

# **Отчет по лабораторной работе №11**

**дисциплина: операционные системы**

**Шмаков Максим Павлович**

# Содержание

Цель работы	5
Задание	6
Выполнение лабораторной работы	7
Выводы	16
Контрольные вопросы	17

## Список иллюстраций

0.1. рис. 1 . . . . .	7
0.2. рис. 2 . . . . .	7
0.3. рис. 3 . . . . .	8
0.4. рис. 4 . . . . .	8
0.5. рис. 5 . . . . .	9
0.6. рис. 6 . . . . .	9
0.7. рис. 7 . . . . .	10
0.8. рис. 8 . . . . .	10
0.9. рис. 9 . . . . .	11
0.10. рис. 10 . . . . .	11
0.11. рис. 11 . . . . .	11
0.12. рис. 12 . . . . .	12
0.13. рис. 13 . . . . .	13
0.14. рис. 14 . . . . .	14
0.15. рис. 15 . . . . .	14
0.16. рис. 16 . . . . .	14
0.17. рис. 17 . . . . .	15

## **Список таблиц**

# Цель работы

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

## Задание

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. 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`).

# Выполнение лабораторной работы

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; `-rшаблон` — 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`.

Сначала создаю файлы `num1.sh`, `textfile1` и `textfile2` (рис. [-@fig:001]) (рис. [-@fig:002])



```
[mpshmakov@fedora ~]$ touch num1.sh
```

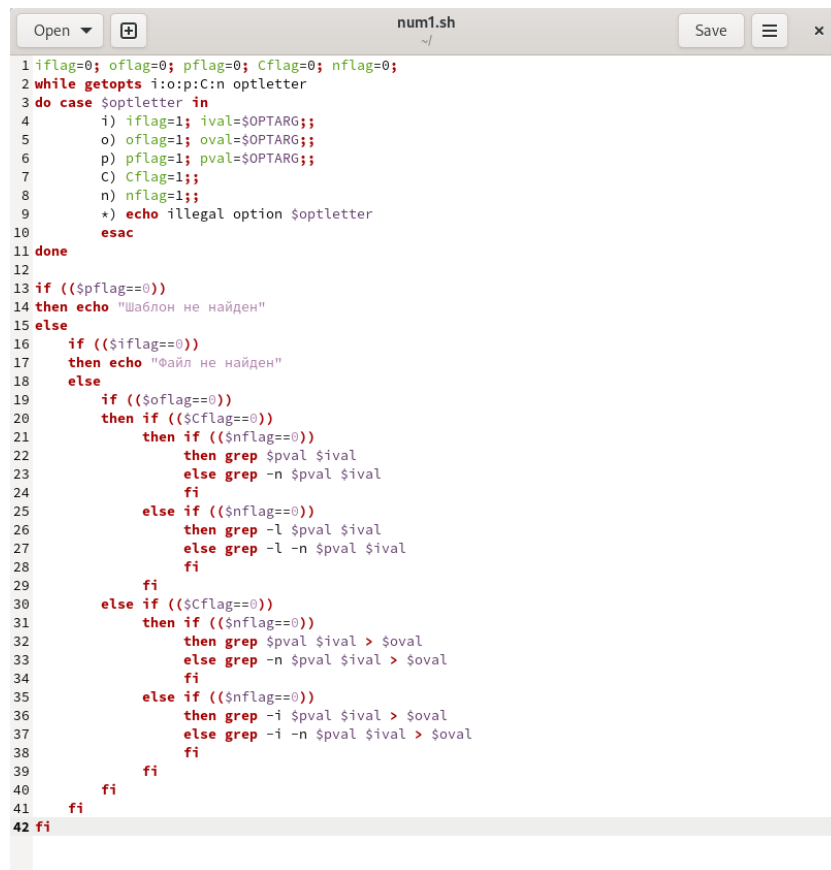
Рис. 0.1.: рис. 1



```
[mpshmakov@fedora ~]$ touch textfile1
```

