

# 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: Довлетмурат Байрамгельдыев

Группа: НФИбд-03-20

Москва, 2021г.

---

#### Цель работы

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

---

#### Введение

Ветвление (условная инструкция) - это конструкция языка программирования, обеспечивающая выполнение определённой команды или набора команд только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд (наборов команд) в зависимости от значения некоторого выражения.

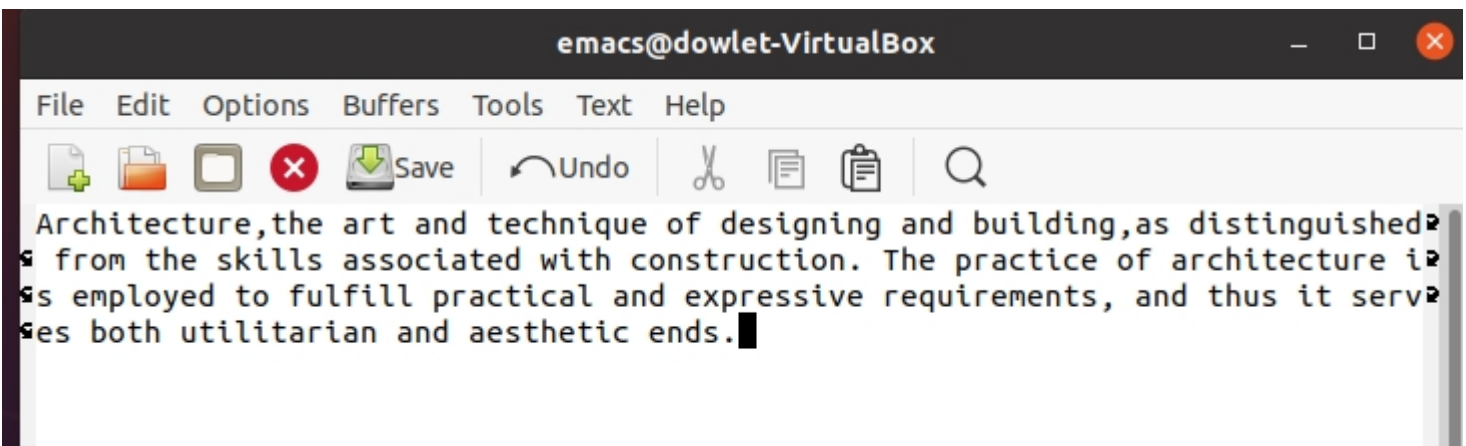
Цикл — это разновидность управляющей конструкции, предназначенная для организации многократного исполнения набора инструкций.

В основе ветвления и отдельных циклов лежат логические операторы сравнения, определяющие необходимость выполнения следующих строк кода или перехода к другим.

