

Липецкий государственный технический университет
Факультет автоматизации и информатики
Кафедра автоматизированных систем управления

Лабораторная работа №5
по Дисциплине «Операционная система Linux»
на тему «Программирование на SHELL.
Использование командных файлов»

Студент

Глебов Д.А.

Группа АИ-18

Руководитель

Кургасов В.В.

к.п.н.

Липецк 2020 г.

Оглавление

Элементы оглавления не найдены.

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

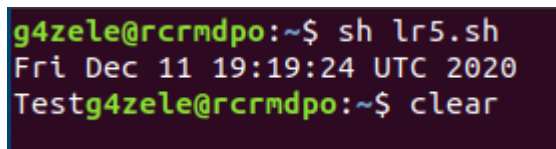
1. Используя команды ECHO, PRINTF вывести информационные сообщения на экран

С помощью команды echo выведем текущую дату на экран, а с помощью команды printf сообщение test printf.



```
echo `date`  
printf "Test"
```

Рисунок 1 - Листинг

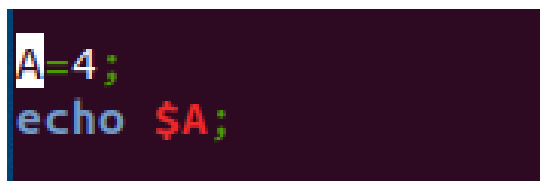


```
g4zele@rcrmdpo:~$ sh lr5.sh  
Fri Dec 11 19:19:24 UTC 2020  
Testg4zele@rcrmdpo:~$ clear
```

Рисунок 2 - Результат работы

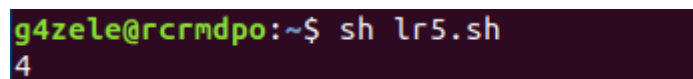
2. Присвоить переменной A целочисленное значение.

Просмотреть значение переменной A В первой строке мы присваиваем переменной A значение 5. С помощью команда echo выводим это значение на экран.



```
A=4;  
echo $A;
```

Рисунок 3 - Листинг



```
g4zele@rcrmdpo:~$ sh lr5.sh  
4
```

Рисунок 4 - Результат работы

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

Аналогично заданию 2 присваиваем переменной A значение 5, затем переменной B присваиваем значение A и выводим значение B на экран.

```
A=3;  
B=$A  
echo $B;
```

Рисунок 5 - Листинг

```
g4zele@rcrmdpo:~$ sh lr5.sh  
3
```

Рисунок 6 - Результат работы

4. Присвоить переменной C значение “путь до своего каталога”.

Перейти в этот каталог с использованием переменной

Переменной C присваиваем путь до каталога используя команду pwd, выводим получившееся значение на экран, затем перейдем в этот каталог.

```
C=`pwd`;  
echo $C;  
cd $C;
```

Рисунок 7 - Листинг

```
g4zele@rcrmdpo:~$ sh lr5.sh  
/home/g4zele  
g4zele@rcrmdpo:~$
```

Рисунок 8 - Результат работы

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

Присваиваем переменной D имя команды date, выводим полученную дату на экран.

```
D=`date`;  
echo $D;
```

Рисунок 9 - Листинг

```
g4zele@rcrmdpo:~$ sh lr5.sh  
Fri Dec 11 19:33:02 UTC 2020  
g4zele@rcrmdpo:~$
```

Рисунок 10 - Результат работы

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

Переменной E присваиваем значение команды для просмотра содержимого файла test. Выводим содержимое файла на экран.

```
E=`cat test`;  
echo $E
```

Рисунок 11 - Листинг

```
g4zele@rcrmdpo:~$ sh lr5.sh  
12345678910  
g4zele@rcrmdpo:~$
```

Рисунок 12 - Результат работы

7. Присвоить переменной F значение “имя команды”, а именно сортировки содержимого текстового файла. Выполнить эту команду, используя значение переменной. Создадим файл, который нам необходимо будет отсортировать.

