

# Отчет по лабораторной работе №12

## Тема:

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Российский Университет Дружбы Народов

Факультет Физико-Математических и Естественных Наук

Дисциплина: *Операционные системы*

Студент: Алхатиб Осама

Группа: НПИбд-02-20

2021г.

## Цель работы

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

## Введение

Операции с блоками кода — ключ к структурированному и упорядоченному сценарию оболочки. Циклы и ветвления являются теми инструментальными средствами, которые предоставляют возможность достигнуть этой цели.

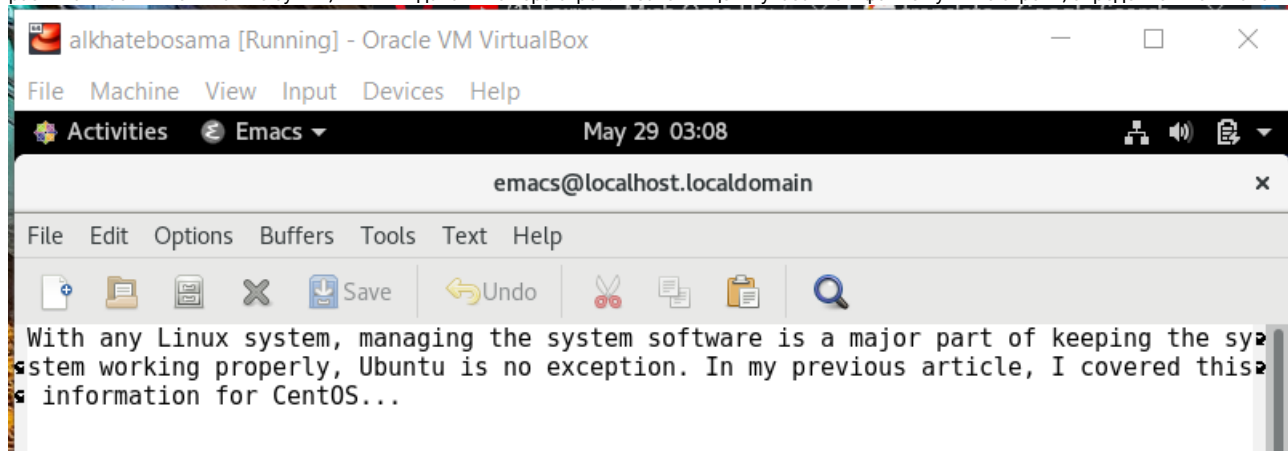
## Циклы

Цикл это блок команд, который повторяется (итерируется) до тех пор, пока не будет выполнено условие выхода из цикла .

## Ход работы:

1. Используя команды `getopts` `grep`, написал 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; \* -С — различать большие и малые буквы; \* -n — выдавать номера строк. А затем ищет в указанном файле нужные строки, определяемые ключом -р.



alkhatebosama [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Emacs May 29 03:09