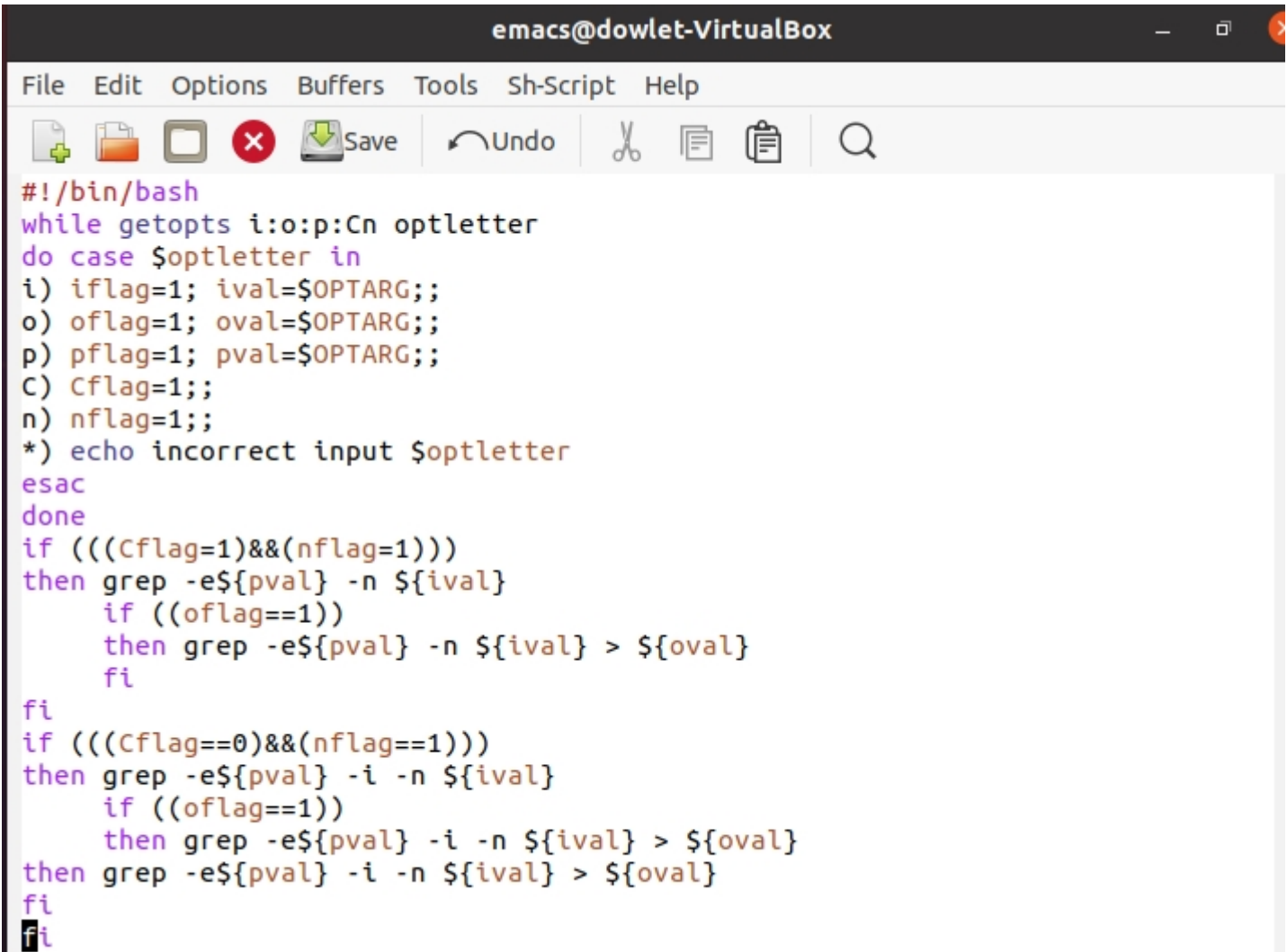
---

#### Ход работы.

1. Для начало я скопировал любой текст из интернета.



- Используя команды `getopts` `grep`, написал командный файл.



- Данный командный файл 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.

```

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

