

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

Отчет по лабораторной работе № 5

«Программирование на SHELL»

по курсу «ОС Linux»

Студент
Группа АИ-18

Грунау Г.Ю.

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.

Содержание

| | |
|--|----|
| Цель работы | 2 |
| 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. | 7 |
| 5. Присвоить переменной D значение “имя команды”, а именно, команды DATE. Выполнить эту команду, используя значение переменной. | 8 |
| 6. Присвоить переменной E значение “имя команды”, а именно, команды просмотра содержимого файла, просмотреть содержимое переменной. Выполнить эту команду, используя значение переменной. | 8 |
| 7. Присвоить переменной F значение “имя команды”, а именно сортировки содержимого текстового файла. Выполнить эту команду, используя значение переменной. | 9 |
| 8. Программа запрашивает значение переменной, а затем выводит значение этой переменной..... | 10 |
| 9. Программа запрашивает имя пользователя, затем здоровается с ним, используя значение введенной переменной. | 10 |
| 10. Программа запрашивает значения двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на экран (использовать команды а) EXPR; б) BC), | 12 |
| 11. Вычислить объем цилиндра. Исходные данные запрашиваются программой. Результат выводится на экран. | 13 |

| | |
|---|----|
| 12. Используя позиционные параметры, отобразить имя программы, количество аргументов командной строки, значение каждого аргумента командной строки | 14 |
| 13. Используя позиционный параметр, отобразить содержимое текстового файла, указанного в качестве аргумента командной строки. После паузы экран очищается. | 15 |
| 14. Используя оператор FOR, отобразить содержимое текстовых файлов текущего каталога поэкранно | 16 |
| 15. Программой запрашивается ввод числа, значение которого затем сравнивается с допустимым значением. В результате этого сравнения на экран выдаются соответствующие сообщения. | 18 |
| 16. Программой запрашивается год, определяется, високосный ли он. Результат выдается на экран..... | 19 |
| 17. Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значения инкрементируются | 20 |
| 18. В качестве аргумента командной строки указывается пароль. Если пароль введен верно, постранично отображается в длинном формате с указанием скрытых файлов содержимое каталога /etc. | 22 |
| 19. Проверить, существует ли файл. Если да, выводится на экран его содержимое, если нет - выдается соответствующее сообщение | 23 |
| 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..... | 24 |
| 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)..... | 26 |
| 22. Если файл запуска программы найден, программа запускается (по выбору). | 27 |
| 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..... | 28 |
| 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. | 30 |
| 25. Написать скрипт с использованием функции, например, функции, суммирующей значения двух переменных. | 32 |
| Вывод..... | 33 |

Ход работы

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

Создадим сценарий, используя редактор `vi script.sh` и введем код:

```
str="$(date)";  
echo "$str";  
printf "%s\n" "$str"
```

Рисунок 1 – Использование ECHO, PRINTF

Присваиваем переменной `str` значение текущей даты. Затем выведем значение переменной с помощью `echo` и `printf`, используя формат строки с переносом `“%s\n”` и занесем переменную окружения в двойные скобки `“$str”`, чтобы дата отобразилась одной строкой.

Запустим сценарий:

```
lovediehatemyubuntuserver:~/lr5$ sh script.sh  
Tue Nov 17 14:07:14 UTC 2020  
Tue Nov 17 14:07:14 UTC 2020
```

Рисунок 2 – Вывод даты

2. Присвоить переменной `A` целочисленное значение. Просмотреть значение переменной `A`.

```
A=5;  
printf "%d\n" $A
```

Рисунок 3 – Присвоение значения

```
lovediehatemyubuntuserver:~/lr5$ sh script.sh  
5
```

Рисунок 4 – Просмотр значения `A`

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

```
A=5;  
B=$A;  
printf "%s\n" "Value of B = $B"
```

Рисунок 5 – Присвоение значения `B`

```
lovediehate@myubuntuserver:~/lr5$ sh scrypt.sh  
Value of B = 5
```

