

Отчет по лабораторной работе №12

Дисциплина: Операционные системы

Морозова Анастасия Владимировна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	16

Список таблиц

Список иллюстраций

2.1	Анализ командной строки	6
2.2	Анализ командной строки	7
2.3	Проверка скрипта	7
2.4	number.c	8
2.5	number.sh	8
2.6	Проверка скриптов	9
2.7	Создание указанного числа файлов	10
2.8	Проверка скрипта	10
2.9	Проверка скрипта	11
2.10	Проверка скрипта	11
2.11	Запаковка в архив	12
2.12	Архивация	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 Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, написала командный файл, который анализирует командную строку с ключами:

- `-i` input file — прочитать данные из указанного файла;
- `-o` output file — 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`. Для данной задачи я создала файл `an.sh` и написала соответствующие скрипты. (рис. -fig. 2.1) (рис. -fig. 2.2)

```
#!/bin/bash
iflag=0; oflag=0; cflag=0; nflag=0; #инициализация переменных-флагов
while getopts i:o:p:C:n optletter #анализ командной строки на наличие опций
do case $optletter in
    i) iflag=1; ival=$OPTARG;; #если опция присутствует
    o) oflag=1; oval=$OPTARG;; #то ей присваивается 1
    p) pflag=1; pval=$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
        fi
    fi
fi
```

Рис. 2.1: Анализ командной строки

```

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
fi

```

Рис. 2.2: Анализ командной строки

Проверила работу написанного скрипта, используя различные опции (команда `./an.sh -I n1.txt -o n2.txt -p capital -C -n`), предварительно добавив право на исполнение файла (команда `chmod +x an.sh`) и создав 2 файла, которые необходимы для выполнения программы: `n1.txt` и `n2.txt` (рис. -fig. 2.3)

```

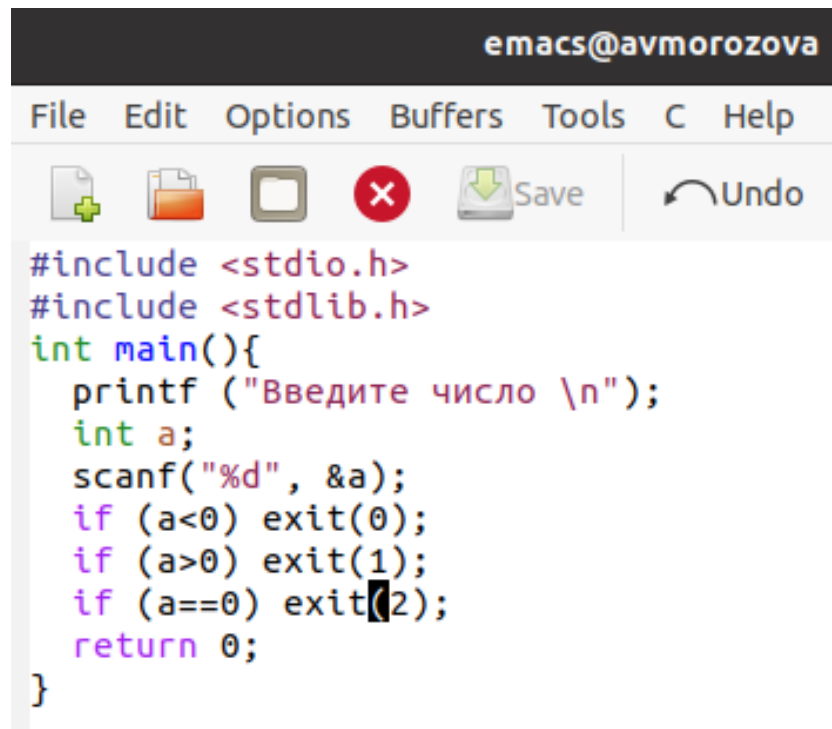
avmorofova@avmorofova:~$ cat n1.txt
my name is Nastya
my NAME is Nastya
I am So happy
avmorofova@avmorofova:~$ ./an.sh -i n1.txt -o n2.txt -p name -C -n
avmorofova@avmorofova:~$ cat n2.txt
1:my name is Nastya
2:my NAME is Nastya
avmorofova@avmorofova:~$ ./an.sh -i n1.txt -C -n
Шаблон не найден
avmorofova@avmorofova:~$ ./an.sh -o n2.txt -p name -C -n
Файл не найден