```

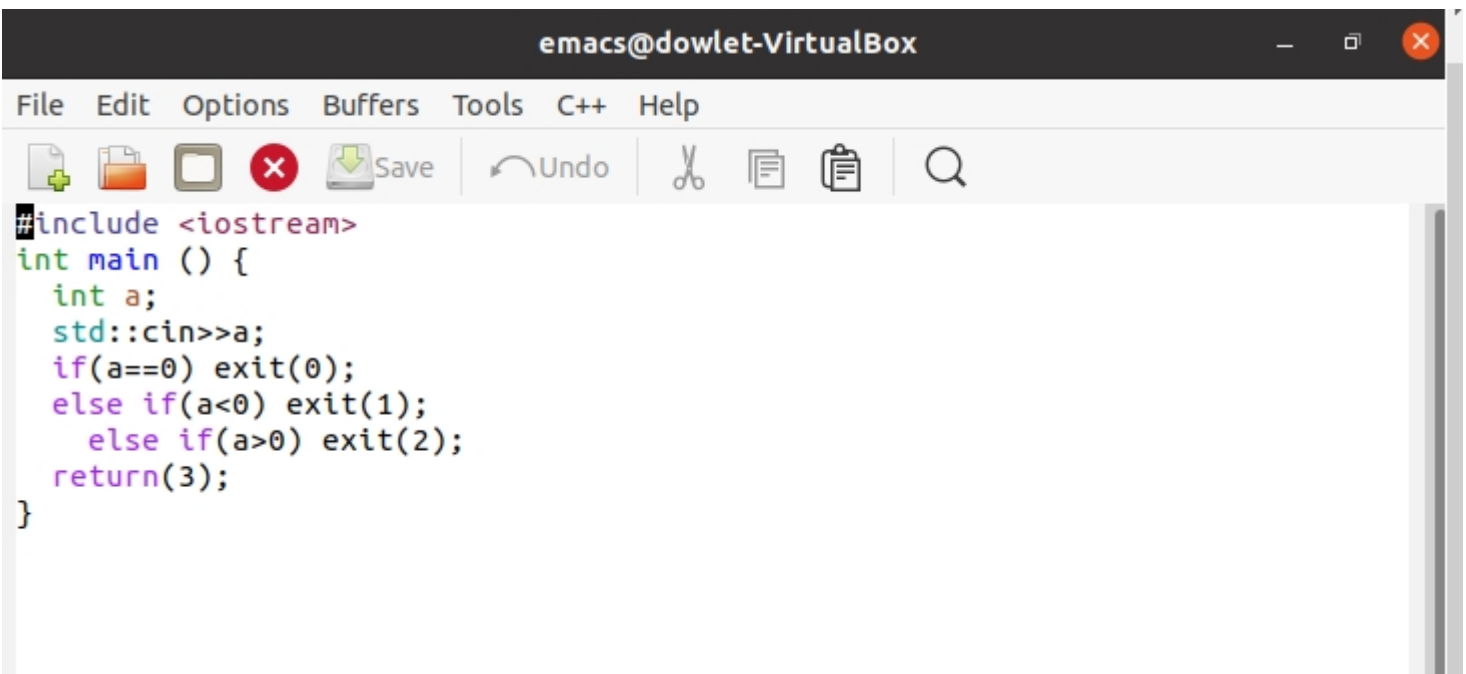
- После я написал слово, которую программа ищет в моем тексте, используя ключ -p

```

dowlet@dowlet-VirtualBox:~$ bash laba12.sh -ilaba12.txt -olaba12_2.txt -pis
1:Architecture,the art and technique of designing and building,as distinguished
from the skills associated with construction. The practice of architecture is
employed to fulfill practical and expressive requirements, and thus it serves b
oth utilitarian and aesthetic ends.
laba12.sh: line 22: syntax error near unexpected token `then'
laba12.sh: line 22: `then grep -e${pval} -i -n ${ival} > ${oval}'
dowlet@dowlet-VirtualBox:~$ bash laba12.sh -ilaba12.txt -olaba12_2.txt -pis -C
1:Architecture,the art and technique of designing and building,as distinguished
from the skills associated with construction. The practice of architecture is
employed to fulfill practical and expressive requirements, and thus it serves b
oth utilitarian and aesthetic ends.
laba12.sh: line 22: syntax error near unexpected token `then'
laba12.sh: line 22: `then grep -e${pval} -i -n ${ival} > ${oval}'
dowlet@dowlet-VirtualBox:~$ bash laba12.sh -ilaba12.txt -olaba12_2.txt -pis -n
1:Architecture,the art and technique of designing and building,as distinguished
from the skills associated with construction. The practice of architecture is
employed to fulfill practical and expressive requirements, and thus it serves b
oth utilitarian and aesthetic ends.
laba12.sh: line 22: syntax error near unexpected token `then'
laba12.sh: line 22: `then grep -e${pval} -i -n ${ival} > ${oval}'
dowlet@dowlet-VirtualBox:~$

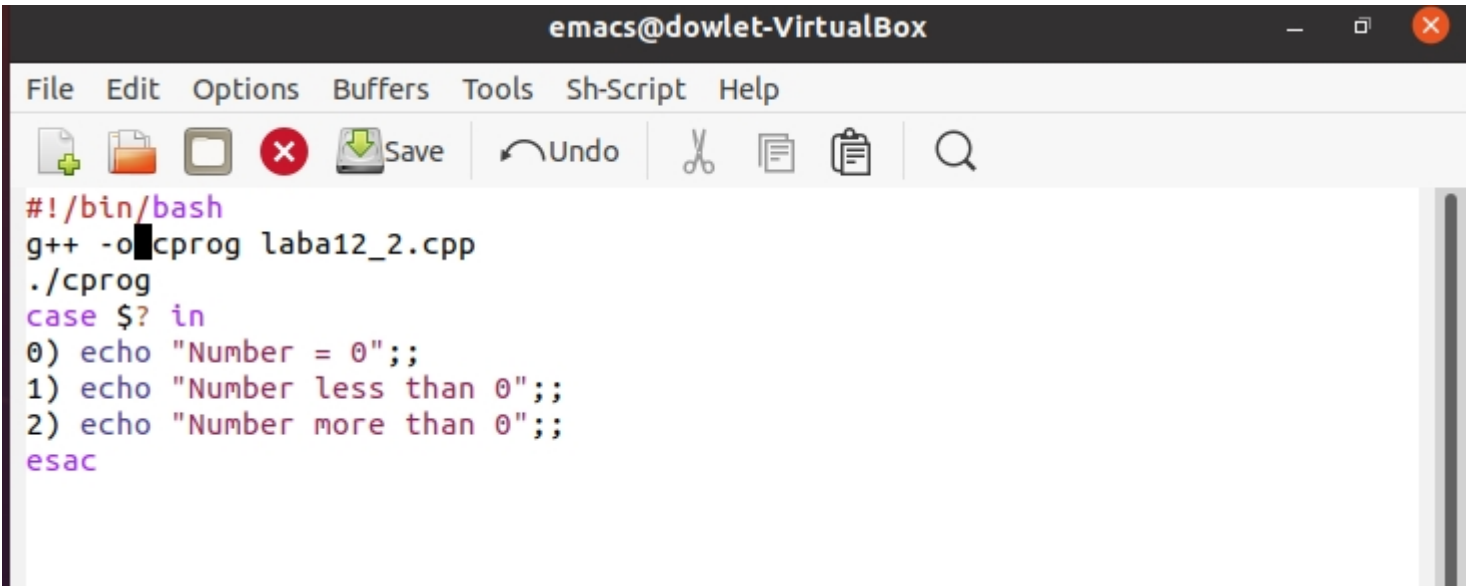
```