Рисунок 6 – Просмотр значения B

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

```
#!/bin/bash  
C=$PWD;  
printf "%s\n" "Current dir: $PWD"  
cd /usr;  
printf "%s\n" "Current dir: $PWD"  
cd $C;  
printf "%s\n" "Current dir: $PWD"
```

Рисунок 7 – Переход в директорию

```
C=$PWD;  
// Присваиваем переменной текущую директорию  
printf "%s\n" "Current dir: $PWD";  
// Выводим текущую директорию  
cd /usr;  
// Переходим в директорию /usr  
printf "%s\n" "Current dir: $PWD";  
// Выводим текущую директорию  
cd $C;  
// Переходим в директорию, присвоенную переменной C  
printf "%s\n" "Current dir: $PWD";  
// Выводим текущую директорию
```

```
lovediehate@myubuntuserver:~$ sudo sh scrypt.sh  
Current dir: /home/lovediehate  
Current dir: /usr  
Current dir: /home/lovediehate
```

Рисунок 8 – Результат

На рисунке 8 видно, что нам удалось перейти в начальную директорию с помощью переменной C.

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

Чтобы узнать текущую дату и время, нужно ввести утилиту `date`.

```
#!/bin/bash
D=`date`;
echo $D
```

Рисунок 9 – Присвоение переменной имени команды

`D=`date`` // Присвоение команды `date` переменной D

`echo $D` // Обращение к переменной D

```
lovediehate@myubuntuserver:~$ sh script.sh
Wed Nov 18 08:26:57 UTC 2020
```

Рисунок 10 – Выполнение скрипта

Как мы видим на рисунке 10, команда `date` успешно сработала по обращению к переменной D.

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

```
E=`cat < '/home/lovediehate/tmpfile'`;
printf "%s\n" "$E";
```

Рисунок 11 – Код скрипта

`E=`cat < '/home/lovediehate/tmpfile'`` // Присваивание переменной E значение выполнения команды `cat < 'file'`

`printf "%s\n" "$E"` // Вывод значения переменной E

```
lovediehate@myubuntuserver:~$ cat tmpfile
interesting text
or not?
lovediehate@myubuntuserver:~$ sh script.sh
interesting text
or not?
```

Рисунок 12 – Выполнение скрипта

На рисунке 12 видно, что наш скрипт `scrypt.sh` успешно вывел содержимое файла `tmpfile`.

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

```
D=`sort '/home/lovediehat/tmpfile'`;
printf "%s\n" "$D";
```

Рисунок 13 – Код скрипта

`D=`sort '/home/lovediehat/tmpfile'`; // Присвоение переменной D результата команды /home/lovediehat/tmpfile.`

`printf "%s\n" "$D" // Вывод значения переменной D`

```
lovediehat@myubuntuserver:~$ cat tmpfile
interesting text
or not?
Aword
Aaword
2zet
/somechar
a
111
lovediehat@myubuntuserver:~$ sh scrypt.sh
/somechar
111
2zet
Aaword
Aword
a
interesting text
or not?
```

Рисунок 14 – Выполнение скрипта

Результат выполнения скрипта: строки файла `tmpfile` успешно выведены и предварительно отсортированы.

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

За пользовательский ввод в BASH-скриптах отвечает встроенная команда `read`, которая считывает одну строку в переменную.

```
echo "Enter string: "  
read F;  
echo $F
```

Рисунок 15 – Код скрипта

Строка `read F` подразумевает, что пользователь должен ввести в консоли строковое значение, которое присвоится переменной `F`.

```
lovediehatemyubuntuserver:~$ sh scrip.sh  
Enter string:  
this is value of F  
this is value of F
```

Рисунок 16 – Выполнение скрипта

Видно, что пользователь (я) после консольного сообщения “Enter string:” ввёл какую-то строку, она считалась в переменную и после этого вывелась.

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

```
echo "What's your name? "  
read Name;  
echo "Hello, $Name";
```

Рисунок 17 – Код скрипта

Строка `read Name` подразумевает, что пользователь должен ввести в консоли строковое значение (имя), которое присвоится переменной `Name`.