emacs@localhost.localdomain

File Edit Options Buffers Tools Sh-Script Help

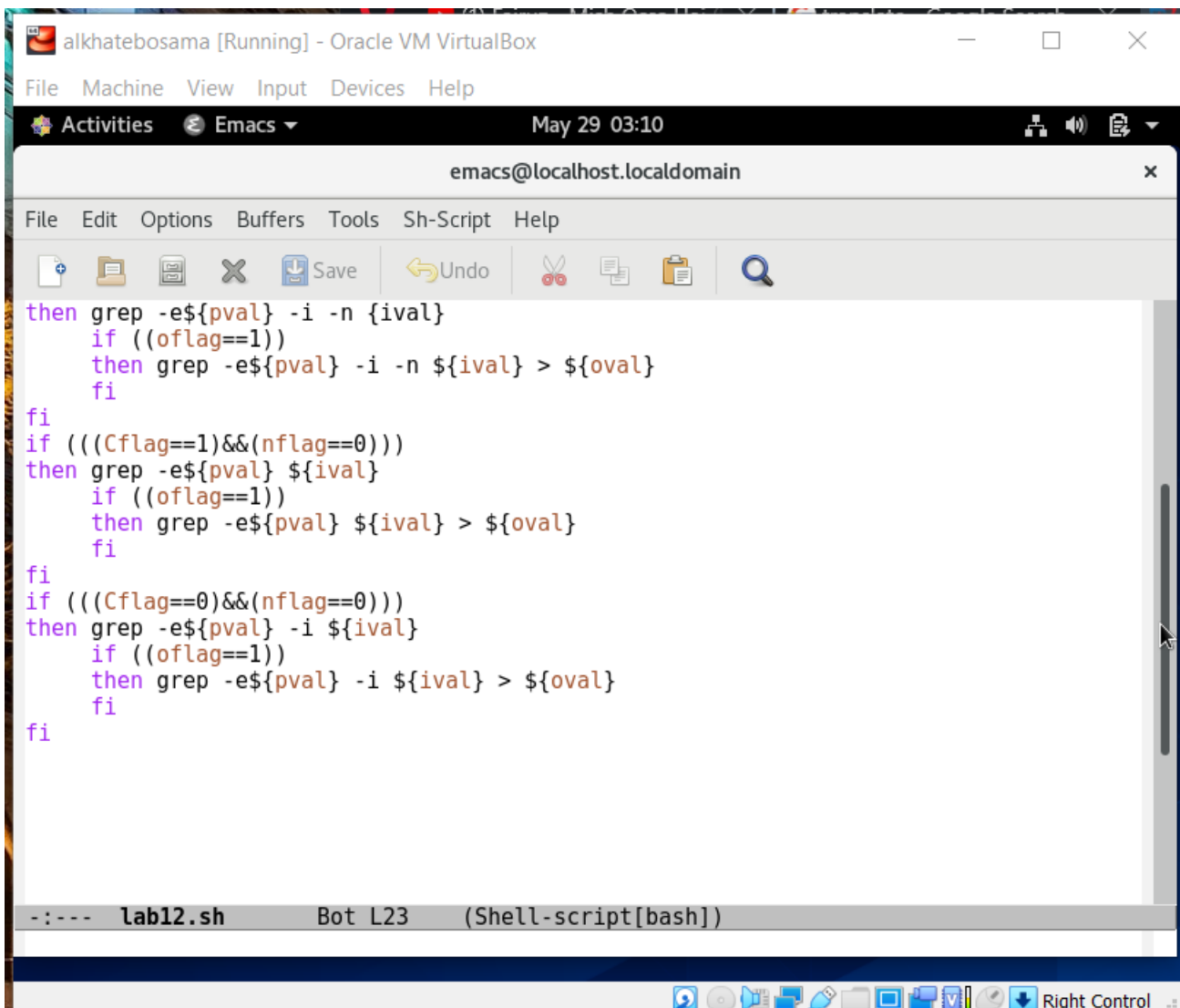
Save Undo

```
#!/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 number nebravtlna vvod $optletter
    esac
done
if (((Cflag=1)&&(nflag=1)))
then grep -e${pval} -n ${ival}
    if ((oflag==1))
    then grep -e${pval} -n ${ival} > ${oval}
    fi
fi
if (((Cflag==0)&&(nflag==1)))
then grep -e${pval} -i -n {ival}
    if ((oflag==1))
    then grep -e${pval} -i -n ${ival} > ${oval}
    fi
fi
fi
```

-:--- lab12.sh Top L23 (Shell-script[bash])