Рис. 0.2.: рис. 2

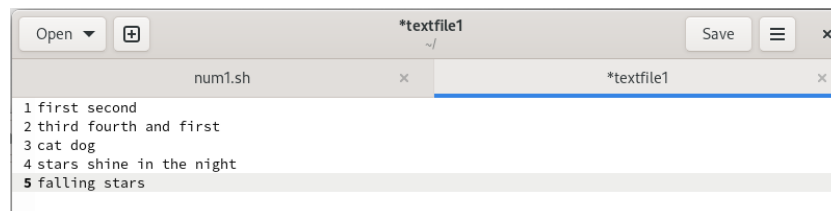
В `num1.sh` пишу скрипт (рис. [-@fig:003])



```
1 iflag=0; oflag=0; pflag=0; Cflag=0; nflag=0;
2 while getopts i:o:p:C:n optletter
3 do case $optletter in
4   i) iflag=1; ival=$OPTARG;;
5   o) oflag=1; oval=$OPTARG;;
6   p) pflag=1; pval=$OPTARG;;
7   C) Cflag=1;;
8   n) nflag=1;;
9   *) echo illegal option $optletter
10      esac
11 done
12
13 if (($pflag==0))
14 then echo "Шаблон не найден"
15 else
16   if (($iflag==0))
17   then echo "Файл не найден"
18   else
19     if (($oflag==0))
20     then if (($Cflag==0))
21          then if (($nflag==0))
22               then grep $pval $ival
23               else grep -n $pval $ival
24               fi
25          else if (($nflag==0))
26               then grep -l $pval $ival
27               else grep -l -n $pval $ival
28               fi
29          fi
30     else if (($Cflag==0))
31     then if (($nflag==0))
32          then grep $pval $ival > $oval
33          else grep -n $pval $ival > $oval
34          fi
35     else if (($nflag==0))
36     then grep -i $pval $ival > $oval
37     else grep -i -n $pval $ival > $oval
38     fi
39     fi
40   fi
41 fi
42 fi
```

Рис. 0.3.: рис. 3

В textfile1 запишу случайный текст с повторяющимися словами, а textfile2 оставлю пустым. (рис. [-@fig:004])



```
1 first second
2 third fourth and first
3 cat dog
4 stars shine in the night
5 falling stars
```

Рис. 0.4.: рис. 4

Даю права на исполнение и проверяю работу скрипта. Все работает исправно. (рис. [-@fig:005])



```
[mpshmakov@fedora ~]$ ./num1.sh -i ~/textfile1 -o ~/textfile2 -p stars -C -n
[mpshmakov@fedora ~]$ cat textfile2
stars shine in the night
falling stars
[mpshmakov@fedora ~]$ ./num1.sh -i ~/textfile1 -o ~/textfile2 -p first -C -n
[mpshmakov@fedora ~]$ cat textfile2
first second
third fourth and first
[mpshmakov@fedora ~]$ ./num1.sh -i ~/textfile1 -o ~/textfile2 -p first -n
[mpshmakov@fedora ~]$ cat textfile2
1:first second
2:third fourth and first
[mpshmakov@fedora ~]$ ./num1.sh -i ~/textfile1 -C -n
Шаблон не найден
[mpshmakov@fedora ~]$ ./num1.sh -o ~/textfile2 -p stars -C -n
Файл не найден
```

Рис. 0.5.: рис. 5

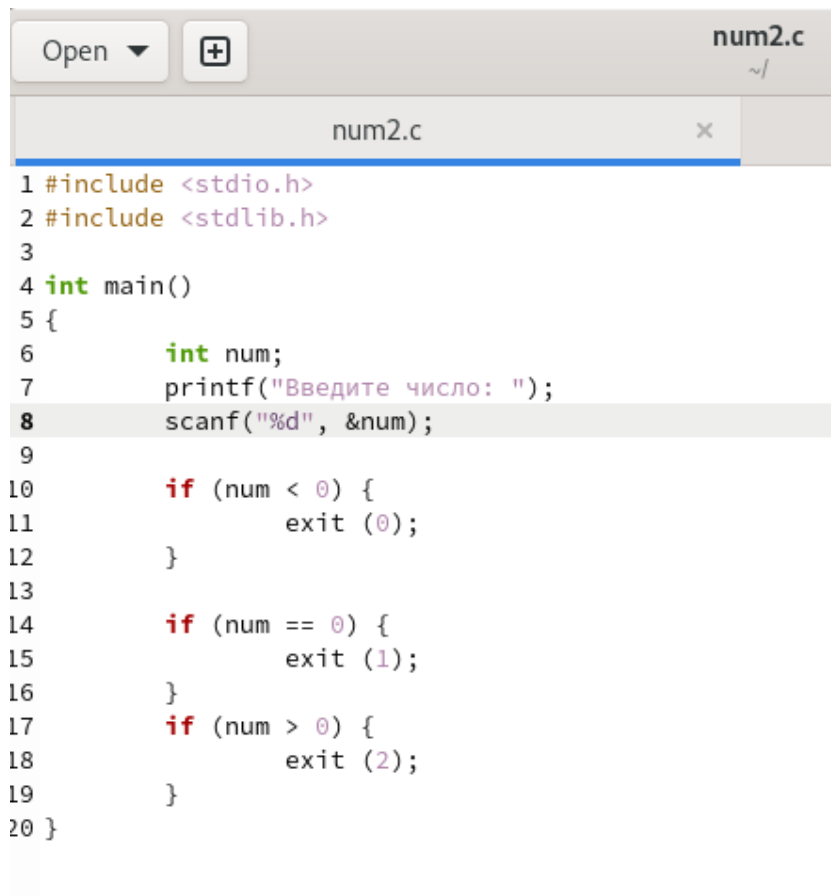
2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.

