

# **Лабораторная работа № 11**

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

**НВЕ МАНГЕ ХОСЕ. Х.М;НКАбд-03-22**

# Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Листинги	15
6	Контрольные вопросы	17
7	Выводы	20
	Список литературы	21

## Список иллюстраций

4.1	Файл . . . . .	8
4.2	Текст программы 1 . . . . .	9
4.3	Файл, в котором выполнялась программа . . . . .	9
4.4	Программа на Си . . . . .	10
4.5	Программный файл . . . . .	11
4.6	Результат . . . . .	11
4.7	Программный файл . . . . .	12
4.8	Результат . . . . .	12
4.9	Файл и запуск . . . . .	13
4.10	Текст программы 4 . . . . .	13
4.11	Созданный архив и файл . . . . .	14
4.12	FILES.txt . . . . .	14

## 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; • `-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 `-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`).

### 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;

- С-оболочка (или csh) — надстройка на оболочкой Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

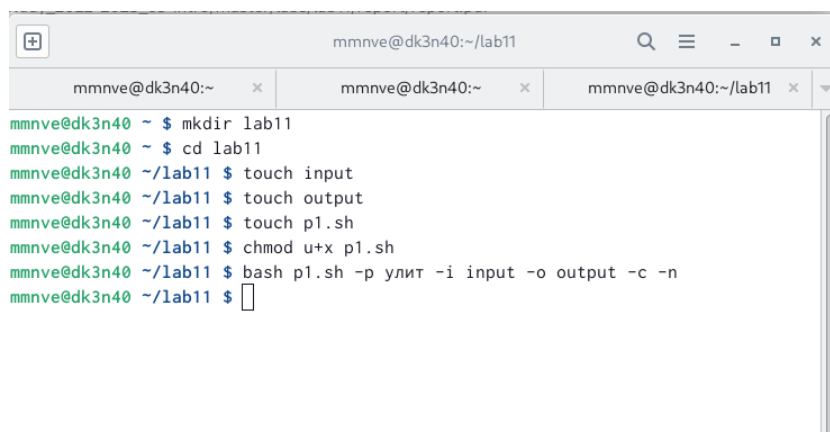
POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода.

## 4 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;
  - `-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`.

(рис. 4.1, 4.2, 4.3)



```
mmnve@dk3n40:~$ mkdir lab11
mmnve@dk3n40:~$ cd lab11
mmnve@dk3n40 ~/lab11 $ touch input
mmnve@dk3n40 ~/lab11 $ touch output
mmnve@dk3n40 ~/lab11 $ touch p1.sh
mmnve@dk3n40 ~/lab11 $ chmod u+x p1.sh
mmnve@dk3n40 ~/lab11 $ bash p1.sh -p улит -i input -o output -c -n
mmnve@dk3n40 ~/lab11 $
```

Рис. 4.1: Файл



```
1 #!/bin/bash
2 while getopts i:o:p:cn optletter
3 do
4 case $optletter in
5     1) iflag=1; ival=$OPTARG;;
6     o) oflag=1; oval=$OPTARG;;
7     p) pflag=1; pval=$OPTARG;;
8     c) cflag=1;;
9     n) nflag=1;;
10    *) echo Illegal option $optletter;;
11    esac
12 done
13 if ! test $cflag
14 then
15     cf=-1
16 fi
17 if test $nflag
18 then
19     nf=-n
20 fi
21 grep $cf $nf $pval $ival >> $oval
```

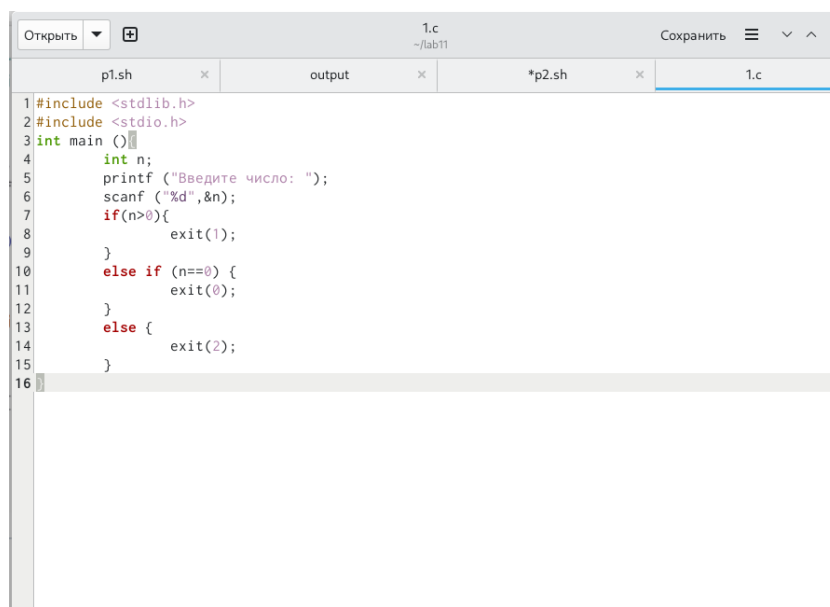
Рис. 4.2: Текст программы 1

```
1 1:Заходит как-то улитка в бар и просит у бармена стакан воды
2 3:через 3 дня улитка снова заходит в бар и снова просит стакан воды
3 7:Бармен не выдерживает и спрашивает улитку
4 9:А улитка отвечает: "у меня дом горит".
```

