

Отчёт по лабораторной работе номер 12

Операционные системы

Нитусова Диана Денисовна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Контрольные вопросы	15
5	Выводы	19

Список таблиц

Список иллюстраций

3.1	Рисунок 1	8
3.2	Рисунок 2	8
3.3	Рисунок 3	9
3.4	Рисунок 4	10
3.5	Рисунок 5	10
3.6	Рисунок 6	11
3.7	Рисунок 7	12
3.8	Рисунок 8	12
3.9	Рисунок 9	13
3.10	Рисунок 10	14
3.11	Рисунок 11	14

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

Написать командные файлы с использованием логических управляющих конструкций и циклов.

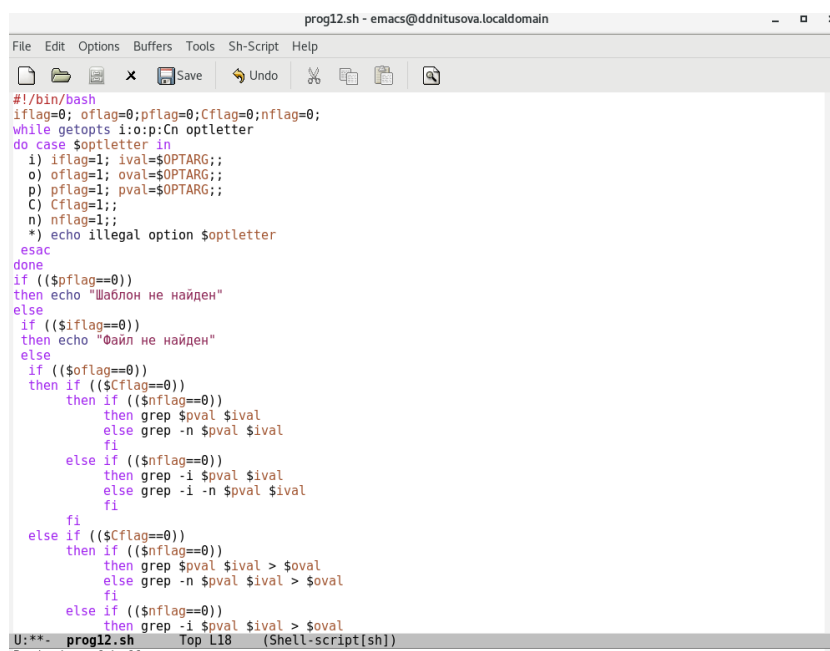
3 Выполнение лабораторной работы

Используя команды `getopts` `grep`, написала командный 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,

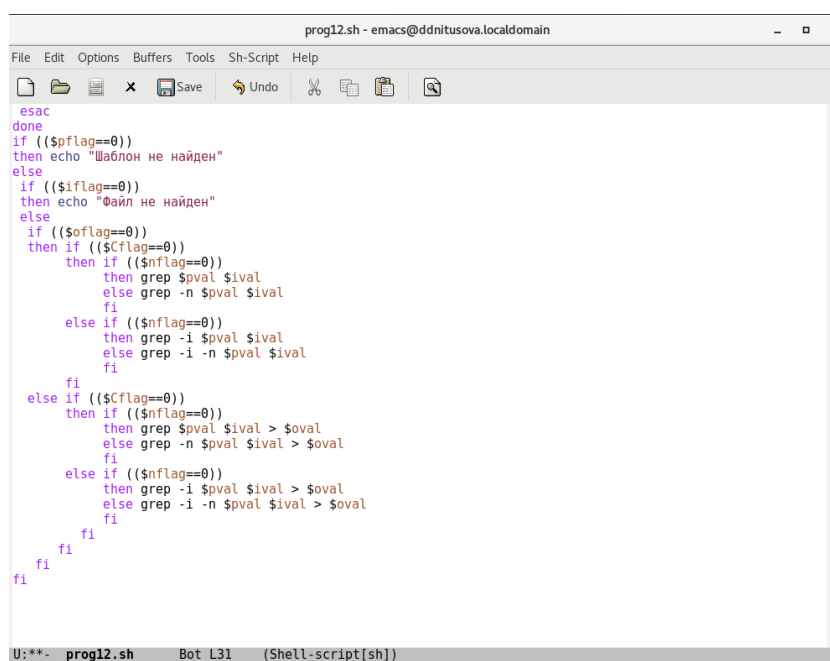
а затем ищет в указанном файле нужные строки, определяемые ключом `-р`.

