

# **Отчёта по лабораторной работе 11**

**Программирование в командном процессоре ОС UNIX. Ветвления и  
циклы**

Абд эль хай мохамад

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
2.1	№1 . . . . .	6
2.2	№2 . . . . .	6
2.3	№3 . . . . .	7
2.4	№4 . . . . .	7
<b>3</b>	<b>Теоретическое введение</b>	<b>8</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	№1 . . . . .	9
4.2	№2 . . . . .	11
4.2.1	Программный код на С . . . . .	11
4.2.2	Bash код программы . . . . .	12
4.3	№3 . . . . .	13
4.4	№4 . . . . .	14
<b>5</b>	<b>Выводы</b>	<b>16</b>
	<b>Список литературы</b>	<b>17</b>

## Список иллюстраций

4.1	-С . . . . .	10
4.2	без -С . . . . .	11
4.3	задание2 . . . . .	12
4.4	создание 10 файлов . . . . .	13
4.5	каталог для работы. . . . .	14
4.6	сжатие файлов . . . . .	15

## **Список таблиц**

# 1 Цель работы

Изучил основы программирования оболочки UNIX. Писал сложные файлы bash, используя логические управляющие структуры и циклы.

## 2 Задание

### 2.1 №1

Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами:

- `-i inputfile` — прочитать данные из указанного файла
- `-o outputfile` — вывести данные в указанный файл
- `-r шаблон` — указать шаблон для поиска
- `-C` различать большие и малые буквы
- `-n` выдавать номера строк.

а затем ищет в указанном файле нужные строки, определяемые ключом `-r`

### 2.2 №2

Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку.

Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено

## 2.3 №3

Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до  $\infty$  (например 1.tmp, 2.tmp, 3.tmp, 4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).

## 2.4 №4

Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).

## 3 Теоретическое введение

Bash — это текстовый файл, содержащий набор команд. Любую команду, которую можно выполнить в терминале, можно поместить в сценарий Bash. Любая серия команд, которые должны быть выполнены в терминале, может быть записана в текстовом файле в таком порядке, как сценарий Bash.



## 4 Выполнение лабораторной работы

### 4.1 №1

Использовал команду `getopts` и `grep`. С помощью команды `getopts` я создал аргументы команды, которые позволяют пользователю взаимодействовать с программой и указывать различные параметры.

`grep` использовался для выполнения поиска.

```
#!/bin/bash
```

```
while getopts i:o:p:nC option
```

```
do case $option in
```

```
    i) iflag=1; ival=$OPTARG;;#input from
```

```
    o) oflag=1; oval=$OPTARG;;#output to
```

```
    p) pflag=1; pval=$OPTARG;;#pattren to seach for "example"
```

```
    C) Cflag=1;;# -i from grep
```

```
    n) nflag=1;;#show line using wc
```

```
    *) echo "Error" && exit 1
```

```
esac
```

```
done
```

```
# -z check if zero
```

```
# ! -z check if not zero
```

```
# [[ ]] work as test command

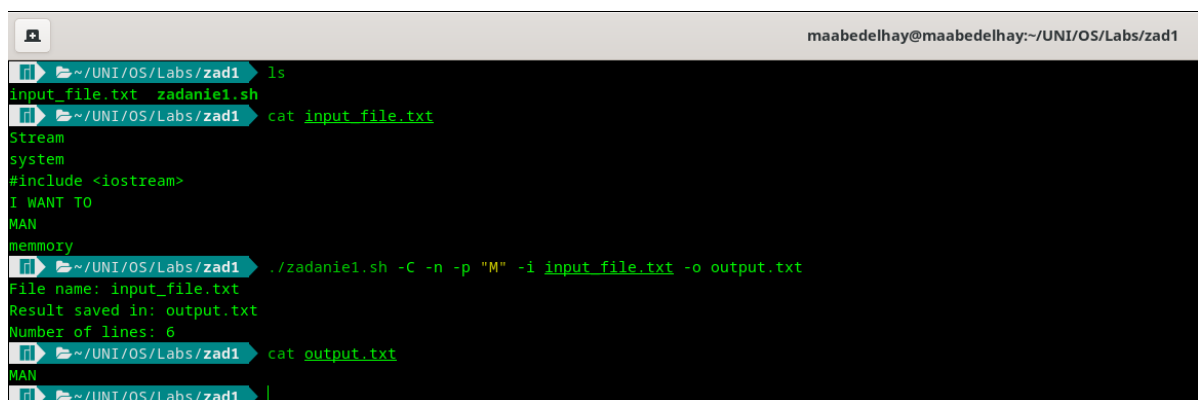
[[ ! -z $iflag ]] && echo "File name: ${ival}"
[[ ! -z $oflag ]] && echo "Result saved in: ${oval}"

# wc print newline, word, and byte counts for each file
n=$(wc -l $ival | awk '{print $1}' )
[[ ! -z $nflag ]] && echo "Number of lines: ${n}"

if [[ ! -z $Cflag ]]
then
    grep "$pval" $ival > $oval
else
    i="-i"
    grep $i "$pval" $ival > $oval
fi

shift $(( $OPTIND - 1 ))
```

