

# **Отчет по лабораторной работе №11**

**Программирование в командном процессоре ОС UNIX. Ветвления и циклы.**

Галацан Николай, НПИбд-01-22

# Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	7
4	Выводы	13
5	Ответы на контрольные вопросы	14

## Список иллюстраций

3.1	Запуск программы №1 . . . . .	8
3.2	Запуск программы №2 . . . . .	10
3.3	Запуск программы №3 . . . . .	11
3.4	Запуск программы №4 . . . . .	12

# 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. 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. Команд- ный 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 ко мандный 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)

### 3 Выполнение лабораторной работы

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`.

Открываю `emacs`. Создаю файл `lab11_1.sh`, набираю текст программы.

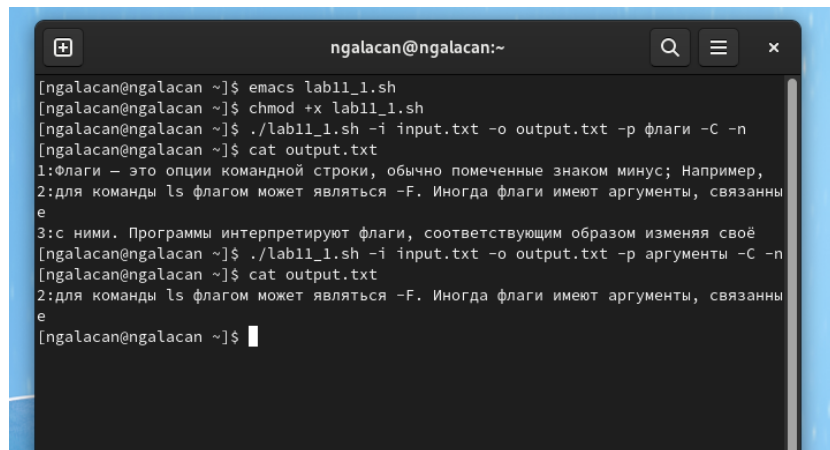
*Листинг программы №1:*

```
#!/bin/bash
iflag=0; oflag=0; pflag=0; Cflag=0; nflag=0;
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 "Недопустимая опция" $optletter
esac
```

done

```
grep -i -n $pval $ival > $oval
```

Даю файлу право на исполнение и запускаю программу. Скрипт использует `getopts` для принятия аргументов из командной строки, которые передаются утилите `grep`. Ищу в файле “input.txt” слово “флаги”, “аргументы” (рис. 3.1).



```
ngalacan@ngalacan:~  
[ngalacan@ngalacan ~]$ emacs lab11_1.sh  
[ngalacan@ngalacan ~]$ chmod +x lab11_1.sh  
[ngalacan@ngalacan ~]$ ./lab11_1.sh -i input.txt -o output.txt -p флаги -C -n  
[ngalacan@ngalacan ~]$ cat output.txt  
1:флаги – это опции командной строки, обычно помеченные знаком минус; Например,  
2:для команды ls флагом может являться -F. Иногда флаги имеют аргументы, связанн  
е  
3:с ними. Программы интерпретируют флаги, соответствующим образом изменяя своё  
[ngalacan@ngalacan ~]$ ./lab11_1.sh -i input.txt -o output.txt -p аргументы -C -n  
[ngalacan@ngalacan ~]$ cat output.txt  
2:для команды ls флагом может являться -F. Иногда флаги имеют аргументы, связанн  
е  
[ngalacan@ngalacan ~]$
```

Рис. 3.1: Запуск программы №1

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. Команд- ный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.

Создаю файл для программы на C `lab11_2_prog.c`, набираю текст программы.

*Листинг программы на C:*

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



```

int main(){
    printf("Введите число: ");
    int n;
    scanf("%d", &n);
    if (n<0) exit(1);
    if (n>0) exit(2);
    if (n==0) exit(3);
    return 0;
}

```

Создаю файл lab11\_2.sh, набираю текст программы.

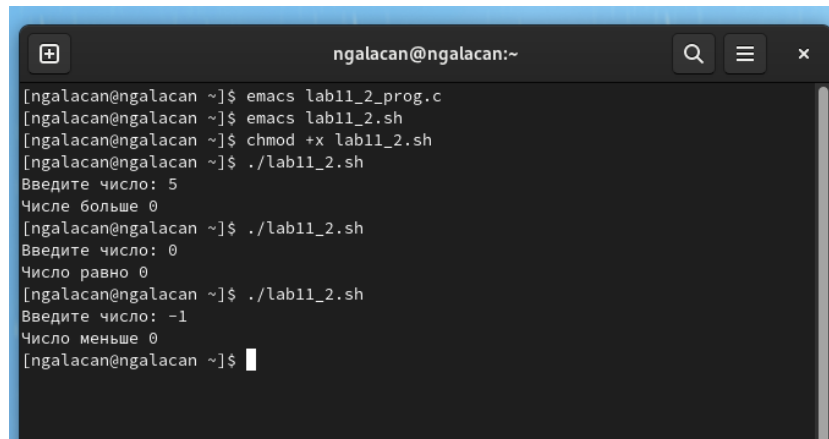
*Листинг программы №2:*

```

#!/bin/bash
gcc lab11_2_prog.c -o lab11_2_prog
./lab11_2_prog
exc=$?
case $exc in
    1) echo "Число меньше 0";;
    2) echo "Числе больше 0";;
    3) echo "Число равно 0";;
esac

```

Даю файлу право на исполнение и запускаю (рис. 3.2).



```
ngalacan@ngalacan:~  
[ngalacan@ngalacan ~]$ emacs lab11_2_prog.c  
[ngalacan@ngalacan ~]$ emacs lab11_2.sh  
[ngalacan@ngalacan ~]$ chmod +x lab11_2.sh  
[ngalacan@ngalacan ~]$ ./lab11_2.sh  
Введите число: 5  
Число больше 0  
[ngalacan@ngalacan ~]$ ./lab11_2.sh  
Введите число: 0  
Число равно 0  
[ngalacan@ngalacan ~]$ ./lab11_2.sh  
Введите число: -1  
Число меньше 0  
[ngalacan@ngalacan ~]$
```

Рис. 3.2: Запуск программы №2

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).