2. Написал на языке C++ 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.

A screenshot of the Emacs editor window titled 'emacs@dowlet-VirtualBox'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'C++', and 'Help'. The toolbar shows icons for file operations and editing. The code in the buffer is a C++ program that reads an integer 'a' and prints a message based on its value: 0 for 'Number = 0', 1 for 'Number less than 0', and 2 for 'Number more than 0'.

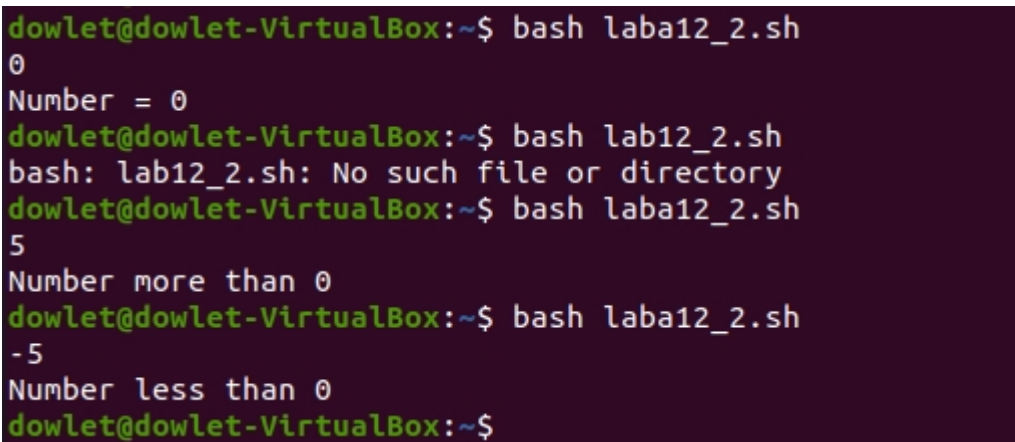
```
#include <iostream>
int main () {
    int a;
    std::cin>>a;
    if(a==0) exit(0);
    else if(a<0) exit(1);
    else if(a>0) exit(2);
    return(3);
}
```