Затем в скрипте присвоим переменной E имя команды для сортировки и имя файла, который нас необходимо отсортировать. Выведем результат сортировки на экран.

```
12  
4244  
1234  
34232532  
34612  
575  
2356  
24232  
24  
1234  
214643  
3222
```

Рисунок 13 - Содержимое файла

```
F=`sort test`;  
echo $F;
```

Рисунок 14 - Листинг

```
g4zele@rcrmdpo:~$ sh lr5.sh
12 1234 1234 214643 2356 24 24232 3222 34232532 34612 4244 575
g4zele@rcrmdpo:~$
```

Рисунок 15 - Результат работы

8. Программа запрашивает значение переменной, а затем выводит значение этой переменной. Для ввода значения переменной А воспользуемся командой `read`. Затем с помощью команды `echo` выведем значение переменной на экран.

```
read var;
echo $var;
```

Рисунок 16 - Листинг

```
g4zele@rcrmdpo:~$ sh lr5.sh
4
4
g4zele@rcrmdpo:~$
```

Рисунок 17 - Результат работы

9. Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введенной переменной. Для ввода имени в переменную А воспользуемся командой `read`. Затем с помощью команды `echo` выведем приветствие и имя на экран.

```
read var;
echo Hi $var;
```

Рисунок 18 - Листинг

```
g4zele@rcrmdpo:~$ sh lr5.sh
42
Hi 42
g4zele@rcrmdpo:~$
```

Рисунок 19 - Результат работы

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

Для ввода значений в переменные A и B воспользуемся командой read. Затем с помощью команды echo выведем значения полученных операций на экран.

```
read A;
read B;
echo `expr $A + $B`
echo $A+$B | bc
echo `expr $A - $B`
echo $A-$B | bc
echo `expr $A \* $B`
echo $A*$B | bc
echo `expr $A / $B`
echo $A/$B | bc
```

Рисунок 20 - Листинг

```
g4zele@rcrmdpo:~$ sh lr5.sh
6
3
9
9
3
3
18
18
2
2
g4zele@rcrmdpo:~$
```

Рисунок 21 - Результат работы

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

Для ввода значений в переменные r (радиус основания) и h (высота цилиндра) воспользуемся командой read. В переменную pi положим значение числа пи (3,14). Затем с помощью команды echo выведем значения полученного объема.

```
echo "R - ?"
read R
echo "R = $R"
echo "H - ?"
read H
echo "H = $H"
pi='3.14'
echo "$pi * $R * $R * $H" | bc
```

Рисунок 22 - Листинг

```

g4zele@rcrmdpo:~$ sh lr5.sh
R - ?
1
R = 1
H - ?
1
H = 1
3.14
g4zele@rcrmdpo:~$

```

Рисунок 23 - Результат работы

12. Используя позиционные параметры, отобразить имя программы, количество аргументов командной строки, значение каждого аргумента командной строки. В переменной \$0 хранится имя программы, а в \$# кол-во аргументов. С помощью цикла for пройдемся по всем аргументам командной строки.

```

echo "Name programm - $0, count arg of command in line $#"
```

```

for arg in $@
do
echo $arg
done

```

Рисунок 24 - Листинг

```

g4zele@rcrmdpo:~$ sh lr5.sh avc 21 dawg
Name programm - lr5.sh, count arg of command in line 3
avc
21
dawg
g4zele@rcrmdpo:~$

```

Рисунок 25 - Результат работы

13. Используя позиционный параметр, отобразить содержимое текстового файла, указанного в качестве аргумента командной строки. После паузы экран очищается. Имя текстового файла будет находиться в переменной \$1. С помощью оператора if производим необходимые проверки: 1. Количество аргументов командной строки единица; 2. Что текстовый файл существует. Выводим содержимое файла командой echo \$(cat \$1). Пауза создается командой sleep, очистка экрана – clear

```

echo $1
if [ $# -eq 1 ]
then
    if [ -s $1 ]
    then
        echo $(cat $1)
        sleep 25
        clear
    else
        echo "ERROR FILE"
    fi
else
    echo "ERROR ARG"
fi

