

# **Лабораторная работа №11**

**Операционные системы”**

Александрова Ульяна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выводы</b>	<b>13</b>

## Список иллюстраций

4.1	Создание файла . . . . .	8
4.2	Код 1ой программы . . . . .	9
4.3	Запуск 1ой программы . . . . .	9
4.4	Текстовый файл 1 . . . . .	9
4.5	Текстовый файл 2 . . . . .	10
4.6	Создание файла 2 . . . . .	10
4.7	Тело 1 . . . . .	10
4.8	Тело 2 . . . . .	10
4.9	Проверка работы 2 . . . . .	11
4.10	Создание файла 3 . . . . .	11
4.11	Код программы . . . . .	11
4.12	Проверка программы . . . . .	12
4.13	Создание файла 4 . . . . .	12
4.14	Проверка программы . . . . .	12

## **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 Задание

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-iinputfile` — прочитать данные из указанного файла; `-ooutputfile` — 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. 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.
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).
4. Написать командный файл, который с помощью команды `tag` 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

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

- оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций; - C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; - оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; - BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).

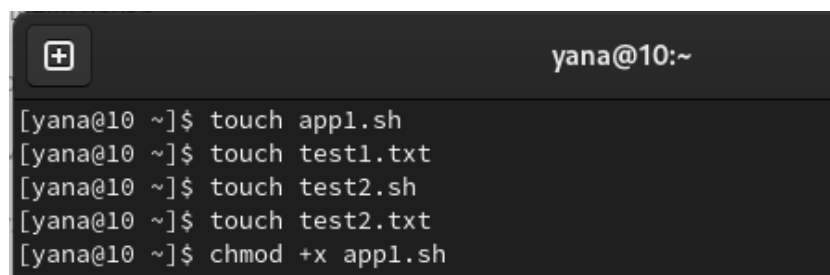
POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.

Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода.

POSIX-совместимые оболочки разработаны на базе оболочки Корна.

## 4 Выполнение лабораторной работы

Создала файл `app1.sh` для первой программы и дополнительные текстовые файлы (рис. fig. 4.1).

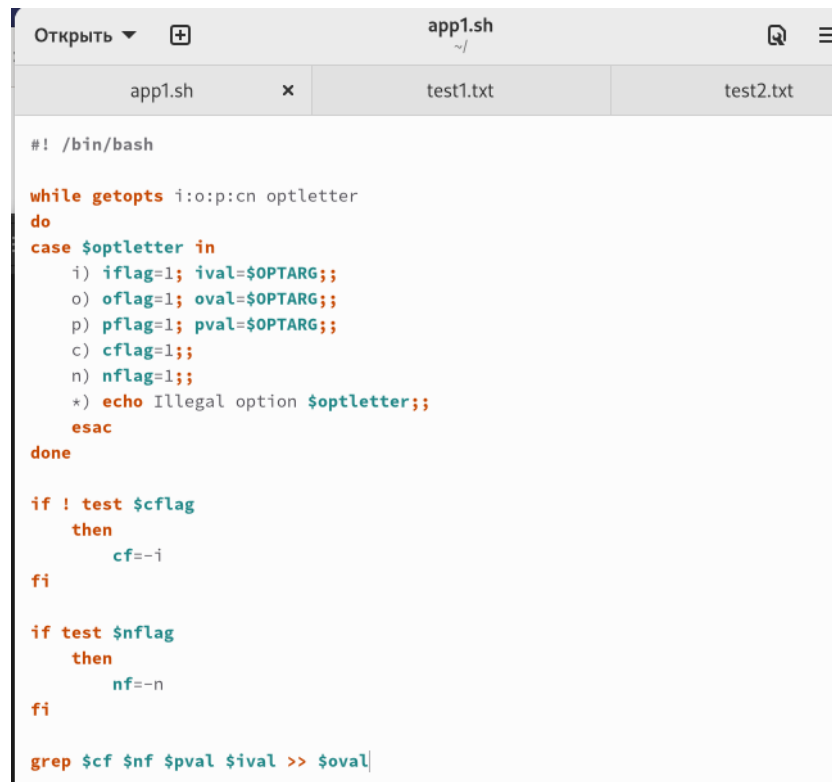
A terminal window with a dark background. The title bar shows a plus icon and the text 'yana@10:~'. The terminal contains five lines of commands and their outputs:

```
[yana@10 ~]$ touch app1.sh
[yana@10 ~]$ touch test1.txt
[yana@10 ~]$ touch test2.sh
[yana@10 ~]$ touch test2.txt
[yana@10 ~]$ chmod +x app1.sh
```