```

Рис. 2.3: Проверка скрипта

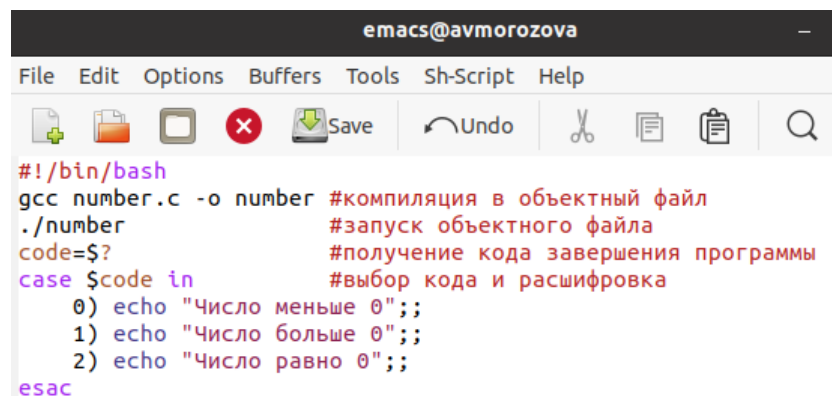
2. Написала на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав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

число было введено. Для данной задачи я создала 2 файла: number.c (рис. -fig. 2.4) и number.sh (рис. -fig. 2.5) и написала соответствующие скрипты.



```
emacs@avmorozova
File Edit Options Buffers Tools C Help
+ Save Undo
#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. 2.4: number.c



```
emacs@avmorozova
File Edit Options Buffers Tools Sh-Script Help
+ Save Undo
#!/bin/bash
gcc number.c -o number #компиляция в объектный файл
./number               #запуск объектного файла
code=$?                #получение кода завершения программы
case $code in           #выбор кода и расшифровка
    0) echo "Число меньше 0";;
    1) echo "Число больше 0";;
    2) echo "Число равно 0";;
esac
```

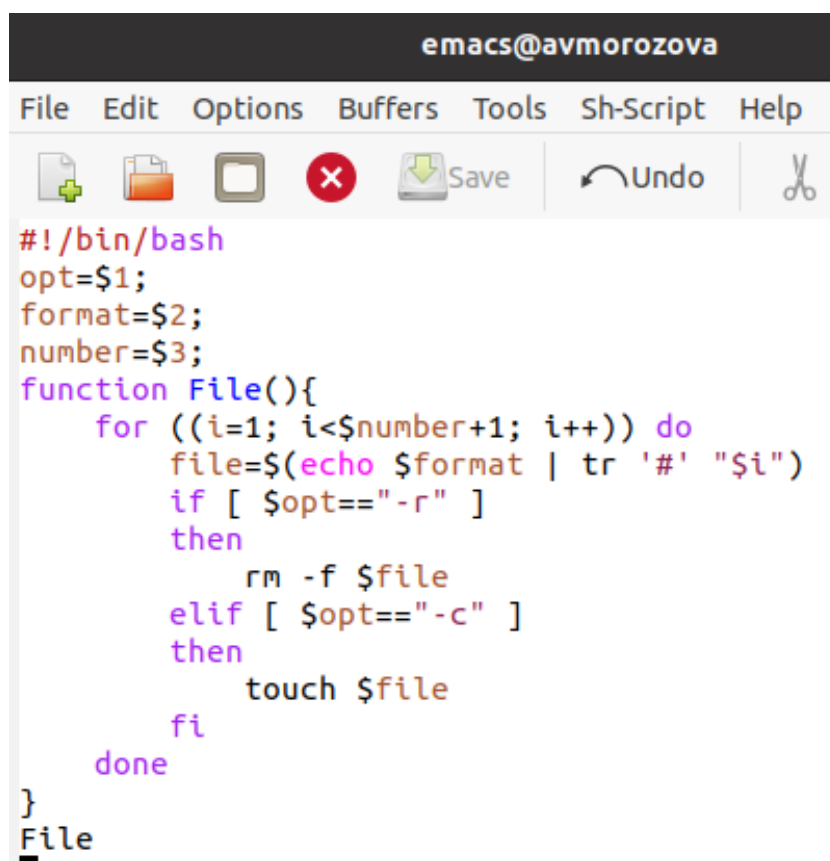
Рис. 2.5: number.sh

Проверила работу написанных скриптов (команда ./number.sh), предварительно добавив право на исполнение файла (команда chmod +x number.sh) (рис. -fig. 2.6)


