

Операционные системы

Лабораторная работа №12

Крупенникова Виктория Александровна

Содержание

1	Цель работы	3
2	Задание	4
3	Выполнение лабораторной работы	5
4	Контрольные вопросы	12
5	Выводы	16

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

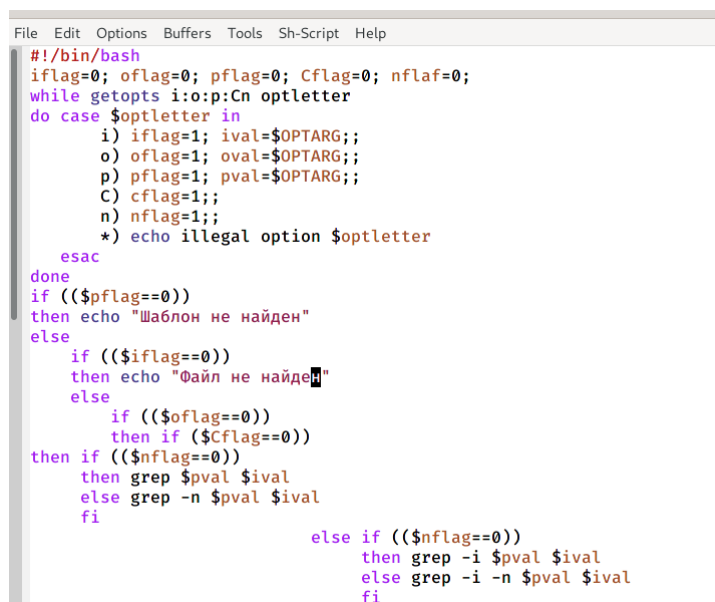
1. Ознакомиться с теоретическим материалом.
2. Изучить основы программирования в оболочке ОС UNIX/Linux.
3. Выполнить упражнения.
4. Ответить на контрольные вопросы.

3 Выполнение лабораторной работы

1)Используя команды `getopts``grep`, написалакомандный файл, который анализирует командную строку с ключами:

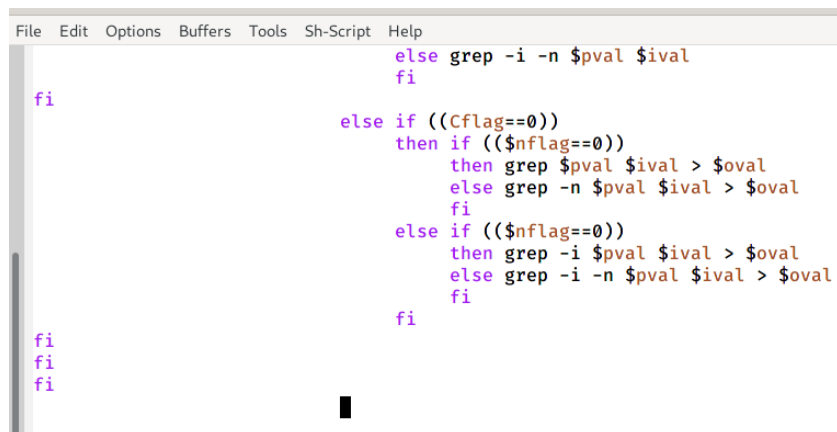
- iinputfile — прочитать данные из указанного файла;
- ooutputfile — вывести данные в указанный файл;
- ршаблон — указать шаблон для поиска;
- C — различать большие и малые буквы;
- n — выдавать номера строк,а затем ищет в указанном файле нужные строки, определяемые ключом -р.