В последней строке скрипт выводит сообщение Hello, `$Name`. Вместо `$Name` выведется строка, которую ввёл пользователь (либо пустота, если пользователь ничего не ввёл).

```
lovediehatemyubuntuserver:~$ sh scrip.sh  
What's your name?  
German  
Hello, German
```

Рисунок 18 – Выполнение скрипта

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

Для вычисления выражений можно использовать команду `expr`. Её можно применять как в командной строке, так и в скриптах.

```
read -p "Enter first number: " H;
read -p "Enter second number: " I;
echo "Summ with EXPR: $(expr $H + $I)"; # summ
echo "Diff with EXPR: $(expr $H - $I)"; # difference
echo "Mult with EXPR: $(expr $H \* $I)"; # multiplication
echo "Div with EXPR: $(expr $H / $I)"; # division
summ=`echo "$H + $I" |bc`;
diff=`echo "$H - $I" |bc`;
mult=`echo "$H * $I" |bc`;
div=`echo "scale=5;$H / $I" |bc`;
echo "Summ with BC: $summ";
echo "Diff with BC: $diff";
echo "Mult with BC: $mult";
echo "Div with BC: $div";
```

Рисунок 19 – Код скрипта

С помощью `expr` доступно только целочисленное вычисление. Для дробных чисел лучше использовать `bc` – Си-подобный интерактивный интерпретатор. Для того, чтобы выводились числа после запятой, нужно использовать переменную `scale`, означающую количество цифр в дробной части (10 строка, рис. 19).

```
lovediehatemyubuntuserver:~$ sh scrip.sh
Enter first number: 10
Enter second number: 3
Summ with EXPR: 13
Diff with EXPR: 7
Mult with EXPR: 30
Div with EXPR: 3
Summ with BC: 13
Diff with BC: 7
Mult with BC: 30
Div with BC: 3.33333
```

Рисунок 20 – Выполнение скрипта

Как мы видим, при делении 10 на 3 EXPR выдал результат 3, а BC выдал результат 3.33333. Всё благодаря тому, что при делении с BC была указана переменная `scale=5`.

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

```
read -p "Enter the height: " h;
read -p "Enter the radius: " rad;
V=`echo "scale=2; $h * $rad * $rad * 3.14" |bc`
echo "V = $V";
```

Рисунок 21 – Код программы

`read -p "Enter the height: " h;` // считывание введенной строки в h

Параметр `-p <PROMT>` - строка приглашения `<PROMPT>`. Без окончного перевода строки, обычно в ней выводят подсказку перед тем, как команда `read` будет считывать данные.

Объем цилиндра вычисляется по формуле $V=3.14*r*r*h$, где r – радиус, h - высота;

Т.к. при вычислении объема цилиндра используется число Пи, нужно использовать калькулятор `bc`, т.к. `EXPR` работает только с целыми числами.

```
lovediehatemyubuntuserver:~$ sh scrip.sh
Enter the height: 2
Enter the radius: 1
V = 6.28
```

Рисунок 22 – Выполнение программы

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

Позиционные параметры — это аргументы командой строки (или функции в скрипте), доступ к которым осуществляется по номеру `${number}`.

Специальные параметры:

`$*` — 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.

```
#!/bin/sh
printf "Name: %s\nQuantity args: %d\n" $0 $#
list="$@";
for i in $list; do
    echo "arg: $i"
done
```

Рисунок 23 – Код скрипта

```
lovediehatemyubuntuserver:~$ ./scrypt.sh 3 4 5
Name: ./scrypt.sh
Quantity args: 3
arg: 3
arg: 4
arg: 5
```

Рисунок 24 – Выполнение скрипта

С помощью `$0` вывелось имя, `$#` — кол-во аргументов, и с помощью цикла и обращения к позиционным параметрам вывелись все введённые аргументы.