Рис. 4.3: Файл, в котором выполнялась программа

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 (рис. 4.5, 4.6, 4.7).



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

Рис. 4.4: Программа на Си

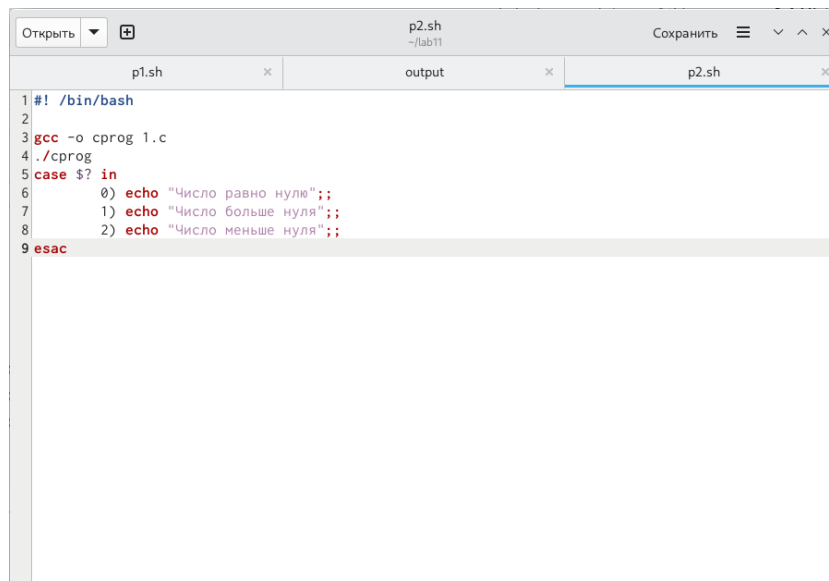


Рис. 4.5: Программный файл

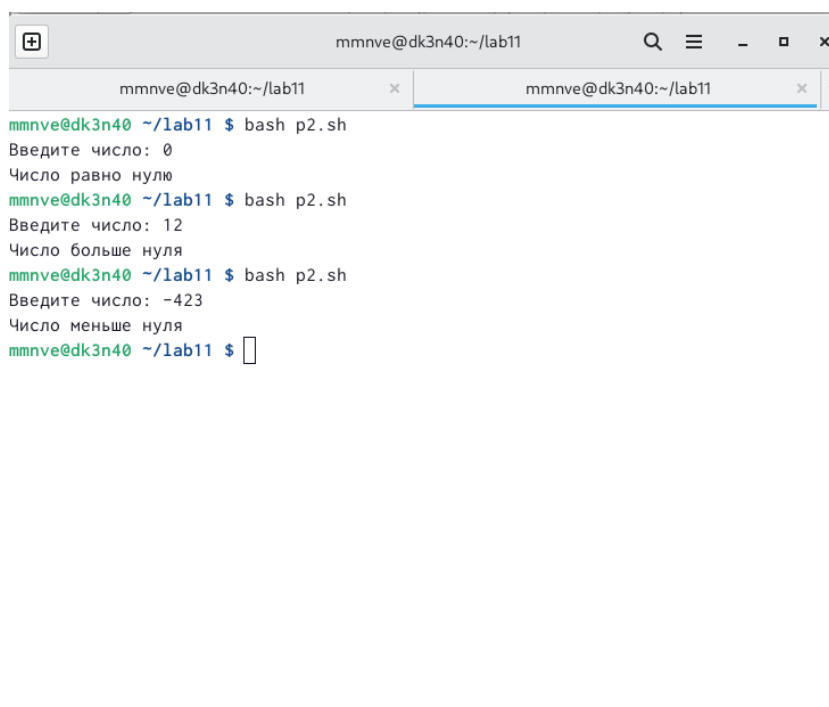


Рис. 4.6: Результат

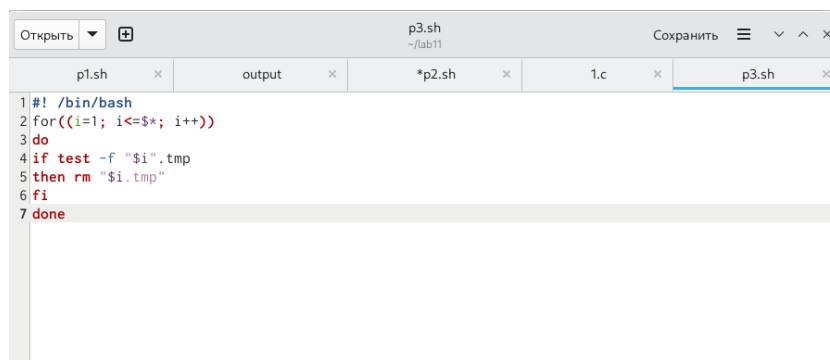
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (например 1.tmp, 2.tmp, 3.tmp, 4.tmp и

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) (рис. 4.8, 4.9).



```
mmnve@dk3n40 ~/lab11 $ bash p3.sh 5
mmnve@dk3n40 ~/lab11 $ ls
1.c  cprog  input  output  p1.sh  p2.sh  p3.sh
mmnve@dk3n40 ~/lab11 $ bash p3.sh 5
mmnve@dk3n40 ~/lab11 $ ls
1.c  cprog  input  output  p1.sh  p2.sh  p3.sh
mmnve@dk3n40 ~/lab11 $
```

Рис. 4.7: Программный файл



```
1 #! /bin/bash
2 for((i=1; i<=5; i++))
3 do
4 if test -f "$i".tmp
5 then rm "$i".tmp"
6 fi
7 done
```

Рис. 4.8: Результат

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`) (рис. 4.10, 4.11, 4.12, ??).

