

# **Лабораторная работа №11**

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

Гисматуллин Артём НПИбд-01-22

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>4</b>	<b>Контрольные вопросы</b>	<b>12</b>
<b>5</b>	<b>Выводы</b>	<b>15</b>
	<b>Список литературы</b>	<b>16</b>

## Список иллюстраций

3.1	Emacs. Командный файл №1 . . . . .	8
3.2	Командная строка. Исполнение командного файла №1 . . . . .	9
3.3	Emacs. Код на языке Си . . . . .	10
3.4	Командная строка. Командный файл №2 и проверка работоспособности . . . . .	10
3.5	Командная строка. Исполнение командного файла №3 . . . . .	11
3.6	Командная строка. Исполнение командного файла №4 . . . . .	11

## **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.

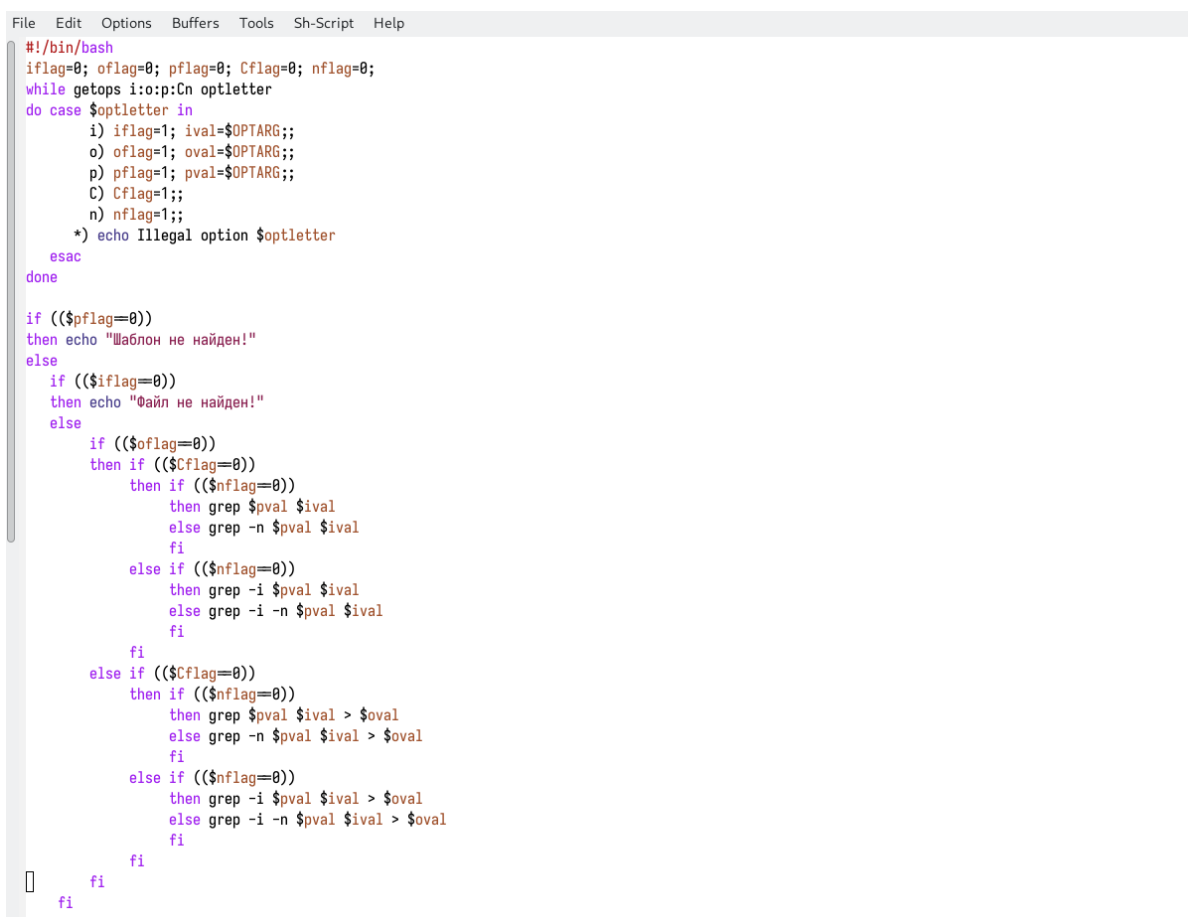
## 2 Задание

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;
  - `-р`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 `-р`.
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 `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 Выполнение лабораторной работы

1. Первое, с чем предстоит столкнуться, это написание командного файла, который анализирует данные файла на основе шаблона и, при случае, производит запись в другой файл. Реализация вышла следующая: (рис. 3.1).