Beginning of buffer

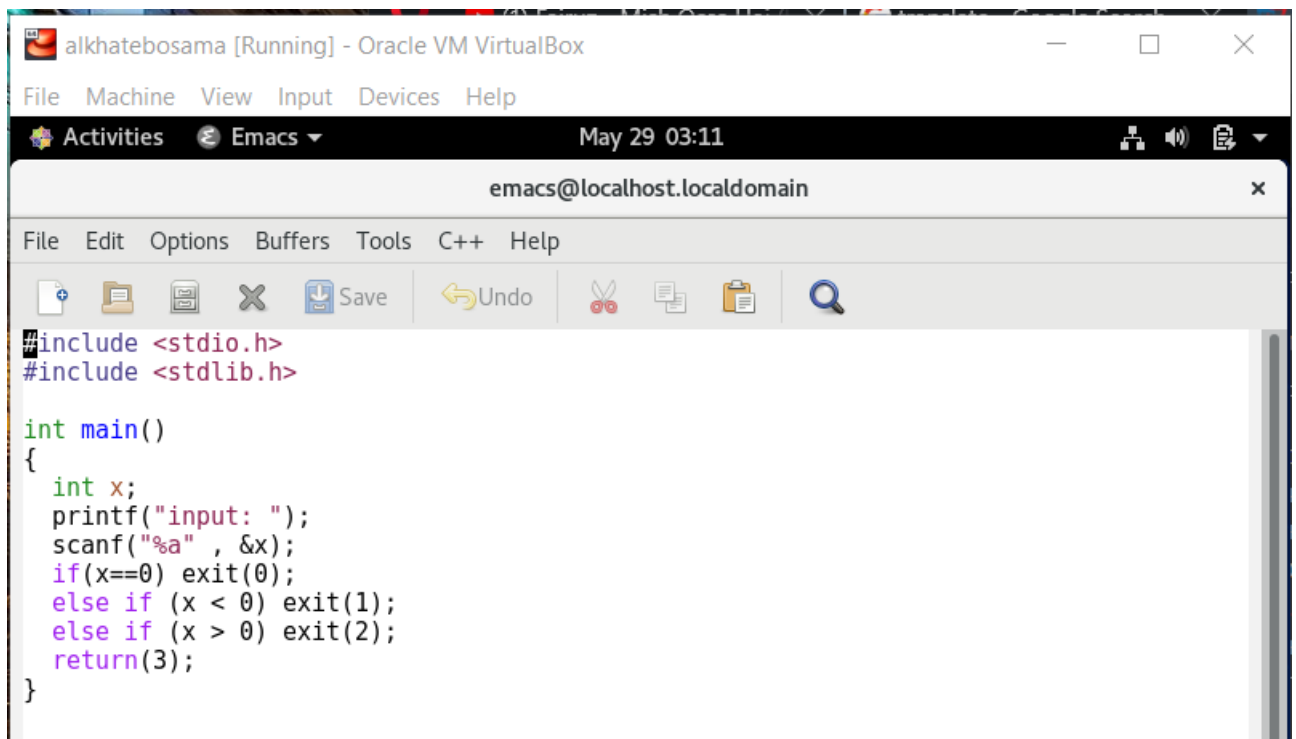
Right Control



```
then grep -e${pval} -i -n {ival}
  if ((oflag==1))
  then grep -e${pval} -i -n ${ival} > ${oval}
  fi
fi
if (((Cflag==1)&&(nflag==0)))
then grep -e${pval} ${ival}
  if ((oflag==1))
  then grep -e${pval} ${ival} > ${oval}
  fi
fi
if (((Cflag==0)&&(nflag==0)))
then grep -e${pval} -i ${ival}
  if ((oflag==1))
  then grep -e${pval} -i ${ival} > ${oval}
  fi
fi
```

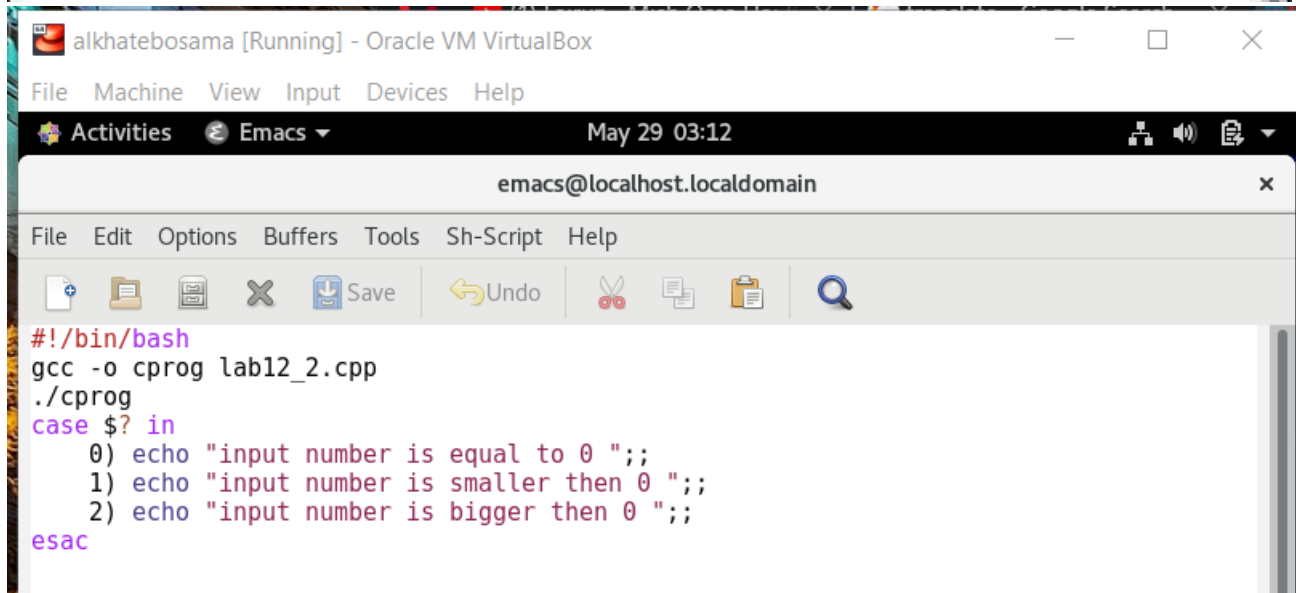
```
[alkahtebosama@localhost ~]$ bash lab12.sh -ilab12.txt -olab12_2.txt -ppart
1:With any Linux system, managing the system software is a major part of keeping
the system working properly, Ubuntu is no exception. In my previous article, I
covered this information for CentOS...
[alkahtebosama@localhost ~]$ bash lab12.sh -ilab12.txt -olab12_2.txt -ppart -n
1:With any Linux system, managing the system software is a major part of keeping
the system working properly, Ubuntu is no exception. In my previous article, I
covered this information for CentOS...
[alkahtebosama@localhost ~]$ bash lab12.sh -ilab12.txt -olab12_2.txt -ppart -C
1:With any Linux system, managing the system software is a major part of keeping
the system working properly, Ubuntu is no exception. In my previous article, I
covered this information for CentOS...
```