Для данной задачи я создала файл `prog12.sh` и написала соответствующие скрипты. (рис. -fig. 3.1) (рис. -fig. 3.2).



```
prog12.sh - emacs@ddnitusova.localdomain
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 $ival
            fi
        fi
    else if (($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
U:*** prog12.sh Top L18 (Shell-script[sh])
Renining of buffer
```

Рис. 3.1: Рисунок 1



```
prog12.sh - emacs@ddnitusova.localdomain
File Edit Options Buffers Tools Sh-Script Help
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
fi
fi
U:*** prog12.sh Bot L31 (Shell-script[sh])
```

Рис. 3.2: Рисунок 2

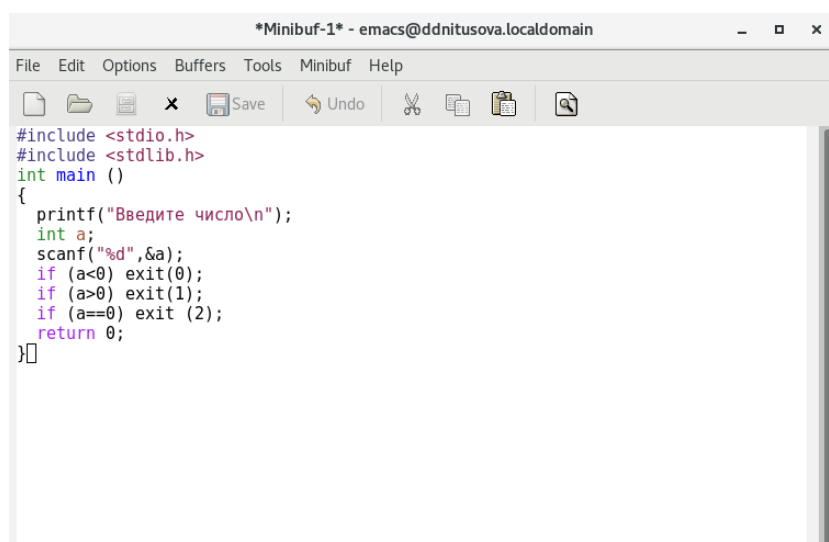
Далее я проверила работу написанного скрипта, используя различные опции (например, команда «./prog12.sh -I a1.txt -o a2.txt -p capital -C -n»), предварительно добавив право на исполнение файла (команда «chmod +x prog12.sh») и

создав 2 файла, которые необходимы для выполнения программы: a1.txt и a2.txt. Скрипт работает корректно (рис. -fig. 3.3).

```
[ddnitusova@ddnitusova ~]$ touch prog12.sh
[ddnitusova@ddnitusova ~]$ emacs &
[1] 3103
[ddnitusova@ddnitusova ~]$ touch a1.txt a2.txt
[1]+  Done                  emacs
[ddnitusova@ddnitusova ~]$ chmod +x prog12.sh
[ddnitusova@ddnitusova ~]$ cat a1.txt
[ddnitusova@ddnitusova ~]$ cat a1.txt
Moscow is the capital of Russia
Paris is the capital of France
Rome is not the CAPITAL of Cananda
[ddnitusova@ddnitusova ~]$ ./prog12.sh -i a1.txt -o a2.txt -p capital -C -n
[ddnitusova@ddnitusova ~]$ cat a2.txt
1:Moscow is the capital of Russia
2:Paris is the capital of France
3:Rome is not the CAPITAL of Cananda
[ddnitusova@ddnitusova ~]$ ./prog12.sh -i a1.txt -o a2.txt -p capital -n
[ddnitusova@ddnitusova ~]$ cat a2.txt
1:Moscow is the capital of Russia
2:Paris is the capital of France
[ddnitusova@ddnitusova ~]$ ./prog12.sh -i a1.txt -C -n
Шаблон не найден
[ddnitusova@ddnitusova ~]$ ./prog12.sh -o a2.txt -p capital -C -n
Файл не найден
[ddnitusova@ddnitusova ~]$
```