```

Рисунок 26 - Листинг

```

g4zele@rcrmdpo:~$ sh lr5.sh test
test
12 4244 1234 34232532 34612 575 2356 24232 24 1234 214643 3222

```

Рисунок 27 - Результат работы

14. Используя оператор FOR, отобразить содержимое текстовых файлов текущего каталога построчно с помощью цикла for проходимся по всем файлам, если файл текстовый, то используем команду cat.

```

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

```

Рисунок 28 - Листинг

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
0  
(END)
```

Рисунок 29 - Результат работы (1)

```
Every night was long and gloomy  
Shadows gathered in the air  
No one ever lisend to me  
No one wondered did I care  
None in all the world to love me  
None to count the stars that hung  
Then the moon came up above me  
And i saw, that it was young  
I wished on the moon  
For something i never new  
Wished on the moon, For more than ever new  
A sweeter rose, a softer sky  
An ai pril day, that would not dance away  
I begged on the star to throw me a beam or two  
Wished on the star and ask for a dream or two  
I looked for every lovliness, it all came true  
    I wished on the moon  
        For  
            You  
(END)
```

Рисунок 30 - Результат работы (2)

```
12
4244
1234
34232532
34612
575
2356
24232
24
1234
214643
3222
(END)
```

Рисунок 31 - Результат работы (3)

```
for f in ./*
do
    if [ -f $f ]
    then
        cat $f | less
    fi
done
(END)
```

Рисунок 32 - Результат работы (4)

15. Программой запрашивается ввод числа, значение которого затем сравнивается с допустимым значением. В результате этого сравнения на экран выдаются соответствующие сообщения. Для ввода значения в переменную var воспользуемся командой read. Затем с помощью if сравниваем значения переменной var с числом 0. В зависимости от сравнения выводим соответствующее сообщение на экран.

```
echo "Var - ?"
read Var
if [ $Var -eq 0 ]
then
    echo "Var is 0"
else
    echo "Var isn't 0"
fi
```

Рисунок 33 - Листинг

```

g4zele@rcrmdpo:~$ sh lr5.sh
Var - ?
0
Var is 0
g4zele@rcrmdpo:~$ sh lr5.sh
Var - ?
14
Var isn't 0
g4zele@rcrmdpo:~$

```

Рисунок 34 - Результат работы

16. Программой запрашивается год, определяется, високосный ли он. Результат выдается на экран В переменную y будет помещен введенный год, проверяем введенный год с помощью условий if. В зависимости от сравнения выводим соответствующее сообщение на экран.

```

echo "Input Year:"
read y
if [ `expr $y % 4` -eq 0 -a `expr $y % 100` -ne 0 ]
then
    echo "Yes"
elif [ `expr $y % 400` -eq 0 ]
then
    echo "Yes"
else
    echo "No"
fi

```

Рисунок 35 - Листинг

```

g4zele@rcrmdpo:~$ sh lr5.sh
Input Year:
2000
Yes
g4zele@rcrmdpo:~$ sh lr5.sh
Input Year:
2010
No
g4zele@rcrmdpo:~$ sh lr5.sh
Input Year:
2020
Yes
g4zele@rcrmdpo:~$

```

Рисунок 36 - Результат работы

17. Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значения инкрементируются С помощью команды read вводим значения. С помощью цикла while инкрементируем переменные a и b пока выполняется условие.

```
echo "Input two nums"
read a
read b
echo "Input interval"
read c
read d
while [ $c -gt $a ] && [ $c -gt $b ] && [ $d -gt $a ] && [ $d -gt $b ]
do
    a=`expr $a + 1`
    b=`expr $b + 1`
    echo "$a ; $b"
done
echo "a - $a; b - $b"
```

Рисунок 37 - Листинг

```
g4zele@rcrmdpo:~$ sh lr5.sh
Input two nums
0
1
Input interval
9
10
1 ; 2
2 ; 3
3 ; 4
4 ; 5
5 ; 6
6 ; 7
7 ; 8
8 ; 9
a - 8; b - 9
g4zele@rcrmdpo:~$
```

Рисунок 38 - Результат работы

18. В качестве аргумента командной строки указывается пароль. Если пароль введен верно, постранично отображается в длинном формате с указанием скрытых файлов содержимое каталога /etc Пароль будет храниться в переменной \$1, в переменной pw хранится пароль с которым будет сравниваться с введенный. Если пароли совпадают, то будет выполнена

команда для отображения в длинном формате с указанием скрытых файлов содержимое каталога /etc.

