

# **Лабораторная работа №11**

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

Чемоданова Ангелина Александровна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>5</b>	<b>Выводы</b>	<b>13</b>
<b>6</b>	<b>Контрольные вопросы</b>	<b>14</b>

## Список иллюстраций

4.1	Скрипт 1 . . . . .	7
4.2	Выполнение скрипта 1 . . . . .	8
4.3	Вводимый файл . . . . .	8
4.4	Выводимый файл . . . . .	9
4.5	Скрипт 2 си . . . . .	9
4.6	Скрипт 2 . . . . .	10
4.7	Выполнение скрипта 2 . . . . .	10
4.8	Скрипт 3 . . . . .	11
4.9	Выполнение скрипта 3 . . . . .	11
4.10	Скрипт 4 . . . . .	12
4.11	Выполнение скрипта 4 . . . . .	12

# 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 N (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`).

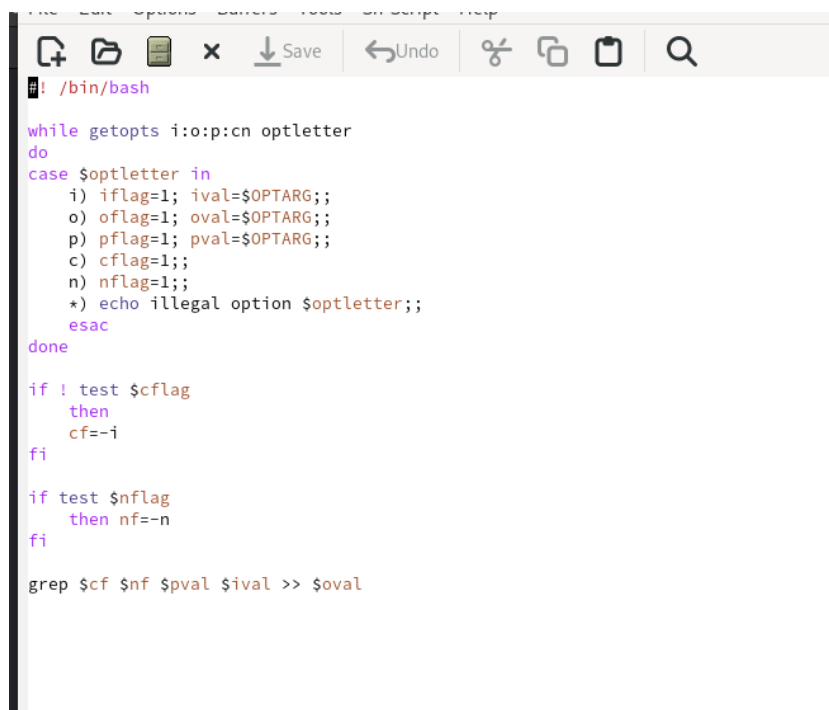
### 3 Теоретическое введение

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.

## 4 Выполнение лабораторной работы

Используя команды `getopts` `grep`, напомним командный файл, который анализирует командную строку с ключами: `-i` — прочитать данные из указанного файла; `-o` — вывести данные в указанный файл; `-r` — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-p`.

Скрипт 1(рис. 4.1).



```
#!/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

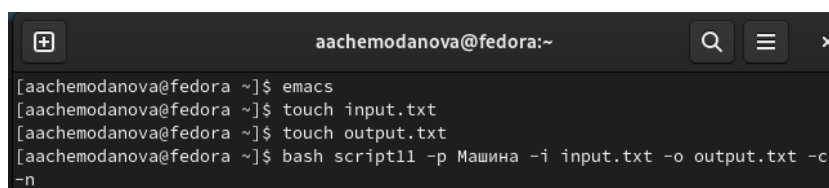
if ! test $cflag
then
cf=-i
fi

if test $nflag
then nf=-n
fi

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

Рис. 4.1: Скрипт 1

Выполнение скрипта 1(рис. 4.2).

A terminal window titled 'aachemodanova@fedora:~' with search, menu, and close icons. It shows the execution of a script: '[aachemodanova@fedora ~]\$ emacs', '[aachemodanova@fedora ~]\$ touch input.txt', '[aachemodanova@fedora ~]\$ touch output.txt', and '[aachemodanova@fedora ~]\$ bash script11 -p Машина -i input.txt -o output.txt -c -n'.