Создаю файлы `num2.c` и `num2.sh`. (рис. [-@fig:006])

```
[mpshmakov@fedora ~]$ touch num2.c num2.sh
```

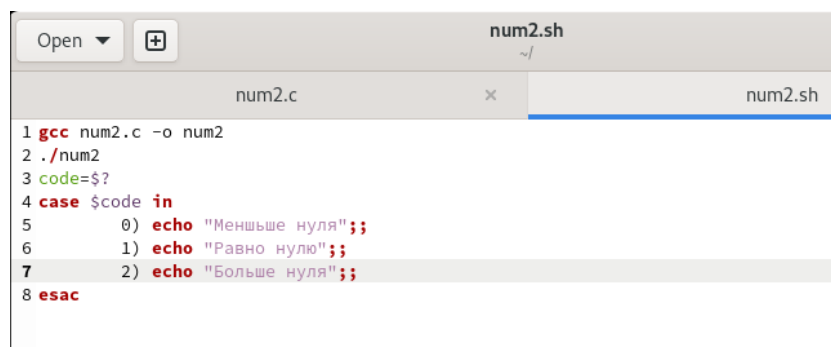
Рис. 0.6.: рис. 6

Пишу в `num2.c` код, который будет возвращать число (0, 1, 2). Это число используется скриптом, чтобы определить, какую строку из 3 ему выводить. (рис. [-@fig:007]) (рис. [-@fig:008])



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     int num;
7     printf("Введите число: ");
8     scanf("%d", &num);
9
10    if (num < 0) {
11        exit (0);
12    }
13
14    if (num == 0) {
15        exit (1);
16    }
17    if (num > 0) {
18        exit (2);
19    }
20 }
```

Рис. 0.7.: рис. 7



```
1 gcc num2.c -o num2
2 ./num2
3 code=$?
4 case $code in
5     0) echo "Меньше нуля";;
6     1) echo "Равно нулю";;
7     2) echo "Больше нуля";;
8 esac
```

Рис. 0.8.: рис. 8

Даю права на исполнение и проверяю работу скрипта. Все работает исправно.  
(рис. [-@fig:009]) (рис. [-@fig:010])

```
[mpshmakov@fedora ~]$ chmod +x num2.sh
```

Рис. 0.9.: рис. 9

```
[mpshmakov@fedora ~]$ ./num2.sh
Введите число: 1
Больше нуля
[mpshmakov@fedora ~]$ ./num2.sh
Введите число: 0
Равно нулю
[mpshmakov@fedora ~]$ ./num2.sh
Введите число: -1
Меньше нуля
```

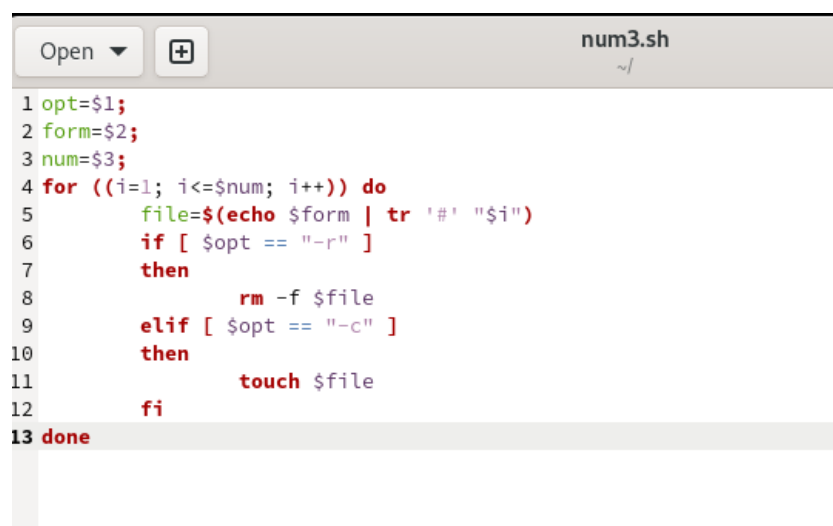
Рис. 0.10.: рис. 10

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).

Создаю файл num3.sh и пишу в него скрипт. (рис. [-@fig:011]) (рис. [-@fig:012])