2. Написал 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 о коде завершения в оболочку. Командный файл вызывает эту программу и, проанализировав с помощью команды `?`, выдает сообщение о том, какое число было введено.

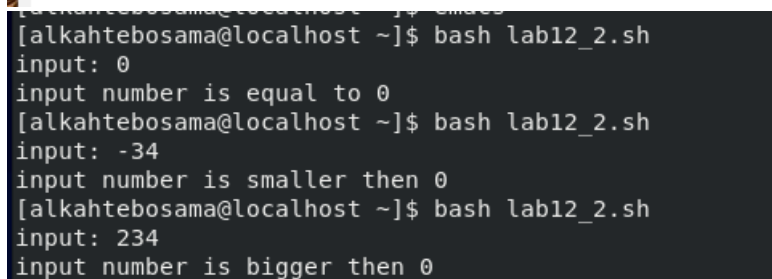


```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int x;
    printf("input: ");
    scanf("%a", &x);
    if(x==0) exit(0);
    else if (x < 0) exit(1);
    else if (x > 0) exit(2);
    return(3);
}
```



```
#!/bin/bash
gcc -o cprog lab12_2.cpp
./cprog
case $? in
    0) echo "input number is equal to 0 ";;
    1) echo "input number is smaller then 0 ";;
    2) echo "input number is bigger then 0 ";;
esac
```



```
[alkhatebosama@localhost ~]$ bash lab12_2.sh
input: 0
input number is equal to 0
[alkhatebosama@localhost ~]$ bash lab12_2.sh
input: -34
input number is smaller then 0
[alkhatebosama@localhost ~]$ bash lab12_2.sh
input: 234
input number is bigger then 0
```

3. Написал 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 удаляет все созданные им файлы (если они существуют).

alkhaitebosama [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Emacs May 29 03:13

emacs@localhost.localdomain

File Edit Options Buffers Tools Sh-Script Help

```
#!/bin/bash
let dflag=0;
while getopts a:d optletter
do case $optletter in
    a) aflag=1; aval=$OPTARG;;
    d) dflag=1;;
    *) echo number nebravelna vvod $optletter
    esac
done
#echo ${aval}
if ((dflag == 0 ))
then for ((i = 1; i<= aval; i++))
do rm ${i}.txt
done
fi
if((dflah == 1))
then for ((i = 1; i <= aval; i++))
do rm ${i}.txt
done
fi
```

-:--- lab12\_3.sh All L1 (Shell-script[bash])