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

```
#!/bin/sh
echo `cat < $1`;
sleep 5;
clear;
```

Рисунок 25 – Код скрипта

Скрипт будет выводить содержание файла (путь к файлу передаётся в первом позиционном аргументе), затем после пятисекундной паузы очистит терминал.

```
lovediehatemyubuntuserver:~$ cat tmpfile
interesting text
or not?
Aword
Aaword
2zet
/somechar
a
111
lovediehatemyubuntuserver:~$ ./scrypt.sh ./tmpfile
interesting text or not? Aword Aaword 2zet /somechar a 111
```

Рисунок 26 – Выполнение скрипта

На рисунке 26 изображены просмотр файла через cat и выполнение скрипта.

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

Для построчного отображения используется команда `less`. С её помощью можно посмотреть содержимое нескольких файлов по очереди, при этом окно терминала не будет засоряться.

Чтобы получить список файлов каталога, используется простая команда `ls`. Цикл пробегается по списку файлов и определяет формат каждого файла с помощью утилиты `file` с параметром `-b`

```
"script.sh" 9L, 123C written
lovediehatemyubuntuserver:~$ file *
H:          empty
I:          empty
Somebody:   ASCII text
arch:       gzip compressed data, max speed, from Unix, original size modulo 2^32 829
cronlog.txt: data
fifo:       fifo (named pipe)
lr2:        directory
lr5:        directory
my_arch:    gzip compressed data, max speed, from Unix, original size modulo 2^32 215
script.sh:  POSIX shell script, ASCII text executable
smbd.err.log: empty
smbd.out.log: data
somebody.err.log: empty
somebody.out.log: ASCII text
tmpfile:    ASCII text
```

Рисунок 27 – Демонстрация утилиты `file`

Параметр `-b` убирает из вывода имя файла, оставляя только его тип. Это требуется для более простого дальнейшего сравнения. Далее, формат каждого элемента списка с помощью условного оператора `if` проверяется, и если формат списка – “ASCII text”, то выполняется просмотр содержимого этого файла с помощью команды `less` “путь/имя файла”. `$PWD` – путь к текущему каталогу, `$i` – имя файла из списка файлов `ls`. На рисунке 27 видно, что у нас всего 3 текстовых файла. Их содержимое скрипт и должен вывести.

```
#!/bin/sh
list=`ls`;
for i in $list; do
    A=`file -b "$PWD/$i"`;
    if [ "$A" = "ASCII text" ]
    then less "$PWD/$i"
    fi
done
```

Рисунок 28 – Код программы


```
interesting text
or not?
Aword
Aaword
2zet
/somechar
a
111
/home/lovediehate/tmpfile (END)
```

Рисунок 31 – Выполнение скрипта, третий файл

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

```
#!/bin/sh
read -p "Enter the number greater then 5: " number;
if [ "$number" -gt 5 ]
then echo "Nice"
else echo "Error"
fi
```

Рисунок 32 – Код скрипта

Скрипт предложит пользователю ввести число. Если оно будет больше 5, то скрипт выдаст сообщение 'Nice', иначе 'Error'.

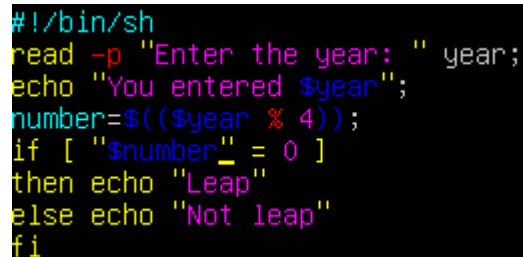
```
lovediehate@myubuntuserver:~$ ./scrypt.sh
Enter the number greater then 5: 2
Error
lovediehate@myubuntuserver:~$ ./scrypt.sh
Enter the number greater then 5: 8
Nice
lovediehate@myubuntuserver:~$ ./scrypt.sh
Enter the number greater then 5: 5
Error
```

Рисунок 33 – Выполнение скрипта

16. Программой запрашивается год, определяется, високосный ли он.

