

Лабораторная работа No 11.

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

Боровиков Даниил Александрович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	13
4	Контрольные вопросы	14

Список иллюстраций

2.1	Создание файла lab11_1.sh	6
2.2	Написание скрипта lab11_1.sh	7
2.3	Право на выполнение и запуск lab11_1.sh	8
2.4	Написание программы на языке Си	8
2.5	Написание скрипта lab11_2.sh	9
2.6	Право на выполнение и запуск lab11_2.sh	9
2.7	Написание скрипта lab11_3.sh	10
2.8	Право на выполнение и запуск lab10_3.sh	11
2.9	Написание скрипта lab11_4.sh	12
2.10	Право на выполнение и запуск lab11_4.sh	12

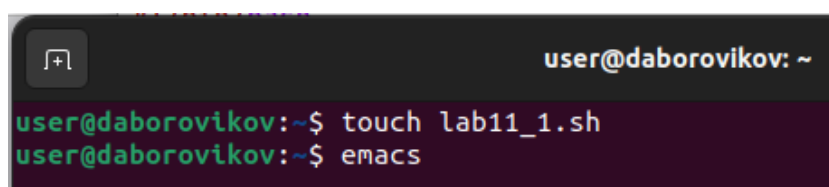
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

Откроем терминал. Создадим в домашнем каталоге файл lab11_1.sh. Откроем emacs.(рис. fig. 2.1).

A screenshot of a terminal window with a dark background. The prompt is 'user@daborovikov: ~'. Two commands are entered: 'touch lab11_1.sh' and 'emacs', each followed by a new line.

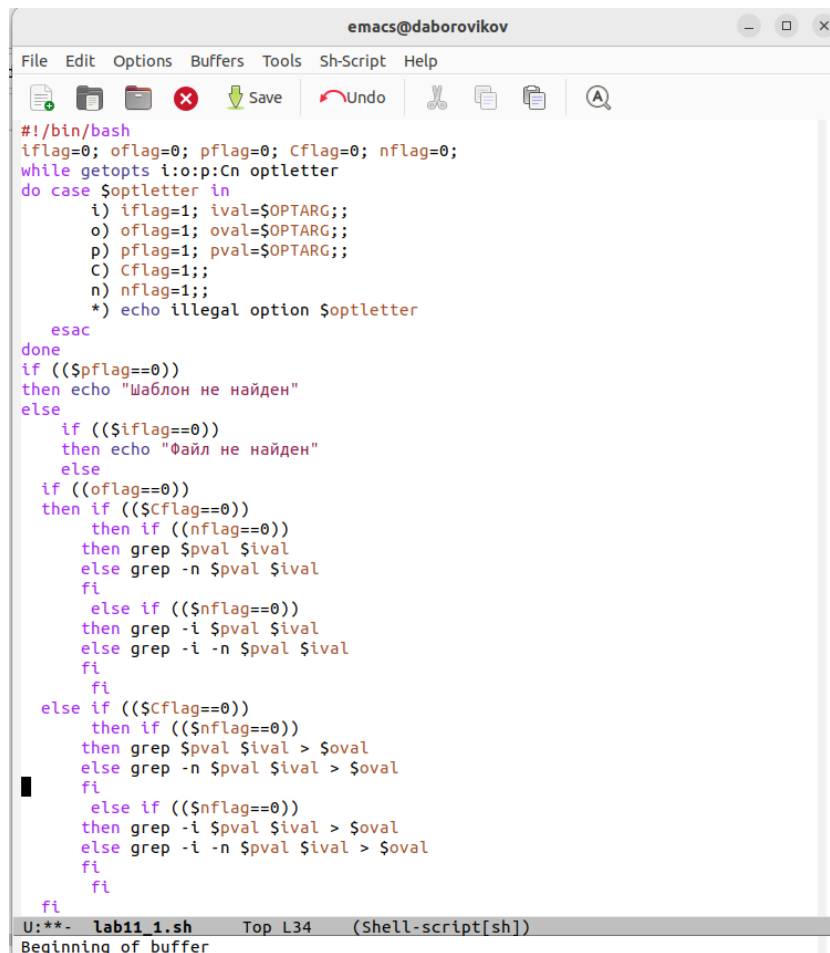
```
user@daborovikov: ~  
user@daborovikov:~$ touch lab11_1.sh  
user@daborovikov:~$ emacs
```

Рис. 2.1: Создание файла lab11_1.sh

Используя команды getoptс grep, напишем командный файл, который анализирует командную строку с ключами:

- -inputfile — 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.(рис. fig. 2.2).



```
emacs@daborovikov
File Edit Options Buffers Tools Sh-Script Help
Save Undo
#!/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 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:*** lab11_1.sh Top L34 (Shell-script[sh])
Beginning of buffer
```

