

Программирование в командном процессоре ОС UNIX. Командные файлы

Отчет по лабораторной работе №12

Башкирова Я.Д

Содержание

1	Теоретическое введение	5
2	Цель работы	6
3	Задание	7
4	Ход работы	8
5	Выводы	16
6	Библиография	17
7	Контрольные вопросы	18

Список таблиц

Список иллюстраций

4.1	Текст	9
4.2	Командный файл, который анализирует командную строку	10
4.3	Анализ командной строки с ключами	10
4.4	Программа на языке Си	11
4.5	Командный файл	12
4.6	Ввод чисел	12
4.7	Командный файл, создающий указанное число файлов	13
4.8	Число файлов от 1 до N и их удаление	14
4.9	Командный файл, запаковывающий в архив все файлы	15
4.10	Архив всех файлов в указанной директории	15

1 Теоретическое введение

Циклы позволяют выполнять один и тот же участок кода необходимое количество раз. В большинстве языков программирования существует несколько типов циклов. Большинство из них поддерживаются оболочкой Bash. Циклы бывают:

- `for` - позволяет перебрать все элементы из массива или использует переменную-счетчик для определения количества повторений;
- `while` - цикл выполняется пока условие истинно;
- `until` - цикл выполняется пока условие ложно.

Циклы Bash, это очень полезная вещь и разобраться с ними будет несложно. Bash позволяет использовать циклы как в скриптах, так и непосредственно в командной оболочке.

2 Цель работы

Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

3 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; `-r`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 `-r`.
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 `tag` 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`).

4 Ход работы

1. Я, используя команды `getopts` `grep`, написала командный файл, который анализирует командную строку с ключами:

- `-iinputfile` — прочитать данные из указанного файла;
- `-ooutputfile` — вывести данные в указанный файл;
- `-р`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 `-р`.

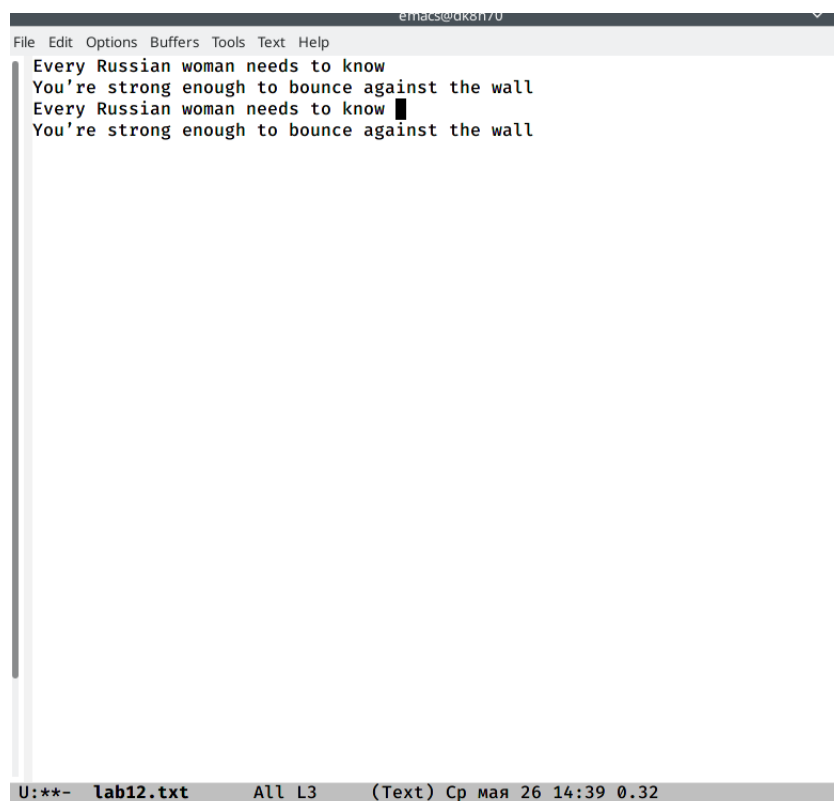
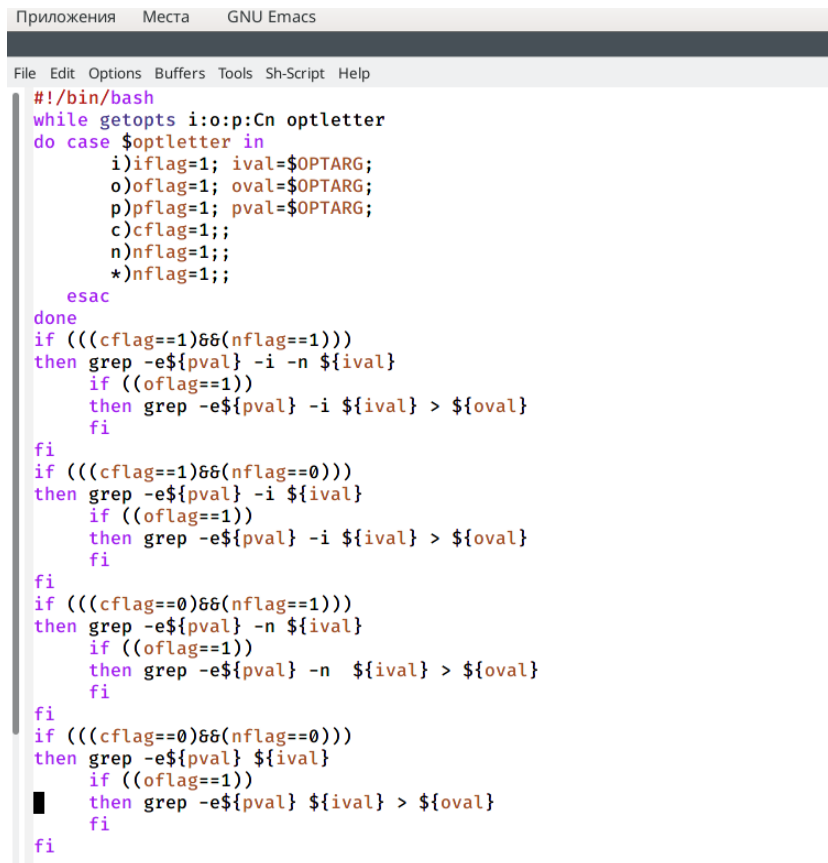
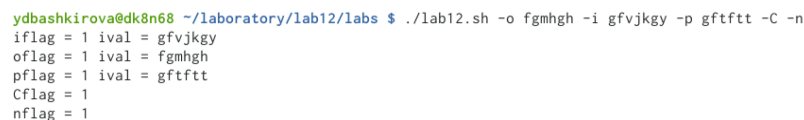