Рис. 4.1: Создание файла

Заполнила код (рис. fig. 4.2).





```
#!/bin/bash

while getopts i:o:p:cn optletter
do
case $optletter in
i) iflag=1; ival=$OPTARG;;
o) oflag=1; oval=$OPTARG;;
p) pflag=1; pval=$OPTARG;;
c) cflag=1;;
n) nflag=1;;
*) echo Illegal option $optletter;;
esac
done

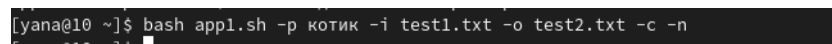
if ! test $cflag
then
cf=-i
fi

if test $nflag
then
nf=-n
fi

grep $cf $nf $pval $ival >> $oval
```

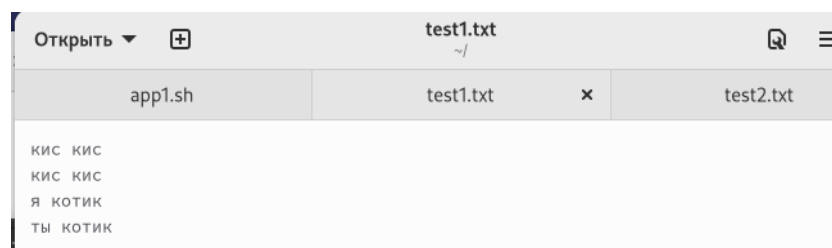
Рис. 4.2: Код 1ой программы

Запустила программу (рис. fig. 4.3). Она работает (рис. fig. 4.4), (рис. fig. 4.5).



```
[yana@i0 ~]$ bash app1.sh -p котик -i test1.txt -o test2.txt -c -n
```

Рис. 4.3: Запуск 1ой программы



```
КИС КИС
КИС КИС
Я КОТИК
ТЫ КОТИК
```

Рис. 4.4: Текстовый файл 1

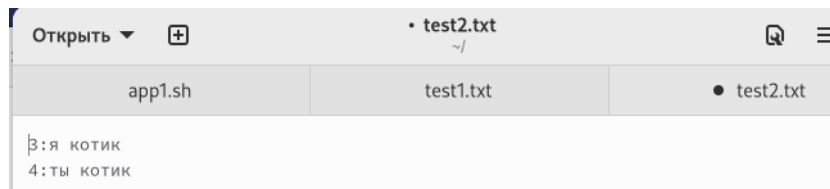


Рис. 4.5: Текстовый файл 2

Я создала второй командный файл, а также дополнительный файл (рис. fig. 4.6).

```
[yana@10 ~]$ touch app2.sh
[yana@10 ~]$ touch app2.c
[yana@10 ~]$ chmod +x app2.sh
```

Рис. 4.6: Создание файла 2

Далее я написала тело обеих программ (рис. fig. 4.7), (рис. fig. 4.8).

