

# **Отчёт по лабораторной работе №13**

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

Гасанов Казимагомед Тимурланович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Вывод</b>	<b>11</b>
<b>4</b>	<b>Контрольные вопросы</b>	<b>12</b>

# Список иллюстраций

2.1	Задание 1 . . . . .	7
2.2	Задание 2 . . . . .	8
2.3	Задание 3 . . . . .	9
2.4	Задание 4 . . . . .	10

## **Список таблиц**

# 1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 2 Выполнение лабораторной работы

1. Используя команды `getopts` `grep` напишем командный файл, который анализирует командную строку с ключами и выполним его: `-i inputfile` — прочитать данные из указанного файла; `-o outputfile` — вывести данные в указанный файл; `-p шаблон` — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк;

а затем ищет в указанном файле нужные строки

```
#!/bin/bash
cflag=0;
nflag=0;
while getopts i:o:p:C:n opt
do
case $opt in
i) ival=$OPTARG;;
o) oval=$OPTARG;;
p) pval=$OPTARG;;
C) cflag=1;;
n) nflag=1;;
esac
done
if [ $cflag -a $nflag ]
then
```

```

grep -n $pval $ival>$oval
elif test $cflag
then
grep $pval $ival>$oval
elif test $nflag
then
grep -n -i $pval $ival>$oval
else
grep -i $pval $ival>$oval
fi

```

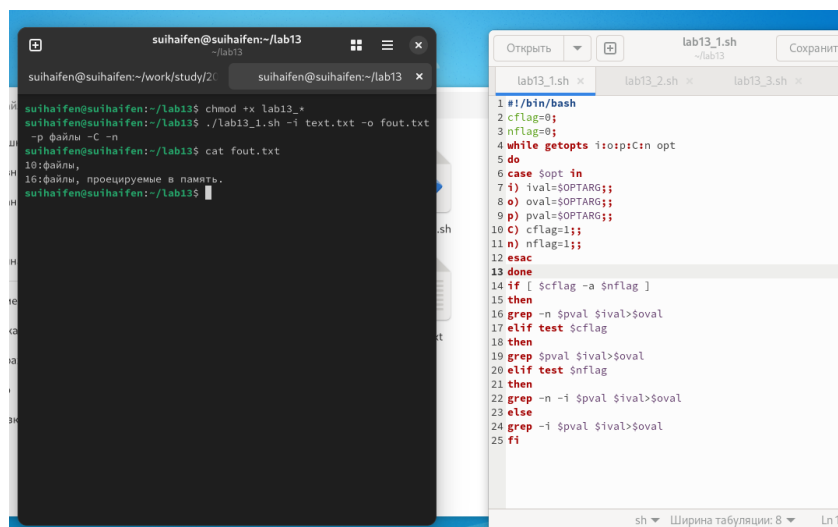


Рис. 2.1: Задание 1

2. Напишем сначала на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем завершим программу при помощи функции `exit(n)`, передавая информацию о коде завершения в оболочку. Командный файл вызовет эту программу и, проанализировав с помощью команды `$?`, выдаст сообщение о том, какое число было введено

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

```
gcc -c script2.c
gcc -o script2 script2.c
./script2
case $? in
    1) echo отрицательное;;
    2) echo равно нулю;;
    3) echo положительное;;
esac
```

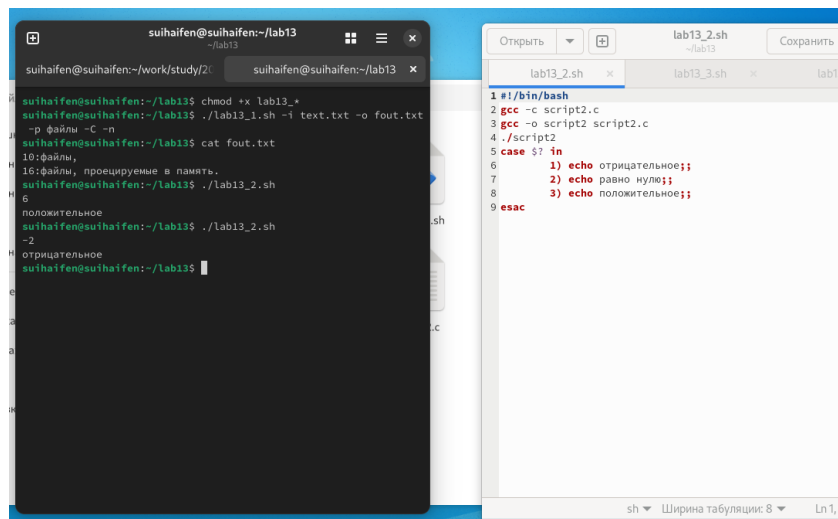


Рис. 2.2: Задание 2

3. Напишем командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N

```
#!/bin/bash
let i=$1+1
while (( i-=1 ))
do touch $i.tmp
done
let j=$2+1;
while (( j-=1 ))
```



```
do rm $j.tmp
done
```

