

Отчет по лабораторной работе №13

Дисциплина: Операционные системы

Иванов Сергей Владимирович

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	7
4	Вывод	15

Список иллюстраций

3.1	Создание файла	7
3.2	Код программы	8
3.3	Результат работы программы	9
3.4	Создание файла	9
3.5	Код программы на Си	10
3.6	Код программы	11
3.7	Результат работы программы	12
3.8	Создание файла	12
3.9	Код программы	12
3.10	Результат работы программы	13
3.11	Код программы	13
3.12	Результат работы программы	14

1 Цель работы

Цель данной лабораторной работы - изучить основы программирования в оболочке ОС UNIX, научиться 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;
 - `-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`.
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 `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`).

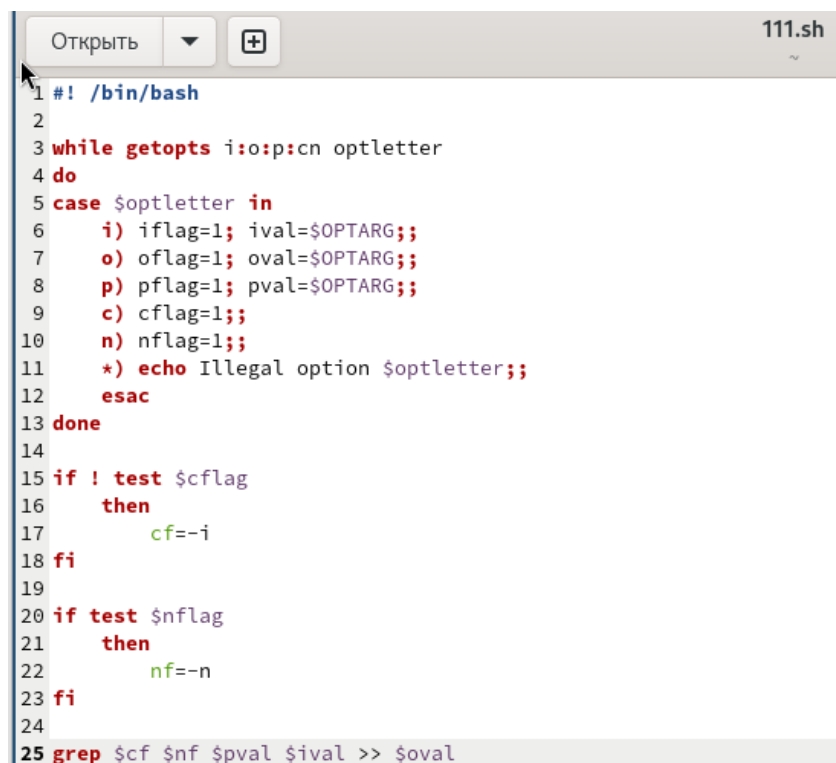
3 Выполнение лабораторной работы

Создаю файл с разрешением на исполнение (рис. 1).

```
[svivanov1@svivanov1 ~]$ touch 111.sh  
[svivanov1@svivanov1 ~]$ chmod +x 111.sh  
[svivanov1@svivanov1 ~]$ bash 111.sh -p улит -i input.txt -o output.txt -c -n
```

Рис. 3.1: Создание файла

Командный файл, с командами `getopts` и `grep`, который анализирует командную строку с ключами: - `-iinputfile` — прочитать данные из указанного файла; - `-ooutputfile` — вывести данные в указанный файл; - `-ршаблон` — 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. 2).



```
1 #!/bin/bash
2
3 while getopts i:o:p:cn optletter
4 do
5     case $optletter in
6         i) iflag=1; ival=$OPTARG;;
7         o) oflag=1; oval=$OPTARG;;
8         p) pflag=1; pval=$OPTARG;;
9         c) cflag=1;;
10        n) nflag=1;;
11        *) echo Illegal option $optletter;;
12    esac
13 done
14
15 if ! test $cflag
16 then
17     cf=-i
18 fi
19
20 if test $nflag
21 then
22     nf=-n
23 fi
24
25 grep $cf $nf $pval $ival >> $oval
```

Рис. 3.2: Код программы

```
#!/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;;
```

```
    *) echo Illegal option $optletter;;
```

```
esac
```

```
done
```



```

if ! test $cflag
then
    cf=-i
fi

if test $nflag
then
    nf=-n
fi

grep $cf $nf $pval $ival >> $oval

```

Результат работы программы в файле output.txt (рис. 3).