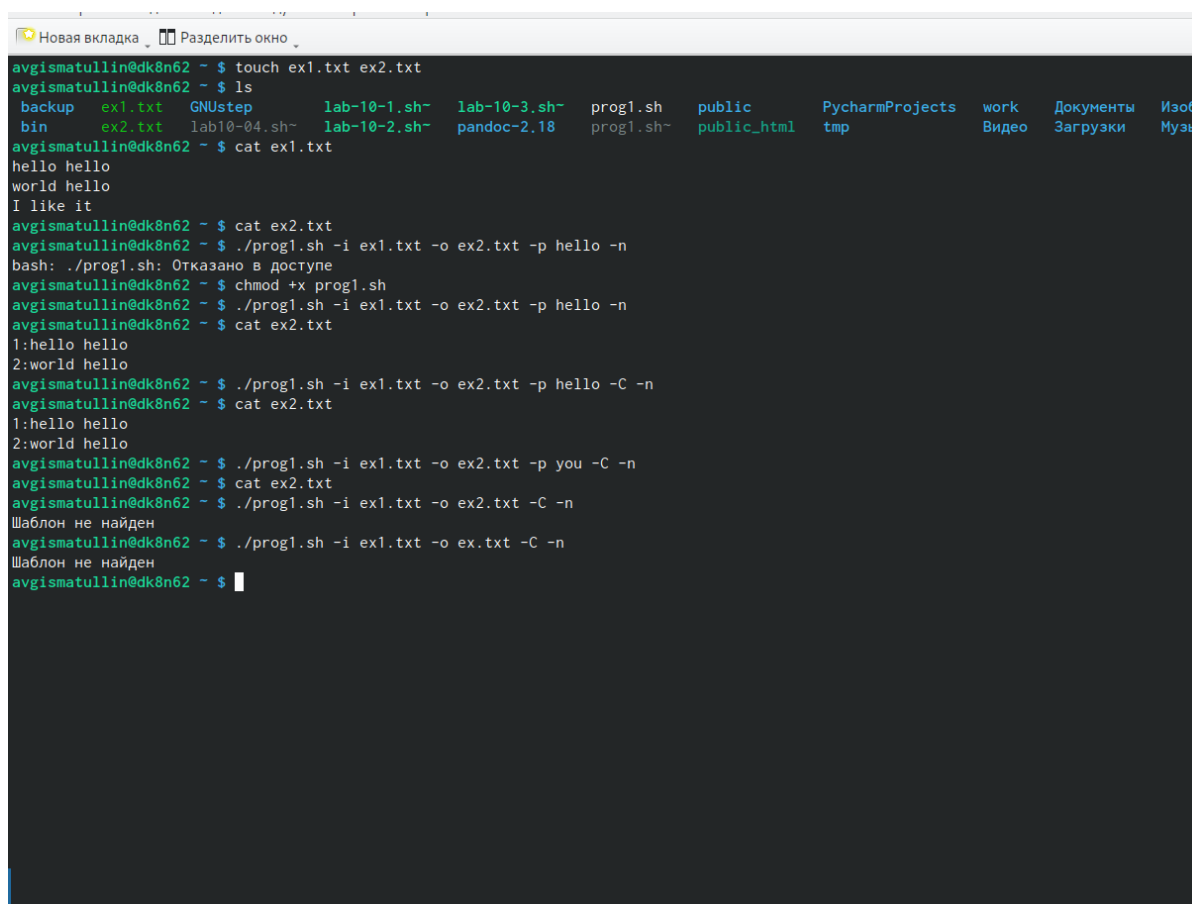
```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
iflag=0; oflag=0; pflag=0; Cflag=0; nflag=0;
while getopts i:o:p:Cn optletter
do case $optletter in
    i) iflag=1; ival=$OPTARG;;
    o) oflag=1; oval=$OPTARG;;
    p) pflag=1; pval=$OPTARG;;
    C) Cflag=1;;
    n) nflag=1;;
    *) echo Illegal option $optletter
esac
done

if (($pflag==0))
then echo "Шаблон не найден!"
else
if (($iflag==0))
then echo "Файл не найден!"
else
if (($oflag==0))
then if (($Cflag==0))
then if (($nflag==0))
then grep $pval $ival
else grep -n $pval $ival
fi
else if (($nflag==0))
then grep -i $pval $ival
else grep -i -n $pval $ival
fi
fi
else if (($Cflag==0))
then if (($nflag==0))
then grep $pval $ival > $oval
else grep -n $pval $ival > $oval
fi
else if (($nflag==0))
then grep -i $pval $ival > $oval
else grep -i -n $pval $ival > $oval
fi
fi
fi
fi
```