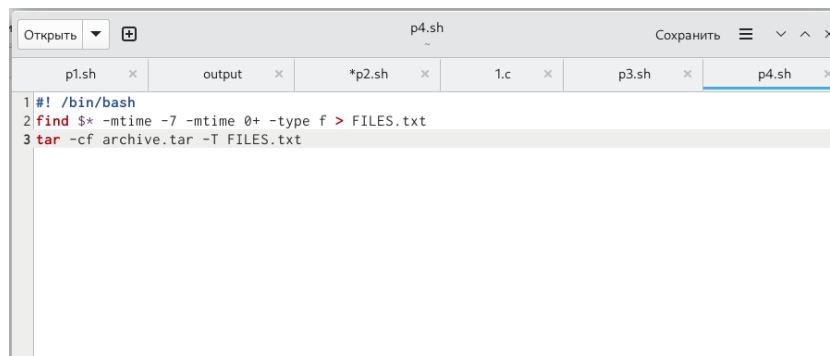


Рис. 4.9: Файл и запуск

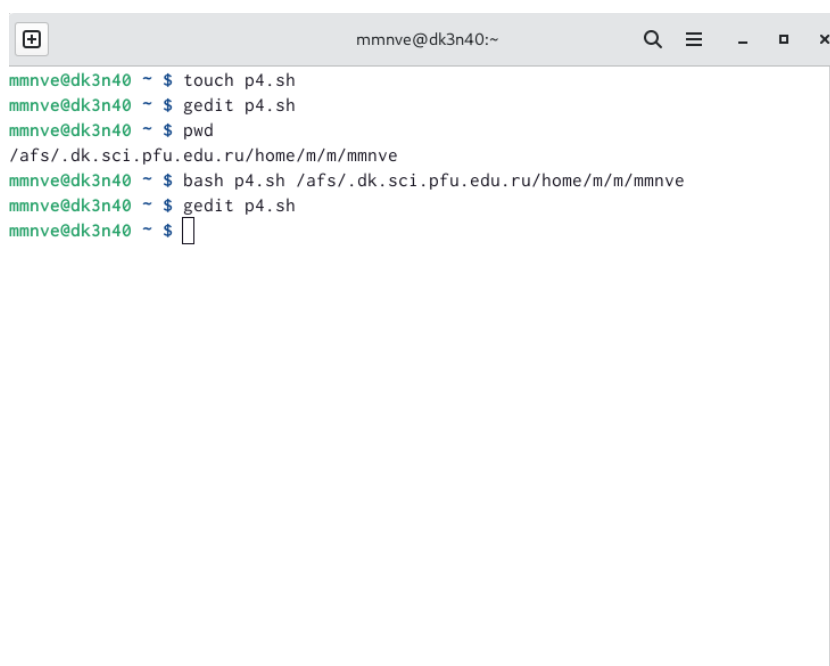


Рис. 4.10: Текст программы 4

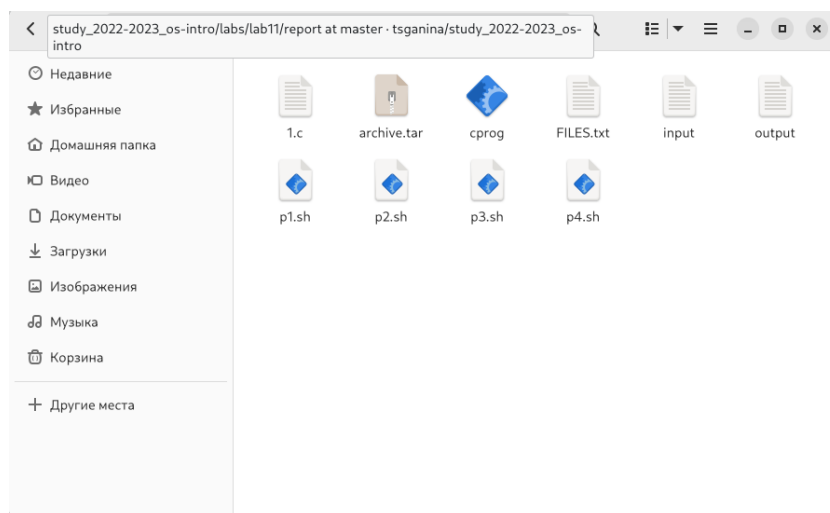


Рис. 4.11: Созданный архив и файл

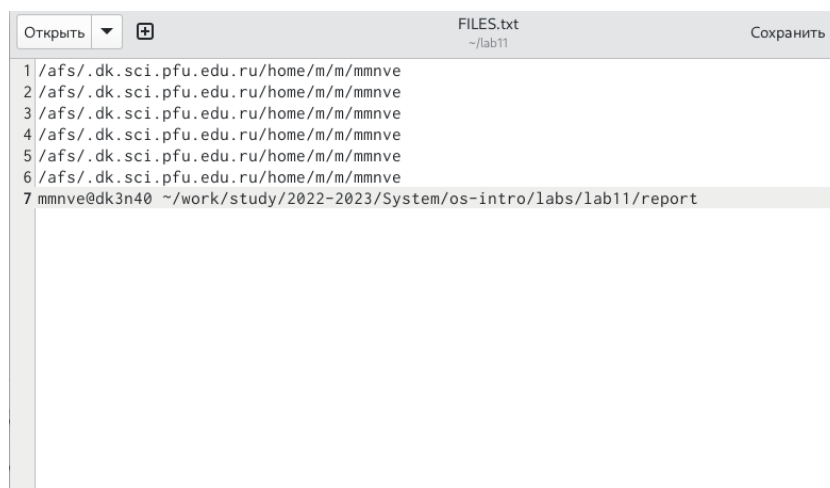


Рис. 4.12: FILES.txt

## 5 Листинги

### 1. Программа 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
```

## 2. Программа 2

```
#!/bin/bash
gcc -o cprog lab11_2.c
./cprog
case $? in
0) echo "Число равно нулю";;
1) echo "Число больше нуля";;
2) echo "Число меньше нуля";;
esac
```

## 3. Программа 3

```
#!/bin/bash
for((i=1; i<=10; i++))
do
if test -f "$i".tmp
then rm "$i".tmp
else touch "$i.tmp"
fi
done
```

## 4. Программа 4

```
#!/bin/bash
find $* -mtime -7 -mtime +0 -type f > FILES.txt
tar -cf archive.tar -T FILES.txt
```



## 6 Контрольные вопросы

1. Каково предназначение команды `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, `-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`.
2. Какое отношение метасимволы имеют к генерации имён файлов? 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. Какие операторы управления действиями вы знаете? 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. 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, не равный нулю (т. е. ложь).

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

## **7 Выводы**

В процессе выполнения данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# **Список литературы**

Руководство к лабораторной работе