```
aachemodanova@fedora:~  
[aachemodanova@fedora ~]$ emacs  
[aachemodanova@fedora ~]$ touch input.txt  
[aachemodanova@fedora ~]$ touch output.txt  
[aachemodanova@fedora ~]$ bash script11 -p Машина -i input.txt -o output.txt -c  
-n
```

Рис. 4.2: Выполнение скрипта 1

Вводимый файл.(рис. 4.3).

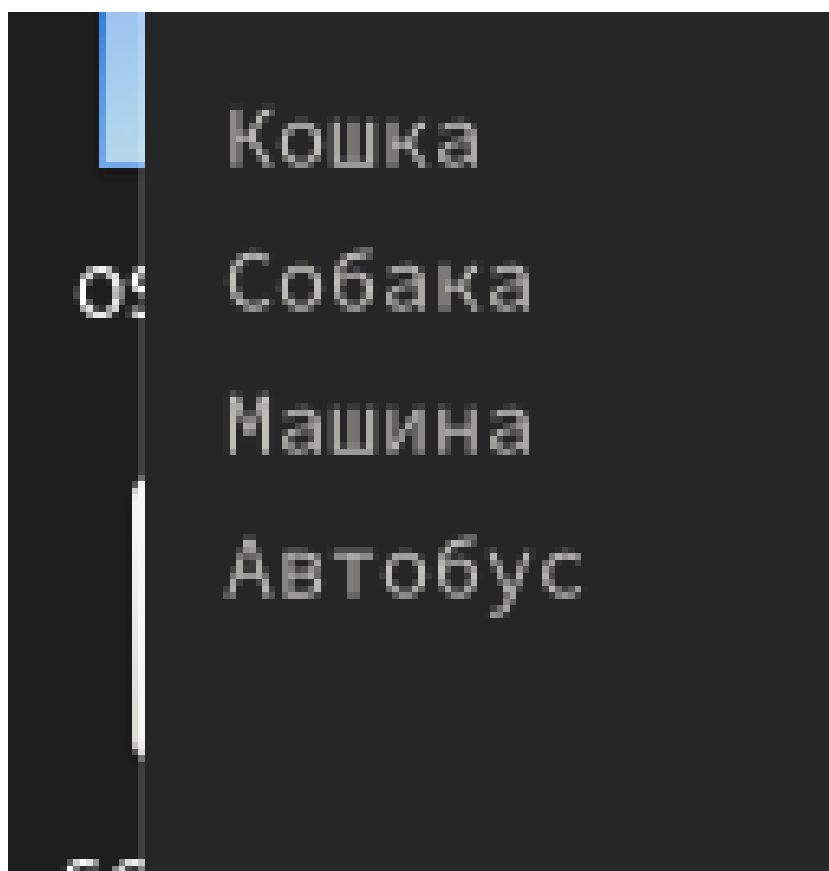


Рис. 4.3: Вводимый файл

