

Лабораторная работа 13

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

Неустроева Ирина Николаевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Вывод	14

Список иллюстраций

3.1	Создание файла	7
3.2	Исполнение	7
3.3	Скрипт	8
3.4	Проверка работы	8
3.5	Скрипт Си	9
3.6	Скрипт	10
3.7	Проверка работы	10
3.8	Создание 7 файлов	11
3.9	Скрипт	11
3.10	Проверка	12
3.11	Скрипт	12
3.12	Проверка работы	13

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 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; – `-rшаблон` — 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 `o` 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 `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`).

3 Выполнение лабораторной работы

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`.

Создадим файл для скрипта, введём в него код (предварительно присвоив ему право на исполнение с помощью команды `chmod`) (рис. 3.1).

```
inneustroeva@inneustroeva:~$ touch 11.1.sh
inneustroeva@inneustroeva:~$ chmod +x 11.1.sh
inneustroeva@inneustroeva:~$ gedit 11.1.sh
inneustroeva@inneustroeva:~$ touch input.txt
bash: touch: команда не найдена
inneustroeva@inneustroeva:~$ touch input.txt
inneustroeva@inneustroeva:~$ touch output.txt
```

Рис. 3.1: Создание файла

Вызовем файл на исполнение (рис. 3.2).

```
inneustroeva@inneustroeva:~$ bash 11.1.sh -p поют -i input.txt -o output.txt -C -n
inneustroeva@inneustroeva:~$
```

Рис. 3.2: Исполнение

Прилагаю скрипт (рис. 3.3).

```

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
26

```

Рис. 3.3: Скрипт

Проверим работу данного файла (рис. 3.4). (рис. 3.5).

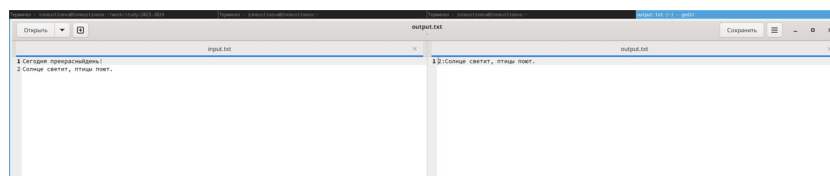
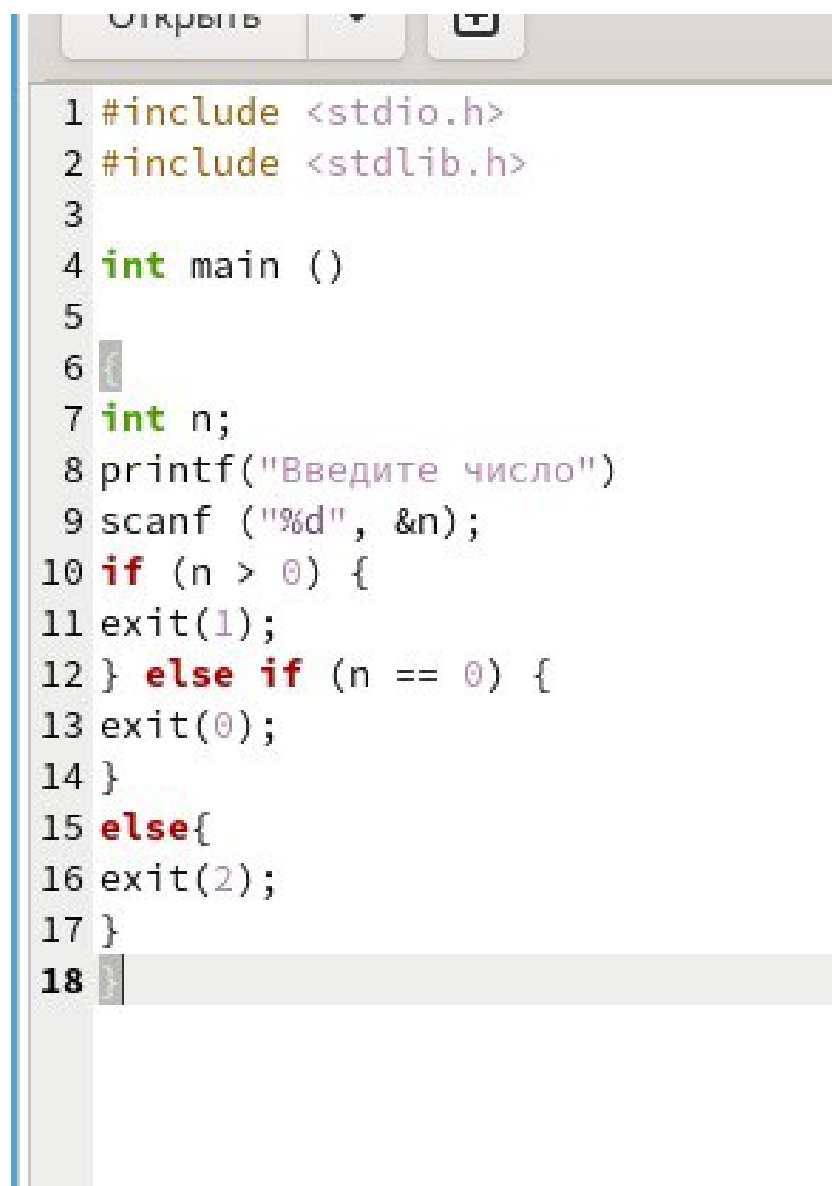


Рис. 3.4: Проверка работы

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. (рис. 3.5).