Indentation setup for shell type bash

Right Control

alkhaitebosama@localhost:~

File Edit View Search Terminal Help

```
[alkhaitebosama@localhost ~]$ ls -l
total 120
drwxrwxr-x. 2 alkaitebosama alkaitebosama 43 May 26 15:58 backup
drwxrwxr-x. 2 alkaitebosama alkaitebosama 23 May 26 15:54 baskup
-rwxrwxr-x. 1 alkaitebosama alkaitebosama 12848 May 29 02:50 cprog
drwxr-xr-x. 2 alkaitebosama alkaitebosama 6 May 26 14:20 Desktop
drwxr-xr-x. 2 alkaitebosama alkaitebosama 6 May 26 14:20 Documents
drwxr-xr-x. 2 alkaitebosama alkaitebosama 6 May 26 14:20 Downloads
-rw-rw-r--. 1 alkaitebosama alkaitebosama 0 May 26 15:46 lab11_0.sh~
-rwxrwxr-x. 1 alkaitebosama alkaitebosama 84 May 26 15:57 lab11_1.sh
-rw-rw-r--. 1 alkaitebosama alkaitebosama 0 May 26 15:56 lab11_1.sh~
-rwxrwxr-x. 1 alkaitebosama alkaitebosama 41 May 26 16:00 lab11_2.sh
-rw-rw-r--. 1 alkaitebosama alkaitebosama 0 May 26 15:58 lab11_2.sh~
-rwxrwxr-x. 1 alkaitebosama alkaitebosama 248 May 26 16:03 lab11_3.sh
-rw-rw-r--. 1 alkaitebosama alkaitebosama 0 May 26 16:01 lab11_3.sh~
-rwxrwxr-x. 1 alkaitebosama alkaitebosama 162 May 26 16:06 lab11_4.sh
-rw-rw-r--. 1 alkaitebosama alkaitebosama 0 May 26 16:04 lab11_4.sh~
-rw-rw-r--. 1 alkaitebosama alkaitebosama 0 May 26 15:50 lab11.sh~
-rw-rw-r--. 1 alkaitebosama alkaitebosama 192 May 29 02:47 lab12_2.cpp
-rw-rw-r--. 1 alkaitebosama alkaitebosama 0 May 29 02:44 lab12_2.cpp~
-rw-rw-r--. 1 alkaitebosama alkaitebosama 200 May 29 02:50 lab12_2.sh
-rw-rw-r--. 1 alkaitebosama alkaitebosama 0 May 29 02:47 lab12_2.sh~
-rw-rw-r--. 1 alkaitebosama alkaitebosama 199 May 29 02:43 lab12_2.txt
-rw-rw-r--. 1 alkaitebosama alkaitebosama 377 May 29 02:58 lab12_3.sh
```

```
alkahtebosama@localhost:~  
File Edit View Search Terminal Help  
[alkahtebosama@localhost ~]$ bash lab12_3.sh -a4 -d  
[alkahtebosama@localhost ~]$ ls -l  
total 120  
drwxrwxr-x. 2 alkahtebosama alkahtebosama 43 May 26 15:58 backup  
drwxrwxr-x. 2 alkahtebosama alkahtebosama 23 May 26 15:54 baskup  
-rwxrwxr-x. 1 alkahtebosama alkahtebosama 12848 May 29 02:50 cprog  
drwxr-xr-x. 2 alkahtebosama alkahtebosama 6 May 26 14:20 Desktop  
drwxr-xr-x. 2 alkahtebosama alkahtebosama 6 May 26 14:20 Documents  
drwxr-xr-x. 2 alkahtebosama alkahtebosama 6 May 26 14:20 Downloads  
-rw-rw-r--. 1 alkahtebosama alkahtebosama 0 May 26 15:46 lab11_0.sh~  
-rwxrwxr-x. 1 alkahtebosama alkahtebosama 84 May 26 15:57 lab11_1.sh  
-rw-rw-r--. 1 alkahtebosama alkahtebosama 0 May 26 15:56 lab11_1.sh~  
-rwxrwxr-x. 1 alkahtebosama alkahtebosama 41 May 26 16:00 lab11_2.sh  
-rw-rw-r--. 1 alkahtebosama alkahtebosama 0 May 26 15:58 lab11_2.sh~  
-rwxrwxr-x. 1 alkahtebosama alkahtebosama 248 May 26 16:03 lab11_3.sh  
-rw-rw-r--. 1 alkahtebosama alkahtebosama 0 May 26 16:01 lab11_3.sh~  
-rwxrwxr-x. 1 alkahtebosama alkahtebosama 162 May 26 16:06 lab11_4.sh  
-rw-rw-r--. 1 alkahtebosama alkahtebosama 0 May 26 16:04 lab11_4.sh~  
-rw-rw-r--. 1 alkahtebosama alkahtebosama 0 May 26 15:50 lab11.sh~  
-rw-rw-r--. 1 alkahtebosama alkahtebosama 192 May 29 02:47 lab12_2.cpp  
-rw-rw-r--. 1 alkahtebosama alkahtebosama 0 May 29 02:44 lab12_2.cpp~  
-rw-rw-r--. 1 alkahtebosama alkahtebosama 200 May 29 02:50 lab12_2.sh  
-rw-rw-r--. 1 alkahtebosama alkahtebosama 0 May 29 02:47 lab12_2.sh~  
-rw-rw-r--. 1 alkahtebosama alkahtebosama 199 May 29 02:43 lab12_2.txt
```