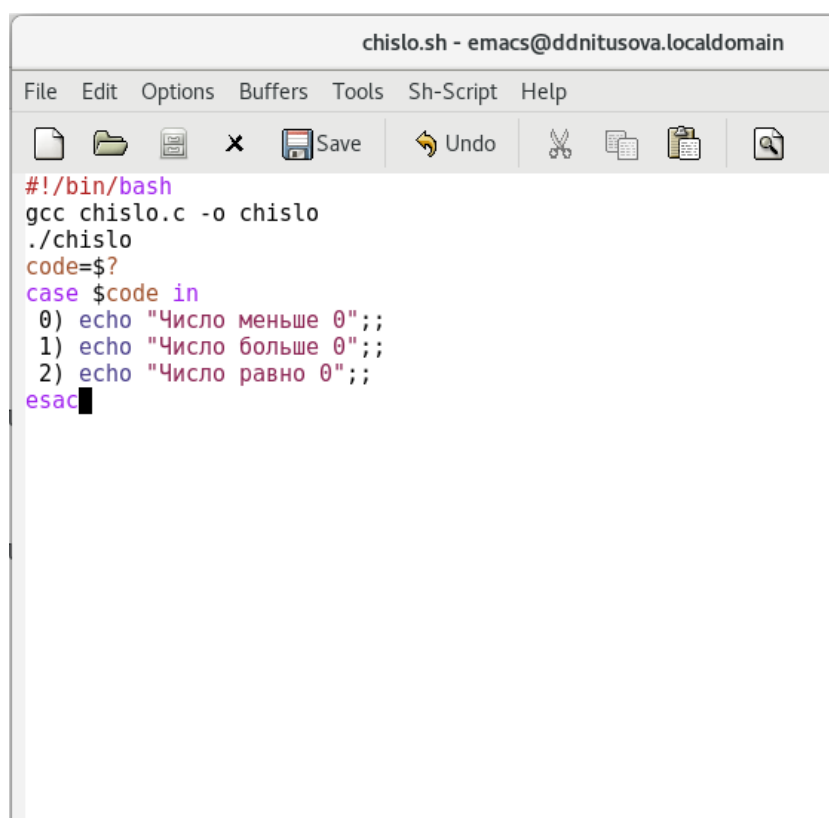
Рис. 3.3: Рисунок 3

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. (рис. -fig. 3.4) (рис. -fig. 3.5).



```
#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. 3.4: Рисунок 4



```
#!/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
```

Рис. 3.5: Рисунок 5

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

Скрипты работают корректно (рис. -fig. 3.6).

```
[ddnitusova@ddnitusova ~]$ chmod +x chislo.sh
[1]+  Done                  emacs
[ddnitusova@ddnitusova ~]$ ./chislo.sh
Введите число
0
Число равно 0
[ddnitusova@ddnitusova ~]$ ./chislo.sh
Введите число
1
Число больше 0
[ddnitusova@ddnitusova ~]$ ./chislo.sh
Введите число
-1
Число меньше 0
[ddnitusova@ddnitusova ~]$ █
```

Рис. 3.6: Рисунок 6

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 (рис. -fig. 3.7).