• Командный файл вызывает эту программу и, проанализировав с помощью команды \$?, выдает сообщение о том, какое число было введено.

A screenshot of the Emacs editor window titled 'emacs@dowlet-VirtualBox'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Sh-Script', and 'Help'. The code in the buffer is a shell script that compiles a C++ program and then uses a case statement to print messages based on the exit status of the program.

```
#!/bin/bash
g++ -o cprog laba12_2.cpp
./cprog
case $? in
0) echo "Number = 0";;
1) echo "Number less than 0";;
2) echo "Number more than 0";;
esac
```

• Проверил свой код.

A screenshot of a terminal window showing the execution of the shell script 'laba12\_2.sh'. The script is run three times with different inputs: 0, 5, and -5. The output shows the corresponding messages: 'Number = 0', 'Number more than 0', and 'Number less than 0'.

```
dowlet@dowlet-VirtualBox:~$ bash laba12_2.sh
0
Number = 0
dowlet@dowlet-VirtualBox:~$ bash laba12_2.sh
bash: laba12_2.sh: No such file or directory
dowlet@dowlet-VirtualBox:~$ bash laba12_2.sh
5
Number more than 0
dowlet@dowlet-VirtualBox:~$ bash laba12_2.sh
-5
Number less than 0
dowlet@dowlet-VirtualBox:~$
```

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.).

File Edit Options Buffers Tools Sh-Script Help



```
#!/bin/bash
let dflag=0;
while getopts a:d optletter
do case $optletter in
    a) aflag=1; aval=$OPTA
       d) dflag=1;;
    *) echo incorret input $optletter
       esac
done
#echo ${aval}
if ((dflag==0))
then for((i=1;i<=aval;i++))
do touch ${i}.txt
done

fi
if ((dflag==1))
then for ((i=1;i<=aval;i++))
do rm ${i}.txt
done
fi
```

U:\*\*- laba12\_3.sh All L20 (Shell-script[sh])

- Число файлов, которые необходимо создать, передаётся в аргументы командной строки

```
dowlet@dowlet-VirtualBox:~$ bash laba12_3.sh -a2
dowlet@dowlet-VirtualBox:~$ la -l
```

