

Отчет по лабораторной работе 13

Дисциплина: архитектура компьютера

Алехин Давид Андреевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	12
	Список литературы	13

Список иллюстраций

4.1	1	8
4.2	2	9
4.3	3	9
4.4	4	10
4.5	5	10
4.6	6	10
4.7	7	11
4.8	8	11
4.9	9	11

Список таблиц

1 Цель работы

2 Задание

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-iinputfile` — прочитать данные из указанного файла; `-ooutputfile` — 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`).

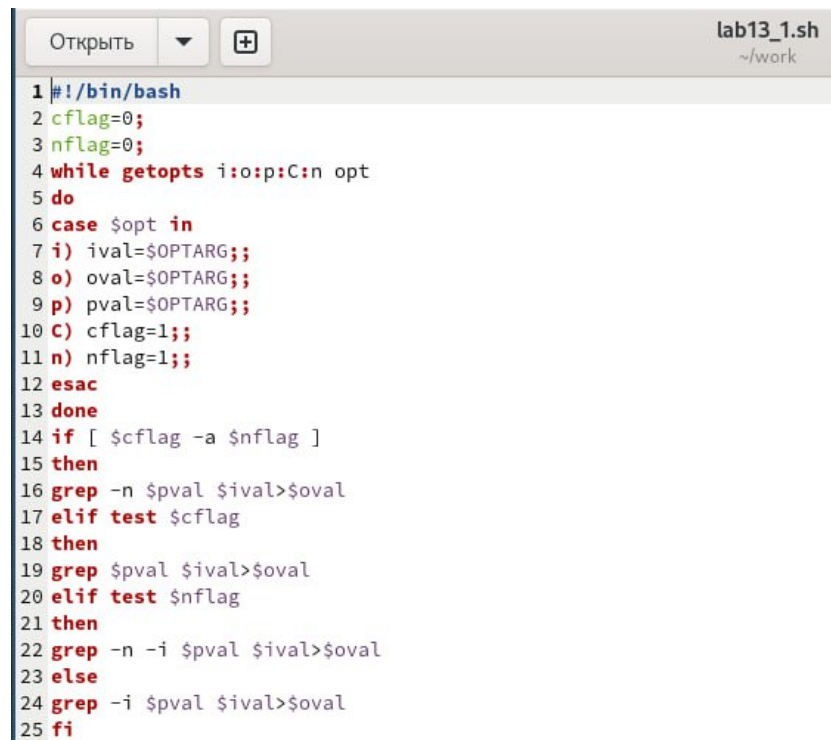
3 Теоретическое введение

4 Выполнение лабораторной работы

Используя команды `getopts` `grep` напишем командный файл, который анализирует командную строку с ключами и выполним его: `-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 (рис. 4.1, 4.2).

```
[davidalekhin@daalekhin work]$ chmod +x lab13_1.sh
[davidalekhin@daalekhin work]$ ./lab13_1.sh -i conf.txt -o fout.txt -p files -C -n
[davidalekhin@daalekhin work]$ |
```

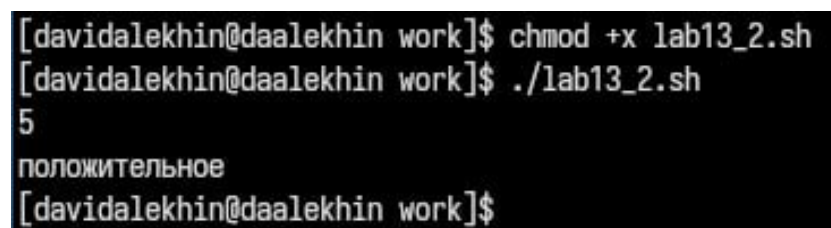
Рис. 4.1: 1



```
1 #!/bin/bash
2 cflag=0;
3 nflag=0;
4 while getopts i:o:p:C:n opt
5 do
6 case $opt in
7 i) ival=$OPTARG;;
8 o) oval=$OPTARG;;
9 p) pval=$OPTARG;;
10 C) cflag=1;;
11 n) nflag=1;;
12 esac
13 done
14 if [ $cflag -a $nflag ]
15 then
16 grep -n $pval $ival>$oval
17 elif test $cflag
18 then
19 grep $pval $ival>$oval
20 elif test $nflag
21 then
22 grep -n -i $pval $ival>$oval
23 else
24 grep -i $pval $ival>$oval
25 fi
```

Рис. 4.2: 2

Напишем сначала на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем завершим программу при помощи функции `exit(n)`, передавая информацию о коде завершения в оболочку. Командный файл вызовет эту программу и, проанализировав с помощью команды `$?`, выдаст сообщение о том, какое число было введено (рис. 4.3, 4.4, 4.5).