Для данной задачи я создала файл `prog1.sh` и написала соответствующие скрипты.



```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
iflag=0; oflag=0; pflag=0; cflag=0; nflag=0;
while getopts i:op: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
fi
```

Figure 3.1: Первый скрипт



```
File Edit Options Buffers Tools Sh-Script Help
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
```

Figure 3.2: Продолжение скрипта

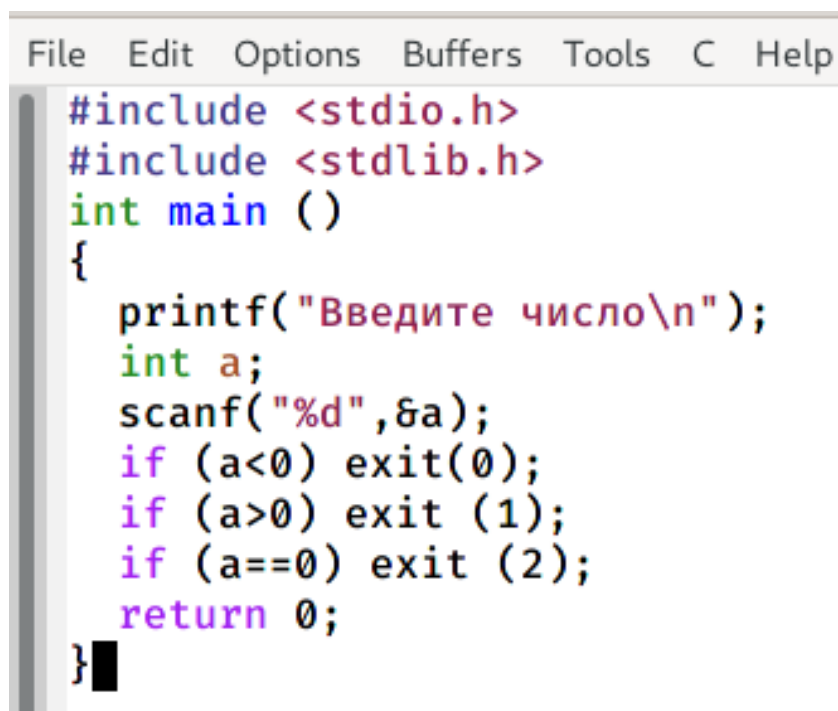
Далее я проверила работу написанного скрипта, используя различные опции (например, команда «./prog.sh-Ia1.txt-oa2.txt-pcapital-C-n»), предварительно добавив право на исполнение файла (команда «chmod+xprog1.sh») и создав 2 файла, которые необходимы для выполнения программы: a1.txtи a2.txt

```
Файл  Правка  Вид  Закладки  Настройка  Справка
vkrupennikova@dk8n75 ~$ touch prog1.sh
vkrupennikova@dk8n75 ~$ emacs &
[1] 3656
vkrupennikova@dk8n75 ~$ touch a1.txt a2.txt
vkrupennikova@dk8n75 ~$ chmod +x prog1.sh
vkrupennikova@dk8n75 ~$ cat a1.txt
vkrupennikova@dk8n75 ~$ cat a1.txt
Hello , WORLD!
vkrupennikova@dk8n75 ~$ ./prog1.sh -i a1.txt -o a2.txt -p hello -C -n
./prog1.sh: строка 20: синтаксическая ошибка рядом с неожиданным маркером «)»
./prog1.sh: строка 20: '      then if ($Cflag==0))'
vkrupennikova@dk8n75 ~$ ./prog1.sh -i a1.txt -o a2.txt -p hello -C -n
vkrupennikova@dk8n75 ~$ cat a2.txt
vkrupennikova@dk8n75 ~$ chmod +x prog1.sh
vkrupennikova@dk8n75 ~$ ./prog1.sh -i a1.txt -o a2.txt -p hello -C -n
vkrupennikova@dk8n75 ~$ cat a2.txt
vkrupennikova@dk8n75 ~$ cat a1.txt
Hello , WORLD!
no
cp
vkrupennikova@dk8n75 ~$ cat a2.txt
vkrupennikova@dk8n75 ~$ ./prog1.sh -i a1.txt -o a2.txt -p hello -C -n
vkrupennikova@dk8n75 ~$ cat a2.txt
vkrupennikova@dk8n75 ~$ cat a2.txt
vkrupennikova@dk8n75 ~$ ./prog1.sh -i a1.txt -o a2.txt -p hello -C -n
vkrupennikova@dk8n75 ~$ ./prog1.sh -i a1.txt -o a2.txt -p Hello -C -n
vkrupennikova@dk8n75 ~$ cat a2.txt
1:Hello , WORLD!
vkrupennikova@dk8n75 ~$ touch chislo.c
vkrupennikova@dk8n75 ~$ touch chislo.sh
vkrupennikova@dk8n75 ~$ emacs &
[2] 9748
vkrupennikova@dk8n75 ~$ emacs &
[3] 10328
[1]  Завершён      emacs
vkrupennikova@dk8n75 ~$ ./prog1.sh -i a1.txt -o a2.txt -p Hello -n
vkrupennikova@dk8n75 ~$ ^C
vkrupennikova@dk8n75 ~$ cat a2.txt
1:Hello , WORLD!
vkrupennikova@dk8n75 ~$ ./prog1.sh -i a1.txt -C -n
Шаблон не найден
vkrupennikova@dk8n75 ~$ ./prog1.sh -i a1.txt -C -n
Шаблон не найден
vkrupennikova@dk8n75 ~$ ./prog1.sh -o a2.txt -p Hello -C -n
Файл не найден
vkrupennikova@dk8n75 ~$
```