Рис. 2.2: Написание скрипта lab11_1.sh

В терминале дадим файлу право на исполнение. Запустим файл и проверим.(рис. fig. 2.3).

```
user@daborovikov: ~  
user@daborovikov:~$ touch lab11_1.sh  
user@daborovikov:~$ emacs  
user@daborovikov:~$ touch a1.txt a2.txt  
user@daborovikov:~$ chmod +x lab11_1.sh  
user@daborovikov:~$ cat a1.txt  
danya vlad vladik  
dima  
cot  
danya danya  
user@daborovikov:~$ cat a2.txt  
danya danya danya  
cot  
cot cot  
danya  
danya 8800  
user@daborovikov:~$ ./lab11_1.sh -i a1.txt -o a2.txt -p danya -n  
user@daborovikov:~$ cat a2.txt  
1:danya vlad vladik  
4:danya danya  
user@daborovikov:~$ ./lab11_1.sh -i a1.txt -o a2.txt -p danya -C -n  
user@daborovikov:~$ cat a2.txt  
1:danya vlad vladik  
4:danya danya  
user@daborovikov:~$
```

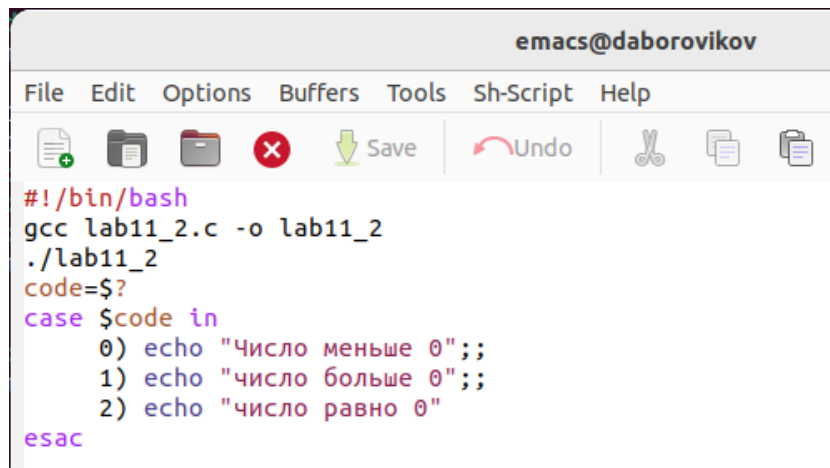
Рис. 2.3: Право на выполнение и запуск lab11_1.sh

Напишем 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(рис. fig. 2.4).

```
emacs@daborovikov  
File Edit Options Buffers Tools C Help  
[Icons]  
#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. 2.4: Написание программы на языке Си

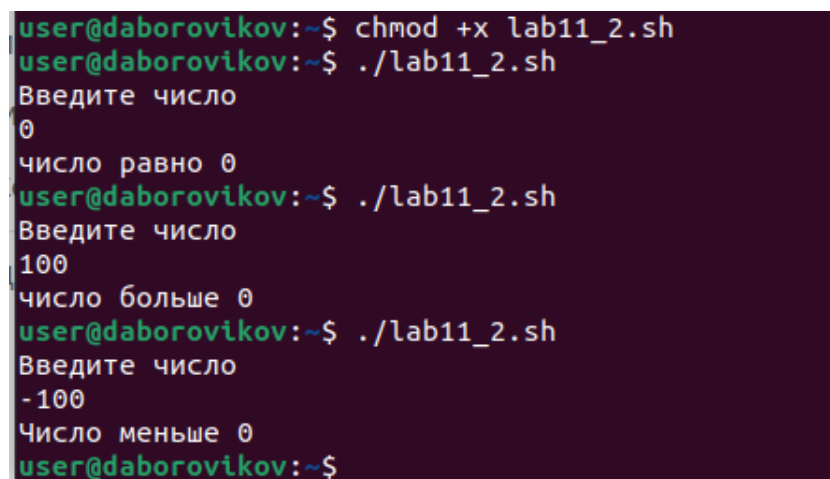
(рис. fig. 2.5).



```
#!/bin/bash
gcc lab11_2.c -o lab11_2
./lab11_2
code=$?
case $code in
0) echo "Число меньше 0";;
1) echo "число больше 0";;
2) echo "число равно 0"
esac
```

Рис. 2.5: Написание скрипта lab11_2.sh

В терминале дадим файлу право на исполнение. Запускаем файл и проверяем.(рис. fig. 2.6).



```
user@daborovikov:~$ chmod +x lab11_2.sh
user@daborovikov:~$ ./lab11_2.sh
Введите число
0
число равно 0
user@daborovikov:~$ ./lab11_2.sh
Введите число
100
число больше 0
user@daborovikov:~$ ./lab11_2.sh
Введите число
-100
Число меньше 0
user@daborovikov:~$
```

Рис. 2.6: Право на выполнение и запуск lab11_2.sh

Напишем 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)(рис. fig. 2.7).