```
pw="Aid123"
echo $1
if [ "$1" = "$pw" ]
then
    cd / | ls -al /etc | less
else
    echo "Wrong Pass"
fi
```

Рисунок 39 - Листинг

```
g4zele@rcrmdpo:~$ sh lr5.sh aid123
aid123
Wrong Pass
g4zele@rcrmdpo:~$ sh lr5.sh Aid123
Aid123
[21]+  Stopped                  sh lr5.sh Aid123
g4zele@rcrmdpo:~$
```

Рисунок 40 - Результат работы

19. Проверить, существует ли файл. Если да, выводится на экран его содержимое, если нет - выдается соответствующее сообщение. С помощью команды read вводим имя файла. Затем с помощью оператора if проверяем существует ли файл. Если существует выполняем команду для просмотра содержимого файла.

```
echo "FileName"
read FN
if [ -e $FN ]
then
    cat $FN
else
    echo "File doesn't exist"
fi
```

Рисунок 41 - Листинг

```

g4zele@rcrmdpo:~$ sh lr5.sh
FileName
test
12
4244
1234
34232532
34612
575
2356
24232
24
1234
214643
3222
g4zele@rcrmdpo:~$ sh lr5.sh
FileName
safsfasfa
File doesn't exist
g4zele@rcrmdpo:~$

```

Рисунок 42 - Результат работы

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. Вводим имя файла или каталога. С помощью операторов if осуществляем необходимые проверки (существование каталога с таким именем, является ли файл каталогом, возможность чтения содержимого каталога)

```

echo "File\Dir"
read fName
if [ -e $fName ]
then
    if [ -d $fName ]
    then
        if [ -r $fName ]
        then
            ls $fName
        else
            echo "Can't read"
        fi
    else
        echo "Open file"
        cat $fName
    fi
else
    "Make new dir"
    mkdir $fName
fi

```

Рисунок 43 – Листинг

```
g4zele@rcrmdpo:~$ sh lr5.sh
File\Dir
test
Open file
12
4244
1234
34232532
34612
575
2356
24232
24
1234
214643
3222
g4zele@rcrmdpo:~$
```

Рисунок 44 - Результат работы (Открытие файла)

```
g4zele@rcrmdpo:~$ ls
1.txt Desktop Downloads Pictures Templates a demo new res_new test
2.txt Documents Music Public Videos clear lr5.sh res snap tmp.txt
g4zele@rcrmdpo:~$ sh lr5.sh
File\Dir
a
lr5-14.sh txt1.txt txt2.txt
g4zele@rcrmdpo:~$
```

Рисунок 45 - Результат работы (открытие Директории)

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) Вводим с помощью команды read имена файлов. С помощью if проверим существование файлов, и права доступа. Перенаправляем содержимое файла 1 в файл 2 с помощью команды cat и перенаправления вывода.

```
echo "Name of file 1:"
read File1
echo "Name of file 2:"
read File2
if [ -e $File1 -a -e $File2 -a -r $File1 -a -w $File2 ]
then
    echo "File1:"
    cat $File1
    echo "File2:"
    cat $File2
    cat $File1 > $File2
    echo "New file 2"
    cat $File2
else
    echo "ERROR"
fi
```

Рисунок 46 - Листинг