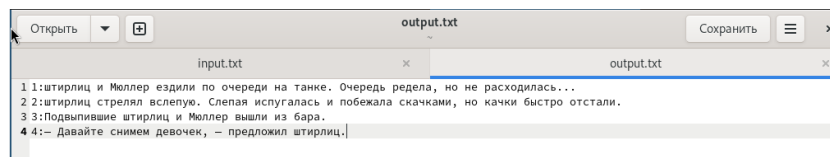


Рис. 3.3: Результат работы программы

Создаю исполняемый файл для второй программы, также создаю файл 12.crr для программы на С (рис. 4).

```

[svivanov1@svivanov1 ~]$ touch 112.sh
[svivanov1@svivanov1 ~]$ chmod +x 112.sh
[svivanov1@svivanov1 ~]$ touch 12.crr
[svivanov1@svivanov1 ~]$ bash 112.sh

```

Рис. 3.4: Создание файла

Пишу программу на языке Си, 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 (рис. 5).

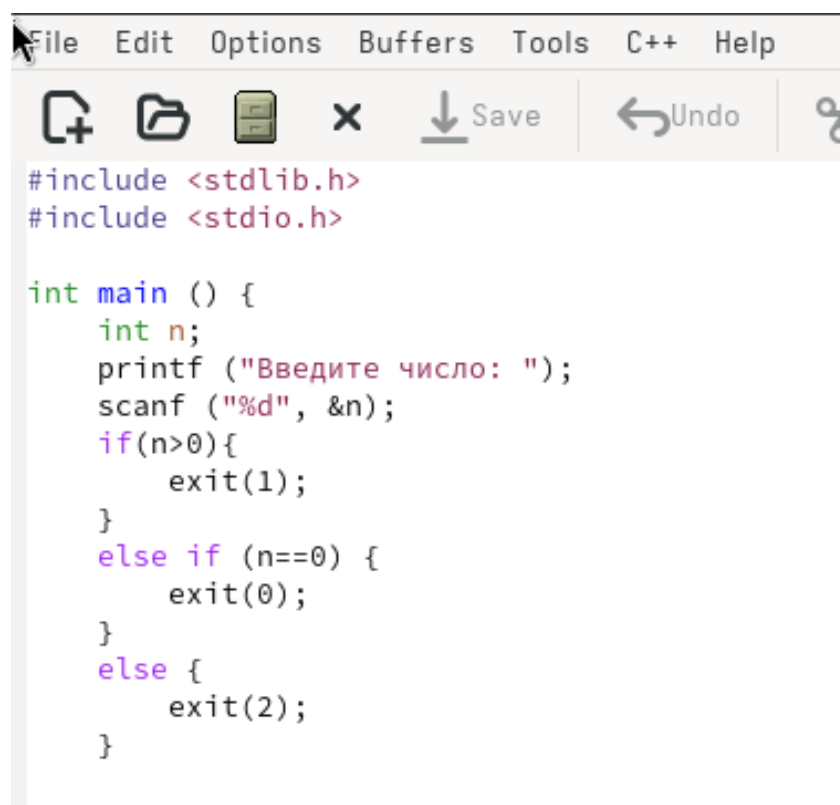


Рис. 3.5: Код программы на Си

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

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

```
int main () {
```

```
    int n;
```

```
    printf ("Введите число: ");
```

```
    scanf ("%d", &n);
```

```
    if(n>0){
```

```
        exit(1);
```

```
    }
```

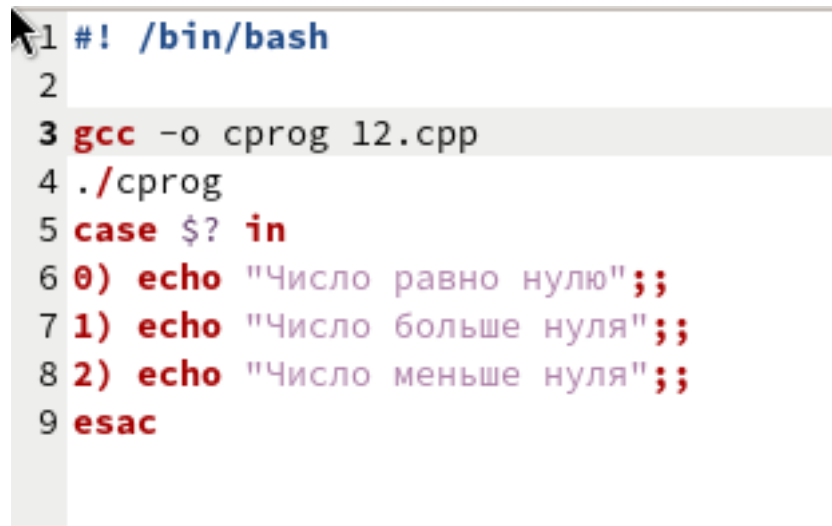
```
    else if (n==0) {
```

```
        exit(0);
```

```
    }
```

```
else {  
    exit(2);  
}
```