Создаём файл для скрипта и файл для программы на языке Си, присваиваем командному файлу право на исполнение, вызываем его. Далее пишем программу на языке Си



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main ()
5
6 {
7     int n;
8     printf("Введите число")
9     scanf ("%d", &n);
10    if (n > 0) {
11        exit(1);
12    } else if (n == 0) {
13        exit(0);
14    }
15    else{
16        exit(2);
17    }
18 }
```

Рис. 3.5: Скрипт Си

Теперь пишем скрипт в командном файле. (рис. 3.6).

```
1 #!/bin/bash
2
3 gcc -o cprog 12.c
4 ./cprog
5 case $? in
6     1) echo равно нулю;;
7     2) echo положительно;;
8     3) echo отрицательно;;
9 esac
```

Рис. 3.6: Скрипт

- Проверим работу данного файла (рис. 3.7).

```
inneustroeva@inneustroeva:~$ bash 11.2.sh
Введите число 5
равно нулю
inneustroeva@inneustroeva:~$ gedit 11.2.sh
inneustroeva@inneustroeva:~$ bash 11.2.sh
Введите число 7
положительно
inneustroeva@inneustroeva:~$ bash 11.2.sh
Введите число 0
равно нулю
inneustroeva@inneustroeva:~$
```

Рис. 3.7: Проверка работы

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).

Для начала создаём командный файл, присваиваем ему право на исполнение. Проверяем, корректно ли отработал код. В домашней папке должны создаться 7

файла при вызове командного файла на исполнение и передаче ему 7 в качестве аргумента (рис. 3.8).

```
inneustroeva@inneustroeva:~$ touch 11.3.sh
inneustroeva@inneustroeva:~$ chmod +x 11.3.sh
inneustroeva@inneustroeva:~$ gedit 11.3.sh
inneustroeva@inneustroeva:~$ bash 11.3.sh 7
inneustroeva@inneustroeva:~$ ls
11.1.sh  11.3.sh  5.tmp      backup    bin
11.2.c   1.tmp    6.tmp      bash1.sh  cc1plus
11.2.C   2.tmp    7.tmp      bash2.sh  conf.txt
11.2.o   3.tmp    abc1       bash3.sh  '#emacs#'
11.2.sh  4.tmp    australia  bash4.sh  feathers
inneustroeva@inneustroeva:~$
```

Рис. 3.8: Создание 7 файлов

Пишем скрипт (рис. 3.9).

```
Открыть ▼ +
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. 3.9: Скрипт

Теперь проверим, удалит ли файлы наш скрипт (при вызове командного файла на исполнение и передаче ему 2 в качестве аргумента созданные только что файлы должны быть удалены)(рис. 3.10).

```
inneustroeva@inneustroeva:~$ ls
11.1.sh  arhiv      backup     cprog
11.2.c   'arhiv (1) ' bash1.sh   '#emacs#'
11.2.o   'arhiv (2) ' bash2.sh   feathers
11.2.sh  'arhiv (3) ' bash3.sh   'feh_002838_000
11.3.sh  arhive     bash4.sh   feh_003914_000
11.4.sh  archive.tar bin         'feh_003974_000
12.c     arhiv.tar  cc1plus    'feh_004014_000
abc1     australia conf.txt    feh_006234_000
inneustroeva@inneustroeva:~$
```

Рис. 3.10: Проверка

4. Написать 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).

Пишем скрипт (рис. 3.11).

```
1 #!/bin/bash
2 find $* -ntime -7 -mtime +0 -type f > FILES.txt
3 tar -cf archive.tar -T FILES.txt
```

Рис. 3.11: Скрипт

Проверяем его работу (рис. 3.12).

```
inneustroeva@inneustroeva:~$ bash 11.4.sh /home/inneustroeva/  
tar: Удаляется начальный '/' из имен объектов  
tar: Удаляются начальные '/' из целей жестких ссылок  
inneustroeva@inneustroeva:~$ ls  
11.1.sh 2.tmp 'arhiv (1)' bash1.sh '#emacs#'  
11.2.c 3.tmp 'arhiv (2)' bash2.sh feathers  
11.2.o 4.tmp 'arhiv (3)' bash3.sh 'feh_002838_000001_ca  
11.2.sh 5.tmp archive bash4.sh feh_003914_000001_fi  
11.3.sh 6.tmp archive.tar bin 'feh_003974_000001_ca  
11.4.sh 7.tmp arhiv.tar cc1plus 'feh_004014_000001_ca  
12.c abc1 australia conf.txt feh_006234_000002_fi  
1.tmp arhiv backup cprog FILES.txt  
inneustroeva@inneustroeva:~$
```

Рис. 3.12: Проверка работы

4 Вывод

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