```

g4zele@rcrmdpo:~$ sh lr5.sh
Name of file 1:
test
Name of file 2:
tmp.txt
File1:

test
File2:
total 60
-rw-r--r-- 1 g4zele g4zele  20 Nov 13 20:57 1.txt
-rw-r--r-- 1 g4zele g4zele 585 Nov 13 21:21 2.txt
drwxr-xr-x 2 g4zele g4zele 4096 Oct 30 16:25 Desktop
drwxr-xr-x 2 g4zele g4zele 4096 Oct 30 16:25 Documents
drwxr-xr-x 2 g4zele g4zele 4096 Oct 30 16:25 Downloads
drwxr-xr-x 2 g4zele g4zele 4096 Oct 30 16:25 Music
drwxr-xr-x 2 g4zele g4zele 4096 Oct 30 17:43 Pictures
drwxr-xr-x 2 g4zele g4zele 4096 Oct 30 16:25 Public
drwxr-xr-x 2 g4zele g4zele 4096 Oct 30 16:25 Templates
drwxr-xr-x 2 g4zele g4zele 4096 Oct 30 16:25 Videos
prw-r--r-- 1 g4zele g4zele   0 Nov 13 19:05 chan
-rwxr-xr-x 1 g4zele g4zele  27 Nov 13 17:53 loop
-rwxr-xr-x 1 g4zele g4zele  39 Nov 13 17:55 loop2
drwxr-xr-x 3 g4zele g4zele 4096 Nov 13 20:41 new
-rw-r--r-- 1 g4zele g4zele 202 Nov 13 19:01 res
-rw-r--r-- 1 g4zele g4zele 121 Nov 13 19:05 res_new
-rw-r--r-- 1 g4zele g4zele   0 Nov 13 21:41 tmp.txt
New file 2

test
g4zele@rcrmdpo:~$

```

Рисунок 47 - Результат работы

22. Если файл запуска программы найден, программа запускается (по выбору) Проверяем количество аргументов командной строки, если файл \$1 существует и может быть выполнен, то выполняем данный файл.

```

if [ $# -eq 1 ]
then
    if [ -e $1 -a -x $1 ]
    then
        sh $1
    else
        echo "ERROR"
    fi
else
    echo "ERROR"
fi

```

Рисунок 48 - Листинг

```

g4zele@rcrmdpo:~$ sh lr5.sh 9.sh
145
Hi 145

```

Рисунок 49 - Результат работы

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. Проверяем количество аргументов, в переменной \$1 будет находиться имя файла. С помощью if проверяем, что файл существует и не пуст. Если размер файла больше нуля, содержимое файла сортируется по первому столбцу по возрастанию, отсортированная информация помещается в файл task23result.

```
if [ $# -eq 1 ]
then
  if [ -s $1 ]
  then
    sort -k1 $1 > "task23result"
    cat "task23result"
  else
    echo "ERROR"
  fi
else
  echo "ERROR"
fi
```

Рисунок 50 - Листинг