Рис. 4.1: Текст



```
#!/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;;
    *) nflag=1;;
    esac
done
if (((cflag==1)&&(nflag==1)))
then grep -e${pval} -i -n ${ival}
    if ((oflag==1))
    then grep -e${pval} -i ${ival} > ${oval}
    fi
fi
if (((cflag==1)&&(nflag==0)))
then grep -e${pval} -i ${ival}
    if ((oflag==1))
    then grep -e${pval} -i ${ival} > ${oval}
    fi
fi
if (((cflag==0)&&(nflag==1)))
then grep -e${pval} -n ${ival}
    if ((oflag==1))
    then grep -e${pval} -n ${ival} > ${oval}
    fi
fi
if (((cflag==0)&&(nflag==0)))
then grep -e${pval} ${ival}
    if ((oflag==1))
    then grep -e${pval} ${ival} > ${oval}
    fi
fi
```

Рис. 4.2: Командный файл, который анализирует командную строку



```
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ ./lab12.sh -o fgmhgh -i gfvjkgy -p gftftt -C -n
iflag = 1 ival = gfvjkgy
oflag = 1 ival = fgmhgh
pflag = 1 ival = gftftt
Cflag = 1
nflag = 1
-
```

Рис. 4.3: Анализ командной строки с ключами

2. Я написала 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 должен вызывать эту программу и, проанализировав с помощью команды `$?` , выдать сообщение о том, какое число было введено.

