artem/txt2.txt
-rw-rw-rw- user/artem      126 2020-11-27 21:24 home/artem/1.txt
-rw-rw-rw- artem/artem    123 2020-11-27 21:23 home/artem/2.txt
-rw-rw-r-- artem/artem    380 2020-11-27 21:35 home/artem/prom.txt
-rw-rw-r-- artem/artem    296 2020-11-13 18:36 home/artem/pvv.txt
-rw-rw-r-- artem/artem     16 2020-10-30 17:57 home/artem/result.txt
drwxrwxr-x artem/artem     0 2020-11-27 21:27 home/artem/test.txt/
-rw-rw-r-- artem/artem    318 2020-11-28 00:00 home/artem/text.txt
-rw-rw-r-- artem/artem  20657 2020-11-13 22:09 home/artem/txt2.txt
Архив собран, результат my.tar
gzip: my.tar.gz already exists; do you wish to overwrite (y or n)? y
Архив сжат, результат my.tar.gz
artem@ubuntuserver:~$
```

Рисунок 54.

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

Код скрипта изображён на рисунке 55. Пример работы скрипта можно увидеть на рисунке 56.

```
summator(){
    echo $1 + $2 | bc
}

value=$(summator $1 $2)
echo Сумма этих чисел равна $value
```

Рисунок 55.

```
artem@ubuntuserver:~$ ./lab5.v2.sh 1 2
Сумма этих чисел равна 3
artem@ubuntuserver:~$ _
```

Рисунок 56.

## Вывод

В ходе лабораторной работы были изучены основные возможности языка программирования Shell с целью автоматизации процесса администрирования системы.

## Список литературы

- [1] Львовский, С.М. Набор и верстка в системе  $\text{\LaTeX}$  [Текст] / С.М. Львовский. М.: МЦНМО, 2006. — 448 с.