

Операционные системы

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Алиев Руслан Нияз оглы

18 апреля 2025

Российский университет дружбы народов, Москва, Россия

Цели и задачи работы

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

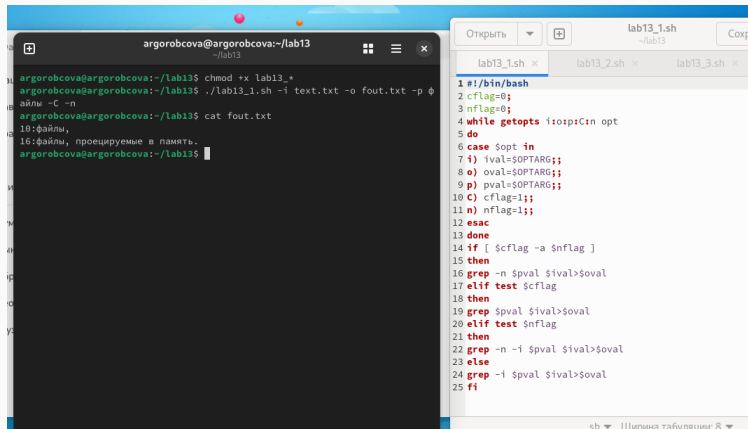
1 Выполнить 4 задания

Процесс выполнения лабораторной работы

1. 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; `-r шаблон` — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк;

а затем ищет в указанном файле нужные строки

Выполнение работы



The image shows a terminal window and a code editor side-by-side. The terminal window, titled 'argorbcova@argorbcova: ~/lab13', displays the following commands and output:

```
argorbcova@argorbcova:~/lab13$ chmod +x lab13_*
argorbcova@argorbcova:~/lab13$ ./lab13_1.sh -i text.txt -o fout.txt -p ф
айлы -C -n
argorbcova@argorbcova:~/lab13$ cat fout.txt
10:файлы,
16:файлы, проецируемые в память.
argorbcova@argorbcova:~/lab13$
```

The code editor, titled 'lab13_1.sh', shows the source code of the script:

```
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. 1: Задание 1

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

Выполнение работы

The image shows a terminal window on the left and a code editor on the right. The terminal window, titled 'argorobcova@argorobcova:~/lab13', shows the execution of two shell scripts. The first script, 'lab13_1.sh', takes 'text.txt' as input and produces 'fout.txt' with the output: '10:файлы,' and '16:файлы, проецируемые в память.'. The second script, 'lab13_2.sh', takes '6' as input and produces the output: 'положительное' and '-7'. The code editor on the right, titled 'lab13_2.sh', shows the source code of the second script, which uses a case statement to echo 'отрицательное;;', 'равно нулю;;', or 'положительное;;' based on the input value.

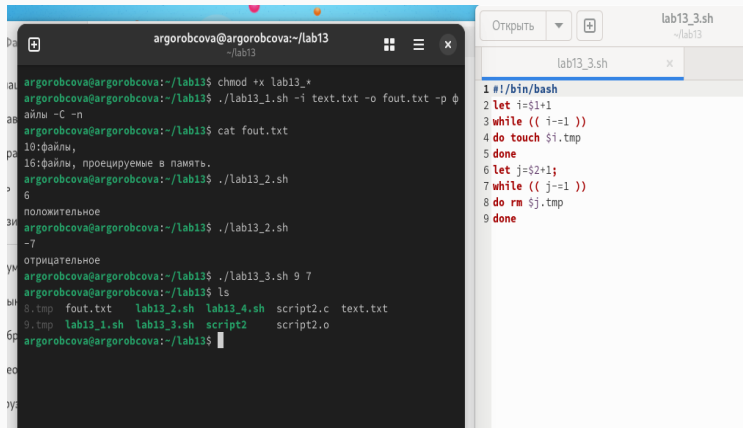
```
argorobcova@argorobcova:~/lab13$ chmod +x lab13_*
argorobcova@argorobcova:~/lab13$ ./lab13_1.sh -i text.txt -o fout.txt -p файлы -C -n
argorobcova@argorobcova:~/lab13$ cat fout.txt
10:файлы,
16:файлы, проецируемые в память.
argorobcova@argorobcova:~/lab13$ ./lab13_2.sh 6
положительное
argorobcova@argorobcova:~/lab13$ ./lab13_2.sh -7
отрицательное
argorobcova@argorobcova:~/lab13$
```

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

Рис. 2: Задание 2

3. Напишем командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N

Выполнение работы



The image shows a terminal window on the left and a file editor on the right. The terminal window has a title bar 'argorobcova@argorobcova:~/lab13' and shows the following commands and output:

```
argorobcova@argorobcova:~/lab13$ chmod +x lab13_*
argorobcova@argorobcova:~/lab13$ ./lab13_1.sh -i text.txt -o fout.txt -p ф
айлы -C -n
argorobcova@argorobcova:~/lab13$ cat fout.txt
10:файлы,
16:файлы, проецируемые в память.
argorobcova@argorobcova:~/lab13$ ./lab13_2.sh
6
положительное
argorobcova@argorobcova:~/lab13$ ./lab13_2.sh
-7
отрицательное
argorobcova@argorobcova:~/lab13$ ./lab13_3.sh 9 7
argorobcova@argorobcova:~/lab13$ ls
8.tmp  fout.txt  lab13_2.sh  lab13_4.sh  script2.c  text.txt
9.tmp  lab13_1.sh  lab13_3.sh  script2    script2.o
argorobcova@argorobcova:~/lab13$
```

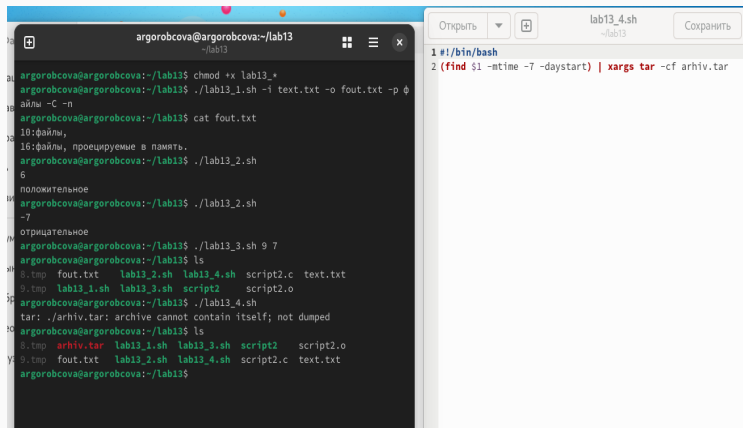
The file editor on the right has a title bar 'lab13_3.sh' and shows the following script code:

```
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. 3: Задание 3

4. 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.

Выполнение работы



The image shows a terminal window and a file manager side-by-side. The terminal window, titled 'argorbcova@argorbcova:~/lab13', displays the following commands and output:

```
argorbcova@argorbcova:~/lab13$ chmod +x lab13.*
argorbcova@argorbcova:~/lab13$ ./lab13_1.sh -i text.txt -o fout.txt -p ф
айлы -C -п
argorbcova@argorbcova:~/lab13$ cat fout.txt
10:файлы,
16:файлы, проецируемые в память.
argorbcova@argorbcova:~/lab13$ ./lab13_2.sh
6
положительное
argorbcova@argorbcova:~/lab13$ ./lab13_2.sh
-7
отрицательное
argorbcova@argorbcova:~/lab13$ ./lab13_3.sh 9 7
argorbcova@argorbcova:~/lab13$ ls
8.tmp  fout.txt  lab13_2.sh  lab13_4.sh  script2.c  text.txt
9.tmp  lab13_1.sh  lab13_3.sh  script2     script2.o
argorbcova@argorbcova:~/lab13$ ./lab13_4.sh
tar: ./arhiv.tar: archive cannot contain itself; not dumped
argorbcova@argorbcova:~/lab13$ ls
8.tmp  arhiv.tar  lab13_1.sh  lab13_3.sh  script2     script2.o
9.tmp  fout.txt  lab13_2.sh  lab13_4.sh  script2.c  text.txt
argorbcova@argorbcova:~/lab13$
```

The file manager window, titled 'lab13_4.sh', shows the following commands:

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

Рис. 4: Задание 4

Выводы по проделанной работе

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