```

files.sh - emacs@ddnitusova.localdomain
File Edit Options Buffers Tools Sh-Script Help
[Icons: New, Open, Save, Undo, Cut, Copy, Paste, Find]

#!/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

```

Рис. 3.7: Рисунок 7

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 -c hi#.txt 3»), удовлетворяющие условию задачи, а потом удалила их (команда «./files.sh -r hi#.txt 3») (рис. -fig. 3.8).

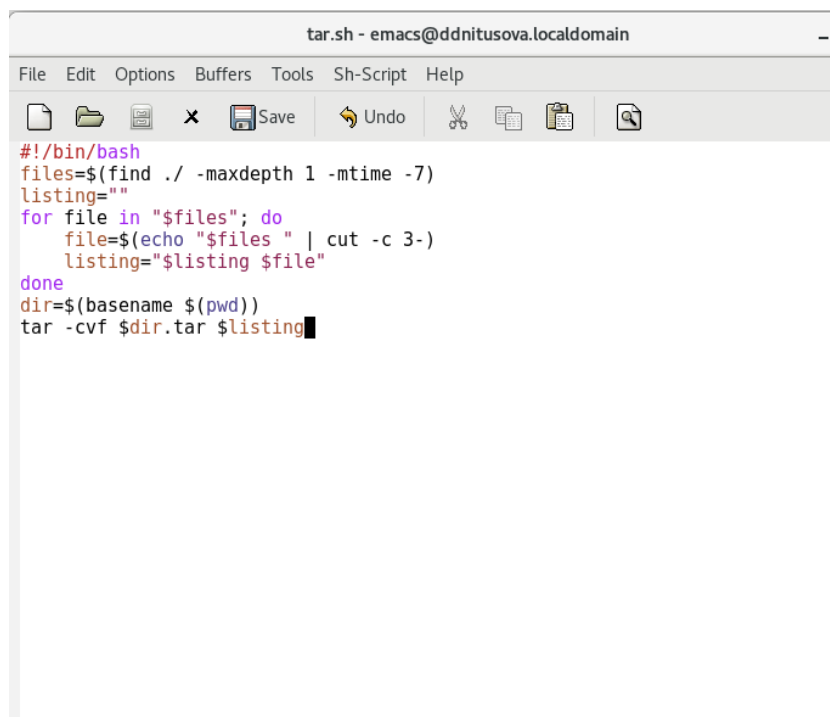
```

[ddnitusova@ddnitusova ~]$ ls
a1.txt  backup.sh-  feathers  lab05  lab09.4.txt  play  ski.plases  Загрузки
a2.txt  chislo     files.sh  #lab07.sh#  may  prog12.sh  text.txt  Изображения
abc     chislo.c  files.sh-  lab07.sh  mewdir  prog12.sh-  usr  Музыка
abc1    chislo.c-  file.txt  lab07.sh-  monthly  prog2.sh  work  Общедоступные
australia chislo.sh  format.sh  lab09.1.txt  my_os  proglis.sh  work.  Рабочий стол
backup  chislo.sh-  format.sh-  lab09.2.txt  nmdir1  prog12.sh-  Video  Шаблоны
backup.sh conf.txt  lab03  lab09.3.txt  OS  reports  Документы
[ddnitusova@ddnitusova ~]$ emacs &
[1] 4027
[ddnitusova@ddnitusova ~]$ ./files.sh -c hi#.txt 3
[ddnitusova@ddnitusova ~]$ ls
a1.txt  backup.sh-  feathers  hi2.txt  lab09.1.txt  my_os  proglis.sh  work.  Рабочий стол
a2.txt  chislo     files.sh  hi3.txt  lab09.2.txt  OS  prog12.sh-  Video  Шаблоны
abc     chislo.c  files.sh-  lab03  lab09.3.txt  OS  reports  Документы
abc1    chislo.c-  file.txt  lab05  lab09.4.txt  play  ski.plases  Загрузки
australia chislo.sh  format.sh  #lab07.sh#  may  prog12.sh  text.txt  Изображения
backup  chislo.sh-  format.sh-  lab07.sh  mewdir  prog12.sh-  usr  Музыка
backup.sh conf.txt  h11.txt  lab07.sh-  monthly  prog2.sh  work  Общедоступные
[ddnitusova@ddnitusova ~]$ ./files.sh -r hi#.txt 3
[ddnitusova@ddnitusova ~]$ emacs
[1]+  Done
[ddnitusova@ddnitusova ~]$ ls
a1.txt  backup.sh-  feathers  lab05  lab09.4.txt  play  ski.plases  Загрузки
a2.txt  chislo     files.sh  #lab07.sh#  may  prog12.sh  text.txt  Изображения
abc     chislo.c  files.sh-  lab07.sh  mewdir  prog12.sh-  usr  Музыка
abc1    chislo.c-  file.txt  lab07.sh-  monthly  prog2.sh  work  Общедоступные
australia chislo.sh  format.sh  lab09.1.txt  my_os  proglis.sh  work.  Рабочий стол
backup  chislo.sh-  format.sh-  lab09.2.txt  nmdir1  prog12.sh-  Video  Шаблоны
backup.sh conf.txt  lab03  lab09.3.txt  OS  reports  Документы
[ddnitusova@ddnitusova ~]$

```

Рис. 3.8: Рисунок 8

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: tar.sh и написала соответствующий скрипт (рис. -fig. 3.9).