Выводимый файл. (рис. 4.4).



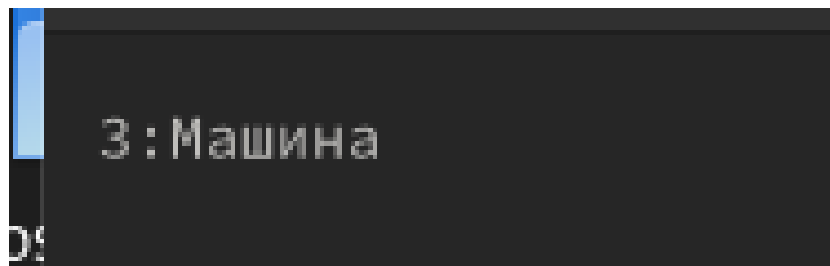


Рис. 4.4: Выводимый файл

Напишем 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.

Скрипт 2 си (рис. 4.5).

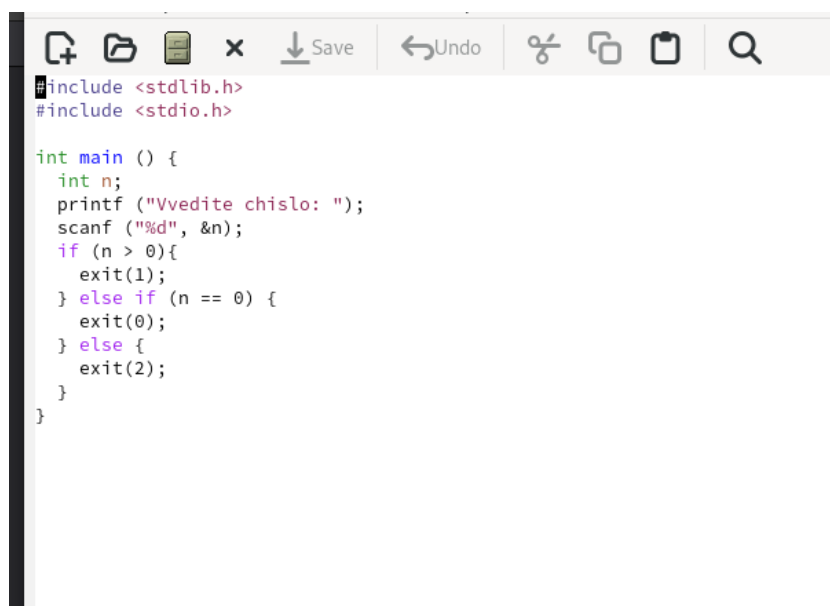
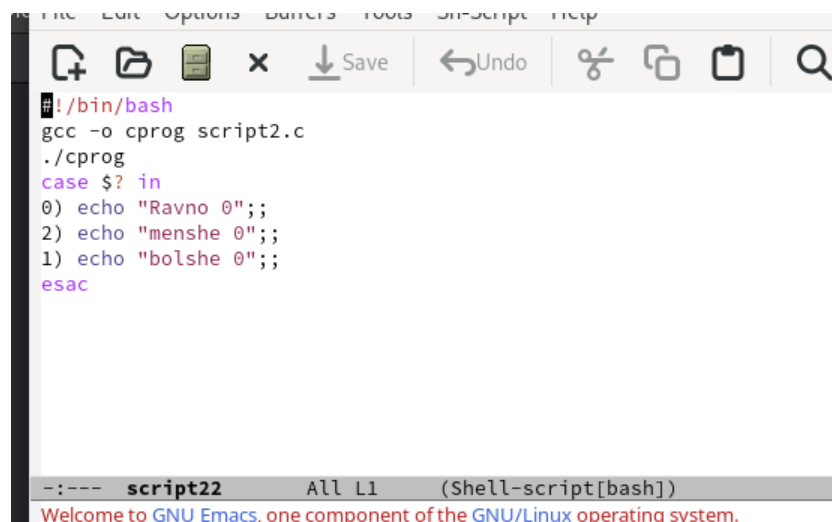


Рис. 4.5: Скрипт 2 си

Скрипт 2 (рис. 4.6).

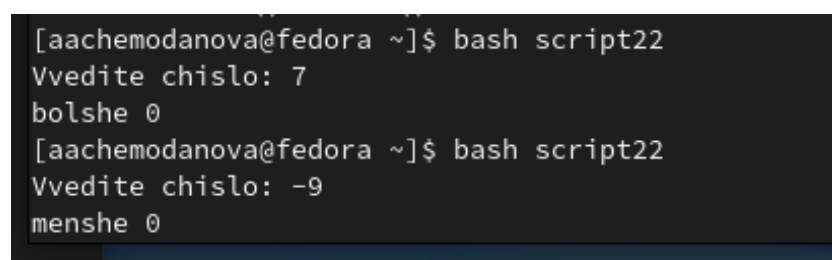


