

Операционные системы

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Анастасия Гончарь

17 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.

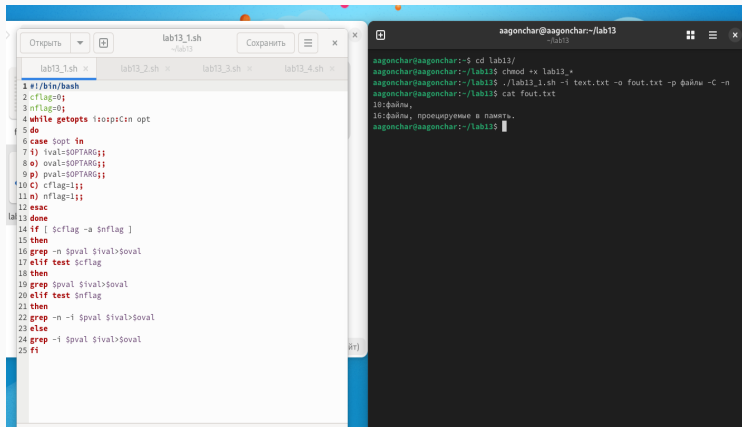
1 Выполнить 4 задания

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; `-p 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 two terminal windows side-by-side. The left window, titled 'lab13_1.sh', displays a shell script with 25 lines of code. The script starts with a shebang, initializes flags, and uses a while loop to process command-line options. It then uses a series of grep commands to check for specific patterns in a file. The right window, titled 'aagonchar@aagonchar:~/lab13', shows the execution of the script. It changes to the lab13 directory, makes the script executable, and runs it with 'text.txt' as input. The output shows the script processing the file and displaying the results.

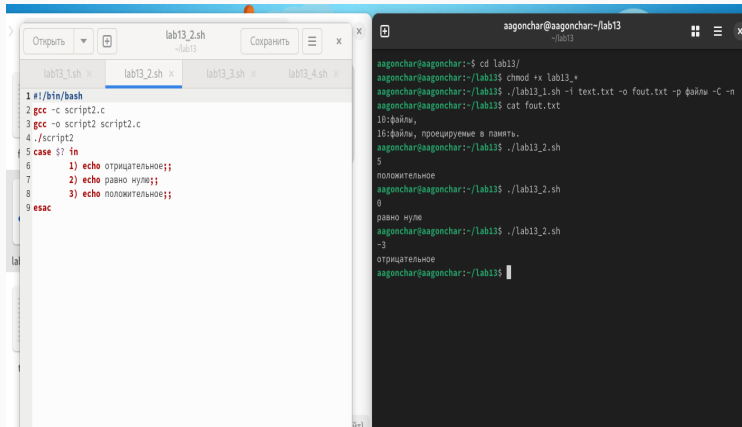
```
lab13_1.sh
1 #!/bin/bash
2 cflag=0;
3 nflag=0;
4 while getopts i:oi:p:C:n opt
5 do
6 case $opt in
7 i) ival=$OPTARG;;
8 o) oval=$OPTARG;;
9 p) pval=$OPTARG;;
10 C) cflag=1;;
11 n) nflag=1;;
12 esac
13 done
14 if [ $cflag -a $nflag ]
15 then
16 grep -n $pval $ival>$oval
17 elif test $cflag
18 then
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
```

```
aagonchar@aagonchar:~/lab13
aagonchar@aagonchar:~$ cd lab13/
aagonchar@aagonchar:~/lab13$ chmod +x lab13.*
aagonchar@aagonchar:~/lab13$ ./lab13_1.sh -i text.txt -o fout.txt -p файлн -C -n
aagonchar@aagonchar:~/lab13$ cat fout.txt
18:файлн,
16:файлн, проецируемые в память.
aagonchar@aagonchar:~/lab13$
```

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 two terminal windows side-by-side. The left window, titled 'lab13_2.sh', contains a shell script with a 'case' statement. The right window, titled 'aagonchar@aagonchar:~/lab13', shows the execution of this script with various arguments, resulting in different outputs based on the case.

```
lab13_2.sh
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
```

```
aagonchar@aagonchar:~/lab13
aagonchar@aagonchar:~$ cd lab13/
aagonchar@aagonchar:~/lab13$ chmod +x lab13_*
aagonchar@aagonchar:~/lab13$ ./lab13_1.sh -i text.txt -o fout.txt -p файлы -C -n
aagonchar@aagonchar:~/lab13$ cat fout.txt
10:файлы,
16:файлы, проецируемые в память.
aagonchar@aagonchar:~/lab13$ ./lab13_2.sh
5
положительное
aagonchar@aagonchar:~/lab13$ ./lab13_2.sh
0
равно нулю
aagonchar@aagonchar:~/lab13$ ./lab13_2.sh
-3
отрицательное
aagonchar@aagonchar:~/lab13$
```