Рис. 3.1: Emacs. Командный файл №1



Проверка работоспособности с различными ключами (рис. 3.2).



```
Новая вкладка  Разделить окно
avgismatullin@dk8n62 ~ $ touch ex1.txt ex2.txt
avgismatullin@dk8n62 ~ $ ls
backup  ex1.txt  GNUstep  lab-10-1.sh~  lab-10-3.sh~  prog1.sh  public  PycharmProjects  work  Документы  Изобр
bin     ex2.txt  lab10-04.sh~  lab-10-2.sh~  pandoc-2.18  prog1.sh~  public_html  tmp  Видео  Загрузки  Музы
avgismatullin@dk8n62 ~ $ cat ex1.txt
hello hello
world hello
I like it
avgismatullin@dk8n62 ~ $ cat ex2.txt
avgismatullin@dk8n62 ~ $ ./prog1.sh -i ex1.txt -o ex2.txt -p hello -n
bash: ./prog1.sh: Отказано в доступе
avgismatullin@dk8n62 ~ $ chmod +x prog1.sh
avgismatullin@dk8n62 ~ $ ./prog1.sh -i ex1.txt -o ex2.txt -p hello -n
avgismatullin@dk8n62 ~ $ cat ex2.txt
1:hello hello
2:world hello
avgismatullin@dk8n62 ~ $ ./prog1.sh -i ex1.txt -o ex2.txt -p hello -C -n
avgismatullin@dk8n62 ~ $ cat ex2.txt
1:hello hello
2:world hello
avgismatullin@dk8n62 ~ $ ./prog1.sh -i ex1.txt -o ex2.txt -p you -C -n
avgismatullin@dk8n62 ~ $ cat ex2.txt
Шаблон не найден
avgismatullin@dk8n62 ~ $ ./prog1.sh -i ex1.txt -o ex.txt -C -n
Шаблон не найден
avgismatullin@dk8n62 ~ $
```

Рис. 3.2: Командная строка. Исполнение командного файла №1

2. Далее необходимо написать небольшой код на языке Си (рис. 3.3), а затем, проанализировав входные данные, произвести соответствующий вывод: (рис. 3.4)

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    printf("Введите число\n");
    int a;
    scanf("%d", &a);
    if (a < 0) exit(0);
    if (a > 0) exit(1);
    if (a == 0) exit(2);
    return 0;}
```

Рис. 3.3: Emacs. Код на языке Си

```
avgismatullin@dk8n62 ~ $ cat prog2.sh
#!/bin/bash
gcc prog2.c -o prog2
./prog2
code=$?
case $code in
    0) echo "Число меньше 0";;
    1) echo "Число больше 0";;
    2) echo "Число равно 0"
esac

avgismatullin@dk8n62 ~ $ ./prog2.sh
Введите число
0
Число равно 0
avgismatullin@dk8n62 ~ $ ./prog2.sh
Введите число
1
Число больше 0
avgismatullin@dk8n62 ~ $ ./prog2.sh
Введите число
-1
Число меньше 0
avgismatullin@dk8n62 ~ $
```

Рис. 3.4: Командная строка. Командный файл №2 и проверка работоспособности

3. После этого создаем файл, цель которого создать определенное количество файлов с некоторой последовательной нумерацией. Результат (рис. 3.5):