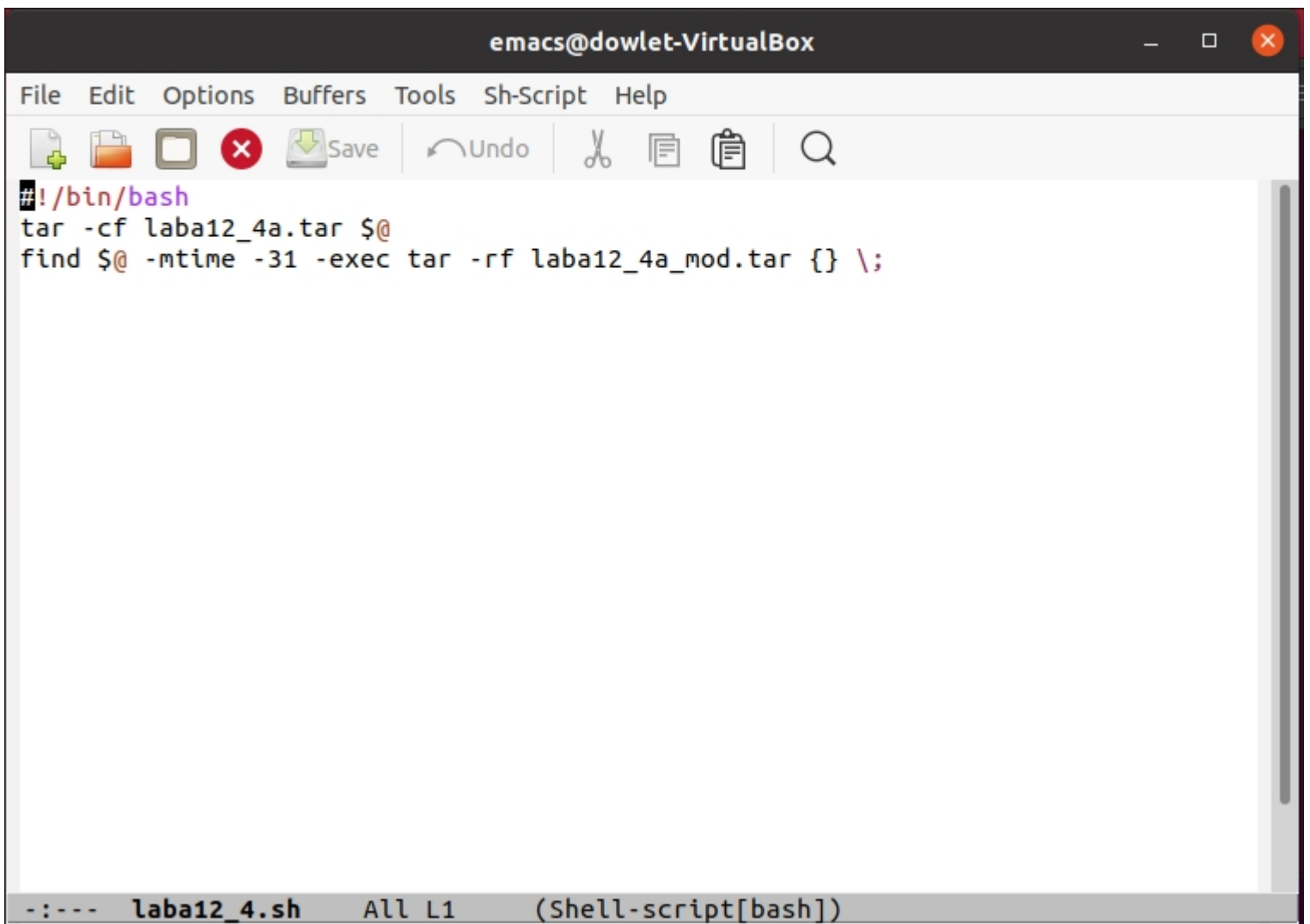


```
total 324
-rw-rw-r-- 1 dowlet dowlet 0 Maý 29 19:42 1.txt
-rw-rw-r-- 1 dowlet dowlet 0 Maý 29 19:42 2.txt
-rwxrwxr-x 1 dowlet dowlet 9536 Noý 1 2020 asdfg
-rw-rw-r-- 1 dowlet dowlet 472 Noý 1 2020 asdfg.asm
-rw-rw-r-- 1 dowlet dowlet 1861 Noý 1 2020 asdfg.lst
-rw-rw-r-- 1 dowlet dowlet 8106 Noý 1 2020 asdfg.map
-rw-rw-r-- 1 dowlet dowlet 2080 Noý 1 2020 asdfg.o
drwxrwxr-x 2 dowlet dowlet 4096 Maý 29 15:05 backup
-rw----- 1 dowlet dowlet 18145 Maý 29 12:32 .bash_history
-rw-r--r-- 1 dowlet dowlet 220 Okt 23 2020 .bash_logout
-rw-r--r-- 1 dowlet dowlet 3771 Okt 23 2020 .bashrc
drwx----- 17 dowlet dowlet 4096 Okt 28 2020 .cache
drwx----- 18 dowlet dowlet 4096 Okt 28 2020 .config
-rwxrwxr-x 1 dowlet dowlet 17200 Maý 29 19:34 cprog
drwxr-xr-x 3 dowlet dowlet 4096 Okt 28 2020 Desktop
drwxr-xr-x 2 dowlet dowlet 4096 Okt 23 2020 Documents
drwxr-xr-x 2 dowlet dowlet 4096 Okt 23 2020 Downloads
drwx----- 3 dowlet dowlet 4096 Maý 22 15:19 .emacs.d
-rw-rw-r-- 1 dowlet dowlet 68 Maý 1 07:02 .gitconfig
drwx----- 3 dowlet dowlet 4096 Maý 29 18:23 .gnupg
drwxrwxr-x 2 dowlet dowlet 4096 Noý 1 2020 lab03a
drwxrwxr-x 2 dowlet dowlet 4096 Noý 1 2020 lab03b
-rw-rw-r-- 1 dowlet dowlet 98 Maý 22 17:46 lab07.sh
-rw-rw-r-- 1 dowlet dowlet 98 Maý 22 17:27 lab07.sh~
-rw-rw-r-- 1 dowlet dowlet 97 Maý 22 17:52 '#lab10.sh#'
-rw-rw-r-- 1 dowlet dowlet 98 Maý 22 17:49 lab10.sh
-rw-rw-r-- 1 dowlet dowlet 0 Maý 29 18:57 lab12 2.txt
```