```
g4zele@rcrmdpo:~$ sh lr5.sh test
0
1
2
3
32
4
5
6
7
8
9
g4zele@rcrmdpo:~$
```

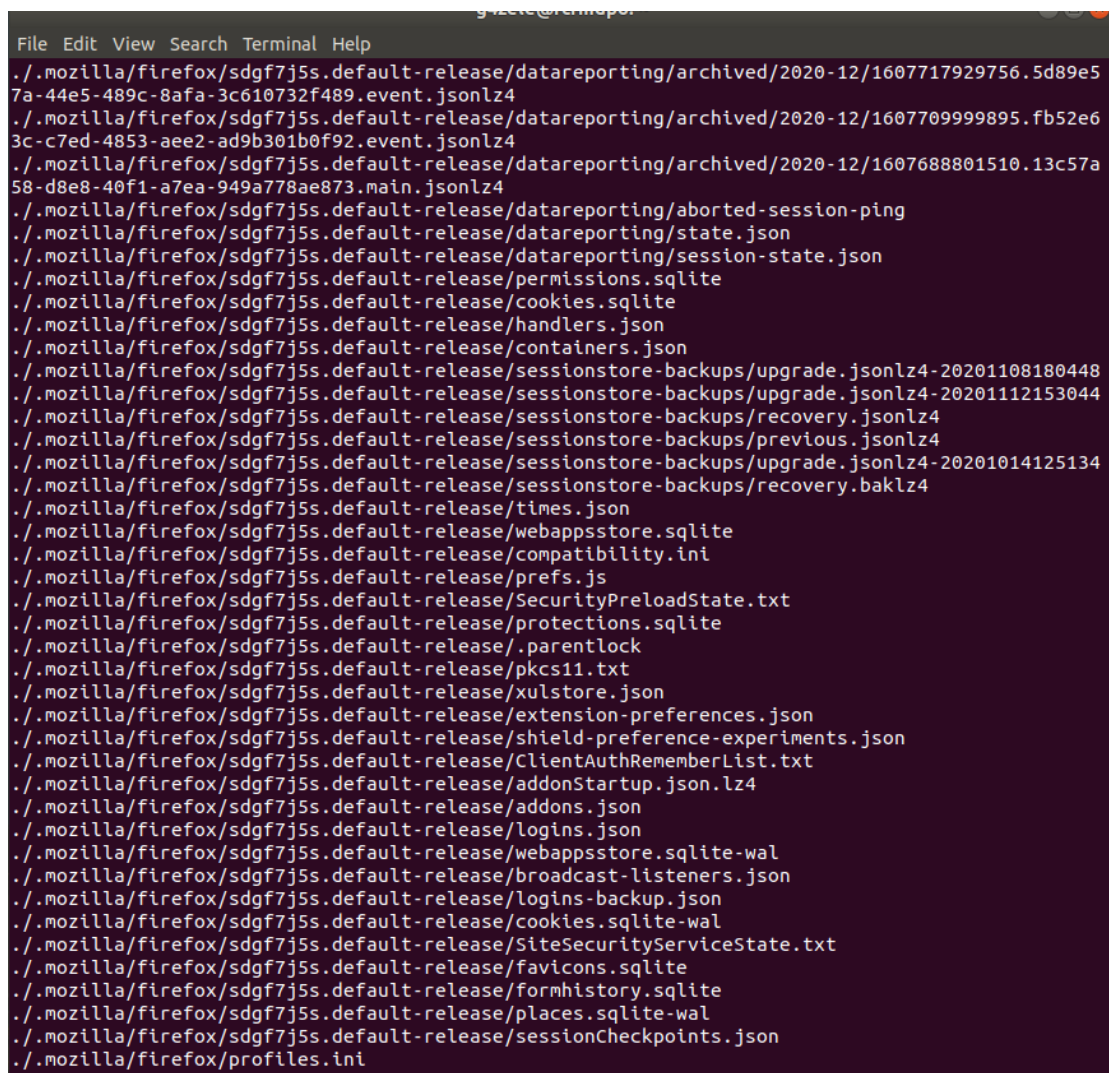
Рисунок 51 - Результат работы

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

Командой `find . -type f` поиск всех текстовых файлов. В переменной `archName` хранится название архивного файла. Командой `tar` осуществляется сборка всех текстовых файлов текущего каталога в архивный файл, после паузы просматривается содержимое файла, затем командой `gzip` архивный файл сжимается

```
find=`find . -type f`  
archName="task24.tar"  
tar -cf $archName $find  
sleep 10  
tar -tf $archName  
gzip $archName
```

Рисунок 51 - Листинг