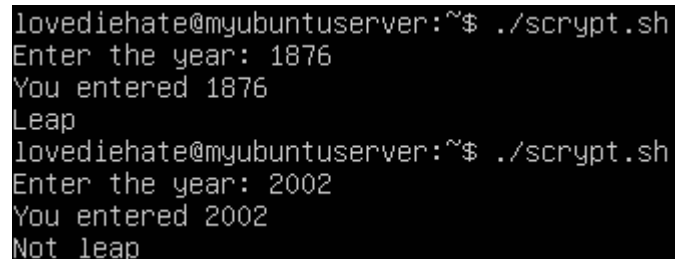
Результат выдается на экран

Високосный год – год, который делится на 4 без остатка. Скрипт делит по модулю входные данные на 4, и если остатка равен нулю, значит год високосный (Leap), иначе не високосный (Not leap).



```
#!/bin/sh
read -p "Enter the year: " year;
echo "You entered $year";
number=$(( $year % 4 ));
if [ "$number" = 0 ]
then echo "Leap"
else echo "Not leap"
fi
```

Рисунок 34 – Код программы



```
lovediehate@myubuntuserver:~$ ./scrypt.sh
Enter the year: 1876
You entered 1876
Leap
lovediehate@myubuntuserver:~$ ./scrypt.sh
Enter the year: 2002
You entered 2002
Not leap
```

Рисунок 35 – Выполнение программы

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

for ((;;)) do – бесконечный цикл.

Далее идут 3 ветки условий if.

Первая ветка – проверка введённых А и В на вход в диапазон.

Вторая ветка – увеличивает А на 1 каждый проход цикла, если А меньше второй границы диапазона.

Третья ветка – такая же, как вторая, только для переменной В.

```
#!/bin/bash
read -p "Enter A: " A;
read -p "Enter B: " B;
read -p "Enter 1 number of range: " C;
read -p "Enter 2 number of range: " D;
for ((;;)) do
if [[ "$A" -ge "$D" || "$A" -le "$C" ]] && [[ "$B" -ge "$D" || "$B" -le "$C" ]]
then
    break
fi
if [ "$A" -gt "$C" ] && [ "$A" -lt "$D" ]
then
    A=$((A + 1))
    echo "A=$A"
    sleep 1
fi
if [ "$B" -gt "$C" ] && [ "$B" -lt "$D" ]
then
    B=$((B + 1))
    echo "B=$B"
    sleep 1
fi
done;
```

Рисунок 36 – Код программы

В первом случае только А входит в диапазон от [1;3], поэтому переменная В не изменилась, а переменная А инкрементировалась (сработало второе условие).

Во втором случае обе переменные вошли в диапазон и увеличились до правой границы диапазона (сработало второе и третье условие).

В третьем случае ни одна переменная не вошла в диапазон, поэтому они не изменились (сработало первое условие – break из цикла).

```
lovediehate@myubuntuserver:~$ ./scrypt.sh
Enter A: 2
Enter B: 4
Enter 1 number of range: 1
Enter 2 number of range: 3
A=3
lovediehate@myubuntuserver:~$ ./scrypt.sh
Enter A: 2
Enter B: 4
Enter 1 number of range: 1
Enter 2 number of range: 5
A=3
B=5
A=4
A=5
lovediehate@myubuntuserver:~$ ./scrypt.sh
Enter A: 2
Enter B: 4
Enter 1 number of range: 1
Enter 2 number of range: 2
```

Рисунок 37 – Выполнение скрипта

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

```
if [ $1 = "secretpass" ]
then
    ls -al /etc | less
else
    echo "error"
fi
```

Рисунок 38 – Код скрипта

```
lovediehatemyubuntuserver:~$ ./scrip.sh invalidpass
error
```