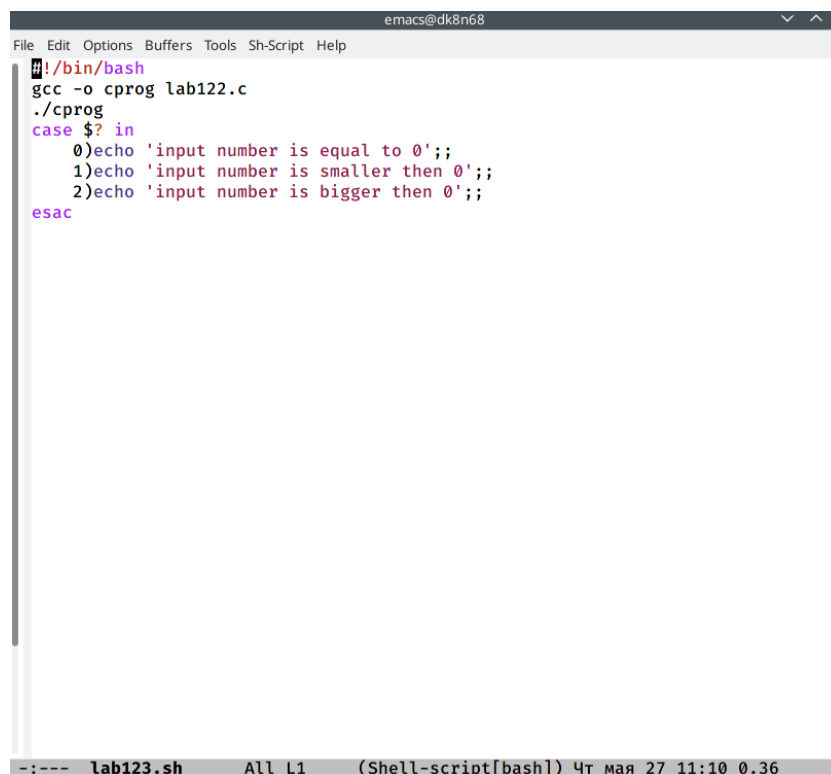
A screenshot of a text editor window showing a C program. The menu bar at the top includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'C', and 'Help'. The code is as follows:

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

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

The status bar at the bottom displays the file path 'lab122.c', the editor mode 'All L1', the encoding '(C/*l Abbrev)', and the timestamp 'Чт мая 27 11:09 0.43'.

Рис. 4.4: Программа на языке Си



```
emacs@dk8n68
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
gcc -o cprog lab122.c
./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

--:--- lab123.sh All L1 (Shell-script[bash]) Чт мая 27 11:10 0.36
```

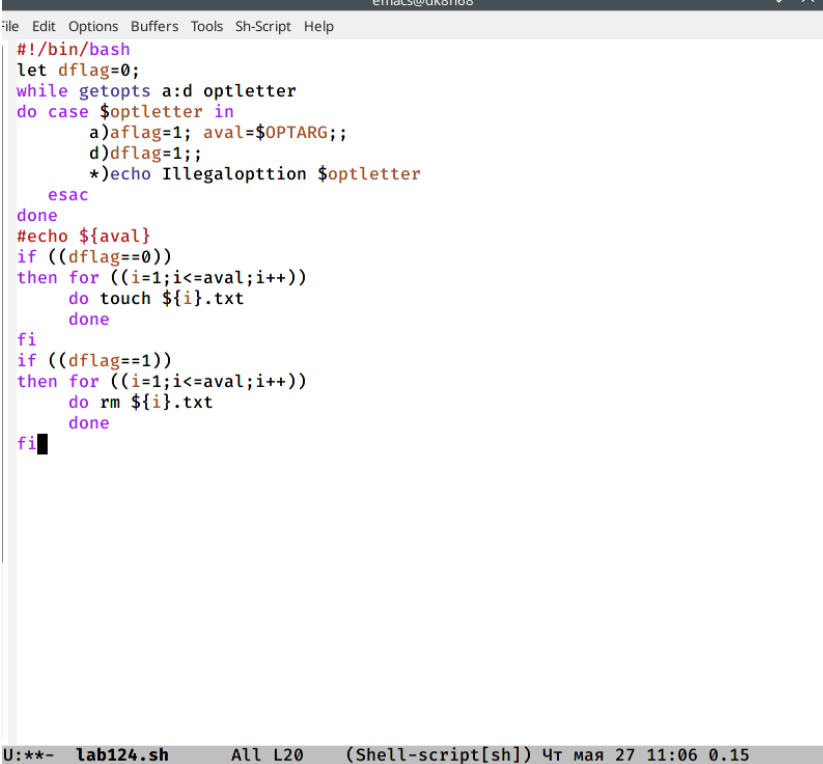
Рис. 4.5: Командный файл

```
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ bash lab123.sh
input: 0
input number is equal to 0
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ bash lab123.sh
input: 19
input number is bigger then 0
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ bas lab123.sh
bash: bas: команда не найдена
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ bash lab123.sh
input: -7
input number is smaller then 0
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $
```

Рис. 4.6: Ввод чисел

3. Я написала 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).



```
#!/bin/bash
let dflag=0;
while getopts a:d optletter
do case $optletter in
    a) aflag=1; aval=$OPTARG;;
    d) dflag=1;;
    *) echo Illegal option $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:*** lab124.sh All L20 (Shell-script[sh]) Чт мая 27 11:06 0.15

Рис. 4.7: Командный файл, создающий указанное число файлов