```
#!/bin/bash
gcc -o cprog script2.c
./cprog
case $? in
0) echo "Ravno 0";;
2) echo "menshe 0";;
1) echo "bolshe 0";;
esac
```

--- script22 All L1 (Shell-script[bash])  
Welcome to GNU Emacs, one component of the GNU/Linux operating system.

Рис. 4.6: Скрипт 2

Выполнение скрипта 2(рис. 4.7).



```
[aachemodanova@fedora ~]$ bash script22
Vvedite chislo: 7
bolshe 0
[aachemodanova@fedora ~]$ bash script22
Vvedite chislo: -9
menshe 0
```

Рис. 4.7: Выполнение скрипта 2

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

Скрипт 3 (рис. 4.8).

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

--:--- **script33** All L1 (Shell-script[bash])  
Welcome to GNU Emacs, one component of the GNU/Linux operating system.

Рис. 4.8: Скрипт 3

Выполнение скрипта 3(рис. 4.9).

```
script33: строка 8: синтаксическая ошибка: неожиданный конец файла
[aachemodanova@fedora ~]$ bash script33 3
[aachemodanova@fedora ~]$ ls
1.tmp      input.txt      script11      script2.c      work          Музыка
2.tmp      lab07.sh~     script11~     script2.c~     Видео         Общедоступные
3.tmp      my_os         '#script2#'   script33       Документы    'Рабочий стол'
backup     os~intro      script22      script33~     Загрузки     Шаблоны
cprog      output.txt    script22~     '~work'       Изображения
[aachemodanova@fedora ~]$ bash script33 3
[aachemodanova@fedora ~]$ ls
backup     os~intro      script22      script33~     Загрузки     Шаблоны
cprog      output.txt    script22~     '~work'       Изображения
input.txt  script11      script2.c     work          Музыка
lab07.sh~ script11~     script2.c~    Видео         Общедоступные
my_os     '#script2#'   script33      Документы    'Рабочий стол'
```

Рис. 4.9: Выполнение скрипта 3

Напишем 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).

Скрипт 4 (рис. 4.10).



Рис. 4.10: Скрипт 4

Выполнение скрипта 4(рис. 4.11).

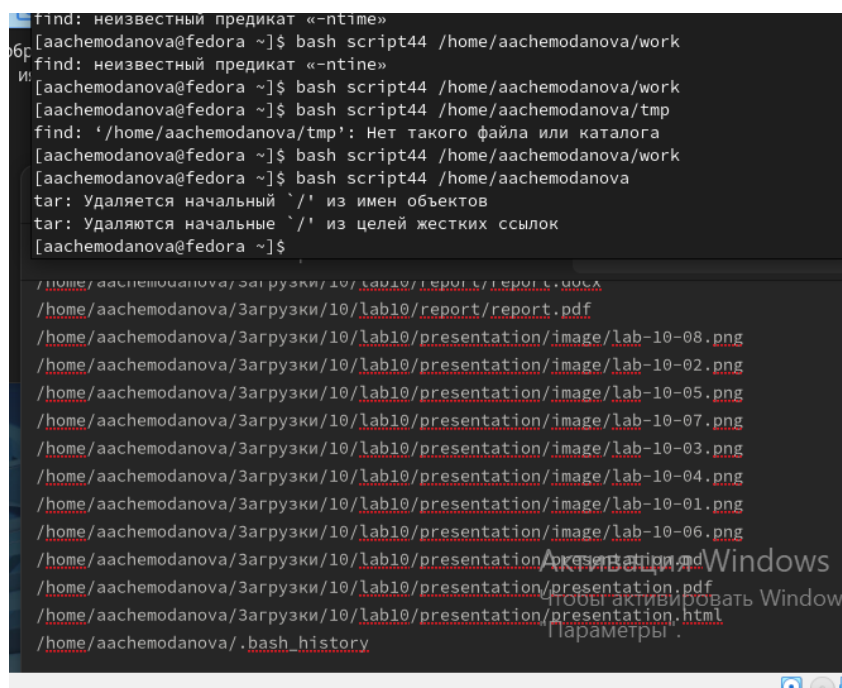


Рис. 4.11: Выполнение скрипта 4

## 5 Выводы

Мы изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 6 Контрольные вопросы

### 1. Каково предназначение команды getoptс?

Весьма необходимой при программировании является команда `getopts`, которая осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ...]` Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Предположим, необходимо распознать командную строку следующего формата: `testprog -infile_in.txt -ofile_out.doc -L -t -r` Вот как выглядит использование оператора `getopts` в этом случае: 

```
while getopts o:i:Ltr optletter do
case
optletter in
o) oflag = 1; oval = OPTARG;;
i) iflag=1; ival=$OPTARG;;
L) Lflag=1;;
t) tflag=1;;
r) rflag=1;;
*) echo Illegal option $optletter
esac
done
```

 Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента (будет равна `file_in.txt` для опции `i` и `file_out.doc` для опции `o`). `OPTIND`

является числовым индексом на упомянутый аргумент. Функция `getopts` также понимает переменные типа массив, следовательно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

## 2. Какое отношение метасимволы имеют к генерации имён файлов?

При перечислении имен файлов текущего каталога можно использовать следующие символы:

- `*` — соответствует произвольной, в том числе и пустой строке;
- `?` — соответствует любому одному символу;
- `[c1-c1]` — соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`.

`echo *` — выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`;

- `ls .c` — выведет все файлы с последними двумя символами, равными `.c`.
- `echo prog.?` — выдаст все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.`
- `[a-z]` — соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

## 3. Какие операторы управления действиями вы знаете?

Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости от результатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет Вам возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути дела являются

операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда.

#### 4. Какие операторы используются для прерывания цикла?

Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестает быть правильным. Пример бесконечного цикла `while`, с прерыванием в момент, когда файл перестает существовать: `while true do if [! -f $file] then break fi sleep 10 done`.

#### 5. Для чего нужны команды `false` и `true`?

`true` : всегда возвращает 0 в качестве кода выхода.

`false` : всегда возвращает 1 в качестве кода выхода.

#### 6. Что означает строка `if test -f mans/i.$s`, встреченная в командном файле?

Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения. Введенная строка означает условие существования файла `mans/i.$s`

#### 7. Объясните различия между конструкциями `while` и `until`?

Если речь идет о 2-х параллельных действиях, то это `while`. когда мы показываем, что сначала делается 1-е действие. потом оно заканчивается при наступлении 2-го действия, применяем `until`.