Figure 3.3: Проверка скрипта

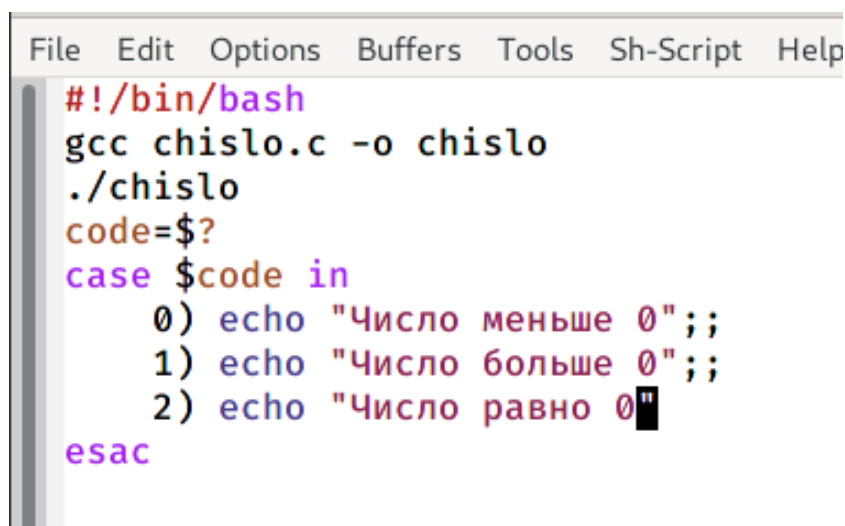
Скрипт работает корректно.

- 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: `chislo.c` и `chislo.sh` и написала соответствующие скрипты.



```
File Edit Options Buffers Tools C Help
#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;
}
```

Figure 3.4: Второй скрипт



```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
gcc chislo.c -o chislo
./chislo
code=$?
case $code in
    0) echo "Число меньше 0" ;;
    1) echo "Число больше 0" ;;
    2) echo "Число равно 0" ;;
esac
```

Figure 3.5: Третий скрипт

Далее я проверила работу написанных скриптов (команда «./chislo.sh»), предварительно добавив право на исполнение файла (команда «chmod+x chislo.sh»)


```
vakrupennikova@dk8n75 ~ $ chmod +x chislo.sh
vakrupennikova@dk8n75 ~ $ ./chislo.sh
Введите число
0
Число равно 0
vakrupennikova@dk8n75 ~ $ ./chislo.sh
Введите число
5
Число больше 0
vakrupennikova@dk8n75 ~ $ ./chislo.sh
Введите число
-5
Число меньше 0
```

Figure 3.6: Проверка скрипта

Скрипты работают корректно.