```
avmogozova@avmogozova:~$ ./number.sh
Введите число
0
Число равно 0
avmogozova@avmogozova:~$ ./number.sh
Введите число
10
Число больше 0
avmogozova@avmogozova:~$ ./number.sh
Введите число
-9
Число меньше 0
```

Рис. 2.6: Проверка скриптов

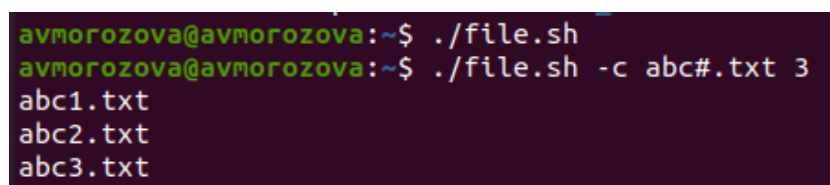
3. Написала командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (1.tmp, 2.tmp, 3.tmp, 4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют). Для данной задачи я создала файл: file.sh (рис. -fig. 2.7) и написала соответствующий скрипт.

The image shows the Emacs editor interface with the title bar 'emacs@avmorozova'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Sh-Script', and 'Help'. The toolbar contains icons for file operations and 'Save', 'Undo', and a scissors icon. The main text area contains a shell script with the following content:

```
#!/bin/bash
opt=$1;
format=$2;
number=$3;
function File(){
    for ((i=1; i<$number+1; i++)) do
        file=$(echo $format | tr '#' "$i")
        if [ $opt=="-r" ]
        then
            rm -f $file
        elif [ $opt=="-c" ]
        then
            touch $file
        fi
    done
}
File
```

Рис. 2.7: Создание указанного числа файлов

Проверила работу написанного скрипта (команда `./file.sh`), предварительно добавив право на исполнение файла (команда `chmod +x file.sh`). Сначала я создала три файла (команда `./file.sh -c abc#.txt 3`), удовлетворяющие условию задачи, а потом удалила их (команда `./file.sh -r abc#.txt 3`)(рис. -fig. 2.8)(рис. -fig. 2.9)(рис. -fig. 2.10)

The image shows a terminal window with the following commands and output:

```
avmorozova@avmorozova:~$ ./file.sh
avmorozova@avmorozova:~$ ./file.sh -c abc#.txt 3
abc1.txt
abc2.txt
abc3.txt
```

Рис. 2.8: Проверка скрипта

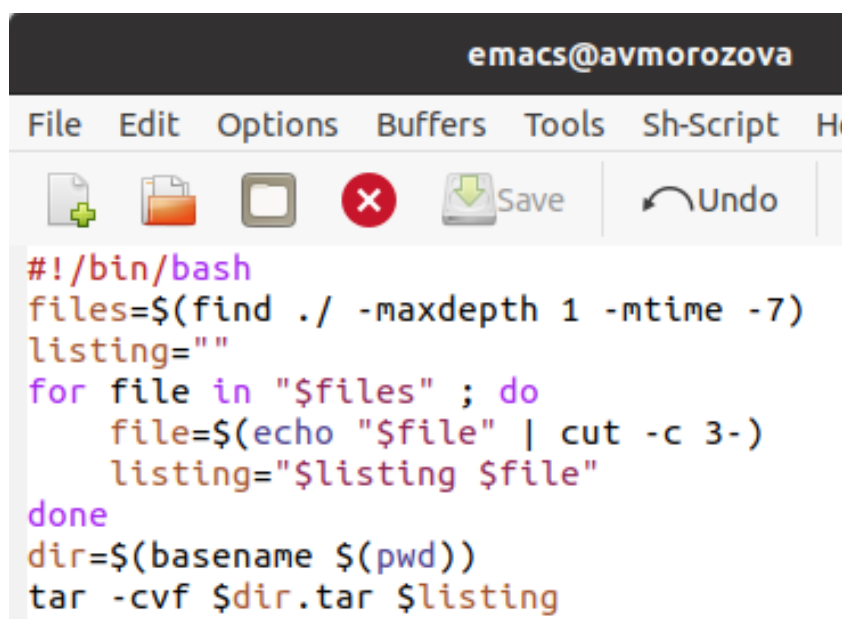
```
avmorofova@avmorofova:~$ ls
abc1.txt      ex1          example2.txt  lab10.sh~    Pictures
abc2.txt      ex1.sh~     example2.txt~ Lab9.md      Public
abc3.txt      ex1.sh~     '#example3.txt#' lab_OS       reports
an.sh         ex2.sh~     example3.txt  Makefile     snap
an.sh~        ex2.sh~     example3.txt~ Music        Templates
backup        ex3.sh~     '#example4.txt#' my_os        usr
backup.sh     ex3.sh~     example4.txt  number       Videos
backup.sh~    ex4.sh~     example4.txt~ number.c     work
Desktop       ex4.sh~     file.sh       number.c~
Documents     example1.txt file.sh~      number.sh
Downloads     example1.txt lab10.sh      number.sh~
```

Рис. 2.9: Проверка скрипта