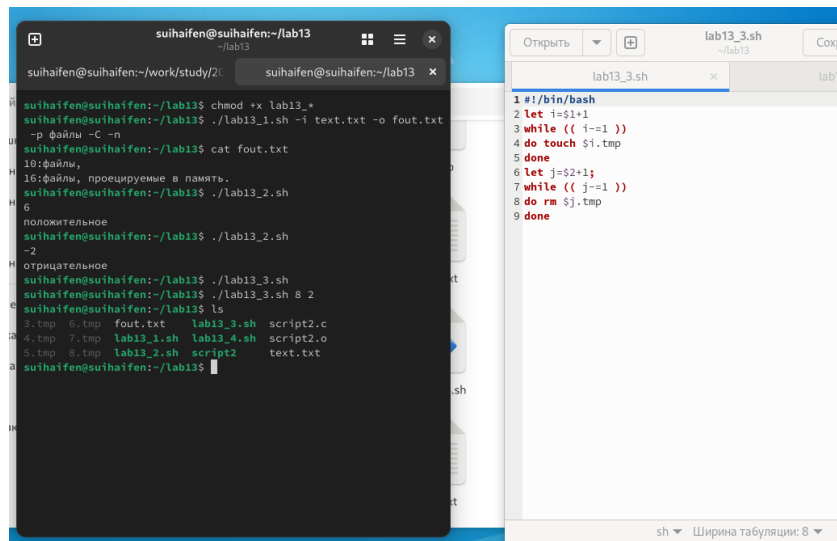


Рис. 2.3: Задание 3

4. Напишем командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории. Модифицируем его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад.

```
#!/bin/bash
(find $1 -mtime -7 -daystart) | xargs tar -cf arhiv.tar
```

The image shows a terminal window on the left and a file editor on the right. The terminal window has a title bar 'suihaifen@~ - /lab13' and shows the following commands and output:

```
suihaifen@suihaifen:~/work/study/2/
suihaifen@suihaifen:~/lab13$ chmod +x lab13.*
suihaifengsuihaifen:~/lab13$ ./lab13_1.sh -i text.txt -o fout.txt
-r файлы -C -n
suihaifengsuihaifen:~/lab13$ cat fout.txt
10:файлы,
15:файлы, проецируемые в память.
suihaifengsuihaifen:~/lab13$ ./lab13_2.sh
6
положительное
suihaifengsuihaifen:~/lab13$ ./lab13_2.sh
-2
отрицательное
suihaifengsuihaifen:~/lab13$ ./lab13_3.sh
suihaifengsuihaifen:~/lab13$ ./lab13_3.sh 8 2
suihaifengsuihaifen:~/lab13$ ls
3.tmp 6.tmp  fout.txt  lab13_3.sh  script2.c
4.tmp 7.tmp  lab13_1.sh  lab13_4.sh  script2.o
5.tmp 8.tmp  lab13_2.sh  script2    text.txt
suihaifengsuihaifen:~/lab13$ ./lab13_4.sh
tar: ./arhiv.tar: archive cannot contain itself; not dumped
suihaifengsuihaifen:~/lab13$ ls
3.tmp 6.tmp  arhiv.tar  lab13_2.sh  script2    text.txt
4.tmp 7.tmp  fout.txt   lab13_3.sh  script2.c
5.tmp 8.tmp  lab13_1.sh lab13_4.sh  script2.o
suihaifengsuihaifen:~/lab13$
```

The file editor on the right has a title bar 'lab13\_4.sh' and shows the following code:

```
1 #!/bin/bash
2 (find $1 -mtime -7 -daystart) | xargs tar -cf arhiv.tar
```

Рис. 2.4: Задание 4

## **3 Вывод**

В данной работе мы изучили основы программирования в оболочке ОС UNIX и писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 4 Контрольные вопросы

1. Каково предназначение команды `getopts`? Ответ: Создание по пользовательским аргументам.
2. Какое отношение метасимволы имеют к генерации имён файлов? Ответ: Используют как файлы так и аргументы.
3. Какие операторы управления действиями вы знаете? Ответ: `if`, `else`, `elif`, `fi`, `while`, `do`, `done`, `until`, `do`, `done`, `for`, `in`, `do`, `done`, `case`, `in`, `esac`
4. Какие операторы используются для прерывания цикла? Ответ:
  - a) `for` – будет выполнять действие до тех пор, пока есть объекты для выполнения.
  - b) `while` – выполняет действие до тех пор, пока условие является истинным.
  - c) `until` – будет выполняться пока условие не станет правдиво.
5. Для чего нужны команды `false` и `true`? Ответ: `until` – будет выполняться до тех пор, пока условие не станет `true`, т.е. пока оно не станет `false`.
6. Что означает строка `if test -f mans/i.$s`, встреченная в командном файле? Ответ: Проверяет если существует файл его размерность и тип с двумя разными расширениями, заменяя через переменные.

7. Объясните различия между конструкциями while и until. Ответ:

while – выполняет действие до тех пор, пока условие является истинным.

until – будет выполняться до тех пор, пока условие не станет истинным, т.е. пока оно false.