```
tar.sh - emacs@ddnitusova.localdomain
File Edit Options Buffers Tools Sh-Script Help
[Icons: New, Open, Save, Undo, Cut, Copy, Paste, Find]

#!/bin/bash
files=$(find ./ -maxdepth 1 -mtime -7)
listing=""
for file in "$files"; do
    file=$(echo "$files" | cut -c 3-)
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
```

Рис. 3.9: Рисунок 9

Далее я проверила работу написанного скрипта (команды «sudo ~/tar.sh»), предварительно добавив право на исполнение файла (команда «chmod +x tar.sh»). Скрипт работает корректно (рис. -fig. 3.10) (рис. -fig. 3.11).

```

[ddnitusova@ddnitusova ~]$ ls -l
итого 88
-rw-rw-r--. 1 ddnitusova ddnitusova 98 май 28 00:41 a1.txt
-rw-rw-r--. 1 ddnitusova ddnitusova 67 май 28 00:46 a2.txt
drwxrwxr-x. 5 ddnitusova ddnitusova 50 май 15 01:54 abc
-rw-rw-r--. 1 ddnitusova ddnitusova 0 май 14 22:04 abcl
drwxr--r--. 2 ddnitusova ddnitusova 6 май 14 22:11 australia
drwxrwxr-x. 2 ddnitusova ddnitusova 43 май 26 11:10 backup
-rwxrwxr-x. 1 ddnitusova ddnitusova 113 май 26 11:07 backup.sh
-rw-rw-r--. 1 ddnitusova ddnitusova 0 май 26 11:01 backup.sh-
-rwxrwxr-x. 1 ddnitusova ddnitusova 8504 май 28 00:58 chislo
-rw-rw-r--. 1 ddnitusova ddnitusova 197 май 28 00:54 chislo.c
-rw-rw-r--. 1 ddnitusova ddnitusova 0 май 28 00:49 chislo.c~
-rwxrwxr-x. 1 ddnitusova ddnitusova 186 май 28 00:57 chislo.sh
-rw-rw-r--. 1 ddnitusova ddnitusova 0 май 28 00:49 chislo.sh-
-rw-rw-r--. 1 ddnitusova ddnitusova 594 май 14 23:31 conf.txt
-rw-rw-r--. 1 ddnitusova ddnitusova 0 май 14 22:12 feathers
-rwxrwxr-x. 1 ddnitusova ddnitusova 261 май 28 01:12 files.sh
-rwxrwxr-x. 1 ddnitusova ddnitusova 251 май 28 01:06 files.sh~
-rw-rw-r--. 1 ddnitusova ddnitusova 3134 май 14 23:27 file.txt
-rwxrwxr-x. 1 ddnitusova ddnitusova 225 май 26 11:29 format.sh
-rw-rw-r--. 1 ddnitusova ddnitusova 0 май 26 11:24 format.sh-
drwxrwxr-x. 3 ddnitusova ddnitusova 18 май 1 15:53 lab03
drwxrwxr-x. 3 ddnitusova ddnitusova 18 май 13 23:54 lab05
-rw-rw-r--. 1 ddnitusova ddnitusova 120 май 16 23:57 #lab07.sh#
-rwxrwxr-x. 1 ddnitusova ddnitusova 119 май 16 23:19 lab07.sh

```

Рис. 3.10: Рисунок 10