```
avmorofova@avmorofova:~$ ./file.sh -r abc#.txt 3
avmorofova@avmorofova:~$ ls
an.sh         ex1.sh~     example2.txt~  lab10.sh~    number.sh~
an.sh~        ex2.sh~     '#example3.txt#' Lab9.md      Pictures
backup        ex2.sh~     example3.txt   lab_OS       Public
backup.sh     ex3.sh~     example3.txt~  Makefile     reports
backup.sh~    ex3.sh~     '#example4.txt#' my_os        snap
Desktop       ex4.sh~     example4.txt   number       Templates
Documents     ex4.sh~     example4.txt~  number.c     usr
Downloads     example1.txt file.sh       number.c~    Videos
ex1           example1.txt file.sh~      number.c~    work
ex1.sh        example2.txt lab10.sh      number.sh
```

Рис. 2.10: Проверка скрипта

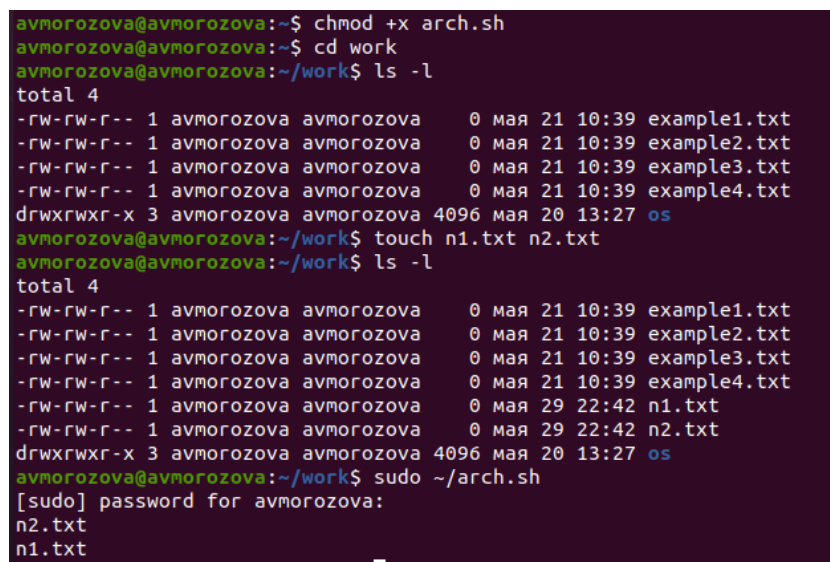
4. Написала командный файл, который с помощью команды `tag` запаковывает в архив все файлы в указанной директории. Модифицировала его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду `find`). Для данной задачи я создала файл: `arch.sh` (рис. -fig. 2.11) и написала соответствующий скрипт.



```
#!/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
```

Рис. 2.11: Запаковка в архив

Далее я проверила работу написанного скрипта (команды `sudo ~/arch.sh` и `tar -tf work.tar`), предварительно добавив право на исполнение файла (команда `chmod +x arch.sh`), перешла в каталог `work`, в котором имелись старые файлы, создала в нем два новых `n1.txt`, `n2.txt`. Файлы, измененные более недели назад, заархивированы не были. (рис. -fig. 2.12)



```
avmorozova@avmorozova:~$ chmod +x arch.sh
avmorozova@avmorozova:~$ cd work
avmorozova@avmorozova:~/work$ ls -l
total 4
-rw-rw-r-- 1 avmorozova avmorozova  0 мая 21 10:39 example1.txt
-rw-rw-r-- 1 avmorozova avmorozova  0 мая 21 10:39 example2.txt
-rw-rw-r-- 1 avmorozova avmorozova  0 мая 21 10:39 example3.txt
-rw-rw-r-- 1 avmorozova avmorozova  0 мая 21 10:39 example4.txt
drwxrwxr-x 3 avmorozova avmorozova 4096 мая 20 13:27 os
avmorozova@avmorozova:~/work$ touch n1.txt n2.txt
avmorozova@avmorozova:~/work$ ls -l
total 4
-rw-rw-r-- 1 avmorozova avmorozova  0 мая 21 10:39 example1.txt
-rw-rw-r-- 1 avmorozova avmorozova  0 мая 21 10:39 example2.txt
-rw-rw-r-- 1 avmorozova avmorozova  0 мая 21 10:39 example3.txt
-rw-rw-r-- 1 avmorozova avmorozova  0 мая 21 10:39 example4.txt
-rw-rw-r-- 1 avmorozova avmorozova  0 мая 29 22:42 n1.txt
-rw-rw-r-- 1 avmorozova avmorozova  0 мая 29 22:42 n2.txt
drwxrwxr-x 3 avmorozova avmorozova 4096 мая 20 13:27 os
avmorozova@avmorozova:~/work$ sudo ~/arch.sh
[sudo] password for avmorozova:
n2.txt
n1.txt
```

Рис. 2.12: Архивация

5. Контрольные вопросы:

- 1) 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 букваданной 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 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 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` и 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 противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.

3 Выводы

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