- Этот же командный файл удаляет все созданные им файлы (если они существуют).

```
dowlet@dowlet-VirtualBox:~$ bash laba12_3.sh -a2 -d
dowlet@dowlet-VirtualBox:~$ ls -l
total 228
-rwxrwxr-x 1 dowlet dowlet 9536 Noý 1 2020 asdfg
-rw-rw-r-- 1 dowlet dowlet 472 Noý 1 2020 asdfg.asm
-rw-rw-r-- 1 dowlet dowlet 1861 Noý 1 2020 asdfg.lst
-rw-rw-r-- 1 dowlet dowlet 8106 Noý 1 2020 asdfg.map
-rw-rw-r-- 1 dowlet dowlet 2080 Noý 1 2020 asdfg.o
drwxrwxr-x 2 dowlet dowlet 4096 Maý 29 15:05 backup
-rwxrwxr-x 1 dowlet dowlet 17200 Maý 29 19:34 cprog
drwxr-xr-x 3 dowlet dowlet 4096 Okt 28 2020 Desktop
drwxr-xr-x 2 dowlet dowlet 4096 Okt 23 2020 Documents
drwxr-xr-x 2 dowlet dowlet 4096 Okt 23 2020 Downloads
drwxrwxr-x 2 dowlet dowlet 4096 Noý 1 2020 lab03a
drwxrwxr-x 2 dowlet dowlet 4096 Noý 1 2020 lab03b
-rw-rw-r-- 1 dowlet dowlet 98 Maý 22 17:46 lab07.sh
-rw-rw-r-- 1 dowlet dowlet 98 Maý 22 17:27 lab07.sh~
-rw-rw-r-- 1 dowlet dowlet 97 Maý 22 17:52 '#lab10.sh#'
-rw-rw-r-- 1 dowlet dowlet 98 Maý 22 17:49 lab10.sh
-rw-rw-r-- 1 dowlet dowlet 0 Maý 29 18:57 lab12_2.txt
-rw-r--r-- 1 dowlet dowlet 741 Noý 2 2020 lab4.asm
-rw-rw-r-- 1 dowlet dowlet 0 Noý 2 2020 lab4.lst
-rwxrwxr-x 1 dowlet dowlet 9184 Noý 2 2020 lab5
-rw-rw-r-- 1 dowlet dowlet 746 Noý 2 2020 lab5.asm
-rw-rw-r-- 1 dowlet dowlet 3051 Noý 2 2020 lab5.lst
-rw-rw-r-- 1 dowlet dowlet 1312 Noý 2 2020 lab5.o
```