Я специально показывал данные вменяемого файла, чтобы было понятно, какой вывод ожидать.



```
maabedelhay@maabedelhay:~/UNI/OS/Labs/zad1
~/UNI/OS/Labs/zad1$ ls
input_file.txt  zadanie1.sh
~/UNI/OS/Labs/zad1$ cat input_file.txt
Stream
system
#include <iostream>
I WANT TO
MAN
memory
~/UNI/OS/Labs/zad1$ ./zadanie1.sh -C -n -p "M" -i input_file.txt -o output.txt
File name: input_file.txt
Result saved in: output.txt
Number of lines: 6
~/UNI/OS/Labs/zad1$ cat output.txt
MAN
~/UNI/OS/Labs/zad1$
```

Рис. 4.1: -C

когда -C включен, программа будет чувствительна к регистру. Как и в приведенном выше примере, шаблон -p "М", и он выводит только слова с большой буквы М.

```
memory
~/UNI/OS/Labs/zad1 ./zadanie1.sh -C -n -p "M" -i input_file.txt -o output.txt
File name: input_file.txt
Result saved in: output.txt
Number of lines: 6
~/UNI/OS/Labs/zad1 cat output.txt
MAN
~/UNI/OS/Labs/zad1 ./zadanie1.sh -n -p "M" -i input_file.txt -o output.txt
File name: input_file.txt
Result saved in: output.txt
Number of lines: 6
~/UNI/OS/Labs/zad1 cat output.txt
Stream
system
#include <iostream>
MAN
memory
~/UNI/OS/Labs/zad1 |
```

Рис. 4.2: без -C

когда -C отключен, программа не будет учитывать регистр. И выведите все строки, содержащие букву m как в заглавной, так и в строчной букве.

## 4.2 №2

Простая программа C для проверки введенного числа, если оно меньше, больше или равно 0. Отправьте введенное число с помощью функции exit () «exit (input number)». Файл Bash прочитает код выхода программы C и распечатает его. Это оно.

### 4.2.1 Программный код на C

```
#include <stdio.h>
#include <stdlib.h>
int main()
```

```

{
    int n;
    printf("Enter number: \n");
    scanf("%d",&n);

    if(n>0){printf("Larger then 0 \n"); }
    else if(n<0){printf("Less then 0 \n"); }
    else {printf("It is 0\n");}
    exit(n);
    return 0;
}

```

## 4.2.2 Bash код программы

```

#!/bin/bash
gcc main.c
./a.out
n=$(echo "$?")
echo "Bash says: the number was entered is ${n}"

```

The screenshot shows a terminal window with the following commands and output:

```

~/UNI/OS/Labs/zad2 ➤ ls
a.out  main.c  zadanie2.sh
~/UNI/OS/Labs/zad2 ➤ ./zadanie2.sh
Enter number:
55
Larger then 0
Bash says: the number was entered is 55
~/UNI/OS/Labs/zad2 ➤

```

Рис. 4.3: задание2

## 4.3 №3

Создан сценарий bash, который будет принимать два аргумента: имя файла и количество файлов для создания. Если файлы уже существуют, они будут удалены, а если нет, будут созданы.

```
#!/bin/bash
fname=$1
n=0
if [[ -f ${fname}_1 ]]
then
    rm ${fname}_*
else
    while [ $n -lt $2 ] # lt = less than
    do
        ((n=n+1))#increment by 1
        touch ${fname}_${n}
    done
fi
```

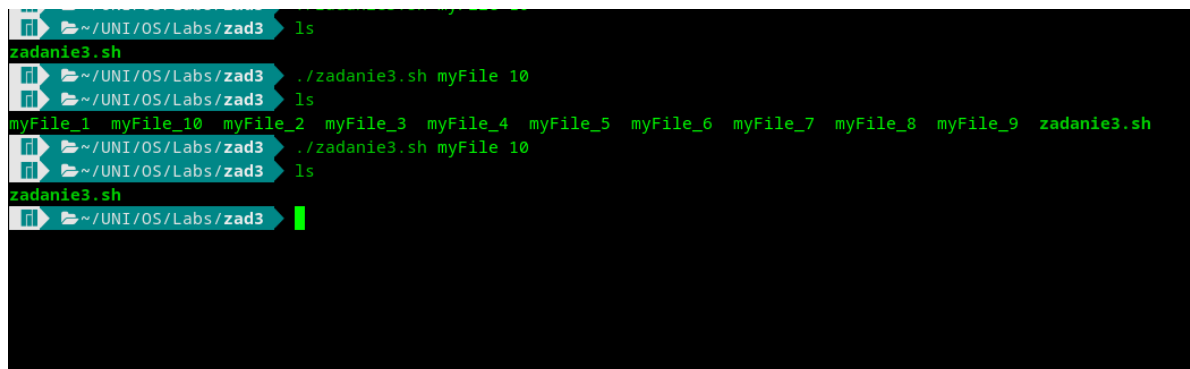
A terminal window with a dark background and light blue text. The prompt is '~ /UNI/OS/Labs/zad3'. The user enters 'ls', showing an empty directory. Then they enter './zadanie3.sh myFile 10'. After another 'ls', the directory now contains files 'myFile\_1' through 'myFile\_10' and the script 'zadanie3.sh'. Finally, they enter './zadanie3.sh myFile 10' again, and after a third 'ls', the directory is empty again, demonstrating the script's ability to delete existing files.