```
avgismatullin@dk8n62 ~ $ cat prog3.sh
#!/bin/bash
opt=$1;
format=$2;
number=$3;
function Files()
{
    for (( i=1; i<=number; i++ )) do
        file=$(echo $format | tr '#' "$i")
        if [ $opt == "-r" ]
        then
            rm -f $file
        elif [ $opt == "-c" ]
        then
            touch $file
        fi
    done
}
Files
avgismatullin@dk8n62 ~ $ ./prog3.sh -c abc#.txt 3
avgismatullin@dk8n62 ~ $ ls
abcl.txt  abc2.txt  bin          lab10-04.sh~  lab-10-2.sh~  pandoc-2.18  prog2      prog2.c~  prog2.sh~  prog3.sh~  public_html  tmp  Видео  Загрузки  Музыка  'Рабочий стол'
abcl.txt  backup    GNUstep     lab-10-1.sh~  lab-10-3.sh~  prog1.sh~    prog2.c    prog2.sh  prog3.sh  public      PycharmProjects  work  Документы  Изображения  Общедоступные  Шаблоны
avgismatullin@dk8n62 ~ $ ./prog3.sh -r abc#.txt 3
avgismatullin@dk8n62 ~ $ ls
backup  GNUstep  lab-10-1.sh~  lab-10-3.sh~  prog1.sh~  prog2.c  prog2.sh  prog3.sh  public  PycharmProjects  work  Документы  Изображения  Общедоступные  Шаблоны
bin     lab10-04.sh~  lab-10-2.sh~  pandoc-2.18  prog2      prog2.c~  prog2.sh~  prog3.sh~  public_html  tmp  Видео  Загрузки  Музыка  'Рабочий стол'
```

Рис. 3.5: Командная строка. Исполнение командного файла №3

4. Последней же задачей является архивация файлов с определенным расширением, а также таких, что их изменения произошли не позднее недели. Создадим новый каталог, добавим туда файлы и проверим работоспособность (рис. 3.6):

```
avgismatullin@dk8n62 ~ $ cd catalogue/
avgismatullin@dk8n62 ~/catalogue $ ls
abcl.txt  abc2.txt
avgismatullin@dk8n62 ~/catalogue $ ~/prog4.sh
bash: /afs/.dk.sci.pfu.edu.ru/home/a/v/avgismatullin/prog4.sh: Отказано в доступе
avgismatullin@dk8n62 ~/catalogue $ chmod u+x ~/prog4.sh
avgismatullin@dk8n62 ~/catalogue $ ~/prog4.sh
abcl.txt
abc2.txt
avgismatullin@dk8n62 ~/catalogue $ ls
abcl.txt  abc2.txt  catalogue.tar
avgismatullin@dk8n62 ~/catalogue $ ls -li
2070481156 abcl.txt  2070481944 abc2.txt  2070479384 catalogue.tar
avgismatullin@dk8n62 ~/catalogue $ cd ..
avgismatullin@dk8n62 ~ $ cat prog4.sh
#!/bin/bash
files=$(find ./ -maxdepth 1 -mtime -7)
listing=""
for file in "$files" ; do
    file=$(echo "$file" | cut -c 3-)
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
avgismatullin@dk8n62 ~ $
```

Рис. 3.6: Командная строка. Исполнение командного файла №4

## 4 Контрольные вопросы

### 1. Каково предназначение команды `getopts`?

Команда `getopts` осуществляет синтаксический анализ командной строки, выделяя флаги, используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg...]` Флаги это опции командной строки, обычно помеченные знаком минус; Например, для команды `ls` флагом может являться `-F`. Строка опций `option-string` это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, то она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введённые данные с помощью оператора `case`. Функция `getopts` включает две специальные переменные среды `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента. Функция `getopts` также понимает переменные типа массив, следовательно, можно использовать её в функции не только для синтаксического анализа аргументов функций, но и для анализа введённых пользователем данных.

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

При перечислении имён файлов текущего каталога можно использовать следующие символы: 1. соответствует произвольной, в том числе и пустой строке; 2. ? соответствует любому одинарному символу; 3. `[c1-c2]` соответствует любому

символу, лексикографически находящемуся между символами `c1` и `c2`. Например, `1.1 echo` выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `1.2. ls.c` выведет все файлы с последними двумя символами, совпадающими с `c`. `1.3. echoprogram?` выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.` `1.4.[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 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, являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда. Команды `OS/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.

### 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 перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов,

но вы можете захотеть продолжить проверять данный блок на других условных выражениях.

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

Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования bash: это команда true, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда false, которая всегда возвращает код завершения, неравный нулю (т.е. ложь). Примеры бесконечных циклов: `while true do echo hello andy done` `until false do echo hello mike done`.

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

Строка `if test -f mans/i.s` проверяет, существует ли файл `mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).

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

Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла `while` служебного слова `while` на `until` условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.

## 5 Выводы

В ходе выполнения лабораторной работы мы изучили основы программирования в оболочке ОС UNIX и научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## **Список литературы**