

# Лабораторная работа 11

Программирование в командном процессоре ОС UNIX.  
Ветвления и циклы

---

Aristova A.O.

2022, 27 May

RUDN University, Moscow, Russian Federation

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

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

- `-I inputfile` — прочитать данные из указанного файла;
- `-o outputfile` — вывести данные в указанный файл;
- `-p 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 `-p`.

2. Написать 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.

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

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, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`.

# Выполнение лабораторной работы

1. Используя команды `getopts` и `grep` я написала первый командный файл, который анализирует командную строку с несколькими ключами, а затем в указанном файле ищет нужные строки, определяемые также ключом и выводит их в указанный файл.

```
while getopts "i:o:p:c:n" opt
do
case $opt in
    i)inputfile="$OPTARG";;
    o)outputfile="$OPTARG";;
    p)shablon="$OPTARG";;
    c)registr="";;
    n)number="";;
esac
done

grep -n "$shablon" "$inputfile" > "$outputfile"
```

Рис. 1: Скрипт к заданию 1.



Затем я добавила права на выполнение файла и выполнила его, указав необходимые опции и аргументы.

```
[aoaristova@fedora lab11]$ chmod +x lab11_1
[aoaristova@fedora lab11]$ ./lab11_1 -i conf.txt -o output.txt -p h -c -n
[aoaristova@fedora lab11]$ ls
compare.cpp  conf.txt  lab11_1  lab11_2  output.txt
[aoaristova@fedora lab11]$
```

Рис. 2: Результат выполнения скрипта 1.

2. На языке программирования C++ я написала вспомогательную программу, которая вводит число и определяет, является оно большим/меньшим/равным нулю. Затем программа завершается, передавая информацию о коде завершения в оболочку, с помощью функции `exit(n)`, где `n` – код.

```
#include <iostream>
using namespace std;

int main(int argument, char *arg[]){
    if (atoi(arg[1]) > 0){
        exit(1);
    }
    else if (atoi(arg[1]) == 0) {
        exit(2);
    }
    else {
        exit(3);
    }
    return 0;
}
```

Рис. 3: Вспомогательная программа на C++ к заданию 2.

Далее я написала командный файл, который вызывает эту программу и, проанализировав с помощью команды \$?, выдает сообщение о том, какое число было введено(большее/меньшее/равное нулю).

```
#!/bin/bash

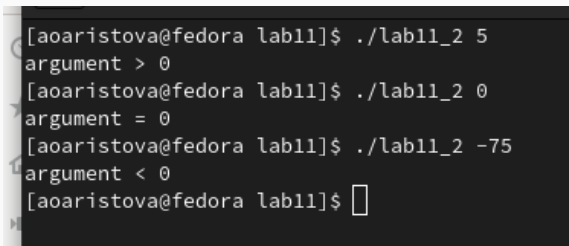
CC=g++
EXEC=compare
SRC=compare.cpp

if [ "$SRC" -nt "$EXEC" ]
then
    echo "Rebuilding $EXEC ....."
    $CC -o $EXEC $SRC
fi

./$EXEC $1
ec=$?
if [ "$ec" == "1" ]
then
    echo "argument > 0"
fi
if [ "$ec" == "2" ]
then
    echo "argument = 0"
fi
if [ "$ec" == "3" ]
then
    echo "argument < 0"
fi
```

Рис. 4: Скрипт к заданию 2.

Затем я добавила права на выполнение файла и выполнила его, указав необходимые опции и аргументы.

A terminal window with a dark background and light-colored text. The prompt is [aoaristova@fedora lab11]\$. The user enters ./lab11\_2 5, followed by argument > 0. Then they enter ./lab11\_2 0, followed by argument = 0. Then they enter ./lab11\_2 -75, followed by argument < 0. Finally, they enter a blank line, indicated by a white cursor box.

```
[aoaristova@fedora lab11]$ ./lab11_2 5
argument > 0
[aoaristova@fedora lab11]$ ./lab11_2 0
argument = 0
[aoaristova@fedora lab11]$ ./lab11_2 -75
argument < 0
[aoaristova@fedora lab11]$
```

Рис. 5: Результат выполнения скрипта 2.

3. Я создала командный файл, который создает  $n$  файлов последовательно пронумерованных (1.tmp, 2.tmp, 3.tmp, 4.tmp и т.д. до  $n$ ), где  $n$  задается как аргумент командной строки. Также этот файл умеет удалять все подобные файлы, если они имеются. Для этого нужно указать другую опцию.

```
#!/bin/bash
while getopts c:r opt
do
case $opt in
c)n="$OPTARG"; for i in $(seq 1 $n); do touch "$i.tmp";done;;
r)for i in $(find -name "*.tmp"); do rm $i; done;;
esac
done
```

Рис. 6: Скрипт к заданию 3.

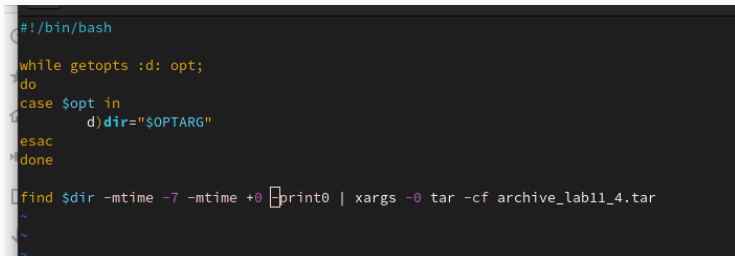
Затем я добавила права на выполнение файла и выполнила его, указав необходимые опции и аргументы.

```
[aoaristova@fedora lab11]$ chmod +x lab11_3
[aoaristova@fedora lab11]$ ./lab11_3 -c 5
[aoaristova@fedora lab11]$ ls
1.tmp 2.tmp 3.tmp 4.tmp 5.tmp compare compare.cpp conf.txt lab11_1 lab11_2 lab11_3 output.txt
[aoaristova@fedora lab11]$ ./lab11_3 -r
[aoaristova@fedora lab11]$ ls
compare compare.cpp conf.txt lab11_1 lab11_2 lab11_3 output.txt
[aoaristova@fedora lab11]$
```

Рис. 7: Результат выполнения скрипта 3.



4. Я создала командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории, модифицировала его так, чтобы он запаковывал только те файлы, который изменялись менее недели тому назад, используя команду `find`.



```
#!/bin/bash

while getopts :d: opt;
do
case $opt in
d) dir="$OPTARG"
esac
done

find $dir -mtime -7 -mtime +0 -print0 | xargs -0 tar -cf archive_lab11_4.tar
```

Рис. 8: Скрипт к заданию 4.

Затем я добавила права на выполнение файла и выполнила его, указав необходимые опции и аргументы.

```
[aoristova@fedora lab11]$ chmod +x lab11_4
[aoristova@fedora lab11]$ ./lab11_4 /home
[aoristova@fedora lab11]$ ls
archive_lab11_4.tar  compare  compare.cpp  conf.txt  lab11_1  lab11_2  lab11_3  lab11_4  output.txt
[aoristova@fedora lab11]$
```

Рис. 9: Результат выполнения скрипта 4.

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