```

ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ bash lab124.sh -a4
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ ls -l
итого 14
-rw-r--r-- 1 ydbashkirova studsci 0 мая 27 11:07 1.txt
-rw-r--r-- 1 ydbashkirova studsci 0 мая 27 11:07 2.txt
-rw-r--r-- 1 ydbashkirova studsci 0 мая 27 11:07 3.txt
-rw-r--r-- 1 ydbashkirova studsci 0 мая 27 11:07 4.txt
-rwxr-xr-x 1 ydbashkirova studsci 8072 мая 27 10:57 cprog
-rw-r--r-- 1 ydbashkirova studsci 0 мая 27 10:19 lab121.txt
-rwxr-xr-x 1 ydbashkirova studsci 191 мая 27 10:48 lab122.c
-rwxr-xr-x 1 ydbashkirova studsci 0 мая 27 10:42 lab122.c~
-rwxr-xr-x 1 ydbashkirova studsci 191 мая 27 10:56 lab123.sh
-rwxr-xr-x 1 ydbashkirova studsci 0 мая 27 10:49 lab123.sh~
-rwxr-xr-x 1 ydbashkirova studsci 353 мая 27 11:06 lab124.sh
-rwxr-xr-x 1 ydbashkirova studsci 0 мая 27 10:58 lab124.sh~
-rw-r--r-- 1 ydbashkirova studsci 803 мая 27 10:11 lab12.sh
-rw-r--r-- 1 ydbashkirova studsci 801 мая 26 15:02 lab12.sh~
-rw-r--r-- 1 ydbashkirova studsci 171 мая 26 14:40 lab12.txt
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ bash lab124.sh -a4 -d
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ ls -l
bash: ls-l: команда не найдена
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ ls -l
итого 14
-rwxr-xr-x 1 ydbashkirova studsci 8072 мая 27 10:57 cprog
-rw-r--r-- 1 ydbashkirova studsci 0 мая 27 10:19 lab121.txt
-rwxr-xr-x 1 ydbashkirova studsci 191 мая 27 10:48 lab122.c
-rwxr-xr-x 1 ydbashkirova studsci 0 мая 27 10:42 lab122.c~
-rwxr-xr-x 1 ydbashkirova studsci 191 мая 27 10:56 lab123.sh
-rwxr-xr-x 1 ydbashkirova studsci 0 мая 27 10:49 lab123.sh~
-rwxr-xr-x 1 ydbashkirova studsci 353 мая 27 11:06 lab124.sh
-rwxr-xr-x 1 ydbashkirova studsci 0 мая 27 10:58 lab124.sh~
-rw-r--r-- 1 ydbashkirova studsci 803 мая 27 10:11 lab12.sh
-rw-r--r-- 1 ydbashkirova studsci 801 мая 26 15:02 lab12.sh~
-rw-r--r-- 1 ydbashkirova studsci 171 мая 26 14:40 lab12.txt
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ █

```

Рис. 4.8: Число файлов от 1 до N и их удаление

4. Я написала 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).

```
emacs@dk8n68
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
tar -cf 12.tar $@
tar -cf 12l.tar
find $@ -mtime -7 -exec tar -rf 12l.tar '{}' ';'
█

U:*** lab125.sh All L5 (Shell-script[sh]) Чт мая 27 11:15 0.31
```

Рис. 4.9: Командный файл,запаковывающий в архив все файлы

```
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ bash lab125.sh /lab12
tar: Удаляется начальный '/' из имен объектов
tar: /lab12: функция stat завершилась с ошибкой: Нет такого файла или каталога
tar: Завершение работы с состоянием неисправности из-за возникших ошибок
tar: Робкий отказ от создания пустого архива
Попробуйте «tar --help» или «tar --usage» для
получения более подробного описания.
find: '/lab12': Нет такого файла или каталога
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ ls
125.sh 12.tar  cprog lab121.txt lab122.c lab122.c~ lab123.sh lab123.sh~ lab124.sh lab124.sh~ lab125.sh lab125.sh~ lab12.sh lab12.sh~ lab12.txt
ydbashkirova@dk8n68 ~/laboratory/lab12/labs $ █
```

Рис. 4.10: Архив всех файлов в указанной директории

5 Выводы

Изучила основы программирования в оболочке ОС UNIX/Linux. Научилась писать небольшие командные файлы.

6 Библиография

https://esystem.rudn.ru/pluginfile.php/1142520/mod_resource/content/3/009-lab_shell_prog_2.pdf
<https://losst.ru/tsikly-bash>

7 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 `mans/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.