```
File Edit View Search Terminal Help  
./mozilla/firefox/sdgyf7j5s.default-release/datareporting/archived/2020-12/1607717929756.5d89e5  
7a-44e5-489c-8afa-3c610732f489.event.jsonlz4  
./mozilla/firefox/sdgyf7j5s.default-release/datareporting/archived/2020-12/1607709999895.fb52e6  
3c-c7ed-4853-aee2-ad9b301b0f92.event.jsonlz4  
./mozilla/firefox/sdgyf7j5s.default-release/datareporting/archived/2020-12/1607688801510.13c57a  
58-d8e8-40f1-a7ea-949a778ae873.main.jsonlz4  
./mozilla/firefox/sdgyf7j5s.default-release/datareporting/aborted-session-ping  
./mozilla/firefox/sdgyf7j5s.default-release/datareporting/state.json  
./mozilla/firefox/sdgyf7j5s.default-release/datareporting/session-state.json  
./mozilla/firefox/sdgyf7j5s.default-release/permissions.sqlite  
./mozilla/firefox/sdgyf7j5s.default-release/cookies.sqlite  
./mozilla/firefox/sdgyf7j5s.default-release/handlers.json  
./mozilla/firefox/sdgyf7j5s.default-release/containers.json  
./mozilla/firefox/sdgyf7j5s.default-release/sessionstore-backups/upgrade.jsonlz4-20201108180448  
./mozilla/firefox/sdgyf7j5s.default-release/sessionstore-backups/upgrade.jsonlz4-20201112153044  
./mozilla/firefox/sdgyf7j5s.default-release/sessionstore-backups/recovery.jsonlz4  
./mozilla/firefox/sdgyf7j5s.default-release/sessionstore-backups/previous.jsonlz4  
./mozilla/firefox/sdgyf7j5s.default-release/sessionstore-backups/upgrade.jsonlz4-20201014125134  
./mozilla/firefox/sdgyf7j5s.default-release/sessionstore-backups/recovery.baklz4  
./mozilla/firefox/sdgyf7j5s.default-release/times.json  
./mozilla/firefox/sdgyf7j5s.default-release/webappsstore.sqlite  
./mozilla/firefox/sdgyf7j5s.default-release/compatibility.ini  
./mozilla/firefox/sdgyf7j5s.default-release/prefs.js  
./mozilla/firefox/sdgyf7j5s.default-release/SecurityPreloadState.txt  
./mozilla/firefox/sdgyf7j5s.default-release/protections.sqlite  
./mozilla/firefox/sdgyf7j5s.default-release/.parentlock  
./mozilla/firefox/sdgyf7j5s.default-release/pkcs11.txt  
./mozilla/firefox/sdgyf7j5s.default-release/xulstore.json  
./mozilla/firefox/sdgyf7j5s.default-release/extension-preferences.json  
./mozilla/firefox/sdgyf7j5s.default-release/shield-preference-experiments.json  
./mozilla/firefox/sdgyf7j5s.default-release/ClientAuthRememberList.txt  
./mozilla/firefox/sdgyf7j5s.default-release/addonStartup.json.lz4  
./mozilla/firefox/sdgyf7j5s.default-release/addons.json  
./mozilla/firefox/sdgyf7j5s.default-release/logins.json  
./mozilla/firefox/sdgyf7j5s.default-release/webappsstore.sqlite-wal  
./mozilla/firefox/sdgyf7j5s.default-release/broadcast-listeners.json  
./mozilla/firefox/sdgyf7j5s.default-release/logins-backup.json  
./mozilla/firefox/sdgyf7j5s.default-release/cookies.sqlite-wal  
./mozilla/firefox/sdgyf7j5s.default-release/ServiceWorkerState.txt  
./mozilla/firefox/sdgyf7j5s.default-release/favicons.sqlite  
./mozilla/firefox/sdgyf7j5s.default-release/formhistory.sqlite  
./mozilla/firefox/sdgyf7j5s.default-release/places.sqlite-wal  
./mozilla/firefox/sdgyf7j5s.default-release/sessionCheckpoints.json  
./mozilla/firefox/profiles.ini
```

Рисунок 52 - Результат работы

25. Написать скрипт с использованием функции, например, функции, суммирующей значения двух переменных С помощью read вводим значения цифр для суммирования. Затем создаем функцию и вызываем её.

```
echo "Input A & B"
read A
read B
sumF()
{
    echo $A+$B |bc
}
echo $(sumF)
```

Рисунок 53 - Листинг

```
g4zele@rcrmdpo:~$ sh lr5.sh
Input A & B
5
5
10
```

Рисунок 54 - Результат работы

Вывод

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