```

drwxr-xr-x. 4 ddnitusova ddnitusova 0 апр 28 19:00 шаблоны
[ddnitusova@ddnitusova ~]$ sudo ~/tar.sh
[sudo] пароль для ddnitusova:
ICEAuthority
Изображения/
Изображения/Снимок экрана от 2021-05-16 22-56-30.png
Изображения/Снимок экрана от 2021-05-16 23-54-00.png
Изображения/Снимок экрана от 2021-05-26 11-00-20.png
Изображения/Снимок экрана от 2021-05-26 11-23-50.png
Изображения/Снимок экрана от 2021-05-01 19-07-38.png
Изображения/Снимок экрана от 2021-05-01 19-14-04.png
Изображения/Снимок экрана от 2021-05-01 19-25-57.png
Изображения/Снимок экрана от 2021-05-13 21-26-36.png
Изображения/Снимок экрана от 2021-05-13 21-28-05.png
Изображения/Снимок экрана от 2021-05-13 21-28-51.png
Изображения/Снимок экрана от 2021-05-13 21-29-49.png
Изображения/Снимок экрана от 2021-05-13 21-30-22.png
Изображения/Снимок экрана от 2021-05-13 21-45-54.png
Изображения/Снимок экрана от 2021-05-13 21-45-58.png
Изображения/Снимок экрана от 2021-05-13 21-49-06.png
Изображения/Снимок экрана от 2021-05-13 23-19-22.png
Изображения/Снимок экрана от 2021-05-14 23-52-29.png
Изображения/Снимок экрана от 2021-05-15 01-43-46.png
Изображения/Снимок экрана от 2021-05-15 01-44-02.png
Изображения/Снимок экрана от 2021-05-15 01-44-39.png
Изображения/Снимок экрана от 2021-05-15 01-45-50.png
Изображения/Снимок экрана от 2021-05-15 01-46-11.png
Изображения/Снимок экрана от 2021-05-15 01-46-22.png
Изображения/Снимок экрана от 2021-05-15 01-46-38.png
Изображения/Снимок экрана от 2021-05-15 01-47-18.png
Изображения/Снимок экрана от 2021-05-15 01-51-04.png
Изображения/Снимок экрана от 2021-05-15 01-54-38.png
Изображения/Снимок экрана от 2021-05-15 01-56-29.png
Изображения/Снимок экрана от 2021-05-15 02-03-24.png
Изображения/Снимок экрана от 2021-05-15 02-05-09.png
Изображения/Снимок экрана от 2021-05-15 02-05-38.png
Изображения/Снимок экрана от 2021-05-15 02-07-17.png
Изображения/Снимок экрана от 2021-05-15 02-07-53.png
Изображения/Снимок экрана от 2021-05-15 02-08-04.png
Изображения/Снимок экрана от 2021-05-15 02-09-49.png
Изображения/Снимок экрана от 2021-05-15 02-13-42.png
Изображения/Снимок экрана от 2021-05-15 02-22-56.png
Изображения/Снимок экрана от 2021-05-15 02-30-41.png
Изображения/Снимок экрана от 2021-05-15 22-04-15.png
Изображения/Снимок экрана от 2021-05-15 22-04-23.png
Изображения/Снимок экрана от 2021-05-15 22-14-26.png

```

Рис. 3.11: Рисунок 11

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