- 3) Написала 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). Для данной задачи я создала файл: files.sh и написала соответствующий скрипт.

```

File Edit Options Buffers Tools Sh-Script Help
#!/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

```

Figure 3.7: Четвертый скрипт

Далее я проверила работу написанного скрипта (команда «./files.sh»), предварительно добавив право на исполнение файла (команда «chmod+x files.sh»). Сначала я создала три файла (команда «./files.sh-cabc#.txt3»), удовлетворяющие условию задачи, а потом удалила их (команда «./files.sh-rabc#.txt3»)

```

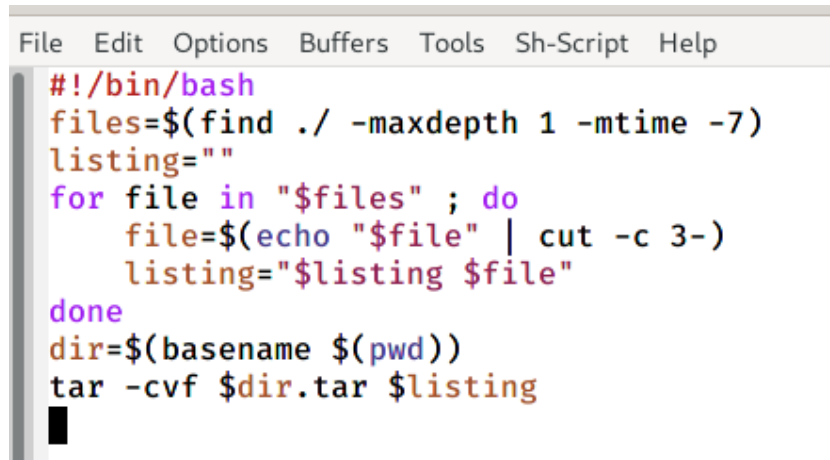
vkrupennikova@dn75 ~$ chmod +x files.sh
vkrupennikova@dn75 ~$ ./files.sh
1.png 3.png 5.png a2.txt chislo.c chislo.sh files.sh lab05 public Видео Изображения проекты
2021-05-29 12:31:36.uv 4.png 7.png backup chislo.c docs GNUstep progrl.sh public_html Загрузки Музыка Общедоступные 'Рабочий стол'
2.png 5.png a1.txt chislo chislo.sh files.sh image progrl.sh tmp
vkrupennikova@dn75 ~$ ./files.sh -C abc#.txt 3
./files.sh: строка 12: [: отсутствует символ *}
./files.sh: строка 12: [: отсутствует символ *}
./files.sh: строка 12: [: отсутствует символ *}
vkrupennikova@dn75 ~$ ./files.sh -C abc#.txt 3
bash: ./files.sh: Нет такого файла или каталога
vkrupennikova@dn75 ~$ ./files.sh -C abc#.txt 3
vkrupennikova@dn75 ~$ ./files.sh -C abc#.txt 3
1.png 3.png 5.png a2.txt abc2.txt chislo.c chislo.sh files.sh lab05 public Видео Изображения проекты
2021-05-29 12:31:36.uv 4.png 7.png abc1.txt backup chislo.c docs GNUstep progrl.sh public_html Загрузки Музыка Общедоступные 'Рабочий стол'
2.png 5.png a1.txt abc1.txt chislo chislo.sh files.sh image progrl.sh tmp
vkrupennikova@dn75 ~$ ./files.sh -r abc#.txt 3
vkrupennikova@dn75 ~$ ./files.sh -r abc#.txt 3
1.png 3.png 5.png a2.txt chislo.c chislo.sh files.sh lab05 public Видео Изображения проекты
2021-05-29 12:31:36.uv 4.png 7.png backup chislo.c docs GNUstep progrl.sh public_html Загрузки Музыка Общедоступные 'Рабочий стол'
2.png 5.png a1.txt chislo chislo.sh files.sh image progrl.sh tmp