Рис. 2: Задание 2

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

Выполнение работы

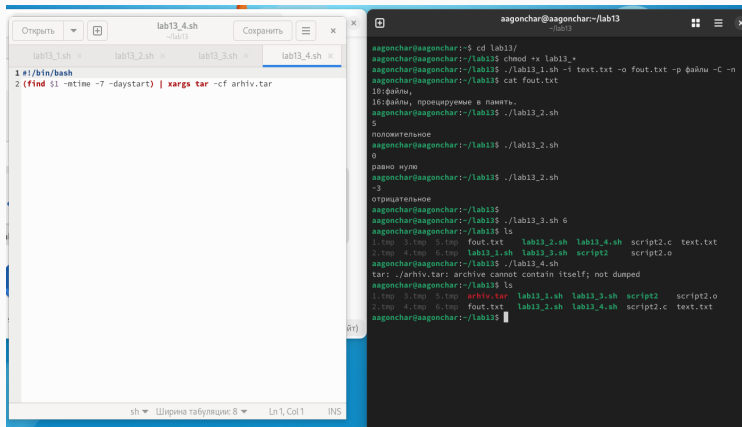
```
lab13_3.sh
~/lab13
lab13_1.sh x lab13_2.sh x lab13_3.sh x lab13_4.sh x
1 #!/bin/bash
2 let i=$1+1
3 while (( i-=1 ))
4 do touch $i.tmp
5 done
6 let j=$2+1
7 while (( j-=1 ))
8 do rm $j.tmp
9 done

aagonchar@aagonchar:~$ cd lab13/
aagonchar@aagonchar:~/lab13$ chmod +x lab13_*
aagonchar@aagonchar:~/lab13$ ./lab13_1.sh -i text.txt -o fout.txt -p файлы -C -n
aagonchar@aagonchar:~/lab13$ cat fout.txt
10:файлы,
16:файлы, проецируемые в память.
aagonchar@aagonchar:~/lab13$ ./lab13_2.sh
5
положительное
aagonchar@aagonchar:~/lab13$ ./lab13_2.sh
0
равно нулю
aagonchar@aagonchar:~/lab13$ ./lab13_2.sh
-3
отрицательное
aagonchar@aagonchar:~/lab13$
aagonchar@aagonchar:~/lab13$ ./lab13_3.sh 6
aagonchar@aagonchar:~/lab13$ ls
1.tmp 3.tmp 5.tmp fout.txt lab13_2.sh lab13_4.sh script2.c text.txt
2.tmp 4.tmp 6.tmp lab13_1.sh lab13_3.sh script2 script2.o
aagonchar@aagonchar:~/lab13$
```

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



```
lab13_4.sh
~/.lab13

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

sh Ширина табуляции: 8 Ln 1, Col 1 INS
```

```
aagonchar@aagonchar:~/lab13

aagonchar@aagonchar:~$ cd lab13/
aagonchar@aagonchar:~/lab13$ chmod +x lab13_*
aagonchar@aagonchar:~/lab13$ ./lab13_1.sh -i text.txt -o fout.txt -p файл -C -n
aagonchar@aagonchar:~/lab13$ cat fout.txt
10:файлы,
16:файлы, проецируемые в память.
aagonchar@aagonchar:~/lab13$ ./lab13_2.sh
5
положительное
aagonchar@aagonchar:~/lab13$ ./lab13_2.sh
0
равно нулю
aagonchar@aagonchar:~/lab13$ ./lab13_2.sh
-3
отрицательное
aagonchar@aagonchar:~/lab13$
aagonchar@aagonchar:~/lab13$ ./lab13_3.sh 6
aagonchar@aagonchar:~/lab13$ ls
1:tap 3:tap 5:tap fout.txt lab13_2.sh lab13_4.sh script2.c text.txt
2:tap 4:tap 6:tap lab13_1.sh lab13_3.sh script2 script2.o
aagonchar@aagonchar:~/lab13$ ./lab13_4.sh
tar: ./archiv.tar: archive cannot contain itself; not dumped
aagonchar@aagonchar:~/lab13$ ls
1:tap 3:tap 5:tap archiv.tar lab13_1.sh lab13_3.sh script2 script2.o
2:tap 4:tap 6:tap fout.txt lab13_2.sh lab13_4.sh script2.c text.txt
aagonchar@aagonchar:~/lab13$
```

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.