```
#!/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. 2.7: Написание скрипта lab11_3.sh

В терминале дадим файлу право на исполнение. Запускаем файл. (рис. fig. 2.8).

```

user@daborovikov: ~
user@daborovikov:~$ chmod +x lab11_3.sh
user@daborovikov:~$ ls
a1.txt      lab10_1.sh  lab11_1.sh  lab11_3.sh~ Музыка
a2.txt      lab10_1.sh~ lab11_1.sh~ snap      Общедоступные
backup      lab10_2.sh  lab11_2     tmp        Рабочий стол
bin         lab10_2.sh~ lab11_2.c   usr        Шаблоны
blog        lab10_3.sh  lab11_2.c~  видео
'##lab07.sh##' lab10_3.sh~ lab11_2.sh  документы
'#lab07.sh#'   lab10_4.sh  lab11_2.sh~ загрузки
lab07.sh      lab10_4.sh~ lab11_3.sh  изображения
user@daborovikov:~$ ./lab11_3.sh -c abc#.txt 3
user@daborovikov:~$ ls
a1.txt      '##lab07.sh##' lab10_3.sh~ lab11_2.sh  документы
a2.txt      '#lab07.sh#'   lab10_4.sh  lab11_2.sh~ загрузки
abc1.txt    lab07.sh       lab10_4.sh~ lab11_3.sh  изображения
abc2.txt    lab10_1.sh     lab11_1.sh  lab11_3.sh~ Музыка
abc3.txt    lab10_1.sh~    lab11_1.sh~ snap      Общедоступные
backup      lab10_2.sh     lab11_2     tmp        Рабочий стол
bin         lab10_2.sh~    lab11_2.c   usr        Шаблоны
blog        lab10_3.sh     lab11_2.c~  видео
user@daborovikov:~$ ./lab11_3.sh -r abc#.txt 3
user@daborovikov:~$ ls
a1.txt      lab10_1.sh  lab11_1.sh  lab11_3.sh~ Музыка
a2.txt      lab10_1.sh~ lab11_1.sh~ snap      Общедоступные
backup      lab10_2.sh  lab11_2     tmp        Рабочий стол
bin         lab10_2.sh~ lab11_2.c   usr        Шаблоны
blog        lab10_3.sh  lab11_2.c~  видео
'##lab07.sh##' lab10_3.sh~ lab11_2.sh  документы
'#lab07.sh#'   lab10_4.sh  lab11_2.sh~ загрузки
lab07.sh      lab10_4.sh~ lab11_3.sh  изображения
user@daborovikov:~$

```

Рис. 2.8: Право на выполнение и запуск lab10_3.sh

Напишем 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).(рис. fig. 2.9).

```
user@daborovikov: ~/catalogi
user@daborovikov:~$ chmod +x lab11_4.sh
user@daborovikov:~$ mkdir catalogi
user@daborovikov:~$ cd ~/catalogi
user@daborovikov:~/catalogi$ ls -l
1418980 a1.txt          1418996 lab10_1.sh  1419001 lab11_2
1418994 a2.txt          1418997 lab10_2.sh  1419002 lab11_2.c
1419006 '##lab07.sh##'  1418998 lab10_3.sh  1419003 lab11_2.sh
1419007 '#lab07.sh#' 1418999 lab10_4.sh  1419004 lab11_3.sh
1418995 lab07.sh    1419000 lab11_1.sh  1419005 lab11_4.sh
user@daborovikov:~/catalogi$ sudo ~/lab11_4.sh
[sudo] пароль для user:
lab11_3.sh
lab11_2.sh
lab11_4.sh
a1.txt
lab11_1.sh
lab11_2.c
a2.txt
lab10_1.sh
lab10_4.sh
lab10_2.sh
lab11_2
lab10_3.sh
user@daborovikov:~/catalogi$ tar -tf catalogi.tar
lab11_3.sh
lab11_2.sh
lab11_4.sh
a1.txt
lab11_1.sh
lab11_2.c
a2.txt
lab10_1.sh
lab10_4.sh
lab10_2.sh
lab11_2
lab10_3.sh
user@daborovikov:~/catalogi$ ls
a1.txt          '#lab07.sh#'  lab10_3.sh  lab11_2.c
a2.txt          lab07.sh      lab10_4.sh  lab11_2.sh
catalogi.tar    lab10_1.sh    lab11_1.sh  lab11_3.sh
'##lab07.sh##' lab10_2.sh    lab11_2     lab11_4.sh
user@daborovikov:~/catalogi$
```

Рис. 2.9: Написание скрипта lab11_4.sh

В терминале дадим файлу право на исполнение. Запускаем файл(рис. fig. 2.10).

```
emacs@daborovikov
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=$(echo "$file" | cut -c 3-)
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
```

Рис. 2.10: Право на выполнение и запуск lab11_4.sh

3 Выводы

В ходе лабораторной работы мы изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

4 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 `s1` и `s2`. 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 `s.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 `OSUNIX` 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.