```

Figure 3.8: Проверка скрипта

- 4) Написала командный файл, который с помощью команды tar 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:

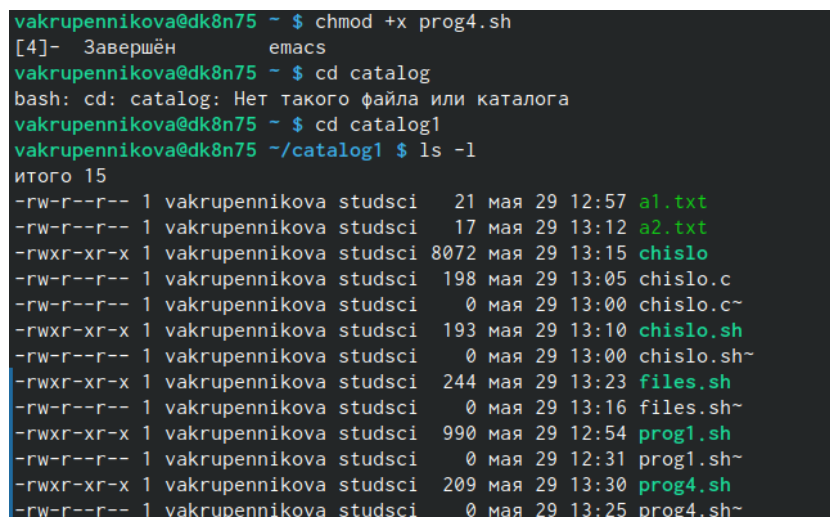
prog4.sh и написала соответствующий скрипт.



```
File Edit Options Buffers Tools Sh-Script Help
#!/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
```

Figure 3.9: Пятый скрипт

Далее я проверила работу написанного скрипта (команды «sudo~/prog4.sh» и «tar-tfCatalog1.tar»), предварительно добавив право на исполнение файла (команда «chmod+xprog4.sh»)и создав отдельный catalog1 с несколькими файлами. Файлы ,измененные более недели назад, заархивированы не были.



```
vakrupennikova@dk8n75 ~ $ chmod +x prog4.sh
[4]- Завершён      emacs
vakrupennikova@dk8n75 ~ $ cd catalog
bash: cd: catalog: Нет такого файла или каталога
vakrupennikova@dk8n75 ~ $ cd catalog1
vakrupennikova@dk8n75 ~/catalog1 $ ls -l
итого 15
-rw-r--r-- 1 vakrupennikova studsci   21 мая 29 12:57 a1.txt
-rw-r--r-- 1 vakrupennikova studsci   17 мая 29 13:12 a2.txt
-rwxr-xr-x 1 vakrupennikova studsci 8072 мая 29 13:15 chislo
-rw-r--r-- 1 vakrupennikova studsci   198 мая 29 13:05 chislo.c
-rw-r--r-- 1 vakrupennikova studsci    0 мая 29 13:00 chislo.c~
-rwxr-xr-x 1 vakrupennikova studsci   193 мая 29 13:10 chislo.sh
-rw-r--r-- 1 vakrupennikova studsci    0 мая 29 13:00 chislo.sh~
-rwxr-xr-x 1 vakrupennikova studsci   244 мая 29 13:23 files.sh
-rw-r--r-- 1 vakrupennikova studsci    0 мая 29 13:16 files.sh~
-rwxr-xr-x 1 vakrupennikova studsci   990 мая 29 12:54 prog1.sh
-rw-r--r-- 1 vakrupennikova studsci    0 мая 29 12:31 prog1.sh~
-rwxr-xr-x 1 vakrupennikova studsci   209 мая 29 13:30 prog4.sh
-rw-r--r-- 1 vakrupennikova studsci    0 мая 29 13:25 prog4.sh~
```

Figure 3.10: Проверка скрипта

Скрипт работает корректно.

4 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 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) 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.

5 Выводы

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