Командный файл должен вызывать эту программу и, проанализировав с помощью команды \$?, выдать сообщение о том, какое число было введено (рис. 6).



```
1 #!/bin/bash  
2  
3 gcc -o cprog 12.cpp  
4 ./cprog  
5 case $? in  
6 0) echo "Число равно нулю";;  
7 1) echo "Число больше нуля";;  
8 2) echo "Число меньше нуля";;  
9 esac
```

Рис. 3.6: Код программы

```
#!/bin/bash
```

```
gcc -o cprog 12.c
```

```
./cprog
```

```
case $? in
```

```
0) echo "Число равно нулю";;
```

```
1) echo "Число больше нуля";;
```

```
2) echo "Число меньше нуля";;
```

```
esac
```

Программа работает корректно (рис. 7).

```
[svivanov1@svivanov1 ~]$ bash 112.sh
Введите число: 95
Число больше нуля
[svivanov1@svivanov1 ~]$
```

Рис. 3.7: Результат работы программы

Создаю исполняемый файл для третьей программы (рис. 8).

```
[svivanov1@svivanov1 ~]$ touch 113.sh
[svivanov1@svivanov1 ~]$ chmod +x 113.sh
[svivanov1@svivanov1 ~]$
```

Рис. 3.8: Создание файла

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) (рис. 9).

```
1 #!/bin/bash
2 for((i=1; i<=$*; i++))
3 do
4 if test -f "$i".tmp
5 then rm "$i".tmp
6 else touch "$i.tmp"
7 fi
8 done
```

Рис. 3.9: Код программы

```
#!/bin/bash
for((i=1; i<=$*; i++))
do
if test -f "$i".tmp
```

```

then rm "$i".tmp
else touch "$i.tmp"
fi
done

```

Проверяю, что программа создала файлы и удалила их при соответствующих запросах (рис. 10).

```

svivanov1@svivanov1 ~]$ bash 113.sh 4
svivanov1@svivanov1 ~]$ ls
111.sh  1.tmp  bin      '##lab07.sh##'  prog1.sh  text.txt  work5  'Общедоступные'
112.sh  2.tmp  blog     '##lab07.sh##'  prog2.sh  touch     Видео  'Рабочий стол'
113.sh  3.tmp  cprog    '##lab07.sh##'  prog3.sh  work      Документы  Шаблоны
12.cpp  4.tmp  fun      my_os           prog4.sh  work2     Загрузки
12.cpp~ australia git-extended  output.txt  ski.places  work3     Изображения
12.cpp~ backup  input.txt  package.json  spring      work4     Музыка
svivanov1@svivanov1 ~]$ bash 113.sh 4
svivanov1@svivanov1 ~]$ ls
111.sh  australia  git-extended  output.txt  ski.places  work3  Изображения
112.sh  backup     input.txt     package.json  spring      work4  Музыка
113.sh  bin        '##lab07.sh##'  prog1.sh    text.txt    work5  'Общедоступные'
12.cpp  cprog      '##lab07.sh##'  prog2.sh    touch       Видео  'Рабочий стол'
12.cpp~ fun        my_os          prog3.sh    work         Документы  Шаблоны
12.cpp~      Загрузки
svivanov1@svivanov1 ~]$

```

Рис. 3.10: Результат работы программы

Создаю исполняемый файл для четвертой программы. Это 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) (рис. 11).

```

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

```

Рис. 3.11: Код программы

```

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

```

Проверяю работу программы (рис. 12).

```
[svivanov1@svivanov1 ~]$ bash 114.sh /home/svivanov1  
tar: Удаляется начальный '/' из имен объектов  
tar: Удаляются начальные '/' из целей жестких ссылок  
[svivanov1@svivanov1 ~]$
```

Рис. 3.12: Результат работы программы

4 Вывод

При выполнении данной лабораторной работы я изучил основы программирования в оболочке ОС UNIX, научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.