4. Написал командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории. Модифицировал его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовал команду `find`).

```
alkahtebosama [Running] - Oracle VM VirtualBox  
File Machine View Input Devices Help  
Activities Emacs May 29 03:15  
emacs@localhost.localdomain  
File Edit Options Buffers Tools Sh-Script Help  
#!/bin/bash  
tar -cf lab12_4_l.tar $@  
find $@ -mtime -7 -exec tar -rf lab12_4_l_modif.tar {} \;
```

```
alkahtebosama@localhost:~  
File Edit View Search Terminal Help  
[alkahtebosama@localhost ~]$ ls  
backup      lab11_3.sh  lab12_3.sh~  lab130_5.sh~  lab19_2.sh  
baskup      lab11_3.sh~ lab12_4.sh   lab130_6.sh   lab19.sh  
cp prog     lab11_4.sh  lab12_4.sh~  lab130_6.sh~  lab12_2.txt~  
Desktop     lab11_4.sh~ lab12.sh     lab130_7.l.tar lab19_2.txt  
Documents   lab11.sh~   lab12.sh~    lab130_7.sh   Music  
Downloads   lab12_2.cpp lab12.txt~   lab130_7.sh~  osama  
lab11_0.sh~ lab12_2.cpp~ lab12.txt~   lab130.sh     Pictures  
lab11_1.sh  lab12_2.sh  lab130_2     lab130.sh~    Public  
lab11_1.sh~ lab12_2.sh~ lab130_5.cpp lab130.txt~   Templates  
lab11_2.sh  lab12_2.txt lab130_5.cpp~ lab130.txt~   Videos  
lab11_2.sh~ lab12_3.sh  lab130_5.sh  lab19_2.cpp
```

## Вывод

Изучил основы программирования в оболочке ОС UNIX, научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

---

## Ответы на контрольные вопросы:

---

1. Команда `getopts` является встроенной командой командной оболочки `bash`, предназначенной для разбора параметров сценариев. Она обрабатывает исключительно однобуквенные параметры как с аргументами, так и без них и этого вполне достаточно для передачи сценариям любых входных данных.
2. При генерации имен используют метасимволы:

\* произвольная (возможно пустая) последовательность символов; ? один произвольный символ; [...] любой из символов, указанных в скобках перечислением и/или с указанием диапазона; `cat f*` выдаст все файлы каталога, начинающиеся с "f"; `cat f` выдаст все файлы, содержащие "f"; `cat program.?` выдаст файлы данного каталога с однобуквенными расширениями, скажем "program.c" и "program.o", но не выдаст "program.com"; `cat [a-d]*` выдаст файлы, которые начинаются с "a", "b", "c", "d". Аналогичный эффект дадут и команды "`cat [abcd]`" и "`cat [bdac]`". 3. Операторы `&&` и `||` являются управляющими операторами. Если в командной строке стоит `command1 && command2`, то `command2` выполняется в том, и только в том случае, если статус выхода из команды `command1` равен нулю, что говорит об успешном ее завершении. Аналогично, если командная строка имеет вид `command1 || command2`, то команда `command2` выполняется тогда, и только тогда, когда статус выхода из команды `command1` отличен от нуля. 4. Оператор `break` завершает выполнение ближайшего включающего цикла или условного оператора, в котором он отображается. 5. Команда `true` всегда возвращает ноль в качестве выходного статуса для индикации успеха. Команда `false` всегда возвращает не-ноль в качестве выходного статуса для индикации неудачи. Во всех управляющих конструкциях в качестве логического значения используется код возврата из программы, указанной в качестве условия. Код возврата 0 – истина, любое другое значение – ложь. Программа `true` – всегда завершается с кодом 0, `false` – всегда завершается с кодом 1. 6. Введенная строка означает условие существования файла `man${s}/${i}.${s}`. 7. Цикл `While` выполняется до тех пор, пока указанное в нем условие истинно. Когда указанное условие становится ложным - цикл завершается. Цикл `Until` выполняется до тех пор, пока указанное в нем условие ложно.