Рисунок 39 – Ввод неправильного пароля

```
total 808
drwxr-xr-x 95 root root      4096 Nov 16 17:44 .
drwxr-xr-x 20 root root      4096 Oct  3 11:53 ..
-rw-r--r--  1 root root         0 Jul 31 16:28 .pwd.lock
drwxr-xr-x  3 root root      4096 Jul 31 16:29 NetworkManager
drwxr-xr-x  2 root root      4096 Oct  7 11:56 PackageKit
drwxr-xr-x  4 root root      4096 Jul 31 16:29 X11
-rw-r--r--  1 root root     3028 Jul 31 16:28 adduser.conf
drwxr-xr-x  2 root root      4096 Jul 31 16:29 alternatives
drwxr-xr-x  3 root root      4096 Jul 31 16:29 apparmor
drwxr-xr-x  7 root root      4096 Jul 31 16:29 apparmor.d
drwxr-xr-x  3 root root      4096 Nov 16 17:43 apport
drwxr-xr-x  7 root root      4096 Oct  3 11:52 apt
-rw-r----- 1 root daemon    144 Nov 12 2018 at.deny
-rw-r--r--  1 root root    2319 Feb 25 2020 bash.bashrc
-rw-r--r--  1 root root      45 Jan 26 2020 bash_completion
drwxr-xr-x  2 root root      4096 Nov 16 17:43 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 Nov 16 17:42 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 11:53 cloud
drwxr-xr-x  2 root root      4096 Oct  3 11:54 console-setup
drwxr-xr-x  2 root root      4096 Jul 31 16:29 cron.d
drwxr-xr-x  2 root root      4096 Nov 16 17:43 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  7 11:58 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
:~
```

Рисунок 40 – Ввод правильного пароля

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

Для проверки существования файла применяется параметр `-s`.

```
#!/bin/bash
if [ -s "$1" ]
then
    less "$1"
else
    echo "error"
fi
```

Рисунок 41 – Код скрипта

```
lovediehatemyubuntuserver:~$ ./scrypt.sh tempfile
error
lovediehatemyubuntuserver:~$ ./scrypt.sh tmpfile
interesting text
or not?
Aword
Aaword
2zet
/somechar
a
111
tmpfile (END)
```

Рисунок 42 – Выполнение скрипта

Файла `tempfile` не существует. Файл `tmpfile` существует, содержимое вывелось.

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.

В самом первом условии файл проверяется на существование. Если не существует, то создается с помощью `mkdir`. Если существует, то будут проверяться вложенные условия.

В первом вложенном условии файл проверяется на то, является ли он каталогом и доступен ли для чтения. Если да, то выводится список его файлов с помощью `ls`.

Во втором вложенном условии файл проверяется на то, является ли он обычным файлом. Если да, то выводится его содержимое с помощью `less`.

```
#!/bin/bash
if [ -s "$1" ]
then
    if [ -d "$1" ] && [ -r "$1" ]
    then
        ls "$1"
    fi

    if [ -f "$1" ]
    then
        less "$1"
    fi
else
    mkdir "$1"
fi
```

Рисунок 43 – Код скрипта

Синим цветом выделяются названия каталогов.

```
lovediehatemyubuntuserver:~$ ls
6 I      arch      fifo  lr5      scrypt.sh  smbd.out.log  somebody.out.log
H Somebody cronlog.txt lr2    my_arch  smbd.err.log somebody.err.log tmpfile
lovediehatemyubuntuserver:~$ ./scrypt.sh lr5
scrypt.sh str
lovediehatemyubuntuserver:~$ ./scrypt.sh tmpfile
interesting text
or not?
Aword
Aaword
2zet
/somechar
a
111
lovediehatemyubuntuserver:~$ ./scrypt.sh new_dir
lovediehatemyubuntuserver:~$ ls
6 I      arch      fifo  lr5      new_dir  smbd.err.log  somebody.err.log  tmpfile
H Somebody cronlog.txt lr2    my_arch  scrypt.sh  smbd.out.log  somebody.out.log
```

Рисунок 44 – Выполнение программы

В первом случае вводится существующая директория lr5. Скрипт выдал содержимое, т.е. список файлов этой директории.

Во втором случае вводится существующий файл tmpfile. Скрипт отображает его содержимое.

В третьем случае вводится несуществующий файл. Скрипт создал каталог с таким названием.