Создаю файл lab11\_3.sh, набираю текст программы.

*Листинг программы №3:*

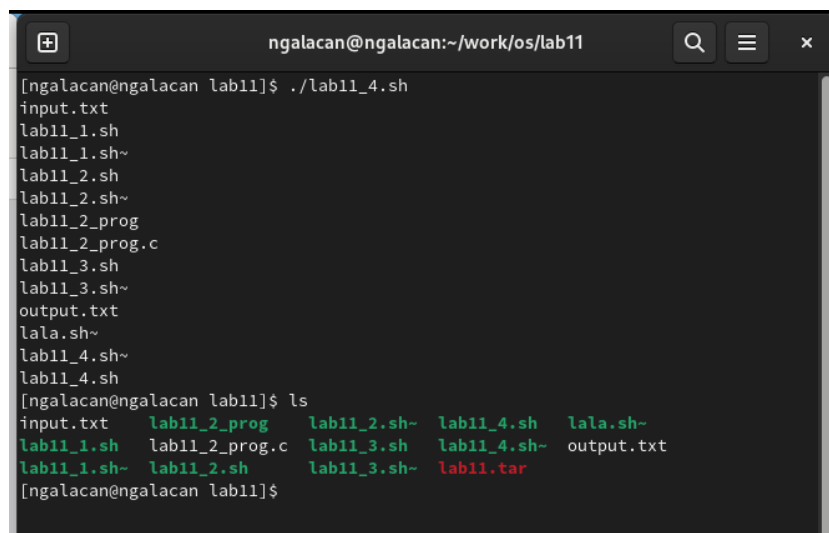
```
#!/bin/bash  
while getopts c:r optletter  
do case $optletter in  
    c) cflag=1; cval=$OPTARG;;  
    r) rflag=1;;  
    esac  
done  
if ((rflag==0))  
then for i in $(seq 1 $cval)  
    do touch "$i.tmp"  
    done
```



*Листинг программы №4:*

```
#!/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
```

Даю файлу право на исполнение и запускаю (рис. 3.4).



The screenshot shows a terminal window with the title bar 'ngalacan@ngalacan:~/work/os/lab11'. The user has executed the script './lab11\_4.sh'. The output of the script is a list of files: 'input.txt', 'lab11\_1.sh', 'lab11\_1.sh~', 'lab11\_2.sh', 'lab11\_2.sh~', 'lab11\_2\_prog', 'lab11\_2\_prog.c', 'lab11\_3.sh', 'lab11\_3.sh~', 'output.txt', 'lala.sh~', 'lab11\_4.sh~', and 'lab11\_4.sh'. The user then runs 'ls', which displays a color-coded listing of the files in the directory.

```
[ngalacan@ngalacan lab11]$ ./lab11_4.sh
input.txt
lab11_1.sh
lab11_1.sh~
lab11_2.sh
lab11_2.sh~
lab11_2_prog
lab11_2_prog.c
lab11_3.sh
lab11_3.sh~
output.txt
lala.sh~
lab11_4.sh~
lab11_4.sh
[ngalacan@ngalacan lab11]$ ls
input.txt  lab11_2_prog  lab11_2.sh~  lab11_4.sh  lala.sh~
lab11_1.sh  lab11_2_prog.c  lab11_3.sh  lab11_4.sh~  output.txt
lab11_1.sh~  lab11_2.sh  lab11_3.sh~  lab11.tar
```

Рис. 3.4: Запуск программы №4

## 4 Выводы

Изучены основы программирования в оболочке ОС UNIX. Приобретены навыки написания более сложных командных файлов с использованием логических управляющих конструкций и циклов.

## 5 Ответы на контрольные вопросы

### 1. Каково предназначение команды `getopts`?

Команда `getopts` осуществляет синтаксический анализ командной строки, выделяя флаги, 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. Какое отношение метасимволы имеют к генерации имён файлов?

При перечислении имён файлов текущего каталога можно использовать следующие символы: 1. соответствует произвольной, в том числе и пустой строке; 2. ? соответствует любому одинарному символу; 3. `[c1-c2]` соответствует любому

символу, лексикографически находящемуся между символами `c1` и `c2`. Например, `1.1 echo` выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `1.2. ls.c` выведет все файлы с последними двумя символами, совпадающими с `c`. `1.3. echoprogram?` выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.` `1.4.[a-z]` соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

### 3. Какие операторы управления действиями вы знаете?

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 `OCUNIX` 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. Какие операторы используются для прерывания цикла?

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. Для чего нужны команды `false` и `true`?

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, неравный нулю (т.е. ложь). Примеры бесконечных циклов: `while true do echo hello andy done` `until false do echo hello mike done`.

6. Что означает строка `if test -f mans/i.$s`, встречаемая в командном файле?

Строка `if test -f mans/i.s` проверяет, существует ли файл `mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).

7. Объясните различия между конструкциями `while` и `until`.

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.