```
[mpshmakov@fedora ~]$ touch num3.sh
```

Рис. 0.11.: рис. 11



```
1 opt=$1;
2 form=$2;
3 num=$3;
4 for ((i=1; i<=$num; i++)) do
5     file=$(echo $form | tr '#' '$i')
6     if [ $opt == "-r" ]
7     then
8         rm -f $file
9     elif [ $opt == "-c" ]
10    then
11        touch $file
12    fi
13 done
```

Рис. 0.12.: рис. 12

Даю права на исполнение и проверяю работу скрипта. Все работает исправно.  
(рис. [-@fig:013])

```

[mpshmakov@fedora ~]$ ./num3.sh -c text#.txt 5
[mpshmakov@fedora ~]$ ls
abc1      'lab 1 shmak max.zip'  test3
australia lsanalog.sh           '#test4#'
backup    lsanalog.sh~          test4
backup.sh Music               testfiletotrans
backup.sh~ my_os                testtest
bin       newcatalog            text
conf.txt  newfile              text1.txt
Desktop  num1.sh              text2.txt
dfsdfs   num2                 text3.txt
Documents num2.c              text4.txt
Downloads num2.sh             text5.txt
feathers  num3.sh             textfile1
filetest  Pictures            textfile2
filetest.txt play               text.sh
file.txt  Public              Videos
finder.sh ski.plases        work
finder.sh~ Templates     work1
index.html '#test1#'          'work (copy)'
index.html.1 test1             zadanie2.sh
'#lab07.sh#' '#test2#'        zadanie2.sh~
lab07.sh  test2              'Операционные системы.zip'
lab11z1   '#test3#'

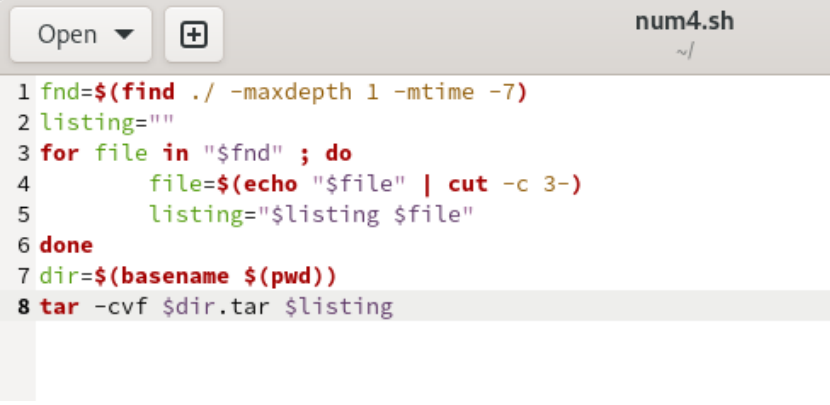
[mpshmakov@fedora ~]$ ./num3.sh -r text#.txt 5
[mpshmakov@fedora ~]$ ls
abc1      finder.sh           num2      '#test4#'
australia finder.sh~      num2.c    test4
backup    index.html          num2.sh   testfiletotrans
backup.sh index.html.1        num3.sh   testtest
backup.sh~ '#lab07.sh#'       Pictures   text
bin       lab07.sh            play      textfile1
conf.txt  lab11z1             Public     textfile2
Desktop  'lab 1 shmak max.zip' ski.plases text.sh
dfsdfs   lsanalog.sh         Templates Videos
Documents lsanalog.sh~        '#test1#' work
Downloads Music          test1     work1
feathers  my_os               '#test2#' 'work (copy)'
filetest  newcatalog          test2     zadanie2.sh
filetest.txt newfile            '#test3#' zadanie2.sh~
file.txt  num1.sh             test3     'Операционные системы.zip'

```

Рис. 0.13.: рис. 13

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).

Создаю файл num4.sh и пишу в него скрипт. (рис. [-@fig:014])