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).

```
lovediehatemyubuntuserver:~$ cat < I
123123213
lovediehatemyubuntuserver:~$ cat < H
somewords
lovediehatemyubuntuserver:~$ ls -l
total 296
-rw-rw-r-- 1 lovediehatemyubuntuserver 0 Nov 19 14:48 6
-r--r--r-- 1 lovediehatemyubuntuserver 10 Nov 19 17:37 H
-rw-rw-r-- 1 lovediehatemyubuntuserver 10 Nov 19 17:39 I
```

Рисунок 45 – Просмотр содержимого файлов и доступа

Сначала первый файл проверяется на существование и на чтение `-r`. Если он прошёл проверку, то проверяется второй файл на запись `-w`. Если проходит проверку, то содержимое первого записывается в конец второго файла.

```
#!/bin/bash
if [ -s "$1" ] && [ -r "$2" ]
then
    if [ -s "$2" ] && [ -w "$2" ]
    then
        cat $1 >> $2;
    else
        echo "$2 does not exist or not available for writing"
    fi
else
    echo "$1 does not exist or not available for reading"
fi
```

Рисунок 46 – Код программы

Файл H доступен только для чтения, значит в него нельзя ничего записать. Файл I доступен для записи, значит в него можно записать.

```
lovediehatemyubuntuserver:~$ ./scrypt.sh I H
H does not exist or not available for writing
lovediehatemyubuntuserver:~$ ./scrypt.sh H I
lovediehatemyubuntuserver:~$ cat < I
123123213
somewords
```

Рисунок 47 – Выполнение программы

Попытка записать из файла I в файл H не увенчалась успехом.

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

```
#!/bin/bash
$1
```

Рисунок 48 – Код скрипта

Весь код скрипта:

```
$1
```

```
lovediehate@myubuntuserver:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

Рисунок 49 – echo \$PATH

Программы из перечня директорий, указанных в `PATH`, могут запускаться без пути к ним. Система ищет в этих папках введенную программу, и если находит, то запускает. Если нет, то выводит ошибку `command not found`.

```
lovediehate@myubuntuserver:~$ ./scrypt.sh "ls -l"
total 300
-rw-rw-r-- 1 lovediehate lovediehate    0 Nov 19 14:48 6
-r--r--r-- 1 lovediehate lovediehate   10 Nov 19 17:37 H
-rw-rw-r-- 1 lovediehate lovediehate   20 Nov 19 17:44 I
-rw-r--r-- 1 root        root          55 Nov 16 18:52 Somebody
-rw-rw-r-- 1 lovediehate lovediehate  255 Oct 30 19:03 arch
-rw-r--r-- 1 root        root        27289 Nov 19 18:06 cronlog.txt
-rw-rw-r-- 1 lovediehate lovediehate    0 Oct 30 19:17 fifo
drwxrwxr-x 3 lovediehate lovediehate  4096 Nov 12 19:37 lr2
drwxrwxr-x 2 lovediehate lovediehate  4096 Nov 17 15:06 lr5
-rw-rw-r-- 1 lovediehate lovediehate   115 Oct 30 19:17 my_arch
drwxrwxr-x 2 lovediehate lovediehate  4096 Nov 19 17:16 new_dir
-rwxrwxrwx 1 lovediehate lovediehate   16 Nov 19 18:01 scrypt.sh
-rw-r--r-- 1 root        root           0 Nov 16 18:53 smbd.err.log
-rw-r--r-- 1 root        root       220340 Nov 19 18:06 smbd.out.log
-rw-r--r-- 1 root        root           0 Nov 16 18:49 somebody.err.log
-rw-r--r-- 1 root        root        4940 Nov 16 18:53 somebody.out.log
-rwxrwxrwx 1 lovediehate lovediehate   59 Nov 18 09:01 tmpfile
```

Рисунок 50 – Выполнение скрипта