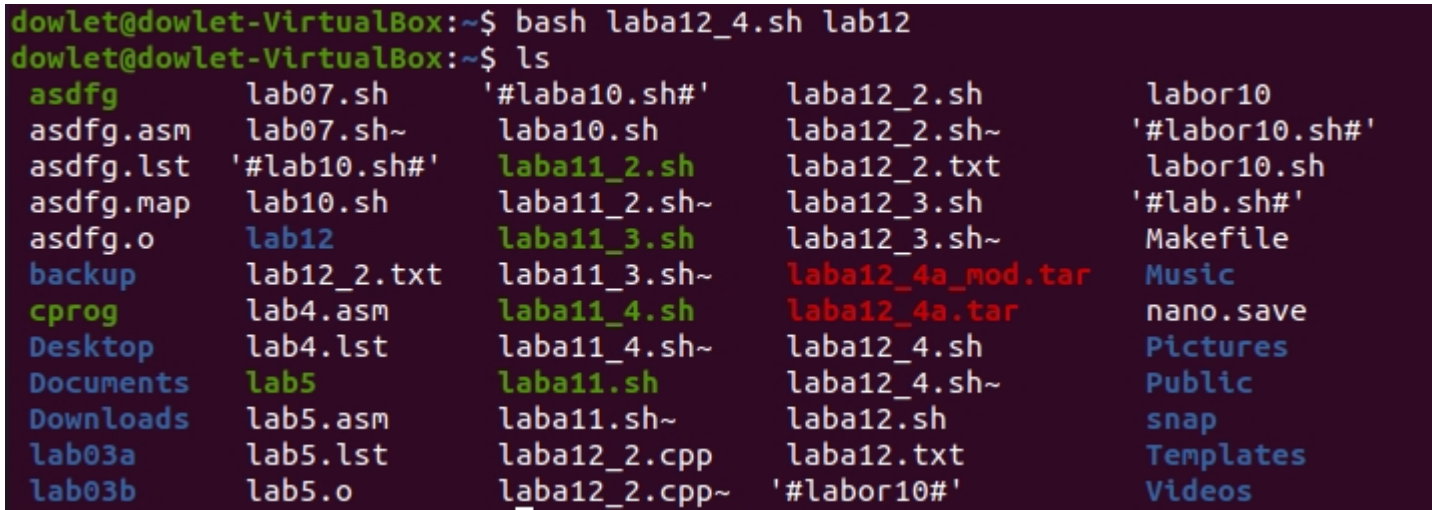
4. Написал командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории.



```
emacs@dowlet-VirtualBox
File Edit Options Buffers Tools Sh-Script Help
Save Undo
#!/bin/bash
tar -cf laba12_4a.tar $@
find $@ -mtime -31 -exec tar -rf laba12_4a_mod.tar {} \;

-:--- laba12_4.sh All L1 (Shell-script[bash])
```

• Модифицировал его так, чтобы запаковывались только те файлы, которые были изменены менее месяца тому назад (использовал команду find).



```
dowlet@dowlet-VirtualBox:~$ bash laba12_4.sh lab12
dowlet@dowlet-VirtualBox:~$ ls
asdfg      lab07.sh      '#laba10.sh#'   laba12_2.sh     labor10
asdfg.asm  lab07.sh~     laba10.sh       laba12_2.sh~    '#labor10.sh#'
asdfg.lst  '#lab10.sh#' laba11_2.sh     laba12_2.txt    labor10.sh
asdfg.map  lab10.sh      laba11_2.sh~    laba12_3.sh     '#lab.sh#'
asdfg.o    lab12         laba11_3.sh     laba12_3.sh~    Makefile
backup     lab12_2.txt   laba11_3.sh~    laba12_4a_mod.tar Music
cprog      lab4.asm      laba11_4.sh     laba12_4a.tar   nano.save
Desktop    lab4.lst      laba11_4.sh~    laba12_4.sh     Pictures
Documents  lab5          laba11.sh       laba12_4.sh~    Public
Downloads  lab5.asm      laba11.sh~      laba12.sh       snap
lab03a     lab5.lst      laba12_2.cpp    laba12.txt      Templates
lab03b     lab5.o        laba12_2.cpp~   '#labor10#'     Videos
```

**Вывод:**

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

**Библиография.**

<http://java-online.ru/java-if-else.xhtml>

**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.