```
1 fnd=$(find ./ -maxdepth 1 -mtime -7)
2 listing=""
3 for file in "$fnd" ; do
4     file=$(echo "$file" | cut -c 3-)
5     listing="$listing $file"
6 done
7 dir=$(basename $(pwd))
8 tar -cvf $dir.tar $listing
```

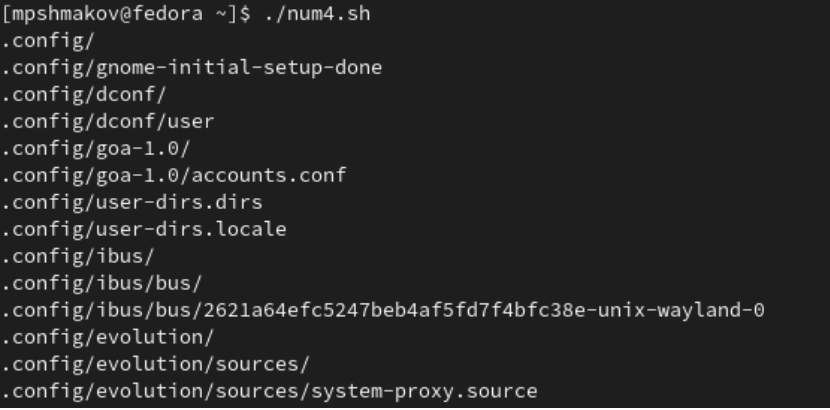
Рис. 0.14.: рис. 14

Даю права на исполнение и проверяю работу скрипта. Все работает исправно.  
(рис. [-@fig:015]) (рис. [-@fig:016]) (рис. [-@fig:017])



```
[mpshmakov@fedora ~]$ chmod +x num4.sh
```

Рис. 0.15.: рис. 15



```
[mpshmakov@fedora ~]$ ./num4.sh
.config/
.config/gnome-initial-setup-done
.config/dconf/
.config/dconf/user
.config/goa-1.0/
.config/goa-1.0/accounts.conf
.config/user-dirs.dirs
.config/user-dirs.locale
.config/ibus/
.config/ibus/bus/
.config/ibus/bus/2621a64efc5247beb4af5fd7f4bfc38e-unix-wayland-0
.config/evolution/
.config/evolution/sources/
.config/evolution/sources/system-proxy.source
```

Рис. 0.16.: рис. 16

```

[mpshmakov@fedora ~]$ ls
abc1      finder.sh~      num2.c      test4
australia index.html      num2.sh     testfiletotrans
backup    index.html.1    num3.sh     testtest
backup.sh '#lab07.sh#'    num4.sh     text
backup.sh~ lab07.sh        Pictures    textfile1
bin       lab11z1         play        textfile2
conf.txt  'lab 1 shmak max.zip' Public       text.sh
Desktop   lsanalog.sh     ski.places  Videos
dfsdfs    lsanalog.sh~    Templates   work
Documents mpshtmakov.tar '#test1#'    work1
Downloads music          test1       'work (copy)~'
feathers  my_os           '#test2#'   zadanie2.sh
filetest newcatalog      test2       zadanie2.sh~
filetest.txt newfile         '#test3#'   'Операционные системы.zip'
file.txt  num1.sh         test3
finder.sh num2            '#test4#'

```

Рис. 0.17.: рис. 17

## **Выводы**

В ходе работы я научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.



# Контрольные вопросы

1. Команда `getopts` 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, 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 введенные 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 введенных пользователем данных.
2. При перечислении имён файлов текущего каталога можно использовать следующие символы:
  - `-` – соответствует произвольной, в том числе и пустой строке;
  - `?` – соответствует любому одинарному символу;

- `[c1-c2]` – соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`. Например,
  - `echo *` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`;
  - `ls *.c` – выведет все файлы с последними двумя символами, совпадающими с `.c`.
  - `echo prog.?` – выведет 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. Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости отрезультатов 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 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.

4. 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. 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, не равный нулю (т. е. ложь). Примеры бесконечных циклов: `while true do echo hello andy done` `until false do echo hello mike done`
6. Строка `if test -f mans/i.s, mans/i.s` является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).
7. 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 противополо-

ложное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.