 A screenshot of a code editor window. The title bar shows 'Открыть' (Open) and a file named 'app2.sh' in the home directory. The editor has two tabs: 'app2.sh' and 'app2.c'. The 'app2.sh' tab is active, showing the following shell script code:
 

```
#!/bin/bash

gcc -o cprog app2.c
./cprog
case $? in
0) echo "N = 0";;
1) echo "N > 0";;
2) echo "N < 0";;
esac
```

Рис. 4.7: Тело 1

 A screenshot of a code editor window. The title bar shows 'Открыть' (Open) and a file named 'app2.c' in the home directory. The editor has two tabs: 'app2.sh' and 'app2.c'. The 'app2.c' tab is active, showing the following C program code:
 

```
#include <stdlib.h>
#include <stdio.h>

int main () {
    int n;
    printf ("Enter the number: ");
    scanf ("%d", &n);
    if (n > 0) { exit(1); }
    else if (n == 0) { exit(0); }
    else { exit(2); }
}
```

Рис. 4.8: Тело 2

Проверила работу приложения (рис. fig. 4.9).

```
[yana@10 ~]$ bash app2.sh
Enter the number: 12
N > 0
[yana@10 ~]$ bash app2.sh
Enter the number: 0
N = 0
[yana@10 ~]$ bash app2.sh
Enter the number: -2
N < 0
```

Рис. 4.9: Проверка работы 2

Создала третий файл app3.sh (рис. fig. 4.10).

```
yana@10:~
[yana@10 ~]$ touch app3.sh
[yana@10 ~]$ chmod +x app3.sh
[yana@10 ~]$ bash app3.sh 3
```

Рис. 4.10: Создание файла 3

Записала код программы (рис. fig. 4.11).

```
Открыть ▾ + app3.sh ~/
#!/bin/bash

for ((i=1; i<=$*; i++))
do
if test -f "$i".tmp
then rm "$i".tmp
else touch "$i.tmp"
fi
done
```

Рис. 4.11: Код программы

Проверила работу кода. Новые файлы создались (рис. fig. 4.12).

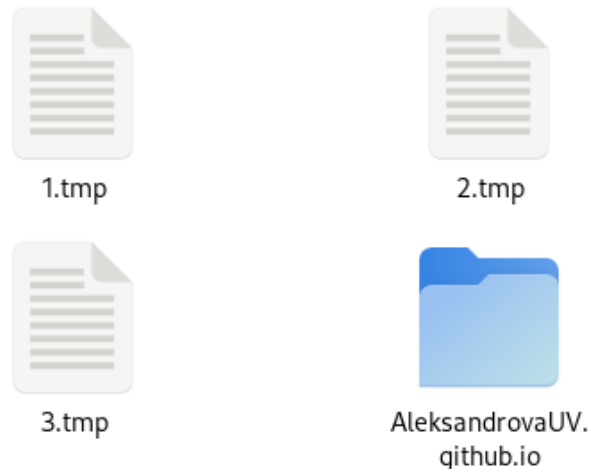


Рис. 4.12: Проверка программы

Я создала последний командный файл app4.sh и заполнила файл (рис. fig. 4.13).



Рис. 4.13: Создание файла 4

Далее запустила приложение. Оно работает исправно (рис. fig. 4.14).

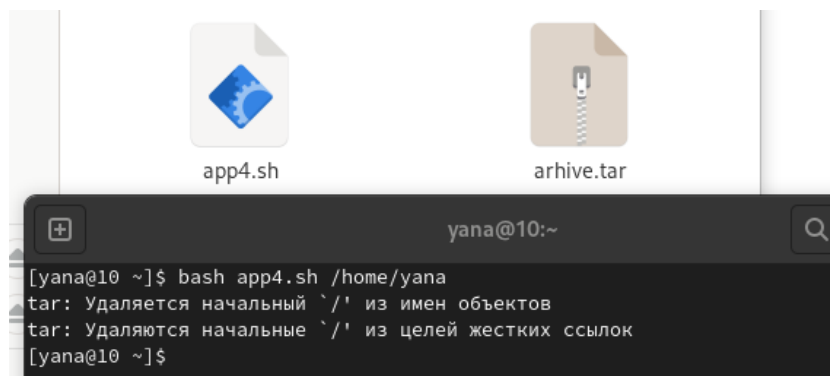


Рис. 4.14: Проверка программы

## 5 Выводы

Я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

...