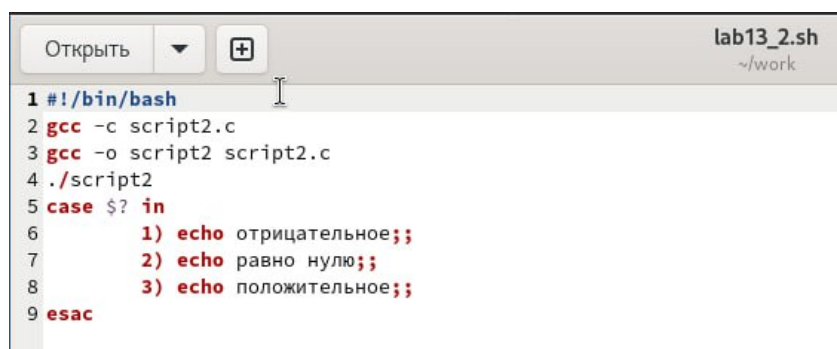
```
[davidalekhin@daalekhin work]$ chmod +x lab13_2.sh
[davidalekhin@daalekhin work]$ ./lab13_2.sh
5
положительное
[davidalekhin@daalekhin work]$
```

Рис. 4.3: 3



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main ()
5 {
6     int o;
7     scanf ("%d", &o);
8     if (o < 0) {
9         exit(1);
10    } else if (o > 0) {
11        exit(3);
12    }
13    exit(2);
14 }
```

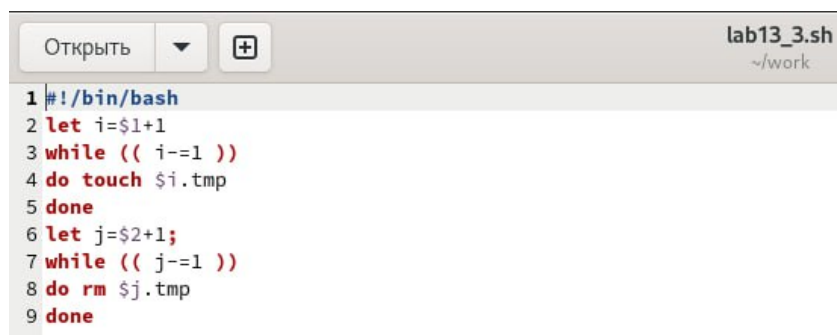
Рис. 4.4: 4



```
1 #!/bin/bash
2 gcc -c script2.c
3 gcc -o script2 script2.c
4 ./script2
5 case $? in
6     1) echo отрицательное;;
7     2) echo равно нулю;;
8     3) echo положительное;;
9 esac
```

Рис. 4.5: 5

Напишем командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (рис. 4.6, 4.7).



```
1 #!/bin/bash
2 let i=$1+1
3 while (( i--=1 ))
4 do touch $i.tmp
5 done
6 let j=$2+1;
7 while (( j--=1 ))
8 do rm $j.tmp
9 done
```

Рис. 4.6: 6

```

[dauidalekhin@daalekhin work]$ chmod +x lab13_3.sh
[dauidalekhin@daalekhin work]$ ./lab13_3.sh
[dauidalekhin@daalekhin work]$ ls
conf.txt  git-extended  lab06      lab12_2.sh  lab12_4.sh  lab13_2.sh  os      script2.c  study
fout.txt  hello.sh      lab12_1.sh  lab12_3.sh  lab13_1.sh  lab13_3.sh  script2  script2.o  text.txt
[dauidalekhin@daalekhin work]$

```

Рис. 4.7: 7

Напишем командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории. Модифицируем его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад. (рис. 4.8, 4.9).



```

1 #!/bin/bash
2 (find $1 -mtime -7 -daystart) | xargs tar -cf arhiv.tar

```

Рис. 4.8: 8

```

[dauidalekhin@daalekhin work]$ chmod +x lab13_4.sh
[dauidalekhin@daalekhin work]$ ./lab13_4.sh
tar: ./arhiv.tar: archive cannot contain itself; not dumped
[dauidalekhin@daalekhin work]$ ls
arhiv.tar  fout.txt  hello.sh  lab12_1.sh  lab12_3.sh  lab13_1.sh  lab13_3.sh  os      script2.c  study
conf.txt  git-extended  lab06      lab12_2.sh  lab12_4.sh  lab13_2.sh  lab13_4.sh  script2  script2.o  text.txt
[dauidalekhin@daalekhin work]$

```

Рис. 4.9: 9

5 Выводы

В данной работе мы изучили основы программирования в оболочке ОС UNIX и писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Список литературы

<https://esystem.rudn.ru/>