```
lovediehatemyubuntuserver:~$ ./scrypt.sh "ps"
  PID TTY          TIME CMD
   881 tty1        00:00:00 bash
  1589 tty1        00:00:02 scrypt.sh
  5023 tty1        00:00:00 scrypt.sh
  7426 tty1        00:00:01 scrypt.sh
 11403 tty1        00:00:00 scrypt.sh
 13649 tty1        00:00:01 scrypt.sh
 17947 tty1        00:00:00 scrypt.sh
 18099 tty1        00:00:00 scrypt.sh
 18144 tty1        00:00:16 scrypt.sh
 18192 tty1        00:00:00 scrypt.sh
 18227 tty1        00:00:00 sleep
 18293 tty1        00:00:23 scrypt.sh
 18330 tty1        00:00:12 scrypt.sh
 19066 tty1        00:00:00 less
 19084 tty1        00:00:00 more
 20371 tty1        00:00:04 find
 20560 tty1        00:00:00 scrypt.sh
 20561 tty1        00:00:00 ps
```

Рисунок 51 – Выполнение скрипта

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.

Вес файла можно узнать с помощью команды `wc -c filename`.

```
#!/bin/bash
size=`wc -c < "$1"`
echo "Size: $size"
if [ "$size" -gt 0 ]
then
    sort -k1 $1 >> $2;
    less $2;
else
    echo "size of file is zero"
fi
```

Рисунок 52 – Код скрипта

```
lovediehatemyubuntuserver:~$ ./scrypt.sh emptyfile sortedfile
Size: 0
size of file is zero
```

Рисунок 53 – Выполнение скрипта

Первый файл пуст, поэтому его нельзя отсортировать.

```
lovediehate@myubuntuserver:~$ cat < sortedfile
h
g
f
d
b
d
c
a
f
g
1
lovediehate@myubuntuserver:~$ ./scrypt.sh sortedfile new_empty_file
22
1
a
b
c
d
d
d
f
f
g
g
h
new_empty_file (END)_
```

Рисунок 54 – Выполнение скрипта

Файл sortedfile имеет вес 22, поэтому он ввёлся в отсортированном виде в файл new_empty_file. Содержимое второго файла отображено.

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.

```
list= ls
string="";

for i in $list; do
    A=`file -b "$PWD/$i"`
    if [ "$A" = "ASCII text" ]
    then
        string="$string $i"
    fi
done
tar --totals -cvf my.tar $string
sleep 2
tar -tf my.tar
```

Рисунок 55 – Код скрипта

Цикл проходит по списку всех файлов и создаёт список текстовых файлов посредством конкатенации `string="$string $i"`. Затем этот список используется для архивирования.

`tar --totals -cvf my.tar $string` // Архивирование с выводом итоговой информации завершенного процесса (`--totals`), с созданием архива my.tar (`-c`), выводом подробной информации процесса (`-v`) и выводом результата в файл (`-f`).

```
lovediehatemyubuntuserver:~$ rm my.tar
rm: cannot remove 'my.tar': No such file or directory
lovediehatemyubuntuserver:~$ ./script.sh
H
I
Somebody
emptyfile
new_empty_file
somebody.out.log
sortedfile
tmpfile
Total bytes written: 20480 (20KiB, 12MiB/s)
H
I
Somebody
emptyfile
new_empty_file
somebody.out.log
sortedfile
tmpfile
```

Рисунок 56 – Выполнение программы

На рисунке 56 видно, что архива не было до запуска скрипта. Он успешно создался в скрипте и заполнился текстовыми файлами.

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

Синтаксис создания функции очень прост:

имя_функции() { список_команд }

```
read -p "write A: " A;
read -p "write B: " B;
summ ()
{
sum=`echo "$A + $B" |bc`;
echo $sum;
}
summ
exit 0
```

Рисунок 57 – Код

В функции sum выполняется сложение двух введенных переменных $A+B$ с помощью калькулятора `bc` и вывод результата.

В предпоследней строке (рис. 57) осуществляется вызов функции `summ`, а в последней осуществляется завершение работы сценария `exit 0`.

```
lovediehatemyubuntuserver:~$ ./script.sh
write A: 4
write B: 2
6
```

Рисунок 58 – Выполнение скрипта

Вывод

Я изучил основные возможности языка программирования Shell с целью автоматизации процесса администрирования системы за счёт написания и использования командных файлов.