Рис. 4.4: создание 10 файлов

Как вы можете видеть на скриншоте выше. Я перечислил содержимое текущего каталога, чтобы убедиться, что он пуст. Выполнил программу, создающую 10 файлов. а затем снова запустил программу, она удалила созданные файлы.

## 4.4 №4

Команда tar будет сжимать файлы из указанного каталога. Только файлы, которые последний раз изменялись более 7 дней.

```
#!/bin/bash
dir=$1
title=$(echo "${dir}" | awk -F "/" '{print $NF}' )
arr=$(cd $dir && find -atime +7 | awk -F "/" '{print $2}' )
cd $dir && tar -czvf ${title}.tar.gz ${arr[@]}
```



Рис. 4.5: каталог для работы.

На этом снимке экрана я показал каталог, с которым мы будем работать, это `~/module-3`. Затем файлы, которые в последний раз изменялись более 7 дней, поэтому мы можем ожидать, какие файлы должны быть сжаты. Таким образом, будет ясно, нормально ли программа выполнялась или нет.

```

~/UNI/computer-modeuling/module-3 cd ~/UNI/OS/Labs
~/UNI/OS/Labs cd zad4
~/UNI/OS/Labs/zad4 ./zadanie4.sh ~/UNI/computer-modeuling/module-3
lab02/
lab02/НПИбд-01-21_Абд_эль_хай_мохамад-_лаб-2_Вычисление-_числа_ПИ.zip
lab02/
lab02/НПИбд-01-21_Абд_эль_хай_мохамад-_лаб-2_Вычисление-_числа_ПИ.zip
re-lab02/
re-lab02/НПИбд-01-21_Абд_эль_хай_мохамад-_лаб-2_Вычисление-_числа_ПИ.zip
re-lab02/
re-lab02/НПИбд-01-21_Абд_эль_хай_мохамад-_лаб-2_Вычисление-_числа_ПИ.zip
lab01/
lab01/НПИбд-01-21_Абд_эль_хай_мохамад_лаб-1.docx
lab01/НПИбд-01-21_Абд_эль_хай_мохамад_лаб-1_quick-sort.py
lab01/
lab01/НПИбд-01-21_Абд_эль_хай_мохамад_лаб-1.docx
lab01/НПИбд-01-21_Абд_эль_хай_мохамад_лаб-1_quick-sort.py
lab01/
lab01/НПИбд-01-21_Абд_эль_хай_мохамад_лаб-1.docx
lab01/НПИбд-01-21_Абд_эль_хай_мохамад_лаб-1_quick-sort.py
lab03/
lab03/НПИбд-01-21_Абд_эль_хай_мохамад-_лаб-3_алгоритм_spigot.docx
lab03/НПИбд-01-21_Абд_эль_хай_мохамад-_лаб-3_алгоритм_spigot.py
lab03/
lab03/НПИбд-01-21_Абд_эль_хай_мохамад-_лаб-3_алгоритм_spigot.docx
lab03/НПИбд-01-21_Абд_эль_хай_мохамад-_лаб-3_алгоритм_spigot.py
lab03/
lab03/НПИбд-01-21_Абд_эль_хай_мохамад-_лаб-3_алгоритм_spigot.docx
lab03/НПИбд-01-21_Абд_эль_хай_мохамад-_лаб-3_алгоритм_spigot.py
lab04/
lab04/НПИбд-01-21_Абд_эль_хай_мохамад_лаб-4_устроены_поисковые_системы.py
lab04/НПИбд-01-21_Абд_эль_хай_мохамад_лаб-4_устроены_поисковые_системы.docx
lab04/
lab04/НПИбд-01-21_Абд_эль_хай_мохамад_лаб-4_устроены_поисковые_системы.py
lab04/НПИбд-01-21_Абд_эль_хай_мохамад_лаб-4_устроены_поисковые_системы.docx
lab04/
lab04/НПИбд-01-21_Абд_эль_хай_мохамад_лаб-4_устроены_поисковые_системы.py
lab04/НПИбд-01-21_Абд_эль_хай_мохамад_лаб-4_устроены_поисковые_системы.docx
~/UNI/OS/Labs/zad4 ls ~/UNI/computer-modeuling/module-3
lab01 lab02 lab03 lab04 module-3.tar.gz re-lab02
~/UNI/OS/Labs/zad4

```

Рис. 4.6: сжатие файлов

Программа выполнилась так, как мы и ожидали. сжал все файлы в файл .tar.gz, и этот файл был создан в том же каталоге.

## 5 Выводы

Bash — очень мощный инструмент. Приобретение сценария bash в качестве навыка повысит производительность и добавит ценности любому работнику в технологической отрасли.



## **Список литературы**